**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Кваліфікаційна наукова праця
на правах рукопису

**ПОНЗЕЛЬ ЯРОСЛАВ ЮРІЙОВИЧ**

УДК 004.9:37.016

ДИСЕРТАЦІЯ

# РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ПОБУДОВИ ІНДИВІДУАЛЬНОГО ПЛАНУ НАВЧАННЯ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

122 «Комп’ютерні науки»
12 «Інформаційні технології»

Подається на здобуття наукового ступеня доктор філософії

Дисертація містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело Я. Ю. Понзель

Науковий керівник:
**ГЛАЗУНОВА Олена Григорівна**,
доктор педагогічних наук, професор

Київ – 2025

# АНОТАЦІЯ

**Понзель Я. Ю. Рекомендаційна система для побудови індивідуального плану навчання здобувача вищої освіти.** Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктор філософії за спеціальністю 122 «Комп'ютерні науки». Національний університет біоресурсів і природокористування України, Київ, 2025.

У дисертації досліджено методи та моделі обробки даних, алгоритми машинного навчання та підходи до побудови рекомендаційних систем у контексті забезпечення індивідуалізації освітнього процесу. Окреслені теоретичні засади формують методологічну основу для створення інтелектуальної системи, що дозволяє автоматизувати формування індивідуального навчального плану (ІНП), сприяючи підвищенню ефективності планування освітньої траєкторії, оптимізації вибору дисциплін та адаптації навчання до професійних інтересів здобувачів. Впровадження розроблених моделей у практику сприятиме цифровізації університетів, зменшенню адміністративного навантаження та підвищенню мотивації студентів через персоналізацію навчання.

На основі реальних даних навчального процесу (з системи «Електронний деканат» та ЄДЕБО) сформовано емпіричну базу дослідження, що охоплює вибірку з 2543 здобувачів вищої освіти. Це дозволило провести глибокий аналіз успішності та освітніх вподобань, виявити приховані закономірності у виборі дисциплін та побудувати надійні прогностичні моделі. Результати аналізу підтвердили необхідність переходу від уніфікованих навчальних планів до гнучких траєкторій, що базуються на інтелектуальному аналізі великих даних.

Встановлено, що найбільш ефективним підходом для прогнозування успішності в умовах розрідженості даних (що є характерним для вибіркових дисциплін) є метод матричної факторизації з регуляризацією за нормою Фробеніуса. Розроблено та програмно реалізовано алгоритм, який мінімізує функцію втрат (RMSE) та дозволяє виявляти потенційні труднощі у навчанні на

ранніх етапах. Експериментальна перевірка на тестовій вибірці показала, що середня похибка прогнозування (RMSE) становить 13.37 балів (на 100-бальній шкалі), а точність потрапляння рекомендованої дисципліни у топ вибору здобувача складає 31,15 %. Такі показники свідчать про здатність системи ефективно ранжувати дисципліни та надавати релевантні рекомендації.

Обґрунтовано математичну модель формування індивідуального навчального плану як задачу лінійного програмування, що враховує обмеження на кількість кредитів, логічну послідовність вивчення дисциплін та вимоги освітньої програми. Це дозволяє автоматизувати процес перевірки валідності плану ще на етапі генерації рекомендацій, усуваючи ризики формування помилкових траєкторій.

Досліджено проблему «холодного старту» для нових користувачів системи та запропоновано гібридний механізм рекомендацій, що поєднує колаборативну фільтрацію з аналізом анкетних даних та кар’єрних цілей здобувача. Виявлено, що інтеграція контекстних даних (ключові слова, хобі, профіль викладача) дозволяє підвищити точність персоналізації порівняно з використанням виключно історичних даних про оцінки.

Розроблено прототип рекомендаційної системи на основі мікросервісної архітектури з використанням технологій ASP.NET Core, Angular та бібліотеки машинного навчання ML.NET. Архітектурне рішення забезпечує модульність, масштабованість та можливість інтеграції з існуючими інформаційними екосистемами університетів. Реалізовано механізми взаємодії між сервісами через оптимізоване API, що покращило продуктивність обміну даними та забезпечило роботу системи в режимі реального часу.

За результатами впровадження та опитування 220 респондентів встановлено високий рівень задоволеності користувачів роботою системи – 74 % позитивних та змішаних відгуків. Здобувачі відзначили зручність інтерфейсу та релевантність наданих рекомендацій. Водночас виявлено потребу у впровадженні методів пояснюваного штучного інтелекту для підвищення довіри до автоматичних рішень та прозорості процесу формування

рекомендацій.

Запропонований організаційно-технічний механізм побудови індивідуальних навчальних планів ґрунтується на диференційованому підході до кожного здобувача, що дозволяє враховувати його академічний бекграунд та професійні прагнення. Практичне застосування результатів дослідження в системах «Особистий кабінет здобувача» створює передумови для повноцінної реалізації студентоцентрованого підходу в вищій освіті України.

**Ключові слова:** інформаційна система, модель, моделювання, рекомендаційна система, машинне навчання, мікросервісна архітектура, освіта, здобувач, індивідуальний план навчання, C#, Microsoft.ML.

# ANNOTATION

**Ponzel Ya. Yu. Recommendation system for building an individual study plan for a higher education applicant.** Qualification scientific work in the form of a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 122 «Computer Sciences». National University of Life and Environmental Sciences of Ukraine, Kyiv, 2025.

The dissertation investigates data processing methods and models, machine learning algorithms, and approaches to building recommender systems within the context of ensuring the individualization of the educational process. The outlined theoretical foundations form the methodological basis for creating an intelligent system that automates the formation of an Individual Study Plan (ISP). This contributes to increasing the efficiency of planning educational trajectories, optimizing the selection of disciplines, and adapting learning to the professional interests of students. The implementation of the developed models will facilitate the digitalization of universities, reduce administrative burdens, and increase student motivation through personalized learning.

An empirical research base was formed using real data from the educational process (sourced from the «Electronic Dean’s Office» system and EDEBO), covering a sample of 2,543 higher education applicants. This allowed for a deep analysis of academic performance and educational preferences, the identification of hidden patterns in course selection, and the construction of reliable predictive models. The analysis results confirmed the necessity of shifting from unified curricula to flexible trajectories based on intelligent big data analysis.

It was established that the most effective approach for predicting academic performance under conditions of data sparsity (typical for elective disciplines) is the matrix factorization method with Frobenius norm regularization. An algorithm minimizing the loss function (RMSE) was developed and implemented, allowing for the detection of potential learning difficulties at early stages. Experimental verification on a test set showed that the average prediction error (RMSE) is 13.37

points (on a 100-point scale), and the accuracy of a recommended discipline appearing in the student's top choices is 31.15 %. These indicators demonstrate the system's ability to effectively rank disciplines and provide relevant recommendations.

A mathematical model for ISP formation is substantiated as a linear programming problem that accounts for constraints on the number of credits, the logical sequence of studying disciplines, and the requirements of the educational program. This allows for automating the process of validating the plan during the recommendation generation stage, thereby eliminating the risk of forming erroneous trajectories.

The «cold start» problem for new system users was investigated, and a hybrid recommendation mechanism was proposed, combining collaborative filtering with the analysis of questionnaire data and the student's career goals. It was found that integrating contextual data (keywords, hobbies, instructor profiles) improves personalization accuracy compared to using historical grade data alone.

A prototype of the recommender system was developed based on a microservice architecture using ASP.NET Core, Angular, and the ML.NET machine learning library. The architectural solution ensures modularity, scalability, and the capability to integrate with existing university information ecosystems. Mechanisms for service interaction via an optimized API were implemented, improving data exchange performance and ensuring real-time system operation.

Based on implementation results and a survey of 220 respondents, a high level of user satisfaction with the system was established – 74 % positive and mixed feedback. Students noted the convenience of the interface and the relevance of the provided recommendations. At the same time, a need for the implementation of Explainable AI methods was identified to increase trust in automated decisions and transparency in the recommendation formation process.

The proposed organizational-technical mechanism for building individual study plans is based on a differentiated approach to each student, allowing for the consideration of their academic background and professional aspirations. The practical application of the research results in «Student Personal Cabinet» systems

creates the prerequisites for the full implementation of a student-centered approach in higher education in Ukraine.

**Key words**: information system, model, modeling, recommendation system, machine learning, microservice architecture, education, student, individual learning plan, C#, Microsoft.ML.

# СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

## Статті у наукових виданнях, включених до міжнародних наукометричних баз даних Web of Science Core Collection та/або Scopus

1. Hlazunova O., **Ponzel Y.** Technologies and algorithms for the implementation of the recommendation system for creating an individual study plan for a higher education student. CEUR Workshop Proceedings. 2024. Vol. 3771. P. 110–117. (Понзелем Я. Ю. проведено літературний науковий пошук, розроблено концепцію та технічні вимоги до рекомендаційної системи, запропоновано та деталізовано алгоритми її функціонування, проведено експериментальні дослідження та проаналізовано отримані результати, підготовлено та оформлено публікацію. Глазуновою О. Г. визначено наукову новизну та теоретичні основи дослідження, надано наукове консультування щодо вибору напрямів та методів дослідження, проведено рецензування та загальне керівництво роботою над статтею).

## Статті у наукових виданнях, включених до Переліку наукових фахових видань України

2. **Понзель Я. Ю.** Архітектура вебсайту з інтегрованою системою для вибору предметів. Наука і техніка сьогодні. 2025. Вип. 1 (42). С. 1331–1343.

3. Глазунова О. Г., **Понзель Я. Ю.** Математична модель обробки даних з використанням комбінованих методів спільної фільтрації та матричної факторизації для рекомендаційних систем в освіті. Технічна інженерія. 2025. № 1 (95). С. 266–273. *(Глазуновою О. Г. сформульовано концептуальну ідею дослідження щодо створення математичної моделі для прогнозування результатів навчання та визначено методологічні підходи до масштабування системи, зокрема запропонувавши шляхи інтеграції контентної фільтрації та врахування додаткових характеристик предметів для підвищення якості рекомендацій. Понзелем Я. Ю. здійснено обґрунтування та вибір інструментарію реалізації, виконавши розроблення програмної складової на базі ML.NET, проведено безпосереднє налаштування процесу факторизації*

*матриць, застосовано методи оптимізації через градієнтний спуск та регуляризацію за нормою Фробеніуса, а також експериментально підтверджено ефективність моделі шляхом мінімізації середньоквадратичної помилки)*.

**Тези наукових доповідей**

4. **Понзель Я. Ю.**, Голуб Б. Л. Формування основного функціоналу для реалізації інформаційної системи в сфері комунікації студента та університету. Інформаційні технології: економіка, техніка, освіта ‘2022: XIII Міжнародна науково-практична конференція молодих вчених, м. Київ, 26–27 жовтня 2022 року: тези доповіді. Київ, 2022. С. 90–91. *(Понзелем Я. Ю. проведено збір та аналіз вимог користувачів, сформовано перелік основних функцій інформаційної системи комунікації, розроблено схеми взаємодії модулів та описано їхнє призначення. Голуб Б. Л. надано консультації щодо структурування та систематизації функціональних вимог, визначення теоретико-методологічних основ розробки інформаційної системи)*.

5. **Понзель Я. Ю.,** Голуб Б. Л. Сутність системи комунікації студента та закладу вищої освіти. Збірник наукових праць за матеріалами Теоретичні та прикладні аспекти розробки комп’ютерних систем: V Всеукраїнська науково-практична конференція студентів і аспірантів, м. Київ, 26 квітня 2023 року: тези доповіді. Київ, 2023. С. 32–33. *(Понзелем Я. Ю. проведено аналіз існуючих комунікаційних процесів між студентом і ЗВО, визначено ключові проблеми та завдання для автоматизації, сформульовано сутність та функціональне призначення системи комунікації, підготовлено публікацію до друку. Голуб Б. Л. надано науково-методичні рекомендації щодо теоретичного обґрунтування концепції системи та визначення її місця в інформаційному просторі ЗВО)*.

6. **Понзель Я. Ю.**, Глазунова О. Г. Підсистема платіжного контролю системи комунікації студента та закладу вищої освіти. Інформаційні технології: економіка, техніка, освіта: IV Міжнародна науково-практична конференція молодих вчених, м. Київ, 28–29 жовтня 2025 року: тези доповіді. Київ, 2025. URL: http://econference.nubip.edu.ua/index.php/itete/XIV/paper/view/3056. *(аналіз*

*актуальності впровадження сучасних цифрових рішень в освітніх установах проведено Понзелем Я. Ю., а також досліджено переваги та недоліки інтеграції повноцінної платіжної системи (Liqpay та ін.) для ЗВО, обґрунтовано необхідність реалізації саме системи платіжного контролю (а не миттєвої оплати), визначено її ключовий функціонал (статистика, зберігання квитанцій, облік стану оплати) для систематизації роботи фінансового відділу, а також підготовлено матеріал для публікації. Глазуновою О. Г. визначено науково-практичне значення дослідження в умовах обмежених фінансових можливостей ЗВО, надано методологічні рекомендації щодо порівняльного аналізу комерційних та освітніх моделей оплати, а також скориговано напрям дослідження у бік ефективної автоматизації внутрішніх фінансових процесів в університеті).*

**Свідоцтво про реєстрацію авторського права на твір**

7. **Понзель Я. Ю.,** Глазунова О. Г. Модель персоналізованих рекомендацій у цифрових сервісах підтримки користувачів: свідоцтво про реєстрацію авторського права на твір № 138537. Комп'ютерна програма. Дата реєстрації 11 серпня 2025 р. *(Понзелем Я. Ю. виконано побудову математичної моделі та її програмну реалізацію, застосувавши підхід матричної факторизації засобами бібліотеки ML.NET, а також реалізовано алгоритми оптимізації та регуляризації моделі з використанням норми Фробеніуса для уникнення перенавчання, а також розроблено програмні модулі для налаштування гіперпараметрів і автоматизованого розрахунку метрик точності (RMSE), забезпечивши працездатність та валідацію комп'ютерної програми. Глазуновою О. Г. сформульовано загальну концепцію дослідження та поставлено задачу щодо створення системи персоналізованих рекомендацій для підвищення ефективності цифрових сервісів підтримки користувачів).*

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

**ІТ** – інформаційні технології.

**БД** – база даних.

**СУБД** – система управління базами даних.

**ШІ (AI – Article Intelligence)** – штучний інтелект.

**API (Application Programming Interface)** – прикладний програмний інтерфейс.

**XAI (Explainable Article Intelligence)** – пояснювальний штучний інтелект.

**ML (Machine Learning)** – машинне навчання.

**ЗВО** – заклад вищої освіти.

**ІНП** – індивідуальний навчальний план.

**RMSE (Root Mean Square Error)** – середньоквадратична помилка**.**

**MAE (Mean Absolute Error) –** середня абсолютна похибка.

# ЗМІСТ

# ВСТУП

**Актуальність теми.** Згідно законопроєкту №10177 «Щодо розвитку індивідуальних освітніх траєкторій та вдосконалення освітнього процесу у вищій освіті» [1], здобувачі вищої освіти зможуть самостійно визначати формат та вектор свого навчання. У здобувача з'являється можливість самостійно формувати обсяги та терміни свого навчання, що наблизить українську освіти до європейського рівня. Це також надасть можливість здобувачу індивідуально будувати план свого навчання, обираючи період вивчення обов'язкових дисциплін та дисциплін, які він сам обирає у межах індивідуального навчального плану за відповідною спеціальністю.

Індивідуальна план навчання здобувача вищої освіти – це персональний шлях розвитку індивіда під час навчання, що формується на основі таких аспектах безпеки освіти, зрозумілості умов освіти, взаємної корисності, а також задоволенні особистих інтересів здобувача, наявність життєвих ситуацій здобувача та його соціокультурного оточення. Мета формування будь-якого індивідуального плану навчання здобувача полягає у створенні ефективного шляху розвитку в системі вищої освіти, який буде персоналізовано та який відповідатиме навчальним потребам особистості. [2]

Побудови індивідуального плану навчання в закладі вищої освіти має такі цілі:

- ефективне оволодіння навчальним матеріалом, а також розвиток практичних навичок й формування компетентностей;
- формування уявлень здобувача про свою особисту роль як суб'єкта освітньої діяльності;
- оволодіння навичками управління пізнавальною діяльністю та задоволення освітніх потреб та інтересів;
- розвиток когнітивно-комунікативних умінь оволодіння інформацією (наприклад, здійснювати пошук, класифікацію, оцінювання, відбір, аналіз, синтез інформації тощо);
- розвиток умінь самоконтролю та рефлексії, що дають змогу

самостійно коригувати навчання за вибраною планом в подальшому часі;

- розвиток співпраці всіх учасників освітнього процесу. [3]

Згідно сучасних норм, особи, які здобувають вищу освіту можуть здійсювати вибір освітніх компонентів у межах, передбачених відповідною освітньою програмою та навчальним планом, в обсязі, що становить не менш як 25 відсотків загальної кількості кредитів ЄКТС, передбачених освітньою програмою (не менше ніж 10 відсотків для спеціальностей, що передбачають доступ до професій, для яких запроваджено додаткове регулювання). При цьому здобувачі певного рівня вищої освіти мають право вибирати освітні компоненти, що пропонуються для інших освітніх програм та рівнів вищої освіти, відповідно до положення про організацію освітнього процесу в закладі вищої освіти.[4]

Тому, беручи до уваги вищеописану норму, заклади вищої освіти відштовхуються від потреби реалізації мінімум 25 відсотків унікальної освітнього досвіду для здобувача та можуть збільшувати це співвідношення на власний розсуд. Так, в Національному університеті біоресурсів і природокористування України перелік дисциплін вільного вибору за уподобаннями здобувачів бакалаврату та магістратури формується навчальним відділом за поданням факультетів та ННІ і становить 8 кредитів, відповідно дві дисципліни по 4 кредити ЄКТС кожна. Дисципліни вільного вибору за уподобаннями здобувачів розроблені кафедрами у широкому спектрі напрямків і спрямовані перш за все на формування так званих гнучких навичок. І завдяки якісному формуванню й реалізації індивідуального навчального плану, забезпечується підвищення особистісного потенціалу здобувачів вищої освіти, надаючи їм можливість істотно доповнити ті компетентності, які сформовані в процесі вивчення обов’язкової складової освітньої програми, враховуючи їх мотивацію, здібності та інтереси.

В залежності від рішення проєктних груп освітніх програм, в університеті реалізуються три варіанти комбінування вибіркових дисциплін:

- загальним списком, з якого необхідно обирати відповідну кількість

дисциплін (кредитів). Такий спосіб вибору найчастіше застосовується для гуманітарних спеціальностей;

- два або ж три блоки вибіркових дисциплін, які є рівноцінними за кількістю кредитів, з яких необхідно обрати один, що частіше за все застосовується для технічних, технологічних та інженерних спеціальностей;
- бінарні блоки, із кожного з яких необхідно обрати одну дисципліну, що також частіше за все застосовується для технічних, технологічних та інженерних спеціальностей [5]

Однак, виникає чимало проблем, які постають перед здобувачем, зокрема, не розуміння того, як правильно побудувати свій шлях до професіоналізму. Внаслідок чого, здобувач може обрати дисципліни, які не будуть найкраще відповідати його вподобанням та можливостям, або ж зробити хибний вибір дисципліни на основі малої вибірки даних, які надали йому знайомі чи близькі.

Заклади вищої освіти, в свою чергу, також зацікавлені в правильній побудові індивідуального навчального плану для здобувача, адже від цього напряму може залежати чимало важливих факторів, таких як: успішність, рівень задоволення університетом, кваліфікаційний рівень спеціаліста та, навіть, рівень щастя особи.

Виходячи із вищеописаного, перед нами постає задача знаходження механізмів та засобів для побудови індивідуального плану навчання здобувача вищої освіти, вірішити яку необхідно для задоволення потреб як для самого закладу так і для здобувача.

Визначимо основні проблеми, які виникають на шляху до вирішення такої задачі:

1) Маштабованість – звичайно, що кожний здобувач потребує побудови графіку персонального навчання. Приміром, в Національному університеті біоресурсів та природокористування України та його відокремлених структурних підрозділах навчається близько 26 тисяч здобувачів, і, хоча першокурсників було близько 2719 особи [6], не тільки першокурсники мріють про побудову індивідуального навчального плану, але й старшокурсники

бажають мати постійний перегляд плану та можливу зміну його вектору. Під такий обсяг робіт, закладу вищої освіти потрібно або набирати великий штат персоналу, який потрібно навчити, організувати процеси та комунікацію із здобувачами та постійно витрачати фінансові ресурси на роботу даної структури, або ж необхідно автоматизувати процеси й довірити це певній системі.

2) Великий обсяг даних – це проблема, яка виникає через необхідність обробки великого масиву даних для надання якісної консультації для побудови індивідуального плану для здобувача. Це може бути як список дисциплін, які рекомендуються для певного роду спеціалістів, успішність здобувачів, які навчалися на цій же спеціальності до самого здобувача, відгуки здобувачів, які пройшли дисципліни, які можуть бути рекомендовані для здобувача і т.д. Дану роботу краще довірити якійсь автоматизованій системі, ніж людині, так як це допоможе зекономити час та ресурси необхідні для надання рекомендації.

3) Зміна моделі обчислення – проблема виникає, коли рекомендації в побудові вектору навчання для здобувача будувалися по одному алгоритму, який опирається на одні дані, і тут виникла необхідність зробити алгоритм більш точним або розширити його кластер даних для обробки. Для автоматизованої системи це не стане якимось проблематичним фактором і для цього прийдеться тільки переписати програмний алгоритм зчитування та обробки даних. Для внесення змін в алгоритм обробки даних в систему, де значну кількість роботи виконує людський ресурс, це викличе плутанину, помилки, необхідність переписування існуючих графіків, тренінги для навчання персоналу й т.д.

4) Доступність – очевидним є той факт, що автоматизована система буде доступна для запитів здобувачів в будь-який час та в будь-якій кількості, що є неможливим для системи надання рекомендації, яка базується на моделі «працівник-здобувач», так як в такому разі це може відбуватися тільки в робочий час, в обмеженій кількості запитів та із втручанням негативних факторів впливу, такі як захворювання, велика відстань між консультантом та

консультуючим, фактори, які пов’язані з війною і т.д.

5) Людський фактор – чинник, який завжди присутній в системі, де присутній людський вплив і чим цей вплив більший, тим ризик винекнення помилки збільшується. [7]

Питання розвитку штучного інтелекту, машинного навчання та інтелектуального аналізу даних висвітлено у фундаментальних працях закордонних вчених, таких як: А. Тюрінг (A. Turing), який заклав основи обчислювального інтелекту; С. Рассел (S. Russell) та П. Норвіг (P. Norvig), Е. Алпайдін (E. Alpaydin), Т. Мітчелл (T. Mitchell), Й. Бенджіо (Y. Bengio), Я. ЛеКун (Y. LeCun), що досліджували теорію навчання машин.

Теоретико-методологічні засади побудови рекомендаційних систем, методів колаборативної фільтрації та матричної факторизації розроблено у працях: Ф. Річчі (F. Ricci), Л. Рокача (L. Rokach), Б. Шапіри (B. Shapira), П. Резніка (P. Resnick), Дж. Констана (J. Konstan), Дж. Ріделя (J. Riedl), Г. Адомовичюса (G. Adomavicius), А. Тузовського (A. Tuzhilin), Є. Корена (Y. Koren), Р. Белла (R. Bell), К. Волінскі (C. Volinsky), С. Функа (S. Funk), Д. Яннаха (D. Jannach), М. Занкера (M. Zanker).

Вагомий внесок у впровадження інформаційних технологій в освітній процес, розвиток індивідуальних освітніх траєкторій та застосування методів Data Mining в освіті зробили вітчизняні науковці: В. В. Литвин, О. А. Баранов, С. А. Венгер, Г. Й. Шевчук, Л. В. Кліх, О. В. Зазимко, Я. М. Рудик, О. Г. Глазунова, Ю. В. Триус, А. В. Спірін, О. М. Глуз, В. П. Лисенко, С. А. Рашкевич, Н. А. Яремчук, О. В. Козенко та інші.

**Зв’язок роботи з науковими програмами, планами, темами, грантами.** Тема дисертації «Рекомендаційна система для побудови індивідуального плану навчання здобувача вищої освіти» безпосередньо пов’язана з пріоритетними напрямами державної політики України у сфері цифровізації освіти, розбудови інформаційного суспільства та впровадження студентоцентрованого підходу в навчання. Дослідження узгоджується із завданнями Стратегії розвитку вищої освіти в Україні на 2022–2032 роки,

положеннями Закону України «Про вищу освіту» (зокрема ст. 62 щодо реалізації права здобувачів на вибір навчальних дисциплін та формування індивідуальної освітньої траєкторії), а також Концепції цифрової трансформації освіти і науки на період до 2026 року, яка передбачає створення та впровадження новітніх цифрових сервісів для учасників освітнього процесу. Тема дисертації повністю відповідає науковим напрямам факультету інформаційних технологій та кафедри інформаційних систем і технологій Національного університету біоресурсів і природокористування України, які охоплюють дослідження у галузі інтелектуальних інформаційних систем, обробки великих масивів даних, машинного навчання, систем підтримки прийняття рішень та інформатизації навчального процесу. Робота також узгоджується із загальною дослідницькою стратегією університету, спрямованою на цифрову трансформацію освітньої діяльності та використання сучасних інформаційних технологій для підготовки висококваліфікованих фахівців. Це підтверджується актом про впровадження у виробництво результатів дисертації на тему: «Рекомендаційна система для побудови індивідуального плану навчання здобувача вищої освіти», де стверджується, що результати дисертації впроваджено в Національному університеті біоресурсів і природокористування України, а також що ці зміни зв’язані із НДР №: 110/11-пр-2020 «Створення моделі гібридного веб-орієнтованого середовища доставки навчального контенту в умовах відкритої університетської освіти».

**Мета і завдання дослідження.** Метою дисертації є розробка та обґрунтування рекомендаційної системи, методів збору й обробки даних, а також алгоритмів машинного навчання для побудови індивідуального навчального плану здобувача.

Для досягнення поставленої мети необхідно розв’язати такі *завдання:*

1. Проаналізувати сучасні моделі рекомендаційних систем, методи та технології штучного інтелекту з метою виявлення особливостей і проблематики їх застосування у процесі формування індивідуального навчального плану здобувачів освіти.

2. Розробити математичні моделі збору й обробки даних, а також підходи до фільтрації та пошуку даних у межах рекомендаційної системи побудови індивідуального плану навчання здобувача.

3. Виконати аналіз та добір методів машинного навчання для пошуку й фільтрації даних, що використовуються в рекомендаційній системі; розробити та програмно реалізувати алгоритм пошуку й фільтрації даних.

4. Розробити прототип рекомендаційної системи для формування індивідуального навчального плану, здійснити експериментальну перевірку його ефективності та на основі отриманих результатів сформулювати рекомендації щодо вдосконалення застосованих методів і алгоритмів.

*Об'єкт дослідження* – процеси збору, обробки, зберігання, представлення даних в рекомендаційних системах.

*Предмет дослідження* – методи та моделі обробки даних та синтез рекомендацій для побудови індивідуального навчального плану на основі алгоритмів машинного навчання.

**Методи дослідження.** В дисертаційному дослідженні використані методи теоретичного (формалізація, аналіз, синтез, аналогія) та емпіричного досліджень (порівняння, опис, моделювання), а також специфічні методи (математичні).

*Інформаційною базою дослідження* стали нормативно-правові документи, професійні стандарти, наукові праці вітчизняних та закордонних вчених, навчально-методичні матеріали, документація існуючих робочих процесів, емпіричні результати власних досліджень моделей та типів інформаційних технологій для побудови рекомендаційних систем.

*Методологічна основа дисертаційного дослідження* побудована на основі фундаментальних, загальнонаукових, термінологічних, функціональних, системних когнітивних принципів та з використанням процесів моделювання.

*Теоретичною основою дисертаційного дослідження* є модель рекомендаційної системи для побудови індивідуального плану навчання здобувача вищої освіти.

**Наукова новизна отриманих результатів.** У дисертаційній роботі отримано такі наукові результати:

*вперше:*

1) розроблено та обґрунтовано модель рекомендаційної системи для побудови індивідуальних навчальних планів здобувачів вищої освіти, яка підвищує рівень персоналізації освітнього процесу та забезпечує оптимальний вибір дисциплін на основі аналізу академічних досягнень;

2) запропоновано новий підхід до прогнозування вибору навчальних дисциплін на основі методів аналізу великих даних та машинного навчання, який дозволяє формувати індивідуальний навчальний план на основі персоналізованого підходу;

*набули подальшого розвитку:*

3) математичні моделі збору й обробки даних, а також підходи до фільтрації та пошуку даних у межах рекомендаційної системи побудови індивідуального плану навчання здобувача;

4) використання методів машинного навчання для автоматизації процесу вибору дисциплін здобувачами, що дозволило зменшити кількість помилок і прискорити реєстрацію на дисципліни;

5) методи взаємодії між інформаційними сервісами університету шляхом оптимізації API та розширення можливостей обміну даними, що покращило узгодженість та продуктивність системи.

**Практичне значення отриманих результатів.** Практичне значення полягає у створенні рекомендаційної системи для побудови індивідуального плану навчання здобувача вищої освіти, розроблення та впровадження якої має безпосереднє практичне значення для університетів, які прагнуть підвищити рівень цифровізації та індивідуалізації освітніх програм, так як запропонований підхід забезпечує підвищення ефективності планування освітньої траєкторії здобувача за рахунок персоналізованих рекомендацій, а також підвищення мотивації здобувача освіти через можливість формувати план відповідно до

власних освітніх цілей і професійних інтересів й покращення якості освітнього процесу за рахунок використання алгоритмів обробки великих даних і сучасних методів штучного інтелекту. А практичне застосування результатів у системах особистого кабінету здобувача освіти або інших інформаційних ресурсах університетів робить дослідження корисним для впровадження у реальних умовах вищої освіти.

**Особистий внесок здобувачки.** Автором самостійно проведено аналіз наукових джерел, нормативно-правової бази та сучасних підходів до організації індивідуалізованого навчання у вищій школі, а також здійснено оцінку стану існуючих рекомендаційних систем в освітньому середовищі. Автором розроблено архітектуру рекомендаційної системи для побудови індивідуального плану навчання, що об’єднує підсистеми збору даних про успішність, аналітики вподобань та генерації персональних рекомендацій. Самостійно сформовано математичну модель рекомендацій на основі методів колаборативного фільтрування, реалізовано алгоритми матричної факторизації для підвищення точності прогнозування рейтингів дисциплін та обгрунтовано методи подолання проблеми «холодного старту» для нових користувачів системи. Автор особисто розробив програмні модулі системи, структуру бази даних та веб-інтерфейс для взаємодії з користувачами. Інтерпретацію результатів, формулювання висновків і практичних рекомендацій здійснено під науковим консультуванням керівника, при цьому всі основні положення дисертації, що виносяться на захист, є самостійним науковим здобутком здобувача. Особистий внесок у публікаціях, виконаних у співавторстві, чітко визначено у списку наукових праць.

**Апробація результатів дослідження.** Основні наукові положення, результати та висновки дисертаційного дослідження Я. Ю. Понзеля пройшли апробацію на міжнародних і всеукраїнських науково-практичних конференціях, що засвідчило їх актуальність, наукову новизну та практичну значущість. Представлення результатів дослідження на фахових форумах дало можливість отримати професійні відгуки науковців і практиків у галузі інформаційних

технологій та екологічного моніторингу, що сприяло уточненню і вдосконаленню теоретичних і прикладних аспектів роботи. Здобувач представив результати дослідження на: XIII Міжнародній науково-практичній конференції молодих вчених «Інформаційні технології: економіка, техніка, освіта ‘2022» (м. Київ, 2022 р.); V Всеукраїнській науково-практичній конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем» (м. Київ, 2023 р.); IV Міжнародній науково-практичній конференції молодих вчених «Інформаційні технології: економіка, техніка, освіта» (м. Київ, 2023 р.); 3rd Workshop on Digital Transformation of Education, co-located with the 19th International Conference on ICT in Education, Research, and Industrial Applications (м. Львів, 2024 р.).

**Публікації.** За темою дисертаційної роботи опубліковано 6 наукових праць, в тому числі: 2 статті – у наукових виданнях, що входять до фахових видань України; 1 стаття опублікована у періодичному науковому виданні держави, яка входить до Організації економічного співробітництва та розвитку Європейського Союзу; 3 тези доповідей опубліковано у збірниках матеріалів конференцій.

**Структура та обсяг дисертації**. Дисертаційна робота складається з анотацій, змісту, вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг дисертації складає 239 сторінок, з них: 41 рисунків по тексту; 9 таблиць по тексту; список використаних джерел із 220 найменувань на 24-х сторінках; 6 додатків на 30-ти сторінках.

# РОЗДІЛ 1
# АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ТА РЕКОМЕНДАЦІЙНИХ СИСТЕМ У ВИЩІЙ ОСВІТІ

## 1.1 Аналіз університетських інформаційних систем

Внаслідок впливів різних зовнішніх та внутрішніх факторів, взаємодія між вищим навчальним закладом та здобувачем освіти постійно змінюється. До прикладу, пандемія COVID-19, майже унеможливила роботу навчальних закладів в режимі офлайн, через загрозу поширення захворення, а повномаштабне вторгнення в Україну спочатку майже повністю дестабілізувало процес навчання, а потім змусило його адаптуватися під постійні повітряні тривоги та тривожний емоційний стан у всіх суб’єктів цього процесу. Комунікація між здобувачем та закладом освіти постійно змінювалася. На початку був певний хаос через нестандартизоване використання безлічі різних застосунків для проведення занять або спілкуванні один з одним. Навчання проводилось в Google Meets, Microsoft Teams, Discord, Zoom і т.д. Там же ж й відбувалися комунікації здобувачів та університетів, а якщо даних програм не вистачало, то на допомогу приходили електронна пошта, месенджери, телефонні дзвінки та інше. Така побудова освітнього процесу була не найефективнішою, так як часто інформація дублювалася, учасники комунікацій часто могли бачити не потрібну їм інформацію або ж зовсім навпаки – з легкістю могли її пропустити. Виникала також проблема із захистом персональних даних, ймовірність витоку яких збільшується із збільшенням способів її передачу. Одвічна проблема запам’ятовування паролю себе яскраво показувала на фоні використання декількох платформ. Офіційні повідомлення від керівництва університету доходили до адресатів тільки через третіх осіб або посередників, що також вносило труднощі у всьому процесі.

Покращити дану ситуації заклади освіти змогли за допомогою стандартизації програмних засобів для взаємодії із здобувачами освіти. Впровадження єдиних цифрових стандартів дозволило замінити хаотичну

комунікацію в різнорідних месенджерах цілісним віртуальним середовищем, що не лише структурувало доступ до навчальних матеріалів, але й суттєво підвищило рівень цифрової безпеки та автоматизувало адміністративну звітність.

Також, багато закладів пішли далі та вклали наявні ресурси в створення власних застосунків для взаємодії або ж із здобувачами, або ж із їх батьками, або ж із потенційними абітурієнтами. Створення кастомних програмних рішень дозволило забезпечити персоналізований сервіс для різних груп користувачів: студенти отримали «університет у смартфоні», батьки – прозорий інструмент моніторингу успішності, а абітурієнти – інтерактивну взаємодію під час вступу, що в сукупності перетворило такі заклади на сучасні клієнтоорієнтовані організації.

Маючи близкучу ідею, важко її плекати та розвивати у вакуумі, не дивлячись на конкурентів або схожі по своїй концепції затосунки. Крім того, щоб побудувати якісний продукт потрібно постійно стежити за новими тенденціями проєктування та розробки застосунків, методології та принципи розвитку проекту, бути в курсі всіх новин та нюансів щодо забезпечення конфіденційності та захищеності даних системи та користувачів.

Слід звернути свою увагу і на закордонні аналоги. Одним із таких є «Harvard Mobile App» – мобільний застосунок відомого приватного Кембриджського університету. Дозволяє вже в мобільному застосунку ознайомитися із університетом, побачити його відомі місця, подорожувати за допомогою геомітки по видатних локаціях. Однак, крім даного застосунку, цей університет також має застосунки для заняття спорту «CrimZone», слідкуванням за здоров’ям «Harvard For Health», рівень продуктивності «Harvard Home-Work Study» та інші [8].

«UCSF Mobile» – мобільний застосунок Університету Каліфорнії, Сан-Франциско, який дає змогу проводити віртуальні тури для абітурієнтів по території закладу. Містить карту парковок, закладів харчування, номери екстренних служб та новини. Вдало показав себе в часи пандемії COVID-19. [9]

«NAUgo» – застосунок Національного Університету Арізони, який надає інформацію про розклад, оцінки, маршрути кампусу, заклади харчування та їх режим роботи, паркінг, банкомати, може підказати про цікаві події, які відбуваються поруч, підтримує бронювання аудиторій для навчання. Також, в застосунку, групою фахівців із різних відділів даного освітнього закладу, було створено спеціальний функціональний модуль «Be Healthy», мета якого полягає в наданні консультацій та порад для здобувачів освіти щодо покращення їхнього здоров'я. Може надати інформацію про тренування, катання на велосипеді на території закладу. [10]

«CSUN» – мобільний застосунок Університету штату Каліфорнія, Нортрідж. Містить функціонал для самообслуговування співробітників, підпис документів, дає доступ до новин та довідників, розміщення місць для парковок, дає можливість забронювати кімнату в бібліотеці. Однією із особливостей даного застосунку, є підтримка функціоналу по збору благодійних коштів для покращення благоустрою університету та життя здобувачів освіти в ньому. [11]

«Oiler Mobile» – мобільний застосунок Університету Фіндлі має спеціальний розклад як для здобувачів освіти та їх батьків, містить карту подій, актуальні новини, функціонал, який допомагає займатися спортом, а також за допомогою PUSH-повідомлень постійно інформує та заохочує здобувачів освіти взяти участь в різних подіях університету. [12]

«UA Mobile» – мобільний застосунок, який був розроблений для Університету Акрона та який допомагає дізнатися базову інформацію про розклад занять, меню їдальні, новини, доступність комп'ютерних аудиторій, час роботи закладу, контактні дані працівників та надає доступ до навчальних курсів та матеріалів. Містить функціональність, яка допомагає відстежувати відвідуваність заходів та проводити анкетування. [13]

«UH Go» – мобільний застосунок Університету Хюстона, за допомогою якого здобувач освіти може переглянути свої оцінки з певної дисципліни, переглянути графік роботи закладів харчування, оплатити за харчування, дізнатися інформацію про різні події, надає можливість приєднуватися до

різних організацій та, навіть, викликати екстренні служби. Містить функціонал цифрового документа персони. [14]

«CSUSM» – застосунок Каліфорнійського державного університету Сан-Маркос, який містить функціонал цифрового документа, інформацію про події та знижки в кампусі, карту з корисними локаціями та перевірку власної університетсьої пошти. [15]

«Cypress Connect» – розробка здобувачів освіти коледжу міста Сайпресс, яких активно залучає керівництво навчального закладу для розробки продукту для власних потреб. Даний застосунок надаї інформацію про новини та події, розкладу, тури в закладі, місця парковок. Має функціонал цифрового документа. [16]

«CIU Mobile» – мобільний застосунок приватного університету на Північному Кіпрі. Містить сервіс трекінгу пересування громадського транспорту, карту кампусу, інформацію про оцінки здобувачів освіти, контактні дані працівників університету, функціонал бібліотеки, програмний модуль для зворотнього зв’язку. [17]

«RU Mobile App» – мобільний застосунок Державного університету в Редфорді, Вірджинія. Містить дані про події в кампусі, графік автобусів, контакти невідкладних служб, карти, персональний ідентифікатор. [18]

«UCF Mobile» – мобільний застосунок університету Центральної Флориди. Його особливістю є те, що він містить функціонал, що допомагає новим здобувачам освіти соціалізуватися в університетський колектив, а також містить логіку, яка дозволяє проводити опитування здобувачів освіти для розуміння настроїв та підтримання духу. Також містить інформацію про розклад, підтримку здоров’я, контактні дані працівників, доступ до свіжих новин, функціонал термінових екстренних викликів, карт, To Do список та доступ до каталогів бібліотеки. [19]

«UNCG Mobile» – мбільний застосунок Державного університету в Грінсборо, Північна Кароліна. Містить мапу кампусу із різними локаціями, контактні дані працівників, розклад місць харчування, оповіщення про

термінові новини чи небезпеку, швидкий доступ до контактів екстренних служб, інформацію про новини. [20]

«Qatar University Mobile» – мобільний застосунок Катарського університету, який дає наступний функціонал: доступ до лекційного контенту та навчального матеріалу, можливість перегляду оцінок, оплата за послуги, віртуальні тури, інформація про новини закладу та події, що в ньому відбуваються, карти та обіднє меню. [21]

«Safe Lancer», «CampusGo», «Lancer Athletics & Recreation», «UWin FAHSS», «U of Windsor Experience» – набір застосунків для здобувачів освіти університету Віндзора. «Safe Lancer» – надає основні інструменти, щоб зробити життя здобувачів освіти в університеті безпечнішим. Містить інформацію про захворювання, контакти екстренних служб, карту злочинів, віртуальні маршрути додому, зворотній зв'язок та службу підтримки. «CampusGo» – програма, яка допомагає здобувачу орієнтуватися на території закладу освіти, шукати потрібні або цікаві локації. Також, за допомогою даного застосунку, можна дізнатися про події, які відбуваються поблизу та актуальні персоналізовані новини. «Lancer Athletics & Recreation» – мобільний застосунок призначений для здобувачів освіти, які полюбляють займатися спортом або відвідувати спортивні події. В ньому є можливість зареєструватися на певні заняття, переглядати графік спортивних подій та режим проведення тренінгів та є можливість бронювати локації. «UWin FAHSS» – програма, яка надає доступ до послуг кампуса. В ній є можливість користуватися картами, курсами, інформацію про обід та інші сервіси на території закладу, спілкуватися із обраним колом осіб на певну тематику, заводити друзів по спільних інтересах, вести свій профіль та календар і т.д. «U of Windsor Experience» – програма для онлайн турів по університету, де разом із сім'єю можна переміщатися по території, дізнатися про унікальні місця та їх історії, з підтримкою карт територій університету, а також відео та фотографій. В сукупності ці всі вищезгадані програми намагються понвістю покрити потреби здобувача освіти як в самому університеті так на його території. [22]

«Istanbul Medipol University» – мобільний застосунок приватного університету в Стамбулі, Туреччина. Розроблений для здобувачів освіти та гостей університету. Включає в себе інформацію про новини та події, інформацію про розклад та пересування транспорту, про заклади харчування, інформацію про курси, завдання та екзамени, є підтримка розкладу. [23]

«Teno App» – застосунок, який використовується в багатьох школах Індії. Містить електронний модуль для навчання, модулі для миттєвих комунікацій, онлайн оплати за курси, цифрової присутності, цифрового календаря та щоденника, інформацію про оцінки за екзамени. Також містить в собі тести для учнів, відеонавчання, головоломки та тренувальні вправи для логіки, аналітику продуктивності. За можливість користуватися даною платформою потрібно заплатити. [24]

«School Plus-School Management» – застосунок, який дозволить краще взаємодіяти один між одним здобувачам освіти, викладачів, персонал управління та батьків. Надає важливу інформацію про час проведення занять та екзаменів, містить функціонал оплати онлайн, модуль коумікації між користувачами, трекінг пересування шкільного автобуса. Програма є платною та найбіль розповсюджена в Індії. [25]

«Helloparent» – розробка також з Індії, яка допоможе батькам бути в курсі всіх подій в школі та стан справ їх дітей в навчанні. Містить календар, модуль комунікацій, підтримку онлайн навчання, щоденник, модуль оплати онлайн, довідник загальних питань, карти та фотоальбоми. Можливість користуватися даним застосунком є платною. [26]

Для нашої тематики перш за все потрібно дивитися на наступних українських конкурентів: «UzhNU Information System», «KNU online» та «Студент ЧДТУ» тощо.

«UzhNU Information System» – проект, який на початку розроблювався як електронний розклад для одного із факультетів Ужгородського національного університету, але у висновку переріс більший проект із набагато більшими функціональними можливостями. Проект містить функціонал для зчитування

QR-коду, що дозволяє переглядати інформацію для відповідного факультету та курсу. Також він містить розділ «Опитування» та власне функцію, яка розроблювалась на самому початку, «Розклад» із інформацією про лекції, викладачів та час проведення. Застосунок реалізовано на основі вебплатформи із використанням node.js, express.js, mongoose та MongoDB. Реліз застосунку планувався на вересень 2022-го року [27].

«KNU online» – застосунок Київського національного університеті імені Тараса Шевченка, який був презентований в рамках проекту «Цифровий університет. Університет у смартфоні» у 2020 році. Доступні функції в проекті: електронний кабінет здобувача освіти, електронний кабінет викладача, електронний деканат, цифрова бібліотека, онлайн-звітність та електронний документообіг, запит довідок, анонімні відгуки про викладачів та підтримувати онлайн-зв'язок [28].

«Студент ЧДТУ» – в 2021 році в рамках проекту «Deanoffice» студентами та викладачами Черкаського державного технологічного університету разом із «Lifecell», Master of Code Global та Інжиніринговою школою Noosphere презентовано даний мобільний застосунок. Його мета спростити інформаційну взаємодію здобувача освіти та університету. Вже реалізований функціонал: авторизація, перегляд профілю користувача, робота з вибірковими дисциплінами, робота зі зразками заяв, можливість вибору вибіркових дисциплін для здобувачів. Повноцінний реліз планувався на кінець 2021-го року – початок 2022-го року [29].

Один із існуючих застосунків для полегшення взаємодії здобувача освіти із університетом є «Univera». Проєкт «Univera» впроваджується в межах програми «Мріємо та діємо» і став можливим завдяки щирій підтримці американського народу через Агентство США з міжнародного розвитку (USAID). Програма «Мріємо та діємо» виконується IREX у партнерстві з Будуємо Україну разом (БУР), Центром «Розвиток корпоративної соціальної відповідальності» (CSR Ukraine), Making Cents International (MCI), Міжнародним республіканським інститутом (IRI) та Zinc Network. [30] . Як

заявляється розробниками, взаємодіючи із державним сервісом «Дія», застосунок вже здатний надавати інформацію про розклад занять, новини, користувачі зможуть отримати довідки від університету, і, навіть, будуть можливість змінити банк для отримання стипендії. Тестування застосунку вже відбувається в таких університетах як: Національний університет «Одеська політехніка», Одеський національний економічний університет, Міжнародний науково-технічний університет імені академіка Юрія Бугая, Львівський національний університет імені Івана Франка, Івано-Франківський національний технічний університет нафти і газу, Державний університет економіки і технологій, Київський університет імені Бориса Грінченка, Харківський національний університет імені Василя Каразіна, Київський політехнічний істнитут імені Ігоря Сікорського. [31]

Сам сервіс «Дія» також надають певні послуги для здобувачів освіти, зокрема це перегляд документів про освіт та студентський квиток. [32] Також існує сервіс «Дія.Освіта», метою якого є розширити для більшого кола осіб якісний освітній матеріал. Даний сервіс пропонує вільний доступ до освітніх матеріалів, тестів, симуляторів, гайдів, вебінарів, подкастів та подій. [33]

В лютому 2024 року Кабінет Міністрів України підтримав постанову по запуску застосунку «Мрія», головна мета якого цифровізувати освіту в Україні та зробити її більш цікавішою та ефективною. Застосунок буде містити такі корисні функції як: інформацію про успішність, відвідування та коментарі вчителів, учнівський квиток, навчальні матеріали, чати для спілкування, інструменти для полегшення рутинної роботи вчителів. В застосунку планують використовувати штучний інтелект для побудови інтерактивного робочого матеріалу та індивідуальної стратегії навчання учня. Початковою цільовою аудиторією є учні та працівники шкіл, але надалі функціонал допрацьовуватиметься й для інших типів закладів освіти. Розробку взяло на себе Міністерством цифрової трансформації України разом із Міністерством освіти і науки України. [34]

«UniShed» – розробка здобувача освіти із Львівського національного

університет імені Івана Франка. Застосунок дає можливість дізнаватися розклад занять, перехід в один на дистанційні заннятя в інших різних сервісах, покликання на на інші сервіси університету, які часто використовуються. Також, можна переглядати прогноз погоди, новини університету та актуальні збори для Збройних Сил України. Написаний за допомогою наступних мов програмування, бібліотек та технологій: ASP.NET Core Blazor, PWA, PHP, фреймворк Leaf, сервіс CloudFlare. Автор застосунку зазначає, що фактор безпеки є найголовнішим в сучасних програмах, якими користуються українці, персональні дані яких є бажаною ціллю для зовнішніх агресорів, саме тому його застосунок не зберігає дані про користувачів на серверах, а розміщує їх пам'яті локального пристрою користувача. [35]

«Мій ДонДУУ» – мобільний застосунок від Донецького державного університету управління, який надає інформацію про університет, його структурні підрозділи, їх історію, освітні програми, розклад, контакти та новини. [36]

«AR Book» – застосунок, який призначений для вчителів та учнів шкіл, і дозволяє використовувати вже готові матеріали до уроків, створювати власні уроки, здійснювати аналітику по залученню та засвоєнню матеріалу та давати рекомендації по додатковому матеріалу, використовувати технологію доповненої реальності для кращого вивчення реальності, а також застосовує гейміфікацію та досягенння для збільшення зацікавленості та мотивації в учнів. Застосунок вміє працювати з календарем, внаслідок чого можна легко створювати розклад занять. Інтегрується до популярних платформ, які використовуються для навчання, як: Moodle, Microsoft Teams, Google Classroom тощо. Базові функції є безкоштовними в застосунку, а за більш просунутий функціонал доведеться заплатити. [37]

Із проаналізованого масиву програмних рішень (понад 25 систем) для детального порівняльного аналізу було відібрано репрезентативну вибірку (табл. 1.1), що охоплює основні класи існуючих систем: закордонні університетські сервіси, платформи для шкільної освіти, українські

університетські розробки та державні екосистеми.

*Таблиця 1.1*

**Порівняльний аналіз класів інформаційних систем**

| Клас систем / Представники | Інтелектуальна складова | Недоліки в побудові ІНП |
|---|---|---|
| Закордонні університетські застосунки (Harvard Mobile, NAUgo, Qatar University та ін.) | Низька. Використовується лише для фільтрації новин або подій | Відсутній механізм вибору дисциплін. Система працює як довідник, а не порадник |
| Комерційні шкільні платформи (HelloParent, Teno App) | Середня. Аналітика успішності, але без персональних рекомендацій щодо траєкторії | Орієнтовані на контроль з боку батьків, а не на самостійний вибір учня/студента. Платний доступ |
| Українські університетські розробки (KNU online, UzhNU, Студент ЧДТУ) | Відсутня. Реалізовано лише автоматизацію документообігу | Вибір дисциплін реалізовано як механічну процедуру («галочки» у списку) без аналізу схильностей чи прогнозів |
| Державні екосистеми (Дія, Мрія) | Висока (Перспективна). Планується впровадження AI для побудови стратегії навчання (Мрія) | Наразі фокус зміщений на середню освіту (школи). Відсутня специфіка вищої освіти (Вибіркові блоку) |
| AR/VR навчальні інструменти (AR Book) | Середня. Рекомендації контенту | Вирішує локальні задачі засвоєння матеріалу, а не глобальну задачу планування семестру |

На початку роботи над дисертацією та перед проведенням дослідження в Національному університеті біоресурсів та природокористування України були три окремі сервіси (рис. 1.1), які разом забезпечували роботу цифрової системи навчального закладу:

1) Навчально-інформаційний портал – програмний модуль, який розроблено на основі навчальної платформи Moodle, та який існує для підтримки навчання. На даному порталі здобувачі освіти можуть ознайомитися із навчальним матеріалом, проходити тести, завантажувати роботи, спілкуватися із викладачем, відмічати свою пристуність і т.д. Важливість даного ресурсу важко переоцінити для університету в часи пандемії та війни.

2) Google Workspace – спеціалізоване хмерне програмне забезпечення, яке використовується для доступу до ресурсів інформаційно-освітнього середовища університету. Воно також надає змогу користуватися месенджерами, редакторами та хмарними сховищами, що дає для здобувачів освіти широкі можливості для навчання.

3) Система електронного деканату – власна розробка університету, яка виконує функції:

a. управління здобувачами освіти, освітніми програмами, спеціалізаціями, дисциплінами й т.д.;

b. зберігання академічної успішності здобувачів освіти, індивідуальних навчальних планів і т.д.;

c. формування навчальних відомостей, довідок та іншого роду документів;

d. інше.

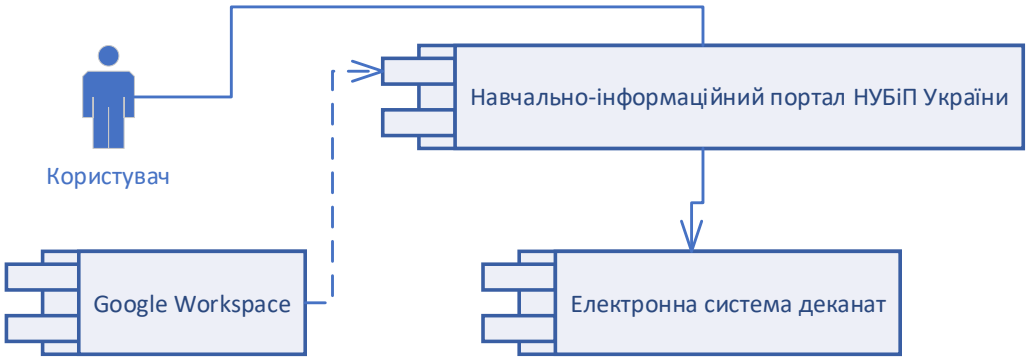


Рис.1.1 – Архітектура взаємодії компонентів екосистеми цифрових сервісів університету до початку дослідження

Попри те, що заклад вищої освіти має порівняно невелику кількість цифрових сервісів, їх функціоналу цілком достатньо для збирання базових статистичних даних про студентів та подальшого проведення аналітичної роботи. Наявні системи фіксують ключові показники освітньої діяльності, що у свою чергу дозволяє формувати узагальнені профілі здобувачів і виявляти тенденції в їхній навчальній поведінці. Адже навіть обмежений обсяг даних

може бути використаний у моделях машинного навчання для побудови первинних прогнозів, виявлення ризиків академічної неуспішності чи формування рекомендацій щодо індивідуального плану навчання. Таким чином, навіть мінімальний набір цифрових інструментів створює достатнє підґрунтя для проведення якісного аналітичного та прогнозного дослідження.

## 1.2 Дослідження застосування штучного інтелекту в інформаційних системах

Прийнято вважати, що робота Алана Тюрінга «Обчислювальна техніка та інтелект» [38], яку він випустив в 1950 році та яка описує «Гру в імітацію» або іншими словами славнозвісний «Тест Тюрінга», і дала старт розвитку такої галузі інформаційних технологій як «Штучний інтелект». Трохи більше ніж 70 років потому, людство вже знайшло широке практичне застосування штучного інтелекту в машинному навчанні, нейронних мережах, глибокому навчанні, когнітивних обчисленнях, комп'ютерному баченні, доказі теорем, розпінаванню зображень, машинному перекладі і розумінні людської мови, ігрових програмах, машинній творчості та експертних системах [39].

В своїй праці «Визначення терміну «Штучний інтелект»» [40] Баранов О. А. пропонує наступне визначення для штучного інтелекту: «штучний інтелект – це певна сукупність методів, способів, засобів та технологій, насамперед, комп'ютерних, що імітує (моделює), когнітивні функції, які мають критерії, характеристики та покзники еквівалентні критеріям, характеристикам та показникам відповідних функцій людини.

Місія штучного інтелекту – це створення певних умов для значного підвищення ефективності всієї соціальної та виробничої діяльності у суспільстві шляхом забезпечення незалежності процесу прийняття якісних (оптимальних) рішень від негативного впливу людського фактору.

Мета впровадження штучного інтелекту – це забезпечення прийняття якісних (оптимальних) рішень та їх подальшої ефективної реалізації.»

В праці «Штучний інтелект: переваги та недоліки» [41] Петрів С. Ю.

наводить певний перелік переваг цієї технології:

- Зменшення людської помилки.
- Можливість постійної безперервної взамодії із штучним інтелектом, так як вона може працювати постійно, на відміну від людини.
- Можливість ефективно використовувати його для виконання небезпечних завдань (ураження ворожої військової сили, ліквідації стихійного чи техногенного лиха й т.д.).
- Швидке прийняття рішень.
- Прискорення пошуку інформації.
- Також автор даної статті наголошує на таких недоліках, як:
- Збільшення безробіття.
- Збільшення витрат на роботу таких систем.
- Збільшення кількості людей, які не прагнуть до роботи та навчання, за рахунок того, що штучний інтелект може виконувати більшу частину їх роботи.
- Відсутність емоцій.
- Відсутність креативного мислення та бачення.
- Постійна потреба в глибокій експертизі та професійному супроводі.

В свою чергу Бриль Ірина Василівна в праці під назвою «Штучний інтелект в реаліях сучасності» [42] наводить наступні ризики, які несе в собі штучний інтелект для людства, та які не згадувалися в роботі автора Петрів С. Ю.:

- Порушення приватності.
- Породження великої кількості неправдивої інформації.
- Помилковий результат через неправильні початкові дані.
- Фундаментування соціально-економічної нерівності.
- Удосконалення систем враження для військових цілей.

Також автор даної роботи вважає за необхідне наголосити й на обмеженнях даної технології:

1. Системи на основі штучного інтелекту виконують лише той тип задач, для якого були створені.

2. Вони не здатні виконувати декілька типів задач та перемикатись між ними в процесі виконання, як це може зробити людина.

3. Для успішного виконання задачі даній технології потрібно витратити певний час та ресурси на навчання, а також мати дані, які будуть для нього істинними та на які він зможе опиратися в подальшій обробці своїх результатів.

Також, слід зауважити, що так як штучний інтелект – це відносно новий інструмент в руках людини, то немає ще потужної законодавчої бази, яка б могла врегулювати всі аспекти використання. В своїй праці «Штучний інтелект: переваги та загрози використання» [43] автори Колесніков А. П. та Карапетян О. М. зазначають, що в Україні на сьогодні ще немає спціального законодавства, яке було б заточене на врегулюванням питаннях використання цієї технології. Або ж приміром автор статті «Штучний інтелект в Україні: правові аспекти» [44], зазначає, що одним із перших вагомих кроків на цьому шляху є Резолюція від ЄС від 16 лютого 2017 року щодо правил цивільно-правового регулювання робототехніки і роботи по цьому питанню продовжують вестися, і що Україні це питання дуже погано розкрите та наводить приклади напрямів для карти правових реформ штучного інтелекту в Україні.

Тому, на основі вищезгаданих плюсів та мінусів та обмежючих факторів, ми можемо сміливо висунити тезу, що штучний інтелект – це інструмент в руках людини, який в залежності від самої людини може принести як велику користь так і непоправну шкоду, тому головне – це вміле та добросовісне його використання.

Напрямок штучного інтелекту для інформаційних технологій не є чимось новим та як він почав розвиватися вже із  років двадцятого століття, але неймовірного розвитку набрала ця технологія за останні роки. Так, в своїй статті «Компанії із найбільшою кількістю патентів пов’язаних із штучним інтелектом» [45] за січень 2023 року Катаріна Бухгольц показує статистику

кількості сімейств патентів компаній в залежності від років. На основі даних в таблиці 1.2, можемо чітко побачити бум створення патентів в сфері штучного інтелекту:

*Таблиця 1.2*

**Дані щодо створення патентів в сфері штучного інтелекту**

| **Компанія** | **2017 рік** | **2021 рік** |
|---|---|---|
| Tencent | 711 | 9614 |
| Bai | 1134 | 9504 |
| IBM | 2921 | 7343 |
| Samsung | 3313 | 6885 |
| Pingan | 46 | 6410 |
| Microsoft | 4373 | 5821 |
| Alphabet | 2033 | 4068 |

Аналізуючи наведену динаміку, варто зазначити, що хоча статистика охоплює період до 2021 року, вона ілюструє фундаментальний зсув технологічного лідерства: китайські технологічні гіганти (Tencent, Baidu) здійснили експоненціальне зростання, випередивши традиційних західних лідерів (IBM, Microsoft). Цей тренд є показовим для розуміння глобального вектора розвитку, який продовжується і сьогодні, трансформуючись у гонку генеративних моделей.

Щодо специфіки сфери освіти, то хоча наведена таблиця відображає загальні патенти, ключові гравці цього списку (Microsoft, Alphabet/Google, IBM) є одночасно і лідерами впровадження ШІ в освітні технології (EdTech). Їхні розробки у галузях обробки природної мови (NLP) та адаптивного навчання лежать в основі сучасних освітніх платформ (Microsoft Teams for Education, Google Classroom AI). Зростання їхньої патентної активності прямо корелює з появою нових інструментів для персоналізації навчання.

У контексті українського ринку IT, ситуація має свою специфіку. Українські компанії рідше фігурують у світових рейтингах за кількістю

патентів (оскільки вітчизняний ІТ-сектор історично орієнтований на аутсорсинг та сервісну модель, або ж інтелектуальна власність стартапів реєструється в юрисдикціях США). Проте, Україна демонструє значний потенціал у практичному застосуванні ШІ, зокрема в EdTech-секторі (компанії Preply, Grammarly), де фокус робиться не на патентуванні базових алгоритмів, а на створенні кінцевих продуктів для користувача. Відсутність українських компаній у топі патентних гігантів свідчить не про відсутність розвитку, а про інтеграцію українських розробників у глобальні екосистеми технологічних лідерів.

В своїй праці «Аналіз національних стратегій розвитку штучного інтелекту» [46] Костеноко О. В. підмічає, що багато країн вже приступили до розробки стратегій по розвитку штучного інтелекту. Наголошується на тому, що якщо це роботи, які за своїми розмірами більші, ніж локальні дослідження, то це потребує значних затрат фінансових ресурсів, людського часу та законодавчого регулювання. Також автор зауважує, що Україні слід звернути увагу на вже вироблені стратегії іншими країнами та виваважено прийняти рішення щодо вибору свого національного курсу по роботі з штучним інтелектом.

Однак, стратегія розвитку штучного інтелекту фактично сама обрала Україну (рис.1.3). У зв’язку з військовими діями, великою перевагою ворога в живій силі та ресурсах Україна починає будувати вектор розвитку штучного інтелекту в безпілотних літальних апаратах. Автори мілітарного журналу «Militarnui» [47] дають короткий опис можливостей такого інструменту «система за допомогою просунутої оптики самостійно розпізнає і фіксує координати техніки ворога (навіть замаскованої) та негайно передає інформацію в командний пункт для прийняття відповідного рішення». Також в статті йдеться про «AirUnit» – компанії, яка спеціалізується на розробці FPV-дрони з системою самонаведення. Таке самонаведення безпілотників планують реалізувати через комп’ютерний зір і побудову алгоритму роботи дрона під час втрати зв’язку з оператором.

Звичайно, що інші країни, які є активними спостерігачами російсько-української війни, також посилили роботу над дослідженням штучного інтелекту у військових цілях. Так, Пентагон в квітні 2023 року оголосив про створення робочої групи з генеративним штучним інтелектом для посилення заходів безпеки національних інтересів в майбутньому [48].

А до такого висновку приходять Хаустова В. Є., Решетняк О. І., Хаустов М. М., Зінченко В. А. в роботі «Напрямки розвитку технологій штучного інтелекту в забезпеченні обороноздатності країни» [49]: «Значний вплив на формування та розвиток військового потенціалу матиме ШІ, вбудований у супутні технології, такі як: ядерні, аерокосмічні, кібернетичні, технології розробки нових матеріалів та біотехнології; віртуальна/доповнена реальність; квантові обчислення; автономність, моделювання, клауди; дослідження матеріалів; виробництво, логістика, стратегічне управління; аналітика великих, малих і широких даних».

В Україні спостерігається сильний розвиток штучного інтелекту. Згідно звіту «Штучний інтелект в Україні» [50] Міністерства цифрової трансформації України станом на 2020 рік Україна займала перше місце в Східній Європі по кількості компаній, які використовують штучний інтелект. Станом на 2021 рік дослідницькі центри штучного інтелекту багатьох корпоративних гігантів розміщувалися в Україні, таких як Amazon, Snapchat, Rakuten Viber, Huawei, Samsung, Google, Lyft, але широкомаштабне вторгнення внесло певні зміни в роботі цих компаній. Grammarly, People.ai, Viewdle, Looksery, AI Factory, Slice – це українські стартапи, які використовуюь штучний інтелект. В різних інших галузях також використовують штучний інтелект компанії, які працюють в Україні: Infopulse, N-iX, Ciklum, Freshcode, GlobalLogic, Epam, Vitech, Sigma, SoftServe, Preply та інші.

Що стосується університетів, то штучний інтелект активно вивчається в закладах вищої освіти. Але для мобільних застосунків українських університетів він поки що не використовувався. В закордонних університетах практика інакша. На приклад, «Harvard AI Sandbox» – програма, яку розробили

в Гарвардському університеті. Вона здатна на своїй базі використовувати декілька інструментів для роботи із штучним інтелектом таких як ChatGPT, Microsoft’s Bing Chat та Google’s Bard, та при цьому головне її покликання в запобіганні витоку приватних даних та інтелектуальної власності університету. Проєкт є порівняно новим та позиціонується як експерементальний. [51]

Хоча прикладів практичного впровадження штучного інтелекту в закордонних закладах освіти наразі небагато, їхня поява здебільшого зумовлена прагненням іти в ногу з сучасними науковими тенденціями. В українських реаліях потреба у таких технологіях набуває значно ширшого змісту: з огляду на складні умови, у яких змушені навчатися здобувачі, інструменти штучного інтелекту здатні не лише підвищити якість освітнього процесу, а й забезпечити додаткову підтримку та безпеку. Саме в цьому контексті особливу цінність набувають методи машинного навчання, які лежать в основі сучасних рекомендаційних систем і дозволяють формувати персоналізовані поради, прогнозувати потреби студентів та оперативно реагувати на потенційні ризики.

**Машинне навчання** – це підтип штучного інтелекту, яке дає змогу оброблювати великі масиви даних та яке дає змогу системі автономно навчатися та вдосконалюватись за допомогою нейронних мереж і глибокого навчання без безпосереднього програмування алгоритмів їх обробки даних. Шляхом постійного збагачення системи різноманітними даними, машинне навчання покращує свою точність в передбаченні або класифікації даних [52].

Починаючи ще з середини 20 століття, з’являються наукові напрацювання пов’язані із машинним навчанням. Так Аллан Тюрінг згадує його у своєму відомому напрацюванні «Тест Тюрінга», а А. Самуель використовував ці знання в своїй програмі перевірки комп’ютера [53].

Як зазначає автор книги [54] у традиційному програмуванні, розробник створює алгоритм, який чітко описує як вирішувати ту чи іншу задачу. Він вказує точні інструкції, які комп’ютер повинен виконувати, для отримання бажаного результату. В той час як в машинному навчанні, комп’ютер не отримує готових інструкцій, а замість цього навчається на даних, щоб знайти

закономірності, що можуть бути використані для вирішення задачі. В такому випадку навчання відбувається за допомогою прикладів, і модель адаптується під ці дані, щоб автоматично приймати рішення або прогнозувати нові ситуації без явного програмування кожної деталі.

Прийнято вважати, що машинне навчання можна поділити на декілька основних частин [55]:

- Процес обчислення – на основі даних, які опрацьовуються алгоритмом, знаходиться найбільш відповідний патерн калькуляції даних для надання прогнозу або класифікації.
- Функція помилок – для оцінки точності даних функція може зробити порівняння наскільки точним був прогноз, на основі вже існуючих прикладів.
- Процес оптимізації моделі – щоб зменшити розбіжність між якимось відомими прикладами та конкретною моделлю, можуть коригуватися вагові коефіцієнти для оптимізації точності прогнозу чи класифікації.

Машинне навчання може знайти своє застосування в різних галузях людської діяльності [56]:

1) Онлайн-шопінг – рекомендаційні системи, патерни аналізу користувацьких інтересів, цільова реклама тощо.
2) Енергетичний сектор – розрахунок енергії, вибір стратегії економії енергії, надання передбачення по потужностях і т.д.
3) Розваги – рекомендаційні системи розважального контенту, цільова реклама тощо.
4) Сектор охорони здоров’я – раннє передбачення хвороб, діагностика, рекомендації по ліванню тощо.
5) Соціальний сектор – знаходження несправжніх користувачів, синтетичний аналіз тощо.
6) Фінансовий сектор – розрахунок ризиків, знаходження елементів шахрайства, розрахунок прибутків тощо.
7) Автентифікація – розпізнавання обличчя та образів, підозрілої активності або ж надання цільової реклами тощо.

8) Автономне управління – розпізнавання об’єктів, автономне управління механізмом, передбачення траекторії тощо.

Алгоритми машинного навчання, методи аналізу великих даних та моделі прогнозування можуть бути ефективно використані для побудови моделі прогнозування індивідуального плану навчання з урахуванням попередніх даних про здобувача освіти. Застосування таких підходів дає змогу виявляти приховані закономірності у навчальній діяльності, визначати потенційні освітні траєкторії та формувати персоналізовані рекомендації. Зокрема, на основі історичних даних можуть створюватися рекомендаційні системи, які прогнозують оптимальні навчальні модулі, ресурси чи стратегії, що сприятимуть підвищенню результативності та персоналізації освітнього процесу.

## 1.3 Дослідження існуючих рекомендаційних систем

ІТ-компанії, як правило, приділяють обмежену увагу підготовці наукових публікацій, зосереджуючись переважно на представленні власних рішень у корпоративних блогах, технічних статтях або на платформах підтримки користувачів. У зв’язку з цим для ідентифікації актуальних прикладів реалізації рекомендаційних систем доцільно розширити спектр джерел інформації, включаючи такі первинні канали комунікації, що дозволяє одержувати найбільш релевантні й практично орієнтовані дані безпосередньо від розробників.

Відома платформа Amazon ділиться історією розвитку своєї рекомендаційної системи. В статті [57] йдеться про те, як на початку нульових виникла ідея пропонувати покупцям схожі товари, аби збільшити продажі. Перший їх алгоритм рекомендацій використовував простий підхід на основі контенту та історії покупок, де товари пропонувалися на основі схожості з тими, що вже були куплені. Але одним із важливих етапів розвитку стало впровадження методу колаборативної фільтрації, що дозволило вже рекомендувати товари на основі покупок інших користувачів, схожих на

поточного покупця. Цей підхід враховував взаємодії між користувачами та продуктами, що допомогло створити більш точні та персоналізовані рекомендації. По ходу розвитку їх система почала враховувати не лише попередні покупки користувача, а й його взаємодії з товарами, відгуки тощо. Потім настав етап в розвитку системи, коли в неї вдалося впровадити нейронні мережі, що допомогло опрацьовувати більш складні дані та давати кращі та точніші рекомендації для користувачів.

Окремим важливим завданням Amazon ставить для себе забезпечення конфіденційності даних користувачів та мінімізування можливих упередженних рекомендацій. Для написання системи, розробники використовували мови програмування Java, JavaScript, Ruby, Python, Perl. [58]

Найвідоміша (на момент написання даної роботи) стрімінгова платформа Netflix також має власну рекомендаційну систему та ділиться інформацією про її розробку [59]. Розробка починалася із простого алгоритму, який на основі 5-ти бальних оцінок користувачів намагався визначити цікавий контент для користувачів. Згодом компанія почала використовувати колаборативну фільтрацію, яка враховувала вподобання користувачів, які схожі між собою за смакими. Такий підхід дозволяв пропонувати новий контент на основі їх попередніх переглядів. З часом Neflix все більше удосконалював систему та впроваджував складніші алгоритми (приміром, матричну факторизацію). Враховуючи різноманітність факторів споживацького контенту, компанія вирішила в розрахунках рекомендації зважати на такі фактори як час перегляду, типи пристроїв, жанр фільму, акторський склад, географічне місцезнаходження тощо.

Так само як і Amazon, Netflix згодом прийшов до рішення про впровадження нейронних мереж у свою систему, так як даних користувачів ставало дуже багато. Також компанія окрему увагу звернула на те, щоб окремих груп аудиторій давались рекомендації на основі алгоритмів, саме тому дорослий глядач може отримати рекомендацію, яка буде відрізнятися від рекомендації, яка була надана дитині. [60] Система написана базуючись на

мовах програмування Java, JavaScript, Scala. [58]

Spotify ділиться [61] тим, що використовує різноманітні алгоритми для створення рекомендацій для своїх користувачів. Так, вони наводять приклад комбінування колаборативної фільтрації (враховується смаки користувачів, які схожі між собою), аналізу контенту (жанр, артисти, треки) та поведінки самого користувача на платформі. Що варто зазначити, Spotify дає користувачам можливість управляти рекомендаціями через власні налаштування. Окремо компанія підкреслює, що усвідомлює те, що її система працює із персональною інформацією користувачів, а тому окремо дбає про захищеність цих даних та їх конфіденційність.

Платформа Youtube [62] є одним із сучасних лідерів серед платформ для перегляду відеоконтенту та має рекомендаційну систему, яка працює за допомогою платформи машинного навчання Google TensorFlow. Система складається з двох нейронних мереж, які відповідають за рекомендацію відео для користувачів. Перша з них – це генерація кандидатів, яка за допомогою колаборативної фільтрації, створює список кандидатів відео на основі сторії переглядів користувача. В своїх розрахунках система опирається на вік кожного елементу навчання, неявний зворотній зв’язок (приміром, переглянуті відео) та рідше бере в розрахунок явний зворотній зв’язок (вподобання, опції не подобається і т.д.). Друга ж нейронна мережа проводить калькуляції рейтингу відео за допомогою логістичної регресії, де основним критерієм є очікуваний час перегляду, а не частота кліків, щоб уникнути промоції клікбейтних відео. Також використовується важена варіація логістичної регресії, де час перегляду для позитивних взаємодій має більшу вагу, ніж для негативних, що допомагає зосередитись на відео, які користувачі насправді дивляться. Дана нейронна мережа виступає більш простішим етапом в роботі системи, так як кількість відео для ранжування менша і є більше даних про кожне відео. [63]

Алгоритми YouTube аналізують перегляди користувачів, пошукові запити, а також взаємодії з відео (лайки, коментарі, підписки), щоб створити персоналізовані пропозиції. Чим більше ви взаємодієте з платформою, тим

точніші та релевантніші рекомендації. Їх система постійно збирає зворотний зв’язок від користувачів, дозволяючи регулювати свої уподобання за допомогою опцій «не цікаво» або «не показувати відео з цього каналу».

Не вийде пропустити таку популярну платформу як TikTok, адже саме за рахунок своїх алгоритмів рекомендації контенту, компанія змогла вибороти собі лідируючі місця в світі соціальних мереж. На сторінці [64] компанії розробники діляться деталями того, як саме працюють їх алгоритми. TikTok використовує складний алгоритм для рекомендації відео на основі взаємодії з платформою (перегляд відео, метаданні про відео, географічна локалізація тощо). Також платформа аналогічно до Youtube дає можливість користувачам самостійно обирати тип контенту, який ті не хочуть бачити в стрічці рекомендацій.

Інший гігант соціальних мереж, компанія Meta, яка володіє відомими продуктами Instagram та Facebook, ділиться в своєму блозі інформацією [65] як працюють їх рекомендаційні системи. Даний алгоритм враховує понад 100,000 (лайки, коментарі, репости, частота відвідування окнкретних сторінок тощо) факторів для того, щоб показати користувачам найбільш релевантний контент у їх новинній стрічці. Система також оцінює, скільки часу користувач проводить на платформі, або ж які пости він переглядає, чи відволікається він на інші публікації, а також які елементи контенту викликають більше взаємодій. Як і TikTok, дає змогу користувачам фільтрувати тип контенту, які вони бачать. Варто також відзначити, що компанія постійно ділиться деталями роботи своєї рекомендаційної системи та дбає про її прозорість, адже дану платформу користувачі можуть сприймати, як простір для споживання новин, так і як маркетплейс. Для написання Facebook використовувались різні мови програмування, серед яких PHP, C++, Python, Java, Perl [58].

Ще одна популярна соціальна мережа Reddit описує в статті [66] деталі рекомендаційної системи. Платформа намагається рекомендувати користувачам контент, на основі того, чи відповідає конкретний пост їхнім інтересам, які обраховаються на основі їхньої взаємодії з платформою (пости, коментарі і

т.д.), а також на основі міри популярності такого контенту. Як і в інших соціальних мережах, користувачі Reddit можуть коригувати рекомендації контенту за допомогою зворотнього зв’язку.

Що стосується соціальної мережі X (колишній Twitter), то в 23-ому році розробники випустили статтю [67], де розповідають про нововведення в алгоритмі рекомендацій твітів. Якщо раніше рекомендації в основному ґрунтувалися на популярних твіттах і акаунтах, то, з часом система еволюціонувала, і тепер акцент зміщений на більшу персоналізацію в пропозиціях. Також системою активно використовується машинне навчання для аналізу користувацьких взаємодій, таких як лайки, ретвіти та коментарі, аби створити більш точні рекомендації. Такі алгоритми дозволяють враховувати індивідуальні переваги користувачів або ж передбачати, тип контенту, в якому вони будуть зацікавлені. Як і всі соціальні мережі Twitter надає можливість для користувачів налаштовувати фільтрацію контенту. Також компанія вирішила відкрити частину свого коду, що надасть можливість стороннім розробникам досліджувати та вдосконалювати алгоритми рекомендацій.

Юрі Бровман – провідний розробник в компанії eBay ділиться в своїй праці [68] секретами роботи рекомендаційної системи його компанії. Він зазначає, що в їхній системі активно використовується Vertex AI Vector Search – технологія, яка дозволяє здійснювати пошук за допомогою векторних уявлень даних, що дозволяє покращувати точність рекомендацій, їх швидкодію наданні, а також персоналізацію та масштабованість. Головна специфіка такого підходу в тому, що кожен товар на платформі перетворюється у вектор, який відображає його характеристики, такі як ціна, категорія тощо.

У своєму блозі компанія Grammarly ділиться ньюансами роботи їх системи [69]. Мета їх рекомендаційної системи є трохи специфічною та відрізняється від інших систем, які в даній роботі представлені, адже вона фокусується на тому, щоб допомогти користувачам покращити їхні тексти, надаючи рекомендації з граматики, стилю, пунктуації та інших аспектів письма. В основі своєї роботи система використовує нейронні мережі – для точнішого

розуміння контексту тексту та виявлення помилок у більш складних конструкціях, а також моделі машинного навчання, які навчаються на великому обсязі текстів для розпізнавання граматичних помилок та неточностей. Разом із пропонованою зміною також надається пояснення даного рішення, що допомагає користувачеві вчитися на власних помилках. Такий набір процесів і методів, які дозволяють користувачам розуміти та довіряти результатам, які були створені алгоритмами машинного навчання, називають пояснюваним штучним інтелекток (XAI).

Пояснюваний штучний інтелект використовується для опису моделі штучного інтелекту, її очікуваного впливу та потенційних упереджень. Це допомагає охарактеризувати точність моделі, справедливість, прозорість і результати в процесі прийняття рішень за допомогою ШІ. Штучний інтелект, який можна пояснити, має вирішальне значення для створення довіри та впевненості під час впровадження моделей ШІ у виробництво. Можливість пояснення ШІ також допомагає організації прийняти відповідальний підхід до розробки ШІ.» [70]

«Пояснювальний штучний інтелект – це інструмент, який здатен надавати відповід на фундаментальні питання «Як?» та «Чому?», які стосуються системи штучного інтелекту. Це, в свою чергу, може бути використано для вирішення правових та етичних питань, важливість та актуальність яких із часом тільки збільшується. Внаслідок цього, дослідники в галузі штучного інтлекту як необхідну характеристику надійного ШІ визначили саме XAI.» [71]

Пояснювальний штучний інтелект використовує техніки, які чудово поєднують в собі точність та інтерпретваність моделей машинного навчання. Це може досягатися двома способами [72]:

1) розробка моделей, які вже є самі по собі інтерпретовані (білі або ж сірі коробки);

2) якщо перші моделі не дали необхідної точності результату, то тоді необхідно для моделей, які є не зрозумілими та закритими (чорними коробками) надати певний рівень інтерпретованості.

Тому на основі цього може зробити, що для нас є два важливі терміни:

1) Інтерпретованість – дозволяє розробникам заглиблюватися у процес прийняття рішень моделі. Така можливість надає розуміння, як модель отримує результати.

2) Пояснювальність – надає пояснення про рішення системи вже не для розробника, а для кінцевого користувача. Це спонукає останнього більше довіряти даній системі, так як він розуміє на чому будувалося рішення системи та чи воно є правильним для нього.

Пояснювальний штучний інтелект базується на чотирьох принципах [73]:

1) Пояснення – здатність системи аргументувати свої результати або процеси. Штучний інтелект або система машинного навчання надає пояснення, чому було вжито цей конкретний результат або дію. Це може включати інформацію про дані, які було використано, застосування конкретної логіки їх обробки або ж інших критеріїв, які вплинули на прийняття рішень.

2) Змістовність – пояснення, які надала система, мають бути зрозумілими для цільового користувача. У випадку систем штучного інтелекту таким користувачем може бути розробник, користувач і т.д. Пояснення повинно бути зрозумілим і відповідати рівню розуміння користувача і не бути занадто технічним або ж абстрактним.

3) Точність пояснення – це те наскільки правильно пояснення системи представляє причину результату або процесу, який використовується для створення цього результату. Пояснення має точно описувати, чому отримано саме такий результат або як система прийшла до конкретного рішення, так як неточні пояснення можуть призвести до недовіри до системи, що є протилежним по своїй суті для XAI.

4) Межі знань – система повинна функціонувати лише в тих умовах, для яких вона розроблена і повинна створювати вихідні дані лише тоді, коли вона має достатній рівень впевненості в правильності своїх результатів. Якщо достовірність системи низька, то пояснення або результати можуть бути ненадійними.

Автори статті [74] поділяють методи пояснюваного ШІ на дві великі категорії: інтерпретовані моделі та методи післяобробки.

1) Інтерпретовані моделі – це моделі, які за своєю природою є зрозумілими та прозорими. Вони надають змогу безпосередньо інтерпретувати результати без необхідності додаткових пояснень.

a. Лінійні моделі та логістична регресія – це моделі, які є простими та інтерпретованими, та в яких можна чітко простежити, як кожна характеристика впливає на результат.

b. Дерево рішень – це інтерпретовані моделі, в яких структура побудована у вигляді дерева. Це дає змогу зрозуміти, як приймаються рішення на основі вхідних даних. Кожен шлях у дереві рішень представляє умови, які були застосовані в прийнятті того чи іншого рішення.

2) Методи післяобробки – методи, які використовуються для пояснення складних моделей, що не мають прямої інтерпретованості.

a. LIME (Local Interpretable Model Agnostic Explanations) – це метод, який створює локальні інтерпретації для кожного прогнозу складної моделі. Він працює шляхом побудови інтерпретованої моделі, яка аппроксимує поведінку складної моделі в межах конкретного прикладу.

b. SHAP (SHapley Additive exPlanations) – метод, який базується на теорії ігор, зокрема на концепції «Шеплі» для визначення важливості кожної ознаки в прийнятті рішення моделлю. Даний метод дозволяє точно оцінити внесок кожної ознаки в фінальний результат.

c. PDP (Partial Dependence Plots) – метод, який використовується для відображення взаємозв’язку між окремими ознаками і результатом моделі, що у свою чергу дозволяє побачити, як зміна конкретної ознаки впливає на прогноз.

d. Saliency Maps – метод, який використовується для інтерпретації моделей глибокого навчання, зокрема, в контексті зображень. Він створює «теплові карти», які вказують на найбільш важливі пікселі в зображенні, що були враховані моделлю для прийняття рішення.

Широко XAI використовується в фінансовій галузі та медицині. Автори

статті [75] відзначають важливу роль таких систем у забезпеченні прозорості, довіри та етики в галузях медицини та фінансів. У медицині це допомагає лікарям краще розуміти рішення штучного інтелекту, що важливо для точності діагнозів та планування лікування. У фінансах XAI сприяє підвищенню довіри до фінансових рішень, забезпечуючи прозорість у процесах кредитування, інвестицій та виявлення шахрайства. Також, автори зазначають, що XAI ще досі є доволі непростим для впровадження в такі специфічні системи, так як обидві галузі стикаються з проблемами балансування точності моделей і забезпечення їх пояснюваності.

У статті [76] йдеться про застосування XAI в IBM Watson Health для того, щоб покращити довіру лікарів до систем, забезпечуючи прозорість в процесах прийняття рішень та надаючи можливість для аудиту і верифікації результатів. Наявність XAI дозволяє лікарям зрозуміти, на яких факторах засновані висновки Watson Health. Це приклад вдалого використання штучного інтелекту, коли така технологія стає потужним інструментом в руках людини, а не її замінником.

Інший приклад застосування в медицині – це Google DeepMind [77], яка пропонує новий підхід до XAI, де вони зосереджуються на створенні моделей, які не лише приймають рішення, але й можуть надати пояснення, що є інтуїтивно зрозумілими для людей. Наприклад, це може бути виділення важливих вхідних даних, які вплинули на результат, або пояснення логіки, яка лежить в основі рішення.

У фінансовому секторі XAI активно використовує FICO [78] – система оцінки кредитного ризику. Вона являє собою платформу для роботи з аналітикою та штучним інтелектом та дозволяє компаніям будувати, тестувати та впроваджувати аналітичні моделі на основі даних, а також здатна забезпечити зручний інтерфейс для роботи з великими даними і алгоритмами AI. Так як багато AI-моделей (зокрема, складні моделі на основі глибокого навчання) можуть бути «чорними ящиками», компанія працює над тим, щоб зробити їх висновки зрозумілішими для користувачів, за рахунок пояснення

рішень моделей машинного навчання та штучного інтелекту.

Або ж вдалий приклад використання XAI в своїй системі має Zest AI [79], яка використовує технології, що дозволяють створювати прозорі та інтерпретовані ШІ-моделі для кредитування. Платформа може надавати пояснення для кожного рішення, що приймається моделлю, щоб кредитори могли зрозуміти, які саме фактори вплинули на результат (наприклад, рівень доходів, кредитна історія і т.д.).

XAI також можуть застосовуватися й сфері освіти. Наприклад, у статті [80] розглядаються впливи розвитку ШІ на освіту і наголошується на важливості XAI в освітньому контексті. Спричинено це тим, що технології ШІ розвиваються швидше, ніж соціальні та правові аспекти їх впровадження, через що і виникає певний рівень недовіри в суспільстві. XAI, як напрямок досліджень, сприяє зменшенню таких проблем, в тому числі й в питаннях справедливості, прозорості та етики. У сфері освіти важливість XAI посилюється через питання автономії учнів, а також аспектів, пов’язаних з індивідуальним оцінюваннями та академічною доброчесністю. У статті підкреслюється, що XAI є ключовим елементом для повного використання можливостей і переваг систем, які працюють на основі штучного інтелекту в освіті, та розвитку людського капіталу, і закликається дослідницька та практична спільнота до активного розвитку цього напрямку.

Європейській Комісії також приділяє значну увагу питанню XAI в освіті. Організація проводить постійні робочі зустрічі (наприклад [81]), де розглядаються питання інтеграції XAI в освітній процес такі як:

1) Способи допомогти освітянам і здобувачам використовувати інструменти ШІ, шляхом спрощення термінології.

2) Створення «балів пояснюваності» для інструментів із штучним інтелектом, що дозволить вибирати зрозумілі і прозорі системи.

3) Підвищення грамотності в ШІ серед викладачів і здобувачів.

4) Розробка ШІ-систем на основі принципів справедливості та прозорості таким чином щоб вони відповідали потребам освітнії суб’єктів.

5) Розробка ШІ-інструментів, які будуть орієнтовані на освітні потреби, шляхом створення партнерств між освітянами, розробниками та політиками.

6) Збільшення інвестицій у дослідження застосування ШІ в освіті.

PinSage [82] – рекомендаційна модель, яка використовує графові згорткові нейронні мережі (GCN, Graph Convolutional Networks), та яка розроблена для покращення ефективності і масштабованості платформи Pinterest в плані рекомендаціях контенту. Вона обробляє інформацію через графи, де вузли графа представляють пінги, користувачів або ж інші елементи. Алгоритм цієї системи використовує графові структури, щоб вивести контекстуальну інформацію про пінги та їх зв’язки, а замість традиційних методів фільтрації, PinSage використовує нейронні мережі для обробки зв’язків між елементами графа, що дозволяє точніше передбачати, які пінги будуть цікаві користувачу.

Компанія Uber Eats розробила рекомендаційну систему [83] для надання користувачам інформації про страви та заклади харчування, які їм можуть сподобатись. Алгоритм системи використовує передові методи машинного навчання для розуміння переваг користувачів та здатен генерувати рекомендації кількох типів: персоналізовані (на основі попередніх замовлень користувача) та контекстуальні (враховується геолокація, час доби тощо). Рекомендаційна система використовує навчання на графах [84], яке є потужним інструментом для обробки складних взаємозв’язків між різноманітними елементами в їхній системі. А для надання точніших рекомендацій, в системі також вирішили використати двовежові вбудовування, які здатні на поєднання характеристик користувача з характеристиками об’єкта [85].

Yelp – це компанія , яка займається схожою справою що і Uber Eats, створила рекомендаційну систему [86], яка на основі взаємодії користувача із платформою (відгуки, рейтинг, збережені заклади, геолокація), намагається передбачити, які заклади або послуги будуть найбільш цікавими для нього. Для створення точних рекомендацій система використовує технології машинного навчання та обробки природної мови, щоб визначити, які відгуки є найбільш

інформативними та корисними для інших користувачів. Таким чином, алгоритм враховує фактори тональності відгуків, їх змістабо ж актуальність інформації.

Fitbit – компанія, яка спеціалізується на фітнес трекерах та програмному забезпеченні, яке з ними пов’язане. Належить компанії Google, в блозі якої і ділиться деталями роботи їх рекомендаційної системи [87]. Дана система надає персоналізовані поради на основі фізичних даних, активності та поведінки користувачів, з метою покращенні здоров’я та підтримки фізичної форми користувача. Алгоритми здатні адаптувати рекомендації відносно звичок користувача та надавати поради, які будуть враховувати стиль життя та рівень активності людини. Дана система враховує персональні потреби та особливості користувача та може підказати коли потрібно збільшити навантаження або коли пора відпочити для оптимального відновлення.

У вересні 2011 року популярна платформа онлайн-книг Goodreads у своєму блозі анонсувала [88] появу в системі рекомендаційної системи, так як хоче покращити для користувачів досвід читання книг шляхом пошуку контенту, який буде найкраще відповідати їх вподобанням. Їх алгоритм враховує книги, які вподобав користувач, оцінки, коментарі проставлені ним, а також береться в розрахунок дані інших користувачів, які мають схожі інтереси.

Стаття [89], яка розмішена на сервісному сайті IMDb, дає корисну інформацію про те, як працюють рекомендації на даній платформі. Приміром, на основі різних факторів, таких як історії переглядів користувача, його оцінок, а також популярності фільмів та серіалів серед інших користувачів, система допомагає знайти новий потенційно цікавий контент для користувача. IMDb також пропонує популярні списки фільмів, які наразі є найпопулярнішими серед глядачів або отримали високу оцінку критиків. Система IMDb створювалась на основі мов програмування ASP.NET, PHP та SQL [58].

Популярна в колах, які пов’язані із ігровою індустрією, платформа Steam випустила новину [90] в своєму блозі, що вони запускають роботу інтерактивного радника. На основі великого об’єму ігрових даних користувачів,

методи машинного навчання опрацьовують інформацію та шукають гравців, які мають схожі між собою вподобання. Таким чином, якщо одному гравцю з певними інтересами гра сподобалося, то й іншому гравцю з такими інтересами гра б мала сподобатись. В системі явно спостерігається проблема холодного старту, так як даний інтерактивний радник не здатен надавати рекомендації по продуктах, в які ще ніхто не грав. Водночас, аналіз кількісних показників впровадження системи (коефіцієнтів переходів та придбань) підтвердив ефективність інтерактивного підходу для вирішення проблеми упередженості популярності. Згідно зі звітом розробників, завдяки можливості користувача самостійно налаштовувати баланс між «популярним» та «нішевим», рекомендаційний механізм забезпечив охоплення «довгого хвоста» каталогу, сприяючи придбанню понад 10 000 різних найменувань ігор, а не лише бестселерів.

Стаття [91] описує рекомендаційну систему для Etsy та те як вона допомагає користувачам знаходити товари, які найбільше відповідають їхнім уподобанням. Система збирає інформацію про історію замовлень користувачі, про товари, які той зберігав та про переглянуті товари, щоб потім використати їх у своєму аналізі. Etsy також враховує вподобання інших користувачів, що мають схожі інтереси, що дозволяє отримувати пропозиції про товари, що можуть бути популярними серед людей із схожими до вподобань цільового користувача.

Стаття [92] на блозі DeepMind на платформі Google проливає трохи світла на те, як покращене машинне навчання допомагає користувачам Google Play Store знаходити персоналізовані додатки, за рахунок глибоких нейронних мереж. Система використовує дані про попередній досвід користувача, а також оцінки та відгуки інших користувачів і т.д. Окрім того, платформа здатна враховувати приховані патерни в поведінці користувачів, що в свою чергу дозволяє забезпечити більш релевантні і точні рекомендації. Для поліпшення персоналізованих рекомендацій, Google Play дозволяє користувачам налаштовувати свій досвід, наприклад, за допомогою встановлення певних

інтересів або оновлення свого профілю, що дозволяє отримувати більш точні рекомендації, що відповідають вашим потребам [93].

Платформа Apple Music також використовує вдосконалені механізми персональних рекомендацій, описані в [94]. Робота системи базується на аналізі двох типів даних: явного зворотного зв'язку, до якого відносяться безпосередні дії користувача (лайки, додавання треків до бібліотеки, створення плейлистів), та неявного – історія прослуховувань, пропуски композицій та тривалість сесій. Такий комплексний підхід дозволяє платформі не лише задовольняти поточні запити користувача, але й реалізовувати стратегію «відкриття», активно пропонуючи нові твори, що семантично близькі до вподобань слухача, тим самим урізноманітнюючи його користувацький досвід та запобігаючи ефекту «фільтраційної бульбашки».

Яскравим прикладом проблеми із упередженими результатами рекомендаційної системи є компанії LinkedIn, ZipRecruiter та Monster. Їх алгоритми хоч і допомагають підбирати кандидатів, групувати резюме або ж надавати рекомендації щодо певних вакансій, але ці системи можуть мати приховані упередження (гендерні, расові чи соціально-економічні), які впливають на рішення щодо кандидатів. Такими деталями у своїй статті [95] ділиться автор та зазначає, що дослідження показали, що деякі алгоритми можуть дискримінувати жінок або расові групи, навіть у випадку, якщо кандидати мають однакові кваліфікації. Що означає, що при проектуванні сучасних рекомендаційних алгоритмів критично важливо не лише оптимізувати метрики точності, а й забезпечувати прозорість прийняття рішень та впроваджувати механізми перевірки на упередженість, щоб гарантувати рівний доступ до можливостей для всіх користувачів.

Проведений аналіз існуючих систем дозволив розділити існуючі підходи на дві групи:

1. Комерційні рекомендаційні системи загального призначення (e-commerce, медіа, соцмережі), які демонструють передові методи обробки великих даних.

2. Спеціалізовані освітні системи (EdTech та наукові інші прототипи систем), які безпосередньо стосуються предметної області нашого цільового дослідження.

*Таблиця 1.3*

**Характеристика методів реалізації рекомендаційних систем у комерційному секторі**

| **Платформа / Компанія** | **Тип даних** | **Еволюція алгоритмів (Основний метод)** | **Технологічний стек** | **Особливості** |
|---|---|---|---|---|
| Amazon, eBay, Etsy | Товари (E-commerce) | Від Item-to-Item CF до глибокого навчання та векторного пошуку | Java, Python, C++, Scala | Використання історії «покупок» (аналог вибору дисциплін), боротьба з холодним стартом |
| Netflix, YouTube, IMDb | Відеоконтент | Гібридні моделі: Матрична факторизація та нейронні мережі (Google TensorFlow). Генерація кандидатів й ранжуванн) | Java, Scala, Python, Node.js, React | Врахування контексту (час доби, пристрій) та багатофакторне ранжування |
| Spotify, Apple Music | Аудіо | Гібридний:NLP (аналіз тексту пісень/жанрів), аналіз аудіо-сигналу | Python, Java | Баланс між «Exploitation» (те, що подобається) та «Exploration» (нові відкриття) |
| Meta, TikTok, X (Twitter), LinkedIn | Соціальні графи | Графові нейронні мережі, аналіз поведінкових патернів (сто тисяч факторів) | PHP, C++, Python, Java, Perl | Висока швидкість обробки та виявлення прихованих зв’язків |
| Pinterest | Візуальний контент | Graph Convolutional Networks (PinSage) | Python | Ефективна робота з графовими структурами даних |
| Grammarly | Текст | NLP, Трансформери, Нейронні мережі | Lisp, Python | Пояснювальні рекомендаційні системи |

Аналіз еволюції алгоритмів провідних платформ (табл. 1.3) свідчить, що метод колаборативної фільтрації був і залишається фундаментальним етапом становлення будь-якої рекомендаційної екосистеми. Перехід до складних

гібридних моделей та глибокого навчання у комерційному секторі відбувався поступово, у міру накопичення «Big Data». Для університетського середовища, де обсяги даних є меншими за масштаби глобальних корпорацій, саме колаборативний підхід є найбільш доцільним варіантом для стартової реалізації системи підтримки прийняття рішень.

**Освітні платформи** також активно використовують рекомендаційні системи, адже це допомагає їх користувачам легше знаходити корисний для них матеріал або ж всебічно розвиватися. Так, яскравим прикладом є платформа Coursera, яка використовуючи GPT і генерацію з доповненням через пошук (RAG), вона адаптує рекомендації до конкретних цілей і переваг користувача. RAG використовує контекстуальну інформацію з великої бази знань для надання чітких пояснень кожної рекомендації, що в свою чергу також вирішує проблему «холодного старту», коли система не має достатньо даних про нових користувачів, використовуючи багатий контекстуальний матеріал з бази знань для генерування рекомендацій. Система є розгорнутою на Hugging Face Spaces і використовує GPT-3.5 Turbo, OpenAI embeddings та ChromaDB в рамках Langchain і Streamlit [96].

Інша популярна освітня платформа Moodle, також має власну рекомендаційну систему, в якої назва схожа на назву системи Coursera, – MoodleRec. Дана система є розширенням для основної платформи та головне її призначення полягає в наданні викладачам більш точних і релевантних навчальних матеріалів. Її алгоритми працюють на основі контентного та колаборативного фільтрування. Також слід зазначити, що вона дозволяє не тільки знайти потрібні об'єкти, а й оцінити, як вони використовуються іншими викладачами в подібних курсах, що робить процес навчання більш персоналізованим і ефективним [97].

Про ще одну відому в освітніх колах платформу Quizlet, розповідає в своїй статті [98] один із розробників системи. Посилання на цю статтю зробила сама компанія Quizlet, тому ми можемо опиратися на інформацію в ній. Принцип роботи їх рекомендаційної роботи полягає в аналізі великої кількості

даних за допомогою машинного навчання та на основі цього надання рекомендацій щодо навчальних карток, які будуть найбільш корисними для користувачів на основі їхніх попередніх взаємодій.

Опираючись на статтю [99] одного із розробників компанії Headway, можемо ознайомитись із деталями реалізації Nibble – новий освітній застосунок, який містить інтерактивні уроки на різні тематики. Дана платформа містить функціональність рекомендаційної системи, головна мета якої полягає в допомозі користувачам в отриманні більше знань за менший час, підвищуючи їх рівень через рекомендації персоналізованих уроків, що відповідають інтересам та цілям користувачів. Система використовує колаборативний фільтр User-Item, створюючи рекомендації на основі взаємодій користувачів з уроками через систему оцінок. Такий фільтр аналізує, які уроки користувачі оцінили високо і надає рекомендації подібним користувачам. Збір даних в системі реалізується за допомогою Amplitude для збору даних в реальному часі, а дані зберігаються у BigQuery і використовуються для навчання моделі машинного навчання. Також автор зазначає, що внаслідок тестування ефективності системи, вони вивявили, що впровадження системи рекомендацій дозволило підвищити середню оцінку уроків на 3 %, завдяки рекомендаціям дані про оцінки стали більш рівномірно розподіленими між різними уроками, що поліпшило аналітику.

Prometheus – це ще один приклад освітньої платформи, яка усвідомила необхідність рекомендаційної системи, метою якою стане аналіз даних про використання матеріалів та педагогічні цілі викладача для надання ресурсів, які найбільше підходять в цьому випадку. Полегшення процесу створення курсів для викладачів відбувається завдяки використанню штучного інтелекту та обробки природної мови, завдяки чому викладачі можуть зосередитися на своєму основному завданні – навчанні, а не витрачати час на пошук та організацію матеріалів для курсу. Також система використовує показники ефективності, щоб відстежувати її вплив на роботу викладачів і допомогти вдосконалити процес навчання. [100]

Також варто зазначити наукові напрацювання в галузі рекомендаційних систем. Приміром ось стаття [101], де описується робота рекомендаційної системи для вивчення іноземних мов. На основі гібридного алгоритму, який складається із випадкового та k-NN алгоритмів, система персоналізовано підбирає навчальний контент для користувача. В матеріалі представлено деякі деталі вебсервісу, який було розроблено на мовах програмування Python, Java-Script на платформі NodeJS, з використанням бази MongoDB та таких інструментів як Redis та Fastify.

У науковій статті [102] йдеться про розробку системи рекомендацій для вступників до вищих навчальних закладів в Україні. Метою даної системи є допомогти абітурієнтам зробити обґрунтований вибір спеціальності, яка відповідає їхнім здібностям, інтересам та фінансовим ресурсам. Розробка базується на методах машинного навчання (наприклад факторизаційних машин), а також на методах майнінгу асоціативних правил, що дозволяє визначити, які спеціальності часто обираються разом.

У своїй роботі [103] автори розглядають створення рекомендаційної системи для ринку електронної комерції. Роботу алгоритму пропонується будувати на основі колаборативної фільтрації та фільтрації по змісту. Основні моменти концепції систем рекомендацій визначаються як персоналізовані рекомендації, рекомендації на основі новин, найкраща покупка та рекомендації для опитування. В статті також пропонується використання методів оцінки продуктів на основі поєднання методів Уїлсона, Байєса та Хакера для нормалізації.

У статті [104] запропоновано підхід до побудови системи рекомендацій курсів, спрямований на мінімізацію ризиків обрання дисциплін, що не відповідають поточному рівню компетентностей здобувача. Дослідження підкреслює обмеженість класичних методів (колаборативної та контентної фільтрації), які, фокусуючись переважно на історії інтересів, можуть ігнорувати об’єктивні показники здатності студента опанувати матеріал. Для вирішення цієї проблеми застосовано алгоритм на основі продукційних правил (IF-THEN),

який забезпечує верифікацію вибору. Механізм роботи полягає у послідовному аналізі пріоритетів здобувача: система зіставляє попередні академічні результати з пререквізитами дисципліни, автоматично відфільтровуючи варіанти, де прогнозується висока ймовірність неуспішності. Такий підхід дозволяє гарантувати академічну відповідність сформованого плану та сприяє покращенню результатів навчання за рахунок більш зваженого розподілу навантаження.

В роботі [105] розглядаються проблеми, з якими стикнулися колаборативні рекомендаційні системи та системи на основі вмісту, та пропонується використання семантичних рекомендаційні системи, що підвищить точність наданих прогнозів та довзолить надати рекомендації для складної предметної галузі або ж коли необхідно явно задати критерії пошуку. Систему реалізовано у вигляді WorkspaceTab-плагіну системи Protégé 5.1.0 з використанням мови програмування Java.

В іншій роботі [106] представлено веборієнтовану рекомендаційну систему для супроводу вивчення музичних творів, що базується на принципі контент-орієнтованої фільтрації. Програмна реалізація виконана з використанням технологічного стеку Apache, Python та MySQL. Валідація запропонованих моделей здійснювалася за допомогою метрик середньої абсолютної (MAE) та середньоквадратичної (RMSE) похибок, низькі значення яких кількісно підтверджують ефективність впровадження системи та підвищення якості персоналізації онлайн-навчання.

Проведений аналіз сучасної науково-технічної літератури засвідчив, що провідні глобальні IT-компанії вкрай дозовано оприлюднюють деталі функціонування своїх алгоритмічних ядер у класичних рецензованих виданнях, розглядаючи їх як комерційну таємницю та ключову конкурентну перевагу. Натомість, основний потік актуальної інформації щодо практичної реалізації, оптимізації швидкодії та методів масштабування рекомендаційних систем змістився у площину корпоративних технічних блогів та офіційних звітів.

У зв’язку з цим, для ідентифікації найбільш релевантних та сучасних

архітектурних рішень, спектр досліджуваних джерел було цілеспрямовано розширено. Окрім академічних праць, до аналізу було залучено технічну документацію та інженерні кейси розробників платформ Amazon, Netflix, Google, Meta тощо. Такий підхід дозволив врахувати не лише теоретичні моделі, а й емпіричний досвід побудови високонавантажених розподілених систем, що функціонують у режимі реального часу.

*Таблиця 1.4*

**Аналіз освітніх рекомендаційних систем та наукових розробок**

| **Система / Джерело** | **Мета системи** | **Застосовані методи** | **Обмеження / Недоліки в контексті побудови ІНП** |
|---|---|---|---|
| Coursera | Рекомендація курсів (MOOC) | GPT-3.5 Turbo, OpenAI embeddings та ChromaDB в рамках Langchain і Streamlit | Орієнтована на додаткову освіту без жорстких структурних обмежень |
| MoodleRec (Moodle) | Рекомендація навчальних матеріалів | Класична колаборативна та контентна фільтрація | Працює на рівні ресурсів всередині курсу, а не на рівні стратегії вибору дисциплін. |
| Headway, Quizlet | Мікронавчання, флеш-картки | User-Item CF, Machine Learning на основі логів взаємодії | Вузька спеціалізація (запам’ятовування), не формує цілісну траєкторію |
| Наукові прототипи | Вивчення мов, вступ до ВНЗ, музика | k-NN, Факторизаційні машини, Асоціативні правила | Зазвичай вирішують ізольовані задачі, мають проблему масштабування |
| Система на правилах (Rule-based) | Вибір курсів | Алгоритми IF-THEN (жорстка логіка) | Занадто жорстка система: гарантує коректність, але не забезпечує персоналізацію інтересів (лише перевірка допуску) |

Проведений аналіз (табл. 1.4) дозволив виявити суттєву прогалину між можливостями сучасних комерційних систем та потребами вищої освіти:

1. Невідповідність цільових функцій, у зв’язку з тим, що комерційні алгоритми оптимізовані для максимізації часу перебування на платформі, тоді як мета освітньої системи – максимізація успішності здобувача та відповідності

його вибору кар’єрним цілям.

2. Існуючі наукові підходи до вибору курсів часто базуються на жорстких правилах, ігноруючи приховані вподобання студента. Водночас, класичні методи колаборативної фільтрації в чистому вигляді можуть порекомендувати дисципліну, яку студент не має права вивчати через відсутність необхідних попередніх знань.

3. Для ефективної роботи в освітньому середовищі недостатньо просто скопіювати алгоритми e-commerce. Необхідна модифікація методу колаборативної фільтрації, яка б накладала на матрицю вподобань жорсткі фільтри структурно-логічної схеми підготовки фахівця.

Саме розробка рекомендаційної системи на основі адаптованої колаборативної фільтрації та обґрунтування стратегії її подальшого розвитку до повноцінної гібридної моделі і є завданням даного дисертаційного дослідження.

## 1.4 Проблематика у розробці та використанні рекомендаційних систем для формування індивідуального навчального плану

**Рекомендаційні системи** – це програмні компоненти, які, на основі статистичних даних та моделей машинного навчання, можуть надати персоналізовані рекомендації для користувачів. Вони використовуються дуже часто в наш час та їх можна зустріти в інтернет-магазинах, стрімінгових платформах і т.д. [107]

Як зазначає автора роботи [108], саме 90-ті роки минулого століття прийнято вважати за початок існування сучасних рекомендаційних систем. З розширенням Інтернету, з’явилися нові сфери застосування такого роду систем, і вже на той час з’явилися перші історії успіху в електронній комерції, наприклад, Amazon.com, який став одним із перших великих користувачів технології рекомендацій. З технічної точки зору, перші системи, приміром GroupLens, використовували підхід із матричним заповненням, де алгоритм машинного навчання передбачав відсутні оцінки на основі нещодавніх взаємодій користувачів. А з часом були розроблені нові методи, які стали

використовувати глибоке навчання та інші підходи для передбачення та ранжування товарів.

Однак, незважаючи на теперішні досягнення, існує опасіння, що багато досліджень ґрунтуються на спрощеннях та надмірних припущеннях, а тому важливо більше зосереджуватись на вивченні того, як рекомендаційні системи впливають на поведінку людей та організацій і як досвід користувацького дизайну впливає на ефективність системи.

Сьогодні персоналізовані рекомендації є невід'ємною частиною онлайн-досвіду, і було опубліковано багато звітів про їхню бізнес-цінність.

Будь-яку рекомендаційну систему можна описати по набору характеристик, таких як [109]:

1. Об'єкт рекомендації – це можуть бути різні товари, фільми, новини, або ж, якщо брати до уваги нашу освітню тематику, дисципліни до вивчення, викладачі й т.д.

2. Мета рекомендації – продаж товару, збільшення переглядів, надання рішень в певній проблематиці, підбір необхідного навчального матеріалу.

3. Джерело рекомендації – це той, хто рекомендує. Це може бути група користувачів близькі по інтересах, група профільних експертів і т.д.

4. Контекст рекомендації – це момент надання рекомендації для цільового користувача.

5. Ступінь персоналізації – рівень аналізу, який був проведений системою, перед наданням прогнозу.

6. Прозорість – наскільки підгрунття для прогнозу, яке використовувала система в своїх розрахунках, є зрозумілим для користувача.

7. Алгоритм розрахунку – певний підхід в способі аналізу та калькуляції даних, щоб надати певний прогноз для користувача.

Рекомендаційні системи також активно використовуються в системах управління навчанням, з метою покращення освітнього процесу, автоматизування вибору навчального матеріалу або ж для підбору більш відповідного вектору навчання для здобувача.

Так автори [110] зазначають, що порівняно із рекомендаційними системами для класичних задач, рекомендаційні системи для освітньої тематики мають унікальні характеристики:

1. Нечіткі вимоги учнів – люди, які проходять навчання, можуть знати те, ким вони хочуть стати в майбутньому, але вони можуть не знати, яких саме навичок чи компетенцій їм потрібно для цього здобути.

2. Стиль навчання – учні можуть мати різні потреби та можливості і рекомендації на основі цих факторів повинні видаватись різні.

3. Навчальні курси повинні бути організовані та залежати один від одного, якщо в цьому є логічна потреба, щоб забезпечити поступове проходження матеріалу учнем та унеможливити складання просунітішої дисципліни перед базовими.

4. Навчальний маршрут – для тих, хто навчається тривалий час, важливо отримати не тільки рекомендацію відносно однієї дисципліни, а отримувати їх стосовно всіх інших та в будь-який момент часу.

Системи управління навчанням (Learning Management Systems, LMS) стають важливим інструментом у сучасних освітніх процесах, так як вони допомагають автоматизувати навчання, відстежувати його прогрес та надавати доступ до навчальних матеріалів в незалежності від того, де перебуває учень [111]. Всі свої переваги такі системи найбільше показали у часи ковіду та широкомаштабного вторгнення росії в Україну, коли обставини змусили освітні заклади перейти на дистанційний формат та впровадити більш гнучке навчання.

У LMS можна відмітити такі основні можливості [112]:

1. Адаптивний дизайн – користувачі повинні мати змогу отримувати доступ до контенту із будь-якого пристрою, не залежно від його розширеня та роздільну здатність. Якісні системе автоматично підбирають той адаптивний вигляд, який буде доцільніше для пристрою користувача.

2. Зручний інтерфейс – користувач повинен бачити логічно розставлені елементи сайту, які відповідають вимогам його візиту на даний ресурс. Навігація до потрібного розділу для користувача має бути інтуїтивно

зрозумілою та швидкою.

3. Аналітика та звітність – На різних етапах учень активно взаємодії із платформою та її функціоналом. Якісна система повинна, за потребою, надавати статистику по роботі певного користувача для адміністратора чи викладача. Такий функціонал може надати аналіз успішності учня або звіт про його прогрес, щоб в подальшому мати можливість скоригувати індивідуальний навчальний план.

4. Ведення каталогу та курсу – люди, в яких є певні права, мають мати можливість створювати курси із навчальними матеріалами та керувати контентом в середині них.

5. Взаємодія та інтеграція вмісту – контент збережений в таких системах має відповідати ряду стандартів, включаючи SCORM [113] та інтерфейи прикладного програмування.

6. Служби підтримки – система має надавати рівень підтримки для користувачів, щоб вирішувати незрозумілі труднощі для користувачів.

7. Підтримка сертифікації та відповідності – функція, яка надає можливість адміністраторам та менеджерам оцінювати реальний рівень навичок в учня.

8. Соціальне навчання – функціонал, який дозволяє взаємодіяти із навколишнім соціумом, за допомогою соціальних мереж. В такому випадку, учні мають змогу знаходити для себе однодумців, партнерів для проекту і т.д.

9. Підтримка ігрового режиму – дана функціональність дозволяє використовувати ігрові механіки в процесі вивчення, що допомагає учням краще засвоювати матеріал. Такий підхід до навчання вже використовується в деяких LMS та може базуватися на стимулювання за допомогою вигранних балів в проходженні завдання, присвоєння перемог учасникам і т.д.

10. Локалізація – такі системи мають підтримувати багатомовність, адже часто в університетах навчають іншомовні здобувачі, які не повинні стикатися із труднощами в користуванні такими системами.

11. Штучний інтелект – один із напрямків, який все більше розвивається

в даній сфері, та якому вдається все більше захопити різноманітних процесів. Зараз він може як допомогати учню у вирішенні проблем, з якими той стикається в процесі вивчення матеріалу, так і може допомогти із формуванням персоналізованого навчального контенту для учня, який буде грунтуватися на основі його вже отриманиих навичок.

Штучний інтелект дозволяє не тільки автоматизувати процеси (автоматичне оцінювання навчальних робіт чи допомога у генерації навчального коненту), а й підвищити ефективність самого навчання за допомогою персоналізованих рекомендацій, аналітики і т.д.

Основні ролі штучного інтелекту в LMS, які ми можемо вивести на основі думок авторів [114] та вже вище розглянутих основних можливостей LMS:

1. Персоналізація навчання – функціонал, який дозволяє створити середовище навчання, яке здатне підлаштовуватися під потреби учня, завдяки чому ті отримують навчальний контент, що найкраще відповідає їх рівню знань та потребам. Це може бути:

a. Адаптивне навчання, яке за допомогою аналізу даних про стдуента, таких як успішність, швидкість опанування матеріалу і т.д. може динамічно адаптувати процес навчання. Приміром, у випадку, коли здобувач часто помиляється в певній темі, система може запропонувати додаткові матеріали для покращення розуміння цього розділу.

b. Персоналізовані рекомендації, які на основі даних про попередній прогрес учня чи групи учнів пропонують дисципліни, матеріали або завдання, які найкраще відповідають їм за певними параметрами.

2. Навчальні асистенти – чат-боти, які побудованим на основі штучного інтелекту та які здатні надати для здобувача допомогу у вирішенні складних питань чи в покращенні розуміння теми в режимі реального часу. Такі асистенти можуть:

a. Відповідати на конкретні питання здобувача.

b. Пояснювати складні концепції.

c. Надавати додаткові матеріали для покращення розуміння.

d. Надати здобувачам зворотний зв’язок щодо процесу виконання завдань.

e. Слідкувати за календарем та нагадувати про важливі події.

3. Створення контенту – штучний інтелект може допомогти викладачу чи адміністратору створити навчальний матеріал або складанні запитань та тестів до цього матеріалу.

4. Розуміння вмісту – штучний інтелект здатний здійснювати аналіз графічних й текстових матеріалів для створення короткого змісту і надання висновків.

5. Автоматизоване оцінювання – дозволяє автоматизувати оцінювання, що звільняє чимало часу на інші задачі для викладача і, навіть, може надати зворотній зв’язок по роботі учня та що в ній було не так і як це можна виправити в майбутньому.

6. Звітність – система здатна за допомогою штучного інтелекту збирати дані про здобувачів та їх активність та на основі них і додаткових факторів робити аналіз щодо ефективності засвоєння навчальних матеріалів. Більш просунуті моделі здатні надати також припущення щодо того, який матеріал може покращити розуміння та сприяняття певної тематики для конкретного здобувача

7. Гейміфікація з використанням ШІ – на основі результатів учня штучний інтелект може змінювати складність задач або ж змінювати самі завдання, що надасть для учня унікальний навчальний досвід та допоможе краще зрозуміти матеріал.

8. Прогнозування успішності здобувачів – аналізуючи велику кількість даних та використовуючи машинне навчання система отримує здатність прогнозувати успішність учнів. На основі заздалегідь визначених факторів можна будувати припущення чи успішно здобувач пройде ту чи іншу дисципліну або навчальний матеріал, що дозволить на ранніх етапах виявляти проблеми та змінювати навчальний матріал або ж взагалі навчальний план для здобувача.

## 1.5 Постановка задачі на основі аналізу університетських інформаційних систем

Виходячи із вищеописаного, можемо зробити висновок, що для університету реалізація програмного рішення є дуже важливим чинником для комунікації із здобувачами та надання тому певних послуг. Університети, які подбали про інформаційно-технологічний фактор своєї інфраструктури, мають більше інструментів для залучення додаткової аудиторії до своїх лав. Основним рішенням для реалізації функціоналу університету, в більшості, обирається вебплатформа – потужне рішення, яке покриває більшість потреб.

Багато з університетів мають мобільні застосунки, так як залучення даного типу платформи в екосистему закладу, може підвищити ефективність навчання та зручність взаємодії між користувачами, так як полегшується доступ до навчальних матеріалів, розкладу, чи комунікації між користувачами системи.

*Таблиця 1.5*

**Порівняльний аналіз функціоналу та типів платформ інформаційних систем**

| **Клас систем / Представники** | **Основний функціонал** | **Тип платформи** |
|---|---|---|
| Закордонні університетські застосунки (Harvard Mobile, NAUgo, Qatar University та ін.) | Інформаційний: навігація (карти), новини, події, базовий розклад, екстрені контакти | Веб/Мобільний |
| Комерційні шкільні платформи (HelloParent, Teno App) | Комунікаційний: чати з батьками, оплата послуг, трекінг автобусів, цифрові щоденники | Мобільний застосунок |
| Українські університетські розробки (KNU online, UzhNU, Студент ЧДТУ) | Адміністративний: електронний деканат, перегляд оцінок, довідки, вибір дисциплін (списком) | Веб/Мобільний |
| Державні екосистеми (Дія, Мрія) | Інтеграційний: документи про освіту, ID-картки | Веб/Мобільний |
| AR/VR навчальні інструменти (AR Book) | Навчальний: візуалізація контенту, гейміфікація | Веб/Мобільний |

Однак підсумувавши загальну картину рішень для університетів зарубіжних (табл. 1.5), то ми з легкістю можемо відмітити, що українська галузь в даному питанні просідає. В нашій країні є невелика кількість університетів, які працюють над власними проєктами для цифровізації процесів в сфері комунікації здобувача та закладу вищої освіти, але, навіть, вони є далекі до того рівня, який демонструють представники за кордоном.

В своїй праці «Особливості розвитку ринку інформаційних технологій в Україні» [115] Ситник Оксана та Дубровський Сергій, аналізуючи стан галузі, виділяють низку системних проблем. Варто зазначити, що хоча повномасштабне вторгнення та воєнний стан суттєво змінили структуру IT-ринку та пріоритети держави (акцент на Military Tech, цифровізація держпослуг), зазначені авторами фундаментальні виклики не втратили своєї актуальності, а в деяких аспектах навіть загострилися в умовах війни:

1) низький рівень освіти та необхідність її реформувати;
2) необхідність консолідації всіх учасників індустрії інформаційних технологій;
3) низька зацікавленість цією сферою на державному рівні;
4) відсутність достатньої кількості кваліфікованих викладачів.

Ситуація навіть є більш критичнішою, ніж відсутність власних мобільних застосунків в університетів. В документі «Стратегії розвитку вищої освіти в Україні на 2021-2031 роки» [116] від Міністерства освіти і науки України в третьому розділі, який присвячений світовим трендам розвитку вищої освіти, в пункті «Цифровізація» зазначається: «Освіта наразі відстає від цифровізації, і необхідно докласти більше зусиль, щоб скористатися інструментами та сильними сторонами нових технологій, одночасно вирішуючи проблеми щодо можливих зловживань, таких як кібервторгнення та проблеми конфіденційності».

Тому, зважаючи на приклади реалізація в інших університетах, а також наявних проблем в сфері освіти, пропонується на початковому етапі створення системи комунікації між здобувачем та закладом вищої освіти реалізовувати

функціонал на вебплатформі та організацію побудови архітектури таким чином, щоб з розвитком проєкту розширити його можливості і для мобільних застосунків.

**Висновки до першого розділу**

У розділі вирішено задачу аналізу сучасного стану моделей та методів штучного інтелекту в освіті, а також досліджено проблематику побудови рекомендаційних систем.

Основні результати розділу полягають у наступному:

1. Проаналізовано роль та динаміку розвитку штучного інтелекту в освітній сфері та встановлено, що інтеграція ШІ в системи управління навчанням дозволяє автоматизувати рутинні процеси, персоналізувати контент та прогнозувати успішність здобувачів. Визначено специфічні характеристики освітніх рекомендаційних систем, які відрізняють їх від комерційних: необхідність врахування логічної послідовності курсів, нечіткість цілей здобувачів та довготривалий характер навчання.

2. Здійснено порівняльний аналіз існуючих рекомендаційних систем та методів їх реалізації. Досліджено досвід провідних технологічних компаній (Netflix, YouTube, Spotify, Amazon) та освітніх платформ (Coursera, MoodleRec, Prometheus) та виявлено, що найбільш ефективними є підходи, що базуються на колаборативній фільтрації та гібридних моделях. Це дозволяє використовувати колективний досвід користувачів для вирішення проблеми вибору в умовах невизначеності, що є релевантним для завдання формування індивідуального навчального плану.

3. Проведено огляд інформаційних екосистем ЗВО (Harvard Mobile, Stanford Mobile, українські аналоги: «KNU online», «Univera», «Студент ЧДТУ») та встановлено, що більшість вітчизняних рішень зосереджені на адміністративних функціях (розклад, новини) і не мають інтелектуальної складової для підтримки прийняття рішень здобувачем освіти.

4. Виявлено недоліки існуючої інформаційної інфраструктури в

університетських системах. Аналіз поточного стану інформаційних екосистем ЗВО показав відсутність єдиного механізму для автоматизованого формування індивідуального навчального плану з використанням інтелектуального аналізу даних. Існуючі інструменти не забезпечують персоналізованих рекомендацій, що ускладнює процес вибору дисциплін.

5. На основі порівняння архітектурних рішень визначено, що для реалізації системи формування індивідуальної освітнього плану найбільш доцільним є створення веб-орієнтованої платформи на початковому етапі з подальшим масштабуванням у мобільний сегмент. Це забезпечить кросплатформеність, легкість інтеграції з існуючими базами даних університету та доступність для всіх учасників освітнього процесу.

# РОЗДІЛ 2
# МАТЕМАТИЧНА МОДЕЛЬ ЗБОРУ І ОБРОБКИ ДАНИХ ДЛЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ФОРМУВАННЯ ІНДИВІДУАЛЬНОЇ ОСВІТНЬОЇ СИСТЕМИ

## 2.1 Проєктування високорівневої структури даних рекомендаційної системи формування індивідуального навчального плану

У наш час освітні системи стикаються з численними викликами, серед яких підтримка високої ефективності навчання, адаптація до індивідуальних потреб здобувачів освіти та необхідність в персоналізації освітнього процесу. Одним із основних моментів персонального навчання є проблема вибору навчальних дисциплін для здобувача. У багатьох випадках існуючі освітні платформи не мають інструментів для рекомендацій дисциплін, які б могли забезпечити максимальний результат для кожного здобувача, враховуючи його попередні досягнення, інтереси або ж схильності.

У зв’язку з цим, важливим кроком до підвищення якості навчання є побудова рекомендаційних систем для освітніх платформ. Один із способів вирішення такого роду проблеми є використання алгоритмів машинного навчання, щоб змогти надати персоналізовані рекомендації для здобувачів на основі визначених факторів. Однак, наявні методи потребують постійного вдосконалення, а також розробки математичних моделей, які б забезпечували точність разом із надійністю рекомендацій.

Проектуючи рекомендаційну систему формування індивідуального навчального плану, ми повинні перш за все визначити як саме вона буде працювати. Дані, на основі яких вона буде робити свої передбачення, є дуже важливі та загалом визначають спосіб тренування моделі даних та кінцевий результат прогнозу.

В системі освіти ледь не найважливішим фактором у навчанні є оцінка. Вона є ідентифікатором того, що здобувач пройшов ту чи іншу дисципліну, вона слугує показником якості, наскільки вдало засвоєно матеріал. Не дарма в

головному документі про отримання певний ступінь освіти фігурують дисципліни та отримані із них оцінки. Вони можуть слугувати основним мотиватором в навчанні, допомагати відслідковувати тенденцію успішності навчання здобувача в часовому проміжку і т.д. Чимало роботодавців звертають увагу на оцінки здобувачів в тому чи іншому напрямку, адже це додатковий показник наполегливості, дисциплінованості, здатності до навчання та рівню розвитку в тій чи іншій сфері. Одним словом, оцінки – це не лише інструмент вимірювання успіху здобувача, а й важливий елемент, що може впливати на його мотивацію, розвиток та кар'єрні перспективи.

Саме тому логічним є побудувати рекомендаційну систему формування індивідуального навчального плану на показникові оцінювання знань здобувача по вже пройдених дисциплінах. Такі дані допоможуть нам зрозуміти реальну компетенцію здобувача в тій чи іншій галузі та оцінити його шанси на здачу дисципліни із схожими характеристиками. Прогноз, наданий рекомендаційною системою, дозволить здобувачу оцінити свої шанси на успіх у певній дисципліні, що інколи є ключовим для мотивації того чи іншого вибору. Також, це надасть здобувачу розуміння, чи є конкретна дисципліна відповідною його рівню підготовки. Наприклад, якщо система прогнозує низьку оцінку через недостатню підготовленість до курсу, здобувач може прийняти рішення не обирати цю дисципліну або додатково попрацювати над підготовкою, що допоможе уникнути ситуацій, коли здобувач освіти обирає курс, який є занадто складним для нього, що може призвести до зниження мотивації і навіть неуспішного завершення курсу.  Ну, і також, якщо вирішити обрати оцінку, як основне джерело даних для рекомендацій, то система надасть прогноз оцінки і здобувач може зробити більш обґрунтований вибір, який базується на реальних можливостях, а не лише на популярності дисципліни або її загальній складності. Врахування прогнозу оцінки дозволяє уникнути ситуацій, коли здобувач вибирає дисципліну через її престижність або цікавість, але потім не може її успішно завершити.

Інші фактори, такі як рівень складності курсу, викладач, який веде

дисципліну, інтереси здобувача чи його майбутня кар'єра, звісно, також є важливими при виборі дисципліни. Однак, прогноз оцінки виступає основним фактором, так як він безпосередньо відображає ймовірність успішного завершення курсу, що є ключовим для здобувача. Інші фактори можна віднести на друге місце, оскільки вони більше стосуються загальних аспектів вибору дисциплін, в той час як передбачення оцінки дає конкретну відповідь на питання, чи зможе здобувач досягти успіху в даному курсі на основі своїх поточних здібностей.

Враховуючи це, нам варто розглянути логічну модель даних рекомендаційної системи. Потребувати буде вона в загальному тільки декілька логічних сутностей:

1) Оцінка – сутність, яка дає інформацію по досягненням конкретного здобувача по тій чи іншій дисципліні. Повинна містити наступні поля:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Дисципліна – унікальний ідентифікатор дисципліни;

c. Здобувача – унікальний ідентифікатор здобувача;

d. Оцінка – результат проходження здобувачем тієї чи іншої дисципліни. Відсутність оцінки чи вцілому запису може свідчити про те, що здобувач ще не пройшов успішно дану дисципліну. Низьке значення даного поля також може слугувати для підтвердження невдалого досвіду здобувача у цьому курсі.

e. Та інші, які залежать від специфічних умов здобувачів, закладу і т.д.

Для кожного здобувача $i$ та кожної дисципліни $j$, ми можемо використовувати матрицю оцінок $R$, де $R_{ij}$ – це оцінка здобувача $i$ за дисципліну $j$. Оцінка тут – це числове значення, яке відображають рівень знання здобувача.

$$R = \begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots \\ R_{m1} & \cdots & R_{mn} \end{bmatrix} \quad (2.1)$$

де:

- $R \in \mathbb{R}$ — це оцінка здобувача $i$ на дисципліну j;

- $m$ — кількість здобувачів;
- $n$ — кількість дисциплін.

Якщо оцінка відсутня, то $R = 0$ або інше значення, яке сигналізує, що здобувач не взаємодіяв з цією дисципліною.

Для більш просунутої рекомендаційної системи звичайно що було б добре враховувати інші параметри та фактори, які б бралися до обчислень в залежності від бажання здобувача, однак одна логічна модель (оцінка) цілком задовільняє потреби побудови базового алгоритму рекомендаційної системи.

2) Рекомендація – це сутність, яка є результатом роботи рекомендаційної системи. Вона містить запропоновані курси для здобувача, виходячи з аналізу історії вибору дисциплін, оцінок здобувача та оцінок інших здобувачів. Може бути представлена у вигляді списку курсів з прогнозованими оцінками або рівнем ймовірності того, що здобувач обере цей курс. Поля моделі:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Здобувач – користувач, для якого надавався прогноз рекомендаційною системою;

c. Значення – дані про те, які курс чи курси потрібно обрати та на основі якої інформації йому це слід робити.

Однак, будуючи рекомендаційну систему, ми усвідомлюємо, що вона не може існувати у вакуумі. Обов’язково повинний бути супутній програмний модуль, в який буде підключена така система або ж вона буде написана безпосередньо в самому функціональному модулі. Це повинен бути компонент, який має бути спорідненим до мети рекомендаційної системи та допомагати будувати вектор навчання для конкретного здобувача.

За таку роль може відповідати функціональність вибору дисципліни. Враховуючи закон про індивідуальний план навчання, про який вже згадувалося в попередньому розділі, така логіка потрібна буде для здобувача протягом всього періоду навчання. Однак, навіть зараз здобувачі проходять щорічний етап вибору дисципліни, який ідеально підходить, якщо йде мова про

пошук місця для інтеграції рекомендаційної системи.

Нам варто розглянути логічну модель даних такої системи, а також розібрати необхідні суності та їх поля, щоб побудувати працюючу систему, за допомогою якої здобувачі зможуть обирати дисципліни та рекомендації по їх спорідненості із можливостями чи вподобаннями здобувача.

Отож, система яка буде давати здобувачам освіти можливість обирати дисципліни за власними вподобаннями повинна містити такі логічні моделі:

3) Дисципліна – сутність $j$, яка дає інформацію про дисципліну, яка доступна до вибору, та яка повинна містити поля $C_j$:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Назва – рядкове поле, яке також слугує ідентифікатором певної дисципліни, але яке не є унікальним, та слугує більше для ідентифікації для користувачів системи;

c. Належність дисципліни до тієї чи іншої групи здобувачів – поле, яке дає розуміння чи є конкретна дисципліна доступна для конкретного здобувача. Допускається можливість використання декількох полів для позначення такої принадлежності до тієї чи іншої вибірки здобувачів. Також може виступати окремою моделлю, що є приміром класичним підходом в проєктуванні баз даних;

d. Часова належність – поле, яке дає розуміння, коли викладатиметься дана дисципліна. Дає розуміння щодо її доступності для здобувачів, а також може виступати важливим фактором у випадку калькуляцій прогнозу рекомендаційної системи (у випадку, якщо нам потрібні будуть свіжі дані);

e. Активність – дисципліна може не проводитись через різний ряд факторів одного року, але проводитись наступного. Такий параметр часто зустрічається в системах, щоб не видаляти потенційно корисні дані сутності, які можуть бути в пригоді в майбутньому. В такому випадку, такому полю проставляють відповідне значення активності, внаслідок чого ці конкретні будуть або братись, або ігноруватись в процесі зчитування інформації;

f. Викладач – опціональний параметр, який не є життєвонеобхідний для вибору дисципліни, але який часто виступає вирішальним фактором у виборі дисципліни певними здобувачами та на основі якого рекомендаційна система може тільки виграти;

g. Кількість кредитів – важлива характеристика дисципліни, так як здобувачі будують свій індивідуальний план навчання на основі кількості кредитів. Може виступати обмежувальним або фільтруючим фактором у виборі дисципліни.

h. Та інші, які залежать від специфічних умов, потреб здобувачів, закладу і т.д.

4) Здобувач – сутність $S_i$, яка відповідає за користувача, який здійснює вибір дисципліни, та для якого надається та чи інша рекомендація. Нехай $S_i$ – це вектор характеристик здобувача $i$: $S_i = (S_{i1}, S_{i2}, S_{i3}, \dots, S_{ik})$, де $k$- кількість характеристик здобувача:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. ПІБ – ідентифікаційні дані здобувача;

c. Ідентифікатор принадлежності до специфічної групи здобувачів – поле або група полів, які даватимуть розуміння до якої вибірки здобувачів належить конкретна персона. Традиційно в університетах використовуються групи, які логічно зв’язані із спеціальністю, курсом, кафедрою, факультетом і т.д. Однак, із все більшим впровадженням індтвідуального плану навчання, дані прив’язки можуть бути недієздатними, а тому ми опишемо дане поле саме таким чином;

d. Ідентифікаційні дані здобувача – паспортні дані, ІПН, код із ЄДЕБО і т.д. Будь-які ідентифікаційні дані, які допоможуть ідентифікувати персону не тільки в системі, а й в реальному житті. Можуть слугувати для реєстрації в системі, пошуку споріднених даних і т.д.

e. Та інші, які залежать від специфічних умови, потреб здобувачів, закладу і т.д.

5) Вибір здобувача – сутність, яка відповідає за вибір здобувача $i$ для

дисципліни $j$ та може бути функцією, яка залежить від характеристик здобувача $S_i$ та характеристик дисципліни $C_j$. Це можна описати через функцію $P(i,j)$, яка виражає зацікавленість здобувача в дисципліні. Така функція може бути як лінійною так і нелінійною, що залежить від способу моделювання:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Лінійна модель, де кожна характеристика здобувача та дисципліни важить однаково:

$$P(i,j) = \alpha_1 \cdot S_{i1} + \alpha_2 \cdot S_{i2} + \cdots + \beta_1 \cdot S_{i1} + \beta_2 \cdot S_{i2} + \cdots \quad (2.2)$$

де $\alpha$ та $\beta$ – це ваги, які можна визначити за допомогою тренування системи.

c. Нелінійна модель, де можна використовувати складні функції (наприклад нейронні мережі), що відображають складнішу взаємодію між характеристиками здобувача та дисципліни:

$$P(i,j) = g(S_i, C_j) \quad (2.3)$$

де $g(S_i, C_j)$ – нелінійна функція.

6) Обмеження – сутність, яка позначає правила та умови вибору здобувачами дисциплін. Так здобувач може мати до вибору тільки дисципліни певного типу, певної кількості кредитів, певного часового проміжку тощо. Нехай $P(i,j)$ – це ймовірність того, що здобувач $i$ вибере дисципліну $j$, обчислена рекомендаційною системою, тоді задача вибору дисциплін для здобувача може бути сформульована як задача лінійного програмування з обмеженнями. Математична модель, якщо максимізуємо загальний інтерес здобувача до вибраних дисциплі (враховуючи рекомендації), буде виглядати наступним чином:

$$max \sum_{j \in C} P(i,j) \cdot x_{ij} \quad (2.4)$$

де $g(S_i, C_j)$ – $x_{ij} = 1$, якщо здобувач $i$, обирає дисципліну $j$, та $x_{ij} = 0$ у протилежному випадку.

Обмеження можуть бути різними, тому розглянемо основні, які спадають на думку:

a. Обмеження на кількість кредитів:

$$\sum_{j \in C} C_j \cdot x_{ij} \leq L \; \forall i \tag{2.5}$$

b. Обмеження на кількість дисциплін:

$$x_{ij} = 0, \text{якщо } P(j) \nsubseteq S_i \tag{2.6}$$

c. Часові обмеження:

$$t_i \cap \mathrm{t}_j = \emptyset, \forall \mathrm{i}, \mathrm{j} \in \mathrm{C}, \mathrm{i} \neq \mathrm{j} \tag{2.7}$$

d. Обмеження на типи дисциплін:

$$\sum_{j \in C} 1_{t_j = \text{університетська дисципліна}} \cdot x_{ij} \leq K_1 \tag{2.8}$$

$$\sum_{j \in C} 1_{t_j = \text{факультетська дисципліна}} \cdot x_{ij} \leq K_2 \tag{2.9}$$

e. Обмеження на попередні дисципліни (пререквізити):

$$x_{ij} = 0, \text{якщо } P(j) \nsubseteq S_i \tag{2.10}$$

Інші сутності також можливі і це залежать від специфічних умов, потреб здобувачів, закладу і т.д. Однак, нами пропонується розглядати саме такі сутності (рис. 2.1) для побудови базового алгоритму рекомендаційної системи.

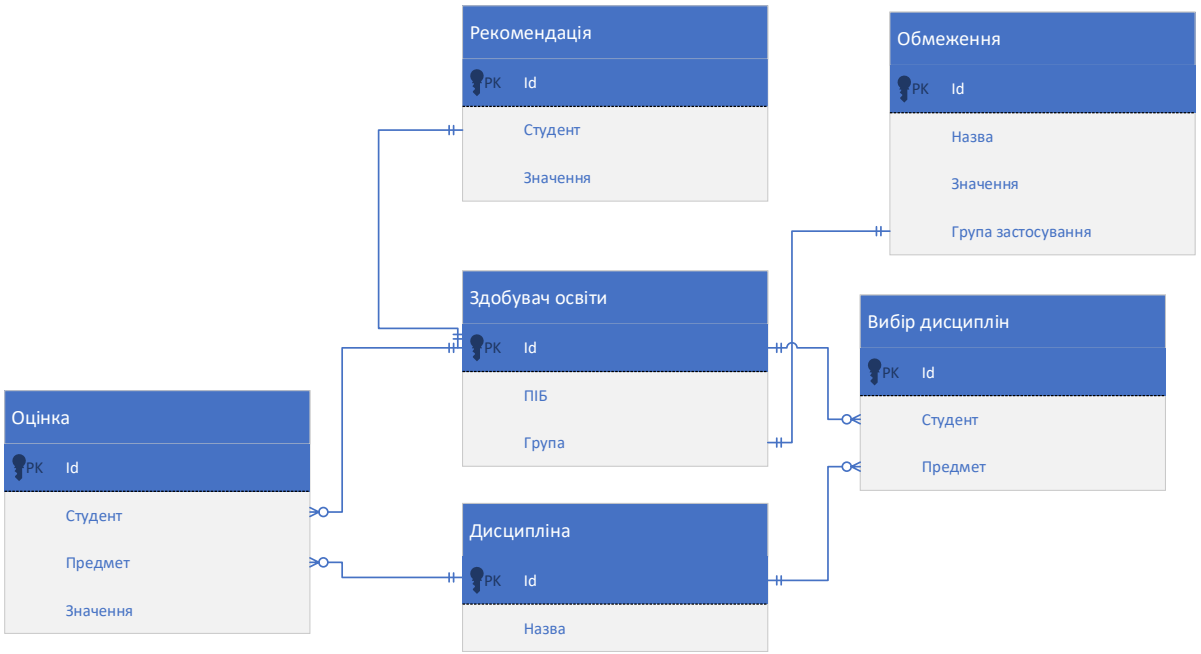


Рис. 2.1 – Логічна модель рекомендаційної системи

Для побудови рекомендаційної системи на основі сутностей, які ми описали раніше, дані беруться з існуючої інформаційної системи деканату університету. Дана система містить структуровану інформацію про дисципліни, здобувачів та їх результати навчання, а також вибір дисциплін, здійснений здобувачами в минулих семестрах. Підключення до такої системи гарантує доступ до завжди актуальних даних та їх постійне оновлення. Ці дані є основою для аналізу переваг здобувачів та побудови математичної моделі, яка дозволить давати рекомендації.

Варто також зазначити, що розроблена система містить набагато більше логічних сутностей, ніж представлено вище. В попередній частині нами було описано тільки моделі, які є базовими та фундаментальними для такого роду систем, однак, сутності та їх поля можуть поступово нашаровуватись на таку систему, задля збільшення її ефективності, покращення точності пргнозів чи варіативності калькуляцій під персональні потреби користувача.До прикладу, наша система містить також додактові сутності: модель анкетних даних, модель відгуку та модель історії вибору дисципліни та рекомендації. Розглянемо кожну з них:

1) Модель анкетних даних – спеціальна сутність, яка була введена в систему для того, щоб користувачі могли класифікувати свої інтереси в системі. Вибираючи теги, які є найближчими до його вподобань, здобувач надає системі додаткові відомості про себе та надає можливість в майбутньому надавати більш точні прогнози, зважаючи на його теперішні інтереси. Маючи такого роду інформацію, а також інформацію про кінцевий вибір здобувача, рекомендаційна система буде здатна з кожним роком краще давати рекомендацію для здобувачів, за рахунок все більшого накопичення інформації про кореляцію вподобань здобувачів та їх вибору. Поля такої сутності наступні:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Здобувач – ідентифікатор користувача, який заповнював анкетні відомості про себе;

c. Вибрані теги – список значень тегів, які обрав користувач;

d. Хобі – інформація, яка потенційно може бути корисна для покращення рекомендацій в майбутньому;

e. Теперішня робота – інформація, в залежності від якої користувачі можуть робити один і той самий вибір через спецефічність вподобань та занять.

f. Майбутня (бажана) робота – інформація про напрям, в якому далі бажає рухатися здобувач, адже теперішній рід зайнятості може кардинально відрізнятися від в майбутнього в силу різних обставин, тому наявність такої інформації для нашої рекомендаційної системи піде тільки на користь.

g. Та інші, які залежать від специфічних умов, потреб здобувачів, закладу і т.д.

2) Модель відгуку – сутність, цінність якої важко переоцінити, адже вона дає нам безпосередній зворотній зв’язок із кінцевим користувачем та містить його враження про роботу функціоналу системи. Володіючи такою інформацією ми можемо отримувати швидкий відгук про те, що варто покращити в системі, які моменти краще покращити, а які функціональність краще взагалі прибрати. В наступних розділах буде розгорнуто описано, які плюси дала ця модель для даної роботи, але поки що зупинимось на опису атрибутів даної моделі:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Здобувач – користувач, який надавав відгук в системі;

c. Корисність – параметр, який відповідає на те, чи була дана функціональність корисна для здобувача чи ні;

d. Коментар – розгорнутий відгук здобувача про функціональність системи. Може бути надзвичайно цінним для подальших покращень як рекомендаційної системи так і всієї функціональності в цілому.

e. Та інші, які залежать від специфічних умов, потреб здобувачів, закладу і т.д.

3) Модель історії вибору дисциплін та рекомендації – сутність, яка являє

собою архів записів про вибір здобувачів тих чи інших дисциплін та дані про те, які рекомендації були надані їм на той момент часу. Може бути корисною інформацією для аналізу того, наскільки вдалим було передбачення рекомендаційної системи із кінцевим результатом (отриманна оцінка здобувачем із обраної дисципліни). Містить наступні поля:

a. Ідентифікатор – поле із унікальним значенням для ідентифікації унікального рядка в системі чи базі даних;

b. Здобувач – користувач, який здійснив вибір дисциплін в системі;

c. Дисципліни – інформація про дисципліни, які були обрані здобувачем;

d. Рекомендовані дисципліни – інформація про дисципліни та значення рекомендацій до них, які були надані системою як найбільш пріоритетні для здобувача, в момент вибору дисциплін;

e. Дата – точні дата та час, коли здійснювався вибір;

f. Та інші, які залежать від специфічних умов, потреб здобувачів, закладу і т.д.

Кожна з цих моделей є частиною складної системи, яка дозволяє персоналізувати освітній процес для здобувачів, надаючи їм курси, які найбільше відповідають їхнім інтересам і попередньому досвіду, а також дозволяє адаптувати рекомендації на основі відгуків та історії. Розширена версія логічної моделі системи виглядає як на рисунку 2.2.

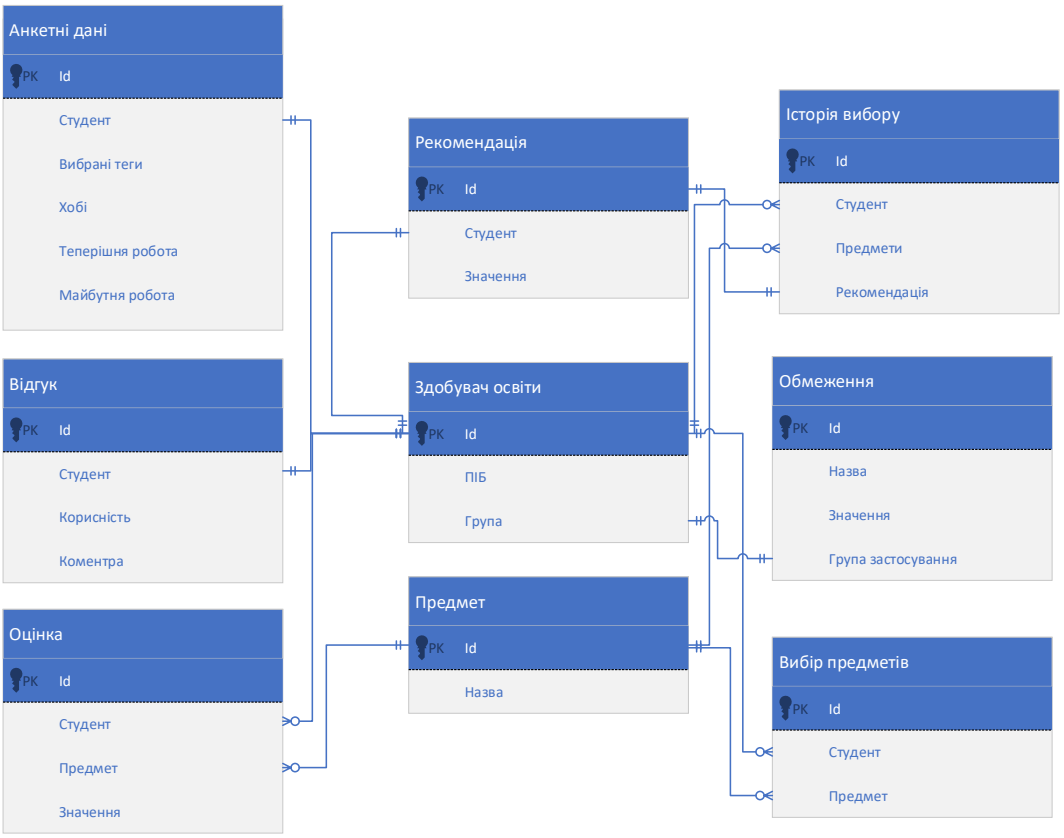


Рис. 2.2 – Розширена логічна модель системи

Важливим етапом роботи рекомендаційної системи є формування релевантної навчальної вибірки. Використання повного масиву історичних даних університету є недоцільним через проблему застарівання освітніх програм та відмінність у специфіці оцінювання між різними спеціальностями.

Для забезпечення високої точності прогнозів та забезпечення швидкодії системи вирішено проводити фільтрацію вхідних даних для рекомендаційної системи (рис. 2.3).

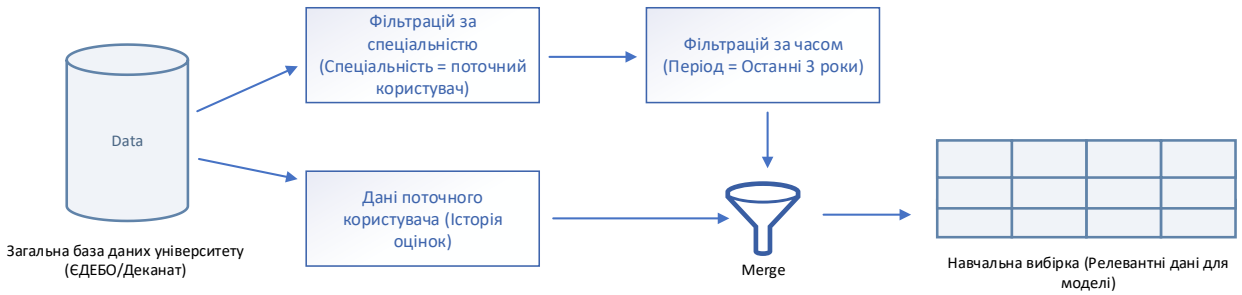


Рис. 2.3 – Схема формування навчальної вибірки для рекомендаційної моделі

Процес відбору базується на двох ключових критеріях:

1. **Контекст спеціальності**, де до вибірки включаються історії успішності лише тих здобувачів, які навчаються на тій же спеціальності, що й поточний користувач. Це забезпечує однорідний характер простору ознак.

2. **Часовий горизонт**, де враховуються дані лише за останні 3 роки. Цей період обрано емпірично як компроміс між достатнім обсягом даних для навчання моделі та актуальністю контенту дисциплін. Дані старші за 3 роки можуть містити оцінки за дисципліни, зміст яких суттєво змінився, що вносить «шум» в модель.

Сформований у такий спосіб масив даних об’єднується з поточною історією успішності активного користувача і трансформується у матрицю взаємодій, яка надалі використовується для факторизації.

## 2.2 Розробка математичної моделі збору і обробки даних для рекомендаційної системи формування індивідуального навчального плану

В ході дослідження параметрів, необхідних для надання рекомендацій щодо вибору навчальних дисциплін, було проаналізовано дані щодо освітньої діяльності здобувача освіти у навчальному закладі. В основі математичної моделі для створення індивідуальних освітніх планів здобувачів запропоновано використати методи спільної фільтрації та матричної факторизації.

Спільна фільтрація – це поширений метод рекомендацій, на основі якого дається прогноз ґрунтуючись на вподобаннях інших користувачів із подібними інтересами. Такий прогноз може бути на основі користувачів або об’єктів (рекомендації на основі схожих об’єктів, які оцінювали інші користувачі) [117].

Матрична факторизація – це спеціальна техніка, яка полягає в розкладі великої матриці оцінок на дві менші, із яких одна представляє собою користувачів, а інша – об’єкти, із якими взаємодіють користувачі. Такі матриці допомагають виявити приховані зв'язки між користувачами та об’єктами, що в свою чергу дозволяє передбачити оцінки і робити точніші рекомендації [118].

Головна мета цієї моделі – дати прогноз оцінки здобувачів для дисциплін,

які доступні йому на вибір, опираючись на попередні оцінки здобувача та оцінки інших здобувачів тієї ж спеціальності (рис. 2.4). В основі моделі лежить матриця оцінок R, де рядки представляють здобувачів, а стовпці – дисципліни. Кожен елемент цієї матриці $R_{ij}$ – це оцінка здобувача $i$ за дисципліну $j$.

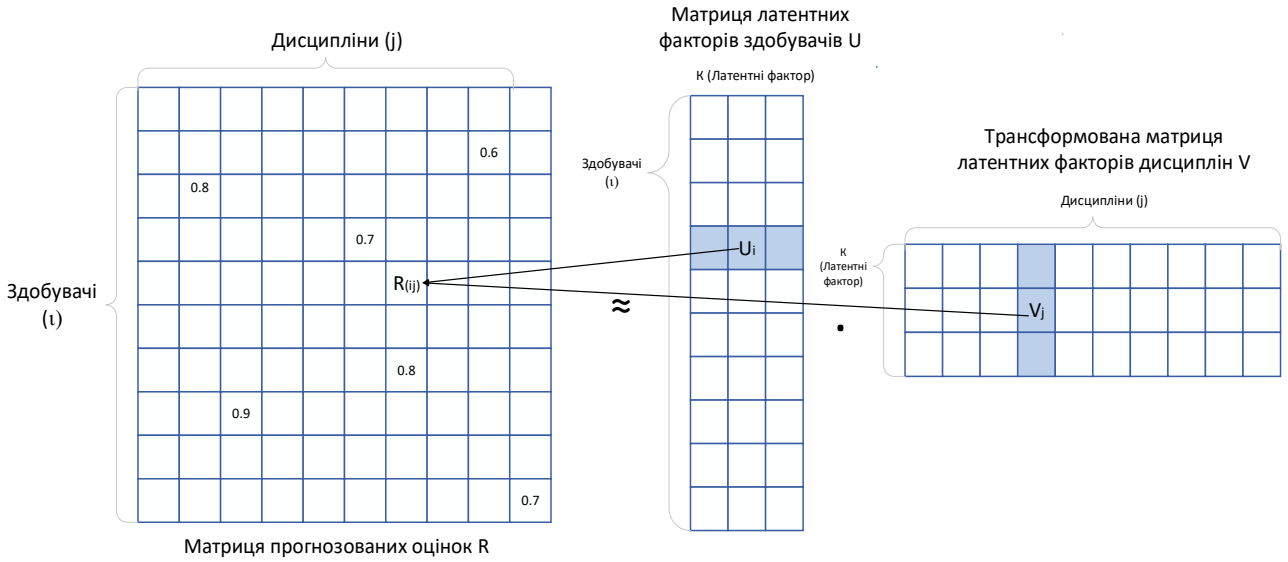


Рис. 2.4 – Схема матричної факторизації для прогнозування оцінок

Математично задача зводиться до апроксимації матриці [119] $R$ за допомогою двох матриць латентних факторів: одна матриця для здобувачів $U$, а інша для дисципліни $V$. Кількість латентних факторів $K$ визначається за допомогою гіперпараметрів моделі. Прогнозування оцінок здійснюється за допомогою скалярного добутку відповідних рядків цих матриць, що в свою чергу дозволяє передбачити оцінки здобувачів для дисциплін:

$$\widehat{R_{\iota J}} = U_i \cdot V_j^T = \sum_{k=1}^{K} U_{jk} \cdot V_{jk} \qquad (2.11)$$

де:

- $U_j$ – вектор латентних факторів здобувача $i$;
- $V_j$ – вектор латентних факторів дисципліни $j$;
- $\widehat{R_{\iota J}}$ – прогнозована оцінка здобувача $i$ за дисципліну $j$.

Також ми будемо використовувати функцію втрат, щоб мінімізувати різницю між реальними оцінками та прогнозованими. Функція втрат – це математична функція, яка вимірює різницю між прогнозованим результатом моделі та фактичними даними. Вона використовується для оцінки точності моделі та мінімізації помилок під час навчання, зменшуючи втрати при оптимізації [120].

Вибір функції втрат для навчання моделі здійснено на основі аналізу розподілу вхідних даних та специфіки задачі прогнозування успішності. Оскільки цільова змінна (рейтинг/оцінка дисципліни) є неперервною величиною в діапазоні [0,1], задача відноситься до класу задач регресії. Серед можливих функцій втрат було обрано середньоквадратичну помилку (MSE) [121], з огляду на наступні фактори:

1. В освітніх рекомендаційних системах «вартість» грубої помилки є високою. Якщо система спрогнозує високий бал, а студент отримає низький, це призведе до хибного вибору та зниження успішності. MSE підносить помилку до квадрату, що змушує алгоритм пріоритетно мінімізувати саме такі великі розбіжності, на відміну від середньої абсолютної помилки (MAE), яка лінійна до викидів. [122]

2. Аналіз вибірки оцінок показує, що відхилення успішності студентів часто підпорядковуються нормальному закону розподілу. З точки зору методу максимальної правдоподібності, саме мінімізація MSE є оптимальною стратегією для даних із шумом. [123]

3. Функція MSE є диференційованою у всій області визначення, що забезпечує стабільну збіжність градієнтних методів оптимізації, які використовуються в реалізованій нейронній мережі.

Наступним чином виглядає наша функція (MSE), яка дозволяє виміряти точність передбачення:

$$MSE = \frac{\sum_{(i,j)\in R_{train}} \left(R_{ij} - \widehat{R_{ij}}\right)^2}{|R_{train}|} \qquad (2.12)$$

де:

- $|R_{train}|$ – кількість відомих елементів у тренувальному наборі даних;
- $R_{ij}$ – реальна оцінка здобувача $i$ за дисципліну $j$.

Також розглянемо розрахунок середньої квадратичної помилки разом із нормою Фробеніуса [124], яка використовується (приміром в праці [125], чи в роботі [126] тощо) для оцінки того, наскільки великі або малі є елементи в матриці, що важливо для регуляризації в методах, таких як матрична факторизація. Моделі, які використовують матричну факторизацію мають велику кількість елементів у матрицях $U$ і $V$ і, якщо не обмежувати ці параметри, то це може призвести до перенавчання моделі. Норма Фробеніуса допомагає підтримувати баланс між складністю і точністю прогнозів, що дозволяє отримати стійкіші моделі, які краще працюють на нових або ж невідомих даних.

Тож ми можемо додати до функції втрат своєрідний штраф за величину елементів в матрицях латентних факторів $U$ і $V$ через норму Фробеніуса. Дані зміни призведуть до того, що модель буде намагатися зробити матриці $U$ і $V$ як можна менш великими, тим самим сприяючи більш простим і стійким до перенавчанням моделям. Така формула буде виглядати наступним чином:

$$MSE = \frac{\sum_{(i,j)\in R_{train}} \left(R_{ij} - \widehat{R_{ij}}\right)^2}{|R_{train}|} + \lambda(\left|\left|U\right|\right|_F^2 + \left|\left|V\right|\right|_F^2) \qquad (2.16)$$

де:

- $|R_{train}|$ – кількість відомих елементів у тренувальному наборі даних;
- $\left|\left|U\right|\right|_F^2 + \left|\left|V\right|\right|_F^2$– це норма Фробеніуса для матриць $U$ та $V$, яку використовуються для запобіганню перенавчанню;
- $\lambda$ – параметр регуляризації, який контролює вплив регуляризації на мінімізацію помилки, тобто наскільки сильно ми хочемо обмежити величину елементів матриць $U$ та $V$. Більші значення значно зменшують величину

латентних факторів, а менші – напротивагу їм, дають більше свободи для цих факторів.

Процес оптимізації полягає в оновленні матриць латентних факторів за допомогою градієнтного спуску, що дозволить поступово зменшувати функцію втрат, поки модель не досягне мінімуму помилки [127]. Інші поширені в рекомендаційних системах методи, такі як метод змінних найменших квадратів (ALS) або сингулярний розклад (SVD), ефективні для лінійних моделей матричної факторизації, проте не дозволяють проводити оптимізацію ваг у багатошарових перцептронах (MLP) через неможливість прямого аналітичного розв'язку при наявності нелінійностей.

Оновлення матриць здійснюється за допомогою похідних функції втрат по кожному з елементів матриць. Ці похідні використовуються для оновлення матриць $U$ та $V$ у процесі стохастичного градієнтного спуску, поки функція втрат не досягне мінімуму та виглядають наступним чином [128]:

$$U_i - t\frac{\partial L}{\partial U_i} \rightarrow U_i - \text{для користувача,} \tag{2.17}$$

$$V_i - t\frac{\partial L}{\partial V_i} \rightarrow V_i - \text{для дисципліни} \tag{2.18}$$

де:

- t – це швидкість навчання;
- $\frac{\partial L}{\partial U_i}$ – похідна по $U_i$.

Розглянемо детальніше, як формуються остання характеристика. Для зручності обчислення конвертуємо $\widehat{R_{ij}} = U_i \cdot V_j^T$ згідно першої формули. Тоді:

- L – функція втрат MSE, яка є середньою квадратичною помилкою:

$$L = \frac{\sum_{(i,j)\in R_{train}} \left(R_{ij} - U_i \cdot V_j^T\right)^2}{|R_{train}|} + \lambda\left(||U||_F^2 + ||V||_F^2\right) \tag{2.19}$$

- $\frac{\partial}{\partial U_i}$ – це загальна похідна по $U_i$, яка складається з:
  - похідної від середньої квадратичної помилки:

$$\frac{\partial}{\partial U_i}\left(\frac{\sum_{(i,j)\in R_{train}}\left(R_{ij}-U_i\cdot V_j^T\right)^2}{|R_{train}|}\right)=2\left(R_{ij}-U_i\cdot V_j^T\right)\cdot\frac{\partial}{\partial U_i}\left(R_{ij}-U_i\cdot V_j^T\right)$$

$$=-2\frac{\sum_{(i,j)\in R_{train}}\left(R_{ij}-U_i\cdot V_j^T\right)(-V_j)}{|R_{train}|} \quad (2.20)$$

- та похідної від регулярізації:

$$\frac{\partial}{\partial U_i}\left(\lambda\left|\left|U\right|\right|_F^2\right)=2\lambda \mathrm{U_i} \quad (2.21)$$

Таким чином отримаємо, що:

$$\frac{\partial L}{\partial U_i}=-2\frac{\sum_{(i,j)\in R_{train}}\left(R_{ij}-U_i\cdot V_j^T\right)(-V_j)}{|R_{train}|}+2\lambda \mathrm{U_i} \quad (2.22)$$

Відповідно похідна для дисципліни виглядатиме:

$$\frac{\partial L}{\partial V_j}=-2\frac{\sum_{(i,j)\in R_{train}}\left(R_{ij}-U_i\cdot V_j^T\right)(-U_i)}{|R_{train}|}+2\lambda \mathrm{V_j} \quad (2.23)$$

Для оцінки ефективності моделі ми використовуємо корінь середньоквадратичної помилки (RMSE) [129], яка дає змогу наочно оцінити, наскільки точно модель відновлює відсутні оцінки. Формула для RMSE виглядає так:

$$RMSE=\sqrt{\frac{\sum_{(i,j)\in R_{train}}\left(R_{ij}-\widehat{R_{ij}}\right)^2}{|R_{train}|}} \quad (2.13)$$

У поточній реалізації ми використовуємо ML.NET на основі мови програмування C#, що дозволяє автоматизувати процес факторизації матриць і мінімізації функції втрат.

Використання вже готових інструментів значно спрощує розробку моделі та її налаштування. Основні етапи цього процесу включають тренування моделі, визначення параметрів тренера (кількість латентних факторів або ж кількість ітерацій), а також оптимізацію моделі через регуляризацію та стохастичний градієнтний спуск. Наприклад, код, який автоматично визначає матричну факторизацію зображено на рисунку 2.5.

```
var metrics = mlContext.Regression.Evaluate(predictions, labelColumnName: "Label", scoreColumnName: "Score");
Console.WriteLine($"Root Mean Squared Error: {metrics.RootMeanSquaredError}");
```

Рис. 2.5 – Калькуляція матричної фактроизації за допомогою інструментів ML.NET

На основі розроблених математичних моделей та методів обробки даних побудовано узагальнений алгоритм функціонування рекомендаційної системи. Логіка роботи системи представлена у вигляді блок-схеми на рис. 2.6.

Процес формування рекомендацій складається з наступних етапів:

1. На вхід подається ідентифікатор активного студента, для якого з репозиторію завантажуються оцінки інших студентів. Відбираються записи лише за останні 3 роки та лише для тієї ж спеціальності, що й у користувача, для забезпечення релевантності навчальної множини. Якщо ж даних немає, алгоритм завершує роботу.

2. Якщо дані для роботи рекомендаційної системи знайдено, то система продовжує роботу та завантажує дані поточного здобувача про його історію оцінок з бази даних та дисципліни, які він ще не проходив.

3. Поділ вибірки на тренувальну та тестову множини та побудова конвеєра обробки, де ідентифікатори студентів та дисциплін перетворюються на числові ключі для утворення матриці суміжності, та навчання моделі.

4. Генерування прогнозів, де нормовані вектори подаються на вхід навченої нейронної мережі, а на виході формується вектор, що містить прогнозовані рейтинги для всіх доступних дисциплін.

5. Ранжування та видача результату, коли список дисциплін сортується за спаданням прогнозованого рейтингу. Топ-N дисциплін формують індивідуальну пропозицію для кінцевого користувача.

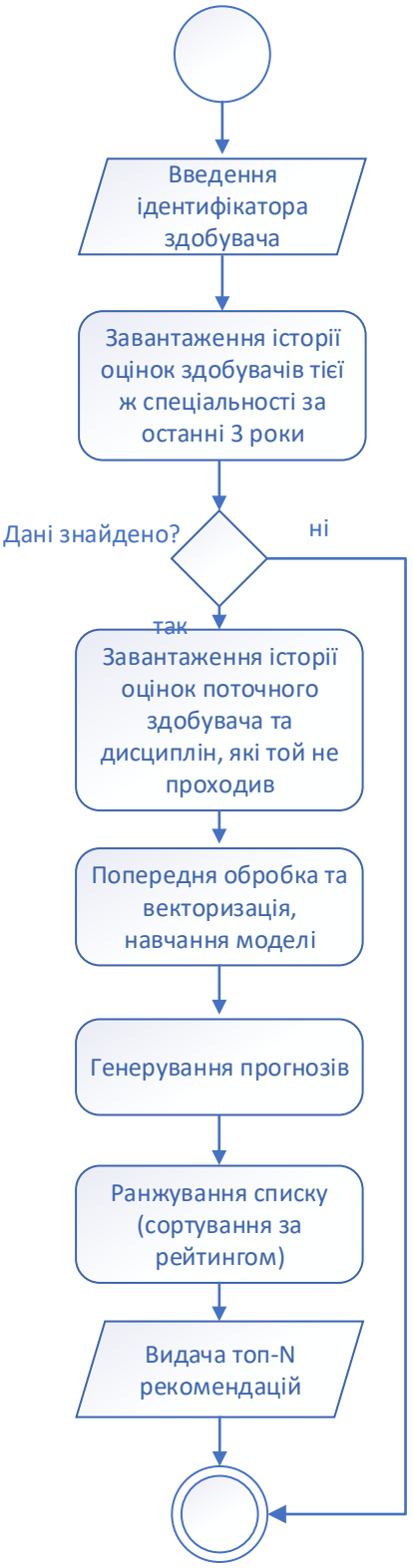


Рис. 2.6 – Блок-схема узагальненого алгоритму формування рекомендацій

Запропонований алгоритм інтегрує етапи відбору релевантних даних, навчання нейронної мережі та ранжування прогнозів у єдиний конвеєр обробки інформації. Це дозволяє автоматизувати процес вибору дисциплін, надаючи студенту науково обґрунтований перелік предметів, що найкраще відповідають

його академічному профілю.

## 2.3 Перспективи розвитку математичної моделі збору і обробки даних для рекомендаційної системи формування індивідуального навчального плану

У результаті проведеного дослідження було розроблено математичну модель рекомендаційної системи для вибору дисциплін для здобувачів на основі їхніх попередніх результатів навчання та оцінок здобувачів тієї ж спеціальності за останні три роки. Запропонована модель ґрунтується на методах матричної факторизації та мінімізації функції втрат. Вона продемонструвала свою ефективність у прогнозуванні оцінок для дисциплін, що здобувачеві пропонуються на вибір.

Продемонстровано як за допомогою використання інструментів ML.NET, процес факторизації матриць і оптимізації моделі був автоматизований, що значно спростило реалізацію та налаштування системи.

Показано як застосування середньоквадратичної помилки для оцінки точності прогнозів дозволило досягти оптимізації результатів і зменшення різниці між реальними та передбаченими оцінками.

Розроблена рекомендаційна система має великий потенціал для використання в освітніх установах, де вона може сприяти формуванню індивідуальних освітніх програм, підвищенню ефективності навчання та наданню здобувачам більш точних та адаптованих під конкретного здобувача рекомендацій. Застосування подібних систем відкриває нові можливості для розвитку сучасного освітнього процесу, забезпечуючи персоналізований підхід до навчання і підвищення якості освіти.

Проте, варто зазначити, що є кілька напрямків для подальшого вдосконалення моделі. Наприклад, можна інтегрувати контентну фільтрацію – метод, який здатен робити пропозиції для користувачів, на основі їх характеристик та вподабань [130]. В нашій ситуації, такий тип алгоритму враховував би наприклад специфікації дисциплін, такі як тип, складність,

викладач і т.д., що дозволить поліпшити точність рекомендацій, зменшуючи залежність від фактору оцінок здобувачів. Це можна реалізувати через комбінування латентних факторів $V_j$ з контентними характеристиками дисциплін $C_j$:

$$\widehat{R_{ij}} = (V_i) \cdot \left(C_j + W_j\right)^T \quad (2.14)$$

де:

- $\widehat{R_{ij}}$ – прогнозована оцінка для здобувача;
- $V_i$ – латентний вектор здобувача $i$;
- $C_j$ – контентні характеристика дисципліни $j$;
- $W_j$ – додаткові ваги або латентні фактори, що представляють специфічні характеристики дисципліни $j$;

Ще одним покращенням може бути обчислення схожості між дисциплінами або здобувачами. Наприклад, дану характеристику можна виразити через косинусну подібність, що дасть змогу обчислювати схожість між дисциплінами через скалярний добуток латентних векторів дисциплін [131]:

$$sim(i, i') = \frac{V_i \cdot V_{i'}}{\left||V_i|\right| \left||V_{i'}|\right|} \quad (2.15)$$

де:

- $V_i$ та $V_{i'}$ – латентні вектори дисциплін;
- $V_i \cdot V_{i'}$ – скалярний добуток між цими векторами;
- $\left||V_i|\right| \left||V_{i'}|\right|$ – норми векторів $V_i$ та $V_{i'}$.

Це дозволить на основі цієї схожості генерувати більш персоналізовані рекомендації. Крім того, можна комбінувати матричну факторизацію з іншими методами, такими як глибоке навчання або додаткові моделі, що будуть використовувати інформацію про навчання здобувача.

Також варто розглянути моделі, які є в розширеній версії математичної моделі системи (модель анкетних даних, модель відгуку та модель історії

вибору дисциплін та рекомендацій, адже їх використання в рекомендаційній системі піде тільки на користь прогнозуванні та точності наданої оцінки.

1) **Математична модель анкетних даних здобувача** $s \in S$ може бути подана як набір атрибутів:

$$Survey(s) = \left(I(s), H(s), W_{currentJob}(s), W'_{futureJob}(s)\right) \quad (2.24)$$

де:

**a) Вибрані теги:**

Нехай $T$ – множина всіх тегів, де кожен тег описує певний аспект інтересів здобувача або дисципліни, приміром:

$$T = \{t_1, t_2, t_3, \dots, t_n\} \quad (2.25)$$

де $t_i \in T$ – це окремий тег («макроменеджмент», «рослинництво», «php» тощо).

Також, нехай $S$ – множина всіх здобувачів, де кожен здобувач $s \in S$ має набір тегів, що відповідають його певним інтересів.

Кожен здобувач може обрати кілька тегів, що відображають його інтереси. Це можна описати через відношення $R_{tags}$ між здобувачами та тегами:

$$R_{tags} \subseteq S \times T \quad (2.26)$$

де $R_{tags}(s,t) = 1$, якщо здобувачкем $s$ було обрано тег $t$, й $R_{tags}(s,t) = 0$, якщо не було обрано.

Тоді інтереси здобувача можуть бути визначені через певний набір тегів:

$$I(s) = \{t \in T | R_{tags}(s,t) = 1\} \quad (2.27)$$

де $I(s)$ – множина тегів, які були обрані здобувачем $s$, та які характеризують його вподобання.

До слова, теги також можуть бути пов’язані із дисциплінами, тому розглянемо й це із математичної точки зору. Нехай $C$ – множина курсів, тоді відношення, яке показує, яка дисципліна має певний тег буде:

$R_{tagsOfDisciplina}(c,t) = 1$ , якщо дисципліна $c$ характеризується тегом $t$ (2.28)

$R_{tagsOfDisciplina}(c,t) = 0$, якщо тег $t$ відсутній в дисципліні $c$ (2.29)

**b) Хобі здобувача:**

Нехай $H$ – це множина можливих хобі, де кожне хобі $h_i \in H$ – це певне заняття чи інтерес здобувача, яке не є академічною дисципліною. Тоді для кожного здобувача можна вивести відношення:

$$R_{hobbies}(s,h) \subseteq S \times H \tag{2.30}$$

де $R_{hobbies}(s,h) = 1$, якщо здобувач $s$ має хобі $h$.

**c) Теперішня робота:**

Нехай W – є множиною можливих типів робіт, а $R_{currentJob} \subseteq S \times W$– це відношення, що показує, який тип роботи має здобувач. Тоді теперішня робота здобувачів може бути описана як:

$$R_{currentJob}(s,w) = 1, \text{якщо здобувач } s \text{ працює на роботі } w \tag{2.31}$$

де $w \in W$.

**d) Майбутня робота** – схожа за описом до теперішньої роботи:

Визначимо множину можливих майбутніх робіт $W'$, де $W' \subseteq W$. Тоді для кожного здобувача можна вивести відношення:

$$R_{futureJob} \subseteq S \times W', \tag{2.32}$$

яке вказує на майбутні роботи:

$$R_{futureJob}(s,w') \subseteq 1, \text{якщо здобувач має бажання в майбутньому працювати в } w' \tag{2.32}$$

де $w' \in W'$.

Для кожного здобувача, на основі його анкетних даних, можна створити свого роду рекомендаційну функцію. Це може бути, приміром, оцінка ймовірності того, що здобувач буде зацікавлений в певних дисциплінах. Так, функція корисності для поєднання тегів, хобі та роботи буде виглядати наступним чином:

$$U(s,r) = w_1 \cdot sim(I(s),r) + w_2 \cdot sim(H(s),r) + w_3 \cdot sim\left(W_{currentJob}(s),r\right) + w_4 \cdot \left(W'_{futureJob}(s),r\right) \tag{2.33}$$

де:

- sim(X,Y) – функція подібності між множинами X й Y;
- $w_1, \dots, w_4$ – ваги кожного з аспектів, які описують інтереси здобувача.

2) **Математична модель відгуку** може бути описана через відношення $R_{rating}$, яке містить запис про відгуки здобувачів до конкретних рекомендацій:

$$R_{rating} \subseteq S \times C \times R \times \{0,1\} \times Date \quad (2.34)$$

де:

- $s \in S$ – це точ чи інший здобувач, який вибирає дисципліни та отримує рекомендації до них;
- $c \in C$ – це дисципліни, які вибирає здобувач;
- $r \in R$ – це рекомендації, які здобувач отримує;
- $\{0,1\}$ — бінарний відгук (0 або 1);
- $Date$ – це дата та час вибору дисциплін здобувачем.

Таким чином, для кожної події відгуку буде створено запис, який містить інформацію про здобувача, відгук та дату, коли цей відгук був залишений. Дані про рекомендовані для нього дисципліни та дисципліни, які він у висновку обрав будуть зберігатися в іншій моделі, але їх можна буде з легкістю співставити між собою, щоб зрозуміти причину конкретного відгуку здобувача.

Тоді математичний опис всієї структури буде наступний:

$$R_{rating} = \{(s, c, r, rating, date) \mid s \in S, c \in C, r \in R, rating \in \{0,1\}, date \in D\} \quad (2.35)$$

Також, якщо ми захочемо вдосконалити й таку модель, то ми можемо замінити оцінення рекомендаційної системи із бінарного значення, на більш просунітішу величину. Так наша система може отримувати оцінку, приміром, по 5-бальній шкалі, тоді в нас ми можемо побудувати функцію $ratingEffectiveness: F => Rating$, що визначає рейтинг ефективності для кожного відкугу:

$$ratingEffectiveness(f) \in \{1,2,3,4,5\} \quad (2.36)$$

де $f \in F$ – це конкретний відгук.

Тоді:

$$R_{rating} \subseteq S \times C \times R \times \{1,2,3,4,5\} \times Date \quad (2.37)$$

де:

- $\{1,2,3,4,5\}$ – це числова оцінка відгуку, де 1 – це найнижче значення, а 5 – відповідно найвище.

Звідси отримаємо математичний опис всієї структури:

$$R_{rating} = \{(s, c, r, rating, date) \mid s \in S, c \in C, r \in R, \\ rating \in \{1, 2,3,4,5\}, date \in D\} \quad (2.38)$$

3) **Математична модель історії вибору дисциплін та рекомендацій** може бути описана через відношенн $H_{recommendationHistory}$, що містить запис про подію вибору дисципліни та рекомендації, яка була надана для момента в той момент часу:

$$H_{recommendationHistory} \subseteq S \times C \times R \times Date \quad (2.39)$$

де:

- $s \in S$ – це точ чи інший здобувач, який вибирає дисципліни та отримує рекомендації до них;
- $c \in C$ – це дисципліни, які вибирає здобувач;
- $r \in R$ – це рекомендації, які здобувач отримує;
- $Date$ – це дата та час вибору дисциплін здобувачем.

При цьому як кожен здобувач $s \in S$ може вибрати дисципліну $c \in C$ в певний момент часу, що буде зберігатися в історії:

$$H_{selection}(s, c) = (s, c, date) \quad (2.40)$$

Так кожеш здобувач може отримати певну рекомендацію при цьому виборі:

$$H_{recommendation}(s, c) = (s, r, date) \quad (2.41)$$

Тоді загальний запис для історії вибору та рекомендацій може бути представлений таким чином:

$$H_{recommendationHistory}(s) = \{(s, c, r, date) \mid \text{для кожної події} \\ \text{в історії вибору та рекомендацій здобувача } s\} \quad (2.42)$$

Врахування великої кількості категоріальних ознак потенційно створює проблему високої розмірності вхідного шару, де кількість параметрів може перевищувати обсяг навчальної вибірки. Для вирішення цієї проблеми в архітектуру нейронної мережі потрібно інтегровати шар вбудовування. Цей шар

має трансформувати вхідні розріджені вектори високої розмірності у щільні вектори фіксованої низької розмірності. Вагові коефіцієнти шару вбудовування будуть навчатися разом з основною мережею, що дозволить системі автоматично формувати семантично близькі представлення для схожих хобі чи професій, суттєво зменшуючи кількість параметрів моделі та запобігаючи перенавчанню.

**Висновки до другого розділу**

У розділі вирішено задачу математичного моделювання та алгоритмічного забезпечення системи.

Основні наукові та практичні результати розділу полягають у наступному:

1. Формалізовано теоретико-множинний опис предметної області у вигляді кортежів сутностей «Здобувач», «Дисципліна», «Оцінка» та розширених структур «Анкетні дані», «Відгук» та «Історія вибору», що дозволило структурувати вхідні дані для рекомендаційної системи та вирішити проблему різнорідності джерел інформації в освітньому середовищі.

2. На основі здійсненого аналізу методів машинного навчання обґрунтовано вибір методу матричної факторизації як базового підходу для прогнозування успішності здобувача. Доведено, що для розріджених матриць оцінок (що є характерним для вибіркових дисциплін) найбільш ефективним є використання розкладу матриці на латентні фактори користувачів та дисциплін з мінімізацією функції втрат.

3. Розроблено алгоритм фільтрації та прогнозування оцінок, зокрема, запропоновано новий підхід до прогнозування, який, на відміну від класичних методів, включає регуляризацію (з використанням норми Фробеніуса) для запобігання перенавчанню моделі на обмежених вибірках даних. Оптимізацію реалізовано через метод стохастичного градієнтного спуску, що забезпечує можливість виявляти потенційні труднощі у навчанні на ранніх етапах, що відповідає заявленій науковій новизні.

4. Формалізовано задачу побудови індивідуального навчального плану як задачу лінійного програмування. Розроблено математичну модель, яка враховує попередні дисципліни, а також обмеження на кількість кредитів, часові обмеження, обмеження на кількість та типи дисциплін, що дозволяє автоматизувати процес перевірки валідності індивідуального навчального плану ще на етапі генерації рекомендацій.

5. Запропоновано підхід до персоналізації через гібридизацію даних. Адаптовано методи машинного навчання шляхом введення функції корисності, яка комбінує колаборативну фільтрацію (оцінки) з контентним аналізом (теги, хобі, кар’єрні цілі), що дозволяє підвищити точність рекомендацій в умовах «холодного старту» та адаптувати вибір дисциплін під позанавчальні інтереси здобувача.

# РОЗДІЛ 3
# МЕТОД ПОШУКУ ТА ФІЛЬТРАЦІЇ ДАНИХ РЕКОМЕНДАЦІЙНОЮ СИСТЕМОЮ ДЛЯ ФОРМУВАННЯ ІНДИВІДАЛЬНОГО НАВЧАЛЬНОГО ПЛАНУ

## 3.1 Аналіз алгоритмів машинного навчання для фільтрації даних

Методи машинного навчання передбачають попередню фільтрацію та структурування даних, що є необхідною умовою для коректного функціонування моделей. Однак, водночас застосування цих технологій супроводжується низкою обмежень: зокрема, високими вимогами до обсягу, якості та репрезентативності даних, а також потенційними ризиками виникнення похибок через неповноту або ж упередженість вхідної інформації.

Зазначають основні проблеми машинного навчання [132]:

1) Великі вимоги до даних – для тренування ефективної моделі потребується багато навчальних прикладів, а отримання даних для таких прикладів може бути складним завданням.

2) Адаптація до змін середовища – модель може погіршити свою ефективність, після її навчання, якщо середовище змінюється.

3) Чорні ящики – чимало моделей машинного навчання є «чорними ящиками» і важко буває зрозуміти, чому саме модель навчилася. Це може виступати проблемою в реальних задачах, де важлива надійність рішень і потреба в людському контролі.

Схожі проблеми у своїй галузі також наводить автор статті [133], де також зазначає що:

1) Використання ML вимагає чималих обсягів даних, а це може бути проблематичним через питання конфіденційності та доступу даних.

2) Моделі машинного навчання можуть бути «чорними ящиками», що ускладнює розуміння того, як вони ухвалюють рішення, і викликає сумніви в їх надійності.

3) Адаптація моделей до різних змін у клінічних умовах може бути складною.

В роботі [134] автор підкреслює проблеми тим, що штучний інтелект в цілому є важкий в плані впровадження в будь-яку структуру, так як досі спостерігається проблема відсутності достатньої компетенції в робітників та користувачів в даній області. А також акцентується на тому, що чергова складність полягає в тому, що для інтеграції таких технологій потрібне потужне технічне обладнання, що дуже часто зустрічається в наш час.

А в статті [135] автор знову піднімає питання ряду проблем, так як проблеми із інтерпретацією, необхідність великих даних та етичні правові питання у використанні приватних даних та їх безпосередня якість. Однак також в роботі наводиться ряд переваг використання машинного навчання:

1) Покращена точність та ефективність – машинне навчання дозволяє створювати прогностичні моделі, які можуть точно прогнозувати медичні результати, що очевидно покращує діагностику і лікування пацієнтів.

2) Аналіз великих даних – моделі машинного навчання мають здатність обробляти та аналізувати величезні обсяги медичних даних, таких як електронні медичні картки, що в свою чергу дозволяє знаходити нові тенденції в здоров'ї пацієнтів.

3) Автоматизація процесів – машинне навчання може автоматизувати чимало медичних процесів (наприклад, обробку результатів тестів, знімків тощо), що може значно скоротити час для прийняття рішень і знизити людські помилки.

4) Персоналізація лікування – алгоритми машинного навчання можуть допомогти розробити персоналізовані підходи до лікування на основі індивідуальних даних пацієнтів.

Таким чином, на основі вищеописаних факторів, можемо узагальнити всі плюси та мінуси машинного навчання:

До мінусів ми можемо віднести:

1) Проблема «чорної коробки» – проблема, яка виникає, коли користувач працює із достатньо складною моделлю та не розуміє, чому вона прийняла той чи інший результат.

2) Необхідність експертизи – із технологіями такого рівня часто має працювати професіонал для надання якісного результату роботи системи.

3) Складність в створенні – розробка ефективної моделі, яка буде спиратися на правильних аргументах в специфічній сфері та оброблювати такі дані за допомогою правильних алгоритмів є досить важкою задачею, яка потребує чимало знань та досвіду розробника.

4) Потужні обчислювальні засоби – проблема, яка спричинена тим, що система має обробити велику кількість даних, що потребує значних ресурсів.

5) Великі даних – важлива вимога в системах такого роду, так як чим більше даних вони мають, тим ефективніше та точніше вони можуть надати прогноз.

6) Вразливість до шаблонів – так як такі моделі не усвідомлюють інформацію, яку вони аналізують та оброблюють, вони з легкістю можуть дати неправильний результат, якщо в даних є неправомірні дані або дані, які можуть нашкодити.

До плюсів ми можемо віднести:

1) Автоматизація процесів – рутинні, однотипні і не тільки завдання можуть бути автоматизованими, що дозволяє економніше витрачати людські ресурси, витрачати час на більш складні задачі та приймати швидше рішення.

2) Обробка великих масивів даних – машинне навчання є ефективним засобом для опрацювання значного об’єму інформації на що в людини пішли б години, дні, а може й місяці.

3) Адаптивність – окрім того, що процеси автоматизуються, моделі машинного навчання також можуть самостійно працювати та покращувати свою роботу без людського втручання.

4) Точність – за допомогою таких моделей можна досягти надзвичайної точності та мінімізувати помилки при калькуляції розрахунків.

5) Відсутність людського фактору – незалежна робота таких систем позбавляє сенсу втручання людини в процеси роботи алгоритму системи.

**Класифікація типів навчання в машинному навчанні** є ключовим елементом теоретичного підґрунтя цієї галузі, так як визначає підходи до формування моделей та способи взаємодії алгоритмів із даними. Різні типи навчання передбачають різні механізми опрацювання інформації, ступінь участі людини в процесі навчання моделі та характер отримуваних результатів. Розуміння цих відмінностей дає можливість обґрунтовано обирати методи, щоб розв'язати певні завдань та забезпечити ефективність побудови інтелектуальних систем.

На основі статті автора [136] розглянемо далі основні типи навчання:

**Навчання з учителем (Supervised Learning)** – підхід, де модель навчається на мічених даних для прогнозування або класифікації. Найвідоміші алгоритми даного типу:

1) Дерева рішень (Decision Trees) – це є один із перших та найпоширеніших алгоритмів машинного навчання, який використовується для задач класифікації та регресії. Шикоро застосовується в астрономії, фінансовому аналізі, системах контролю, прогнозуванні захворювань та інших сферах. В такому алгоритмі дані діляться на підмножини в залежності від вхідних ознак, поки не досягнуть листового вузла, який містить клас або кінцевий результат. Його переваги: здатність працювати з категоріальними та числовими даними, легкість у розумінні, мінімальна обробка даних і швидкість. Його недоліки: нестабільність та схильність до локальних оптимумів замість глобальних [137].

2) Наївний байєс (Naive bayes) – ефективний алгоритм машинного навчання, який використовує теорему Байєса для класифікації даних та основна ідея якого полягає в припущенні, що всі характеристики є незалежними, що спрощує розрахунки та дозволяє швидко будувати моделі навіть при великих обсягах даних. Є ефективним для задач класифікації, таких як фільтрація спаму, аналіз настроїв у текстах і медична діагностика. Хоча він і вважається

доволіпростим, проте він часто дає хороші результати, особливо при великих наборах даних з категоріальними ознаками [138].

3) Підтримка векторних машин (Support Vector Machine) – алгоритм, який працює шляхом знаходження гіперплощини, яка максимально розділяє різні класи в даних з максимальним відступом між точками різних класів. Він використовується в таких сферах, як розпізнавання образів, фінансовий аналіз та біоінформатика. Алгоритм добре справляється з великою вибіркою даних та може працювати в нелінійних умовах за допомогою ядрових функцій, а також фективний у задачах, де важливо знайти оптимальні межі для класифікації [139].

4) Лінійна регресія (Linear Regression) – алгоритм машинного навчання, що використовується для прогнозування значень на основі лінійної залежності між змінними. Він шукає найкращу пряму лінію, яка мінімізує відстань між прогнозованими та реальними значеннями. Даний алгоритм може застосовуватися для передбачення числових значень. Подібно до баєсівського алгоритму він також вважається доволі простим, однак є ефективним лише, коли залежність між змінними є лінійною [140].

5) Логічна регресія (Logistic Regression) – це алгоритм, що використовується для вирішення задач класифікації, головна мета яких передбачити ймовірність належності спостереження до певного класу. Він використовує сигмоїдну функцію для перетворення лінійної комбінації вхідних змінних у значення ймовірності, що лежить між 0 і 1. Може шикоро застосовуватися в дослідженнях медичних, маркетингових, фінансових тощо [141].

**Навчання без учителя (Unsupervised Learning)** – підхід, в якому модель виявляє структуру в немічених даних. Він зазвичай використовується для кластеризації. Найвідоміші алгоритми даного типу:

1) Метод кластеризації K-середні (K-Means clustering) – це алгоритм, який використовується для групування схожих даних у кластери та працює за рахунок розбиття даних на задану кількість кластерів, мінімізуючи відстань між

точками в кожному кластері і їх центром. Він ітераційно коригує центри кластерів, поки не досягне оптимального поділу. Даний алгоритм широко застосовується в сферах маркетингу аналізі зображень, біоінформатика тощо [142].

2) Метод кластеризації K-середні (K-Means clustering) – це алгоритм, який використовується при роботі із набором даних, які відокремленні один від одного. Він працює за рахунок розбиття даних на задану кількість кластерів, мінімізуючи відстань між точками в кожному кластері і їх центром, а також ітераційно коригує центри кластерів, поки не досягне оптимального поділу [143].

3) Алгоритми головних компонент (PCA) – це статистичний метод, який використовується для зменшення розмірності даних, зберігаючи при цьому найбільш важливу інформацію. Він трансформує початкові змінні в нові, які є лінійними комбінаціями оригінальних змінних, таким чином зменшуючи кількість ознак, а це у свою чергу дозволяє покращити ефективність аналізу та моделювання. Широко своє застосовується даний метод знайшов у візуалізації даних, стисканні інформації, покращення продуктивності моделей та зменшення впливу мультиколінеарності в задачах машинного навчання [144].

4) Apriori – алгоритм, який працює на основі того, що всі підмножини частого набору повинні бути частими, що дозволяє ефективно скорочувати кількість перевірок і зменшувати обсяг обчислень [145].

**Напівконтрольоване навчання (Semi-supervised Learning)** – підхід, який є поєднанням двох вищезгаданих, в якому модель використовує мічені дані для навчання, але може також використати немічені дані для покращення результатів:

1) Трансдуктивні SVM (Transductive support vector machine) – використовує інформацію на основі немічених зразків, аби клафікувати та забезпечити кращу продуктивність класифікації, аніж звичайна опорна векторна машина (SVM). Однак, такий алгоритм має очевидний недолік, так як кількість позитивних зразків має бути визначена до навчання, і вона не

змінюється під час фази навчання [146].

2) Генеративні моделі (Generative models) – моделі, які використовуються для створення нових даних, які є схожими на вже існуючі. Генеративні моделі навчаються на основі реальних даних і можуть генерувати нові зразки, які мають подібні характеристики. Вони використовуються в різних сферах, таких як створення зображень, текстів, музики та є основною в генеративному штучному інтелекті [147]. IBM – це компанія, як вже було зазначено вище, яка є одним із передових гравців в аспекті вивчення штучного інтелекту. В своєму блогу, який присвячений результатам пошуків та досліджень, вони присвятили окрему тему для генеративного штучного інтелекту [148]. Автори статті заявляють, що «генеративний ШІ відноситься до моделей глибокого навчання, які можуть генерувати високоякісний текст, зображення та інший вміст даних, на основі яких його навчали». Також автори спрощено пояснюють, як дана технологія здатна створювати контент на основі вже спожитої інформації: «на високому рівні генеративні моделі кодують спрощене представлення навчальних даних і використовують їх для створення нової роботи, схожої, але не ідентичної вихідним даним». Але, навіть, одні із основних просувачів ідеології генеративного штучного інтелекту не можу зазначити ризики, які той може породити: «генеративний ШІ має величезний потенціал для створення нових можливостей і цінностей для підприємства. Однак це також може створити нові ризики, будь то юридичні, фінансові чи репутаційні. Багато генеративних моделей, у тому числі ті, що використовують ChatGPT, можуть видавати інформацію, яка звучить авторитетно, але не відповідає дійсності (іноді її називають «галюцинаціями») або є небажаною та упередженою. Генеративні моделі також можуть ненавмисно поглинати особисту інформацію або інформацію, захищену авторським правом, у свої навчальні дані та виводити її пізніше, створюючи унікальні проблеми для законів про конфіденційність та інтелектуальну власність.»

3) Самонавчання (Self-Training) – самонавчання поєднує елементи навчання з учителем та без його участі, дозволяючи таким чином моделям

навчатися на основі не тільки мічених, але й немічених даних. Підхід такого роду використовує існуючі мічені дані для створення початкової моделі, а потім покращує її, використовуючи немічені дані, що допомагає підвищити точність моделі при обмеженому доступі до мічених даних. Такий спосіб навчання корисний для задач, де мічених даних мало або вони важко доступні [149].

4) Підкріплене навчання (Reinforcement Learning) – це підхід, в якому алгоритм вчиться через взаємодію з середовищем, отримуючи винагороду за правильні дії або ж покарання за неправильні. Головними характеристиками підкріпленого навчання є: відсутність прямого нагляду за діями, необхідність взаємодії з середовищем та врахування довгострокових наслідків від вибору дій. У порівнянні з іншими методами машинного навчання, такими як контрольоване та неконтрольоване навчання, підкріплене навчання фокусується на максимізації винагороди в умовах невизначеності, де важливо балансувати між дослідженням нових дій і використанням вже відомих [150].

У статті також згадуються інші важливі концепції та методи, що сприяють покращенню результатів:

**Навчання з множинними задачами (Multi-Task Learning)** – підхід, коли модель одночасно вирішує кілька різних пов’язаних між собою задач. На противагу від навчання для одного завдання, даний алгоритм пропонує ряд переваг, таких як спрощена архітектура моделі, покращена продуктивність або ж здатність до узагальнення між різними доменами. MTL включає різні методи, такі як регуляризація, навчання взаємозв’язків, пропагування ознак, оптимізація і попереднє навчання. З часом підхід став гнучким і здатним адаптуватися до нових завдань без обмежень за типом чи модальністю завдання, включаючи можливості навчання без явних міток (zero-shot learning). Цей підхід став популярним протягом останніх двадцяти років і активно використовується в таких галузях, як комп’ютерне бачення, обробка природної мови, системи рекомендацій, діагностика хвороб і робототехніка [151].

**Ансамблеве навчання (Ensemble Learning)** – метод, в якому комбінується використання кількох моделей. Це дозволяє покращити точність і

стабільність прогнозів, за рахунок зниження ймдвірності використання моделі, яка погано себе показує в конкретній ситуації. Найбільш відомі алгоритми – це усереднення (averaging), пакетування (bagging), випадковий ліс (random forest), укладання (stacking) та підвищення (boosting) [152].

**Нейронні мережі (Neural Networks)** – широковідомий в загальних масах алгоритм, що імітує структуру та функціонування людського мозку, і використовуються для виявлення складних патернів у великих наборах даних. Їх побудова складає ться з багатьох шарів обчислювальних одиниць, званих нейронами, які взаємодіють між собою. Поділяється на кілька типів, так як здатний працювати із вчителем (Supervised Neural Networks), без вчителя (Unsupervised Neural Networks) та із підкріпленим навчанням (Reinforced Neural Networks). Використовує в своїй роботі Глибоке навчання (Deep Learning) – підкатегорія машинного навчання, де використовуються багатошарові нейронні мережі, які здатні вирішувати дуже складні завдання (розпізнавання зображень, обробка природної мови тощо), оскільки вони можуть виявляти складні зв’язки та абстракції в даних [153].

**Навчання на основі прикладів (Instance-Based Learning)** – це метод, який здатен зберігати дані навчання і використовувати їх без створення абстракцій. Рішення в такому методі приймаються на основі подібності нових даних до найближчих екземплярів з навчальної вибірки. Найвідоміший його алгоритм – це алгоритм K-найближчих сусідів (K-Nearest Neighbors), який використовується для класифікації або ж регресії, і де нові дані класифікуються на основі найближчих сусідів у навчальній вибірці [154].

Дуже зручно в своїй роботі [155] автори згрупували методи по сфері їх застосуванні:

1) Лінійна регресія – оцінювання робіт, знаходження раку.

2) Логічна регресія – персональне навчання, виявлення ретинопатії, виявлення ботів, прогнозування поведінки криптовалюти, соціальні медія та вакцини, соціальні медія та політика.

3) Нейронні мережі – оцінювання робіт, персональне навчання,

виявлення раку, оцінка кредитних ризиків, прогнозування поведінки криптовалюти.

4) Підтримка векторних машин – оцінювання робіт, персональне навчання, передбачення наслідків ліків, виявлення ринопатії, виявлення раку, мережева система виявлення вторгнення, виявлення ботів, оцінка кредитних ризиків, прогнозування поведінки криптовалюти, передбачення кризи валюти, фармаконагляд, соціальні медія та вакцини, соціальні медія та політика.

5) Наївний байєс – оцінювання робіт, розумний супровід, персональне навчання, виявлення ринопатії, прогнозування поведінки криптовалюти, соціальні медія та політика.

6) Дерева рішень – оцінювання робіт, попередження відрахування, персональне навчання, дослідження діабету, виявлення ринопатії, виявлення раку, мережева система виявлення вторгнення, виявлення ботів, мережева система виявлення вторгнення в авто, оцінення кредитних ризиків, прогнозування поведінки криптовалюти, передбачення кризи валюти, соціальні медія та політика.

7) K-середніх – рекомендування курсів, персоналізоване навчання, оцінення кредитних ризиків.

8) Апріорі – рекомендація курсів, персональне навчання.

9) K-найближчих сусідів – персональне навчання, виявлення ринопатії, виявлення раку, мережева система виявлення вторгнення, оцінення кредитних ризиків, прогнозування поведінки криптовалюти.

10) Глибоке навчання – оцінювання робіт, передбачення наслідків ліків, виявлення ринопатії, виявлення раку, мережева система виявлення вторгнення, виявлення ботів, мережева система виявлення вторгнення в авто, оцінення кредитних ризиків, прогнозування поведінки криптовалюти, передбачення кризи валюти, фармаконагляд, соціальні медія та політика.

**Фільтрування даних** є важливим етапом у процесі машинного навчання, так як перед тим, як використовувати дані для навчання моделей, дуже важливо опрацювати та очистити їх від небажаних або неактуальних значень. Це

дозволяє підвищити точність і ефективність моделей.

Фільтрування даних – це спосіб зменшення вмісту шуму або помилок у виміряних даних процесу. Таке завдання є дійсно важливим, адже шум вимірювання може маскувати важливі характеристики в даних та обмежувати їхню корисність на практиці [156]. В сучасному світі, де все більшого панування набуває штучний інтелект разом із машинним навчанням, існують численні алгоритми, які можуть бути використані для фільтрації даних в залежності від типу інформації та конкретних завдань.

В контексті машинного навчання такий процес може включати:

1) Видалення шумів [157];
2) Виявлення аномалій [158];
3) Оброблення та заповнення пропущених параметрів [159];
4) Редукція вимірів [160];

Методи машинного навчання можна поділити на дві основні категорії: навчання з учителем та навчання без учителя. У методах навчання з учителем вхідні дані розподіляються на наперед визначені класи. Для навчання таких класифікаторів необхідна навчальна вибірка з помаркованими прикладами різних класів. Методи ж навчання без учителя не потребують маркованих даних і не асоціюють вхідні дані з конкретними класами, а виявляють закономірності в даних і виконують їх групування у кластери, які схожі між собою [161].

Машинне навчання включає в себе декілька алгоритмів машинного навчання для фільтрації даних:

1) Алгоритми класифікації – це один із основних методів, який може бути використаний для фільтрування даних. Такі алгоритми можуть бути застосовані для виявлення та відділення так званих «шумових» даних або ж аномалій від нормальних спостережень. Типи таких алгоритмів [162]:

a. Метод опорних векторів (SVM) – метод, який дозволяє ефективно фільтрувати «неправильні» або ж некоректні дані.

b. Логістична регресія – метод, який може використовуватися для оцінки ймовірності приналежності елементів до певного класу.

c. k-найближчих сусідів – це метод, який фільтрує аномалії на основі схожості з іншими елементами.

2) Алгоритми кластеризації – алгоритми, що дозволяють згрупувати схожі елементи в даних, що може бути використано для фільтрації схожих чи аномальних значень. Типи таких алгоритмів [163]:

a. k-середніх (k-means) – популярний метод кластеризації, який прагне мінімізувати середню квадратичну відстань між точками в одному кластері та може бути використаний для групування даних та виявлення аномалій через порівняння відстаней між точками.

b. Ієрархічна кластеризація – метод, який є ефективним для кластеризації та фільтрації даних з різною щільністю, де шуми можуть бути відокремлені від основних кластерів. При його роботі будується не одне розбиття вибірки на непересічні класи, а систему вкладених розбиттів.

3) Методи відбору ознак – це метод, в якому фільтрація даних відбувається за допомогою вибору ознак, що дозволяє зменшити розмірність та, що важливо, зберегти тільки найбільш важливі компоненти. Типи таких алгоритмів [164]:

a. Метод головних компонентів або ж PCA – метод, який використовується для того, щоб зменшити розмірность та фільтрувати найбільші вагомі ознаки.

b. Реалізація через регуляризацію – тип, який відомий своїми методами (L1 (Lasso) та L2 (Ridge)), де регуляризація може бути використана для відбору найважливіших ознак та відкидання малозначущих.

4) Алгоритми детекції аномалій – алгоритми для виявлення та фільтрації викидів або аномальних значень, які можуть бути застосовані для знаходження аномалій. Такі алгоритми використовують статистичні методи або підходи на основі навчання. Типи таких алгоритмів [165]:

a. Алгоритм локальної детекції аномалій (LOF) – це метод, що дає оцінку наскільки віддаленими є точки від своїх сусідів. Він використовується для виявлення аномалій у великих наборах даних.

b. Автокодувальники (Autoencoders) – це нейронні мережі, що можуть бути використані для виявлення аномальних спостережень, якщо модель не може відновити вхідні дані через відмінності від стандартного шаблону.

На основі розглянутих парадигм, робимо висновок, що для задач освітньої сфери, де дані можуть бути різнорідними та неповними, критично важливим є вибір алгоритму, здатного працювати з великими масивами інформації та виявляти приховані закономірності. Також є важливим попередня обробка даних перед подачею їх на вхід рекомендаційного алгоритму. А також бачимо, що фільтрація даних за допомогою методів класифікації та кластеризації є необхідною передумовою для побудови ефективної моделі, а використання методів зменшення розмірності або регуляризації дозволяє підвищити точність прогнозу та уникнути перенавчання. Таким чином ми можемо перейти до обґрунтованого вибору конкретного підходу (колаборативної фільтрації та матричної факторизації) для реалізації рекомендаційної системи, що забезпечить баланс між точністю рекомендацій та обчислювальною ефективністю.

### 3.2 Вибір підходу до побудови ефективної рекомендаційної системи

Машинне навчання, як один із ключових підтипів штучного інтелекту, широко застосовується для розроблення рекомендаційних систем. Завдяки здатності алгоритмів виявляти закономірності у даних та прогнозувати поведінкові патерни користувачів, ці технології забезпечують формування персоналізованих рекомендацій і підвищують ефективність взаємодії з інформаційними ресурсами.

Саме тому даний підтип широко використовується як власниками соцмереж для рекомендації трендів для свої користувачів, так і правохоронними органами для пошуку артефактів злочинних дій в глобальній мережі. В загальному, використання машинного навчання можна поділити на такі галузі: розпізнавання голосу, надання послуг в обслуговуючому сервісі, комп’ютерному баченні, роботизованому процесі автоматизації,

автоматизованій торгівлі акціями, виявленню шахрайства та в найважливішій сфері для нашої статті – рекомендаційних системах.

Автори публікацій [166, 167] зазначають, що існує декілька підходів у створенні рекомендаційних систем в LMS та кожен з яких має свої переваги та недоліки:

1) Фільтрація за змістом – спосіб, коли користувачу надається рекомендація на основі схожості із матеріалами, які той вже оцінив в минулому. Цей метод базується на гіпотезі, що користувач зацікавиться об'єктами, які мають схожі характеристики з тими, що були ним позитивно оцінені у минулому. У контексті освітньої системи такими атрибутами можуть виступати метадані дисципліни: назва, опис, ключові слова, кафедра, викладач або перелік компетентностей. Система створює профілі користувачів, які відображають їхні інтереси через ваги відповідних характеристик. Процес рекомендації зводиться до обчислення міри подібності між вектором профілю студента та векторами доступних дисциплін. Аналіз сильних та слабких сторін цього підходу наведено в таблиці 3.1.

*Таблиця 3.1*

**Переваги та недоліки рекомендаційних систем, побудованих на основі фільтрації за вмістом**

| **Переваги** | **Недоліки** |
|---|---|
| Ключ для отримання відповідної рекомендації для кожного користувача | Елементи, які представлені для системи одним і тим самим набором ключових слів є однаковими |
| Підтримує різні типи даних для співставлення інтересів користувачів | Складно надати якісну рекомендацію для тих, хто переглядав велику кількістю продуктів в системі |
| Не потрібні дані про інших користувачів | Немає доступу до історії, коли користувач починає роботу із системою |
| Немає проблем із холодним стартом | Еволюцію інтересів користувача потрібно постійно враховувати |
| Система здатна надати унікальну рекомендацію, які відповідає специфічним вподобанням користувача | Для надання точних оцінок, користувач повинен давати оцінку та відгук вже пройдених пропозицій |

2) Фільтрація на основі знань – на відміну від фільтрації за змістом, де проводиться аналіз характеристик самих елементів і порівнюють їх з

уподобаннями користувача, використовують чітко визначені знання про елементи та вимоги користувачів для створення рекомендацій. Такі системи для надання рекомендації не потребують історії взаємодії або ж відгуків користувачів та є особливо корисними в складних сферах, де товари купуються не часто, наприклад, автомобілі, квартири чи спеціалізоване обладнання [168]. Аналіз позитивних та негативних аспектів використання цього методу представлено в таблиці 3.2.

*Таблиця 3.2*

**Переваги та недоліки рекомендаційних систем, побудованих на фільтрації на основі знань**

| **Переваги** | **Недоліки** |
|---|---|
| Відсутність проблеми «холодного старту» | Необхідність збору та підтримки знань |
| Підходять для систем, де рідко купуються товари | Не враховує взаємодії між користувачами |
| Більш точні рекомендації на основі бази знань про зв’язки між елементами | Не завжди точне моделювання потреб користувачів |
| Можливість налаштування під конкретні потреби | Високі витрати на розробку та обслуговування |

3) Колаборативна фільтрація – спосіб генерації рекомендації, який грунтується на схожих уподобаннях інших користувачів. Як і попередні дві вищезгадані фільтрації має свої плюси та мінуси, які наведені в таблиці 3.3. Тип генерації рекомендації може будуватися за трьома категоріями:

a. На основі сусідів вибираються шляхом порівняння зі схожістю активних користувачів.

b. Виявляє подібність елементів на основі матриці.

c. Дає рекомендації на основі того, наскільки схожим користувачам той чи інший продукт сподобався.

*Таблиця 3.3*

**Переваги та недоліки рекомендаційних систем, побудованих на основі колаборативної фільтрації**

| **Переваги** | **Недоліки** |
|---|---|
| Використовує оцінки інших для оцінки елементу іншому користувачу | Проблема холодного старту |
| Знаходить групу користувачів, які є близькими за своїми вподобаннями до інтересів користувача, для якого дається рекомендація | Система потребує все більше часу та ресурсів на обчислення, чим більше користувачів до неї доєднується |
| Чим більше користувачів, тим більш якісна рекомендація буде надаватися | Система рекомендацій може стикнутися із низькою щільністю матриці |
| | Не можливість рекомендації нових не схожих продуктів |

4) Гібридні системи – ті, які поєднують в собі попередні підходи, щоб поліпшити точність рекомендації. Вони можуть компенсувати недоліки кожного із попередніх способів генерації рекомендації та покращити точність передбачення, однак є складнішими в інтеграції та часто мають високі вимоги до обчислювальних ресурсів.

5) Демографічні рекомендаційні системи – ті, які враховують демографічні характеристики користувачів. Даний тип рекомендаційних систем рідко згадується, але може зустрічатися в деяких працях. [169]

Як зазначає автор [170]: «Усі сучасні рекомендаційні системи схильні до проблеми бульбашки фільтрів, що виникає, коли алгоритм формування списку рекомендацій підбирає інформацію, яку користувач хотів би бачити, і, в результаті, користувачі відділяються від інформації, яка їх не цікавить або їм не подобається, фактично ізолюючи їх у власних «бульбашках».» та наводить вимоги, якими повинна характеризуватися вибірка рекомендацій:

1) Різноманітність – елементи прогнозу, який надала рекомендаційна система, не повинні бути майже однакові, а мати відмінності для надання різного роду послуг для клієнта, або ж вони повинні доповнювати одна одну.

2) Новизна – нові об’єкти, які ще на мають оцінок в системі, також мають рекомендуватися для кінцевого користувача.

3) Неочікуваність – серед елементів прогнозу, які надала система,

можуть бути неочікувані елементи. Такі елементи мають нести для користувача новий досвід та бути несхожим на попередні рекомендації.

Якщо даний приклад застосувати для нашої ситуації, то можемо бачити, що перший тип здатен давати рекомендації на основі особистих вподобань самого користувача, другий – здатен давати рекомендацію на основі вподобань підгрупи користувачів, які є схожими на цільового користувача, а третій – поєднує підходи фільтрації на основі контенту та колаборативної фільтрації для покращення якості рекомендацій. Фільтрування на основі контенту бере до уваги тільки уподобання здобувача, що не є бажаним для рекомендаційної системи для закладу вищої освіти, так як хоч в університетах і можуть виховувати спеціалістів вузького профілю, але володіння базовими різноманітними знання в їх спеціалізованій сфері є обов'язковим для створення висококваліфікованого спеціаліста. Також вагомим фактором у виборі подальшого плану навчання є досвід минулих поколінь, так як здобувачем, більш за все, буде обрано дисципліни, вірогідність здачі успішно яких буде якомога вищою, а відгуки здобувачів старших курсів будуть якомога кращими. Тому під вимоги базового алгоритму для першої версії рекомендаційної систем найкраще підходить спільна фільтрація, адже саме вона здатна вирішити нашу цільову задачу по рекомендації дисциплін на основі власної академічної успішності та результатів інших здобувачів.

### 3.3 Проблеми та виклики рекомендаційних систем

З огляду на те, що рекомендаційні системи охоплюють широкий спектр підходів та методів, у науковій літературі вони класифікуються за різними ознаками. У праці «Систематичний огляд і перспектива дослідження рекомендаційних систем» [171] Deepjyoti Roy та Mala Dutta підкреслюють, що для всебічного аналізу цих технологій необхідно розмежовувати їх за типами відповідно до принципів формування рекомендацій, природи вхідних даних та механізмів обробки інформації. Автори виділяють кілька основних типів рекомендаційних систем, кожен з яких ґрунтується на специфічних

методологічних підходах і має свої переваги та обмеження:

1) Система рекомендацій на основі вмісту – цей тип базується на параметрах продуктів системи, на основі яких далі надаються рекомендації для клієнта, в залежності від того, чи вони схожі своїми властивостями на продукти, які вподобав користувач, чи ні.

2) Система рекомендацій на основі спільної фільтрації – основна алгоритму даної системи лежить в пошуку групи користувачів, вподобання яких схоже до вподобань цільового користувача. Проаналізувавши ці дані, система здатна роботи передбачення, які наступні продукти можуть бути вподобані цільовим користувачем, на основі того, як дані продукти вподобала вибірка користувачів, яка була підібрана на попередніх кроках. Таким чином, основною ідеєю цього алгоритму є гіпотеза того, що користувач буде мати такі ж самі вподобання в майбутньому, як і група користувачів, попередні вподобання яких є схожими до вподобань цільового користувача.

3) Гібридна фільтрація – це поєднання двох попередніх типів, для надання якомога точнішої оцінки прогнозування для користувача. Даний підхід також може покращити не тільки точність передбачення, а я продуктивність та швидкодію системи, що в проєктах певного типу може бути критичним для кінцевого користувача.

Хоча рекомендаційна система й дає чимало переваг для компаній, які реалізовують в себе даний функціонал, проте вони викликають чимало головного болю, коли доходить до вирішення типових проблем цих систем. Таких проблем може бути чимало, тому в праці «Дослідження викликів та рішень для рекомендаційних систем» [172] автори Marwa Hussien Mohamed, Mohamed Khafagy та Mohamed Hasan Ibrahim спробували згрупувати їх в певні підтипи:

1) Приватність – це проблема, яка полягає у використанні персональних даних користувачів, які використовує рекомендаційна система для прогнозування. Проблема може бути дуже серйозною, так як це може призвести до витоку конфіденційних даних користувачів, що може стати

критичним для певного проєкту.

2) Проблема холодного старту – це проблема, яка стосується нових користувачів, які ще не виконували ніяких дій в системі та щодо них немає якихось даних, які б могли їх характеризувати. В такому випадку рекомендаційна система стикається з проблемою точності рекомендування продуктів для користувача, які той може вподобати, так як вона не має на основі чого будувати свої передбачення.

3) Надмірна спеціалізація – це тип проблеми, коли для користувача формуються передбачення завжди схожі та на одну чи декілька спільних тематик, так як користувач або оцінював тільки такі вузьспеціалізовані продукти, або система тільки на такі фактори опирається при розрахунку рекомендації. Внаслідок винекнення такої проблеми, користувач стикається з проблемою пошуку нових продуктів, які потенційно йому можуть сподобатися, але вони лежать поза сферою його теперішніх вподобань в системі.

4) Масштабування – пов’язана дана проблема з поступовим збільшенням даних, на основі яких рекомендаційна система робить прогнозування. Може здатися, що чим більше даних, тим краще для кінцевого користувача, адже передбачення для нього буде створене на основі великої вибірки, що дасть всеосяжну повноту критеріїв та оцінок продуктів, але це в свою чергу може викликати доволі серйозну проблему, так як збільшення часу на обробку інформації, що може бути негативно оцінено кінцевим користувачем.

5) Розрідженість даних – коли в користувача є значний обсяг продуктів, але він їх не оцінює, хоча рекомендаційна система сильно залежить від оцінки користувача, на основі якої, вона може давати рекомендації для інших користувачів. В певній мірі, дана проблема є протилежною до проблеми масштабування, так як в одному випадку даних для якісного передбачення є дуже багато, а в іншому замало.

6) Різноманітність – проблема, яка виникає, якщо рекомендаційна система будує своє прогнозування на одному типі даних, через що результат

може бути неточний та однобічний.

7) Новизна – вибірка даних для опрацювання рекомендаційною системою повинна постійно поповнюватися свіжими даними, що даватиме більш актуальну та точну оцінку для кінцевого користувача.

8) Щаслива випадковість – інколи, є ймовірність, що користувач може отримати неочікувану рекомендацію, навіть, якщо вона вдала.

9) Щаслива випадковість – інколи, є ймовірність, що користувач може отримати неочікувану рекомендацію, навіть, якщо вона вдала.

10) Атаки на систему – проблема виникає, коли дані, які використовуються для прогнозу, є пошкоджені через навмисні дії деяких користувачів в необ'єктивному оціненні певних продуктів для зміни їхнього рейтенгу.

11) «Сіра вівця» – проблема виникає, коли користувач по своїх вподобаннях не підходить під жодну вибірку користувачів, а тому не отримує якісного передбачення системою.

Однак, варто зауважити, що у вже згаданій праці «Систематичний огляд і перспектива дослідження рекомендаційних систем» [171] Deepjyoti Roy та Mala Dutta також додатково визначають такі проблеми:

1) Схожість продуктів – ситуація часто виникає, коли в нас є дуже схожі або ідентичні продукти, які відрізняються наприклад своїм іменуванням. Чимало рекомендаційних систем не вміють вирішувати цю проблему, через що кінцевий користувач може отримати менш якісне передбачення.

2) Проблема рекомендації нових продуктів – часто виникають ситуації, коли новий елемент не рекомендується користувачу, адже через свою новизну, він має мало оцінювальних характеристик, порівняно із більш «старішими» продуктами, через що він менше рекомендується для кінцевого користувача, а це призводить до зациклення проблеми, так як менше рекомендацій – менше оцінювання, менше оцінювання – менше рекомендацій.

Розглянемо вищезгадані проблеми в автоматизованій системі для побудови індивідуального плану навчання здобувача вищої освіти:

1) Приватність – для коректної роботи рекомендаційної системи нам потрібні перш за все дані про успішність певних здобувачів за певний відрізок часу. В тому випадку, якщо така інформація внаслідок неправомірних дій потрапить до третіх осіб, то це не спричинить серйозних проблем, адже така інформація не є надзвичайно конфеденційною та закритою.

2) Проблема холодного старту – проблема, яка може бути актуальна у випадку рекомендації дисциплін до вивчення для осіб, які тільки-но розпочали навчання. У зв’язку з тим, що в здобувача немає ще результатів хоч з якихось університетських дисциплін, то система немає факторів, на які б могла опиратися при наданні найбільш підходящої наступної дисципліни. Дана проблема зникає, якщо система буде застосовуватися тільки для здобувачів старших курсів, адже певні результати в них вже є, які допоможуть у наданні найкращої рекомендації. Або ж цю дилему можна вирішити за рахунок введення анкетування першокурсників та на основі наданих відповідей будувати навчальний план для здобувачів. Таке анкетування могло б містити як вподобання здобувачів як в загальному плані, так і в плані спеціалізації, його хобі або ж успіхи в школі чи спортивних змаганнях.

3) Надмірна спеціалізація – це проблема, яка з легкістю може виникнути в системі такого профілю, але вирішувати чи добре це, чи погано необхідно власнику продукту. В наш час цінуються як фахівці, які мають широкі всебічні знання у своїй сфері, так і вузькопрофільні спеціалісти, які мають майже до досконалості відточені навички в певній специфічній галузі. Тому алгоритм обробки даних має залежати від рішення творців проєкту, але однозначним є той факт, що користувач такої автоматизованої системи, повинен завжди мати можливість вибрати самостійно ту дисципліну, яку він сам захотів, а не який йому підібрав спеціальний алгоритм. Рекомендаційна система повинна бути такою, що інформує та надає консультації, а не бути наказово-примусовим інструментом у побудові навчального вектору. Впровадивши таку альтернативну можливість в систему, ми досягнемо всестороннього задоволення потреб користувача.

4) Масштабування – проблема, яка при необачній фільтрації даних може швидко стати критичною для рекомендаційної системи навчального вектору здобувача. Небажані наслідки, такі як повільна робота система чи надмірне споживання апаратних потужностей серверу, можуть виникнути тоді, коли береться надмірна кількість даних для аналізу та пошуку найкращого рішення. Так, якщо для прогнозування найбільш відповідного до вподобань здобувача дисципліни, будуть братись в обробку оцінки здобувачів всіх років та всіх спеціальностей, то ми отримаємо завелику вибірку, що і призведе до вищезгаданих проблем. І чим більшим навчається здобувачів в закладі вищої освіти, тим серйознішою буде дана проблема. В такій системі фільтрація даних для аналізу має визначатися або за умов творця системи, або бути динамічним показником (із заздалегіть визначеними лімітами значень), який зможуть задавати користувачі системи або ж бути взяті за останні три роки та тільки стосуватися спеціалізації тієї персони, для якої проводиться аналіз. Дані часові рамки запропоновані у зв’язку з тим, що чотири роки це вже завеликий термін для аналізу, так як за цей час може змінитися як методи ведення певної, так і сам викладач дисципліни, а обмеження вибірки здобувачів тільки із тієї ж спеціальності базується на гіпотезі того, що із великою ймовірністю здобувача певної спеціальності будуть цікави ті ж дисципліни, що й інших здобувачів цієї спеціальності.

5) Розрідженість даних – дана проблема не становить загрози, коли в аналіз беруться тільки дані, які відповідають за оцінки здобувачів із певної вибірки дисциплін. Однак, дана проблематика може виникнути, якщо в розрахунок прогнозу також беруть анкетні дані якості освіти. Такі дані містять особисті враження та суб’єктивну оцінку здобувачів, які можуть виступати вагомим чинником в наданні прогнозу для інших здобувачів у виборі, але й також можуть внести похибку в точність прогнозу, через неоднорідність надання відгуків щодо пройдених дисциплін.

6) Різноманітність – при побудові алгоритму обробки даних, важливо будувати його на різних типах даних, що для здобувача не так вже й важко

зробити, якщо враховувати також його вподобання, відгуки й т.д. Якщо рекомендаційну систему побудвати занадто простою, то може виникнути ситуація, коли для різних здобувачів, не залежно від їх вмінь та вподобань, відсоток рекомендацій одних і тих же дисциплін є занадто великим. Така поведінка є не бажаною в моделі, яка надає індивідуальний прогноз, а тому її слід уникати, якщо не на початкових етапах існування, то на його подальших ітераціях розвитку.

7) Новизна – проблема, коли база даних не буде поповнюватись свіжою інформацією, не має виникнути в рекомендаційних системах, які базуються на аналізі оцінок здобувачів з певних дисциплін. Це зумовлено тим фактором, що процес навчання є постійним, а отже система мінімум раз в півроку повинна отримувати свіжі дані результатів успішності здобувачів, на основі яких алгоритм буде мати змогу давати точніший та більш актуальний прогноз.

8) Щаслива випадковість – даний фактор матиме тим частіше місце в системі, чим більше різнопланових даних використовується для надання прогнозу. Така випадковість не є критичною для даної рекомендаційної системи, адже специфіка пропозиції наступної дисциплін не є критичним фактором, який впливає на життя, здоров’я, фінансовий стан і т.д. здобувача. Також така подія може вплинути й позитивно вплинути на вибір здобувача, адже може запропонувати щось нове для здобувача, що у свою чергу частково вирішить проблему надмірної спеціалізації.

9) Атака на систему – проблема, яка може серйозно вплинути на систему, якщо її алгоритм бере в розрахунок, приміром, суб’єктивну оцінку здобувача відносно певної дисципліни та якщо ця оцінка містила в собі навмисне спотворену та неправдаву інформацію. Дану проблему можна вирішити або за рахунок фільтрації даних на етапі їх збору від здобувачів, або ж за рахунок числених вбудованих методів машинного навчання.

10) Проблема «сірої вівці» – випадок, який може мати місце в рекомендаційній системі по вибору вектору навчання здобувача, але який з

легкістю компенсується функціоналом, який надасть здобувачеві самому обрати дисципліну для подальшого навчання. Після такого вибору, система отримає нові дані, які вже дозволять отримати новий більш точний прогноз.

11) Схожість продуктів – дана ситуація може виникнути в системі, коли беруться в аналіз дисципліни, які є близькими по своїй тематиці чи сфері застосування. Для вирішення даної проблематики можна будувати прогноз на додатковому факторі спорідненості дисциплін.

12) Рекомендації нових продуктів – проблема, яка виникає в залежності від алгоритму фільтрації та аналізу даних. Якщо система бере в розрахунок, як було запропоновано вище, оцінки з дисциплін за останні три роки, то логічним виглядає те, що нова дисципліна повинна спочатку пройти пробний період, виправити свої неточності в форматі навчання, отримати перші оцінки для здобувачів освіти в системі, а вже потім рекомендуватися профільною рекомендаційною системою.

Отож, ми розглянули типові проблеми для рекомендаційних систем, прирівняли їх до освітньої сфери та дали відповідь на них. Висновки на основі цих відповідей, допоможуть обрати правильний спосіб реалізації алгоритму рекомендаційної системи, а також допоможуть покращувати та доповнювати його в майбутньому. Дані відповіді не є обов’язковими для реалізації в кожній системі, яка надає рекомендації дисциплін до вивчення, так як багато ще чого може залежати від супутніх факторів. Однак, якщо притримуватись більшість із тих відповідей, які було сформовано, то можна отримати програмний модуль, який здатен надавати якісні рекомендації в побудові для здобувача вищої освіти індивідуального плану навчання, шляхом підбору найбільш відповідних для нього дисциплін.

Також, слід зазначити, що сформувавши відповіді на типові проблематики рекомендаційних систем, можна дійти висновку, що для побудови якісної рекомендаційної системи, як мінімум, варто звертати увагу на колаборативну фільтрацію або ж на гібридні системи, так як фільтрація за змістом є недостатньою для надання рекомендацій із розрахунком на

індивідуальні потреби та бажання здобувача освіти. Система рекомендацій на основі вмісту враховує лише дані здобувача, що не може задовільнити всі аспекти в побудові індивідуального плану навчання здобувача вищої освіти, оскільки хоч університети можуть готувати фахівців вузького профілю, але володіння базовими різнобічними знаннями у своїй спеціалізації є обов'язковим для створення висококваліфікованого спеціаліста. Також, слід зауважити, що досвід попередніх поколінь є важливим фактором у виборі майбутнього плану навчання, оскільки здобувачі, швидше за все, обиратимуть дисципліни по-перше, які з найбільшою вірогідністю вони можуть успішно здати, а по-друге, якіі мають найкращі відгуки серед здобувачів, які вже пройшли даний етап навчання. Ось чому система рекомендацій на основі спільної фільтрації дуже добре відповідає нашим потребам та дає змогу вирішити наше цільове завдання – надавати персональні рекомендації дисциплін для здобувачів спираючи на числення фактори, серед яких особистий досвід здобувача, успішність здобувачів, які знаходяться в одному й тому ж колі інтересів, інтереси й т.д.

### 3.4 Програмна реалізація алгоритму рекомендаційної системи

Найбільш непростий вибір стосується вибору технології для реалізації рекомендаційної системи. Обрана технологія має задовільняти всі вимоги до рекомендаційних систем (швидкість, точність, надійність і т.д.), а також повинна підтримувати інтеграцію із обраними нами вище технологіями. Адже, навіть якщо певна технологія здається надзвичайно потужною в даній сфері, її інтеграція може виявитися складною через проблеми сумісності з іншими частинами інфраструктури. Тому, наш вибір має бути базуватися не лише на теоретичних характеристиках технологій, а й на практичних вимогах нашого проєкту. Тільки за таких умов можна досягнути балансу між високою ефективністю та можливість працювати в рамках існуючої інфраструктури.

Тепер нам важливо розглянути, які мови програмування найкраще підходять для обробки великих обсягів даних, реалізації алгоритмів машинного навчання, забезпечення високої продуктивності в реальному часі та інтеграції із

існуючою системою. Як зазначається в дослідженні [171] найбільш популярні платформи моделювання, які використовуються для розробки різних систем рекомендацій та можуть підходити під наші задачі це: Java, Python, мова програмування R, TensorFlow, Weka тощо. Тож розглянемо найбільш популярні і не тільки мови програмування для створення рекомендаційних систем:

1) Python – найпопулярніша мова програмування для створення рекомендаційних систем. В інтернеті з легкістю можна знайти численні реалізації такого типу проєктів саме за допомогою даної технології. Python вважається нодією із найпростіших мов програмування. Має величезну кількість вбудованих методів у складі стандартної бібліотеки. Серед основних особливостей Python: відкритий код, високорівнева мова програмування, незалежність від платформи, процедурно і об'єктно-орієнтований [173]. З найбільш використовуваних модулів і бібліотек за допомогою яких можна побудувати рекомендаційну систему разом із Python [174]:

a. Pandas – пакет для маніпуляційних дій над даними, який можна використовувати для аналізу та підготовки даних, а також для очищення та попередньої обробки інформації.

b. NumPy – пакет для наукових обчислень, який підтримує операції з масивами та матрицями, а також математичні функції.

c. Scikit-learn – пакет, який можна використовувати для створення алгоритмів рекомендаційної системи, так як він підтримує багато методів, таких як класифікація, регресія та кластеризація.

d. Natural Language Toolkit – модуль для обробки природної мови, який підтримує лематизацію, токенізацію, стемінг та інші методи.

e. Gensim – модуль для тематичного моделювання та обробки природної мови. Він підтримує такі алгоритми машинного навчання, як LSI, LDA, Word2Vec.

f. Matplotlib – пакет Python для візуалізації даних, який дозволяє створювати різноманітні графіки, діаграми та схеми.

g. Selenium – бібліотека, що допомагає працювати з

вебпереглядачами та автоматизувати їх.

h. Joblib – бібліотека для паралельних обчислень, яка підтримує розподіл завдань між кількома процесорами. Часто використовується для прискорення тренування моделей машинного навчання.

i. Streamlit – це популярний пакет для проєктів з аналізу даних та машинного навчання та призначений для швидкого створення інтерактивних вебзастосунків з використанням простих скриптів на мові Python.

j. Pickle – стандартний пакет Python для серіалізації та десеріалізації об’єктів, що дозволяє зберігати об’єкти у файли та передавати їх по мережі.

k. Модуль «re» – пакет призначення якого це робота з регулярними виразами.

2) Java – об’єктно-орієнтована мова програмування, яка для роботи використовує віртуальну машину, що забезпечує універсалізм програм, які були написані за допомогою неї [175]. Дана мова програмування характеризується високою продуктивність та продуктивністю, має велику кількість бібліотек та фреймворків як для класичних задач із програмування, так і для рекомендаційних систем, приміром:

a. Apache Mahout – проєкт із відкритим кодом, який забезпечує алгоритми машинного навчання, у тому числі для колаборативної фільтрації, класифікації та кластеризації [176].

b. Weka – бібліотека, яка використовується для машинного навчання та аналізу даних, в академічних дослідженнях, промисловості або ж для навчання. Має в арсеналі велику колекцію сучасних алгоритмів для регресії, класифікації, кластеризації, пошуку асоціацій, візуалізації та попередньої обробки даних [177].

c. Deeplearning4j – бібліотека для глибинного штучного навчання, яка реалізує нейронні мережі на віртуальній машині Java, використовуючи API для Java та C++ для обчислень, та спеціалізується на ефективному тренуванні глибинних моделей в умовах високої паралельності та великих обсягів даних.

Підтримує різні типи нейронних мереж, зокрема рекурентні нейронні мережі, згорткові мережі, обмежені машини Больцмана, глибокі мережі віри, глибокі автоенкодери та поверхневі нейронні мережі. Інтегрується з Hadoop та Spark, що дозволяє масштабувати навчання моделей для великих обсягів даних, а також підтримує паралельне тренування нейронних мереж як на процесорах, так і на графічних процесорах [178].

3) C++ – найпоширеніша мова програмування для операційних систем на Window та Unix. Вона є розширенням мови програмування C, але також підтримує об’єктно-орієнтоване програмування, простір імен, перевантаження операторів та імен функцій тощо. Серед недоліків, які можна відмітити: складний синтаксис, макроси, неінтуїтивні типи перетворень і т.д. [179] Також широко застосовується в побудові рекомендаційних систем, та за результатами деяких досліджень [180] може вигравати у швидкодії, але програвати в точності в порівнянні із Python.

4) TensorFlow – це система машинного навчання, яка працює на великому масштабі та в гетерогенних середовищах. Вона використовує графи потоку даних для представлення обчислень, спільного стану та операцій, які змінюють цей стан, а також розподіляє вузли графа потоку даних між багатьма машинами в кластері або в межах окремої машини на кілька обчислювальних пристроїв. Розробники TensorFlow зазначають, що дана технологія має відкритий код, а тому вже понад 8 000 осіб створили репозиторії вихідного коду, бінарне розповсюдження було завантажено 500 000 разів, а їх користувачі опублікували десятки моделей машинного навчання, що використовують цю систему. [181]

5) R – це мова програмування та програмне середовище, які знаходяться у відкритому доступі, та використовуються для аналізу емпіричних даних, графічного представлення та звітності [182]. Широко використовується в рекомендаційних системах та має багато спеціальних пакетів, серед яких один із найпопулярніших Recommenderlab – один з основних пакетів, який надає інструменти для реалізації різних методів, таких як колаборативна фільтрація,

методи на основі контенту, метод латентних факторів а також можливість створювати гібридні системи [183].

6) Scala – це маштабована, функціональна, об'єктно-орієнтована мова програмування, що використовується для роботи з великими даними, особливо разом із Apache Spark. Дана мова програмування схожа на Java, але має більш лаконічний синтаксис та більше бібліотек, які орієнтовані на функціональне програмування. [184]

7) JavaScript – це інтегрована мова створення сценаріїв, яка дозволяє вбудувати прикладний код безпосередньо в HTML-сторінку. Завдяки тому, що JavaScript виконується безпосередньо в браузері, він дозволяє будувати системи рекомендацій, які можуть працювати в реальному часі і адаптуватися до дій користувачів. Може використовуватися у вже вищезгаданому нами TensorFlow.js. [185]

8) ML.NET – це відкритий фреймворк, який є розробкою компанії Microsoft для побудови моделей машинного навчання на основі .NET. Дана технологія побудована на шаблоні конвеєра, який є ідеальним для розробки рішень машинного навчання. У цьому шаблоні елементи обробки розташовані так, що вихід кожного елемента є вхідним для наступного, що аналогічно фізичному конвеєру. Фреймворк дозволяє користувачам створювати власні моделі машинного навчання за допомогою мов програмування C# або F# для різних задач, таких як регресія, класифікація, аналіз настроїв, прогнозування часового ряду, рекомендації тощо. [186]

Дивлячись на всі описані рішення, їх переваги та недоліки та нюанси застосування, можемо зробити висновок, що вибір C# та ML.NET для побудови рекомендаційної системи (рис. 4.10) може бути чудовим рішенням і ось чому:

1) C# та ML.NET є частиною екосистеми .NET, що дає змогу зручно інтегрувати рекомендаційну систему з проєктом написаним на ASP.NET та з іншими інструментами Microsoft.

2) C# і ML.NET дозволяють створювати надійні та добре масштабовані рішення, які легко обслуговувати і розвивати.

3) Висока продуктивність, яку C# забезпечує завдяки компіляції в машинний код, що може бути критичним для великих та високонавантажених систем.

4) Надає широкі можливості для керованого навчання, класифікації, регресії, кластеризації, нейронних мереж та інших типів задач машинного навчання. Також, є підтримка глибокого навчання через інтеграцію з бібліотеками, такими як TensorFlow.

5) Вбудовані алгоритми для колаборативної фільтрації і контентної фільтрації, що дозволяє легко створити рекомендаційну систему без необхідності самостійно імплементувати алгоритми з нуля.

6) C# є однією з найбезпечніших мов програмування завдяки автоматичному керуванню пам’яттю, строгій типізації та багатьом іншим характеристикам. Це може бути важливим для розробки нашої рекомендаційної системи, що працює із конфіденційними даними здобувача.

7) Інтеграція з Azure Machine Learning, що дозволяє використовувати хмарні ресурси для тренування і розгортання моделей машинного навчання.

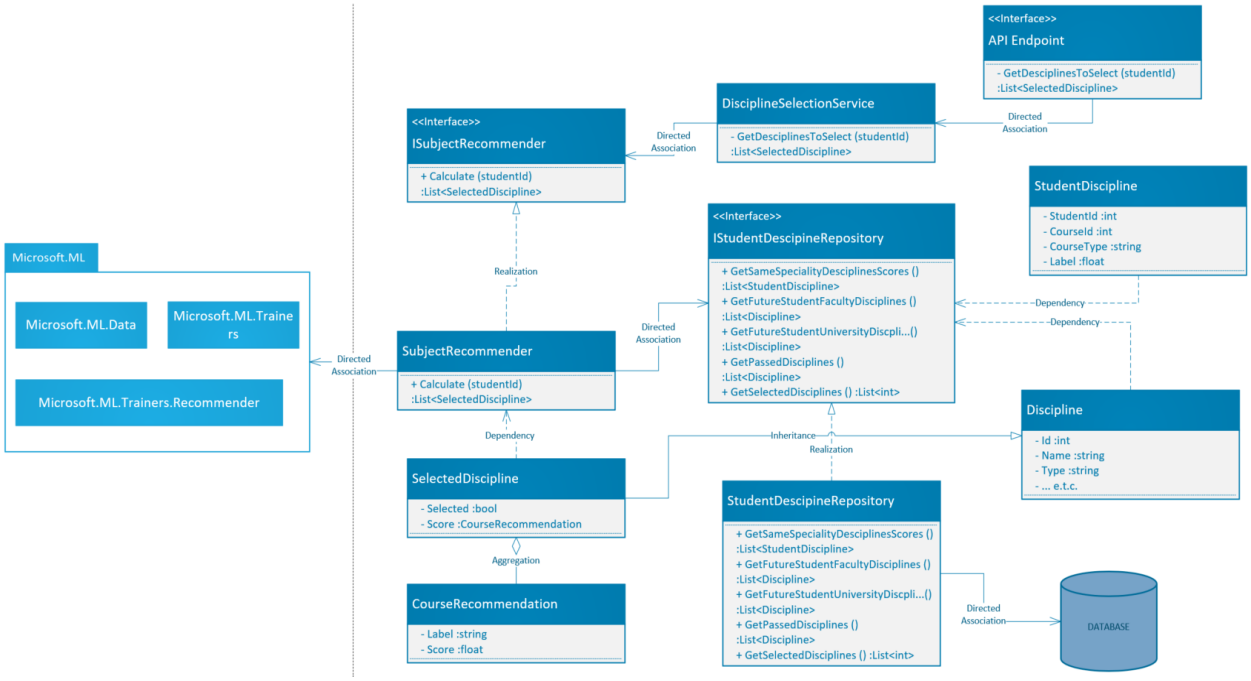


Рис. 3.1 – Діаграма класів рекомендаційної системи

Використовуючи матричну факторизацію та методи оцінювання точності оцінки була побудована базова логіка рекомендаційної системи на мові програмування C# та з використанням бібліотеки Microsoft.ML, яка представлена в Додатку Г. Розглянемо та пояснимо складові цього алгоритму.

Для того, щоб рекомендаційна система розроблялась паралельно та не в залежності від основної системи, було створено окремий проєкт із власним простором імен ML (рис. 3.2). Даний проєкт ми можемо підключити до системи та використовувати за потреби. Однією з основних переваг є також те, що в ході розробки нового покращеного алгоритму не прийдеться переписувати весь функціонал, а просто замінити залежності між ними. Головна умова такої підміни – це реалізації такого ж контракту, який реалізовував попередній алгоритм. Саме тому головний клас рекомендаційної системи SubjectRecommender реалізує інтерфейс ISubjectRecommender та використовується завдяки механізмам DependencyInjection [187]. Таке рішення дає в майбутньому гнучкість між основним модулем та рекомендаційним в плані розробки, тестування чи заміни.

```
namespace ML
{
    1 reference | Yaroslav Ponzel, 15 hours ago | 1 author, 16 changes
    public class SubjectRecommender(IStudentDesciplineRepository studentDesciplineRepository) : ISubjectRecommender
    {
```

Рис. 3.2 – Простір імен рекомендаційної системи

Для повноцінного функціонування алгоритму, імпорту даних для розрахунків, а також використання існуючих реалізацій алгоритмів машинного навчання проєкт потребував додавання сторонніх бібліотек (рис. 3.3):

1. Microsoft.ML – основний простір імен у ML.NET, що надає базові функції для роботи з даними, навчання моделей, переоцінювання та оцінки. Включає в себе основні класи та методи для створення та застосування моделей машинного навчання в різних проєктах на основі .NET.

2. Microsoft.ML.Data – простір імен, який містить класи та інтерфейси для роботи з даними. Надає засоби для опису даних (приміром, клас IDataView для роботи з великими наборами даних тощо), а також типи даних для

навчальних наборів, прогнозів та результатів.

3. Microsoft.ML.Trainers – простір імен для алгоритмів тренування моделей машинного навчання. В ньому зібрані різні алгоритми навчання, які можуть бути використані для класифікації, регресії, кластеризації тощо.

4. Microsoft.ML.Trainers.Recommender – це простір імен, що містить містить алгоритми тренування моделей для різних типів рекомендаційних моделей. Одним з таких є матрична факторизація – алгоритм, який є популярним методом у рекомендаційних системах для аналізу великих наборів даних.

5. Models – розроблений в рамках даної системи простір імен, який містить моделі, які описують дані, що використовуються для розрахунків та передбачень.

6. Models.Enums – розроблений в рамках даної системи простір імен, який містить певні класифікатори, що допомагають фільтрувати та структурувати дані.

7. Repositories – розроблений в рамках даної системи простір імен, який відповідає за взаємодію з базою даних та надає необхідні дані для обчислень та передбачень.

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Trainers;
using Microsoft.ML.Trainers.Recommender;
using Models;
using Models.Enums;
using Repositories;
```

Рис. 3.3 – Зовнішні бібліотеки, які використовує рекомендаційна система

Рекомендаційна система потребує власних класів для повноцінної роботи з даними. Тому в системі було визначено два класи StudentDiscipline та CourseRecommendation (рис 3.4), які описують дані для тренування моделі та дані, які описують вигляд передбачення моделі відповідно. StudentDiscipline містить властивості, які описують унікальний ідентифікатор здобувача, унікальний ідентифікатор дисципліни, назву дисципліни та дані про оцінку здобувача по даній дисципліні. CourseRecommendation містить властивості для

опису дисципліни, яка рекомендується, та значення передбачення для неї.

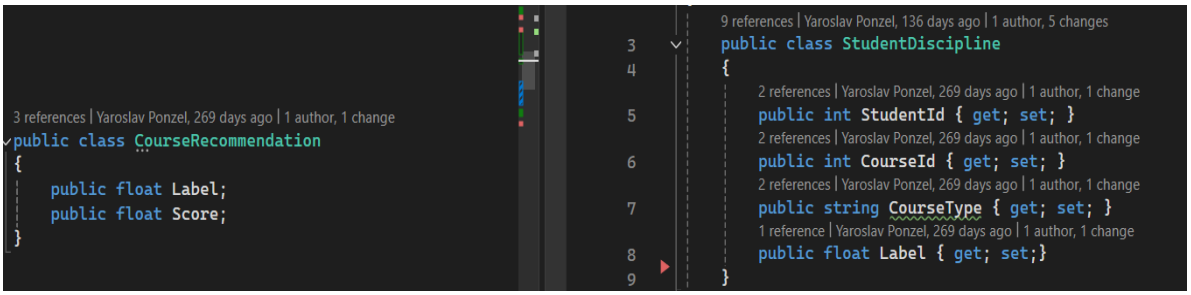


Рис. 3.4 – Основні класи рекомендаційної системи

Робота рекомендаційної системи починається із приймання параметру, який відповідає за унікальний ідентифікатор здобувача, для якого система має надати рекомендацію (рис 3.5). Із цим параметром система намагається витягнути дані із бази даних, які містять в собі інформацію про успіхи здобувачів із навчальних дисциплін за останні три роки. Такі дані можливо отримати за рахунок використання зовнішнього репозиторію IStudentDesciplineRepository, який ми отримали за рахунок механізму DependencyInjection, та який відповідає за абстракції доступу даних щодо дисциплін, які вивчають здобувачі.

Часовий проміжок розміром в 3 роки було обрано для опрацювання тільки актуальних даних та з метою унеможливлення роботи моделі із надмірною кількістю даних. У випадку ж, якщо такі дані не були знайдені в системі, то жодних процесів, які пов’язані із рекомендаційною системою, не відбувається, а система йде по своєму стандартному алгоритму вибору дисциплін. Така ситуація може трапитися, якщо здобувач освіти вперше навчається в університеті, а отже дані про хоча б якусь навчальну успішність відсутні в базі даних, або він навчався не в часову проміжку вибірки.

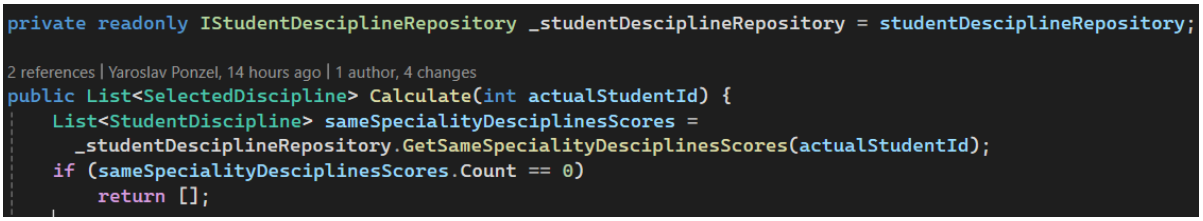


Рис. 3.5 – Сигнатура методу рекомендаційної системи

Дані взяті із репозиторію є такими, що не підходять для опрацювання машинним навчанням. А тому метод LoadFromEnumerable перетворює отримані дані для подальшої обробки та навчання моделей (рис. 3.6). Даний метод повертає об’єкт типу IDataView, який є основним абстрактним типом даних в ML.NET. Даний тип оптимізовано для роботи з великими наборами даних, оскільки дозволяє працювати з ними по частинах, не завантажуючи все в пам’ять одразу.

Іншим важливим етапом при створенні моделей машинного навчання є розділення набору даних на дві частини: тренувальний та тестовий, що досягається за рахунок використання методу TrainTestSplit. Це є важливим етапом при створенні моделей машинного навчання, оскільки дає змогу тренувати модель на одну наборі даних, а оцінювати її ефективність на інших, таким чином уникаючи перенавчання.

```
var mlContext = new MLContext();
IDataView trainingData = mlContext.Data.LoadFromEnumerable(sameSpecialityDesciplinesScores);
DataOperationsCatalog.TrainTestData dataSplit = mlContext.Data.TrainTestSplit(trainingData, testFraction: 0.2);
IDataView trainSet = dataSplit.TrainSet;
IDataView testSet = dataSplit.TestSet;
```

Рис. 3.6 – Ініціалізація даних для рекомендаційної моделі

Наступний крок в програмі – це ініціалізації параметрів для алгоритму матричної факторизації (рис 3.7), основною метою якої є зменшення розміру матриці шляхом пошуку латентних факторів, які можуть бути використані для передбачення відсутніх значень. Клас такої моделі містить чимало параметрів для конфігурації навчання, серед яких використовуються наступні [188]:

- MatrixColumnIndexColumnName – ім’я стовпця в наборі даних, який містить індекси для стовпців матриці. В нашому випадку буде мати значення StudentIdEncoded, а отже в кожному стовпці будуть знаходитися ідентифікатори здобувачів.
- MatrixRowIndexColumnName – ім’я стовпця в наборі даних, який містить індекси для рядків матриці. В нашому випадку буде мати значення CourseIdEncoded, а отже в кожному рядку будуть знаходитися ідентифікатори

дисциплін.

- LabelColumnName – ім’я стовпця, що містить значення оцінки або взаємодії між користувачем і об'єктом. У нашому випадку це оцінка здобувача для дисципліни.
- NumberOfIterations – кількість ітерацій для навчання моделі. Вказано значення 10, а отже алгоритм матричної факторизації буде виконувати 10 циклів роботи, щоб оптимізувати фактори та мінімізувати похибку між передбаченнями та реальними значеннями.
- ApproximationRank – кількість факторів, які будуть використані для апроксимації оригінальних оцінок або взаємодій у матриці. Велика кількість факторів можуть підвищити точність моделі, але й можуть також привести до переобучення, якщо даних недостатньо.
- Lambda – параметр L2-регуляризації, який виступає алгоритмічною ланкою корекції моделі. Він необхідний для нівелювання зміщеності оцінок в умовах розріджених даних (приміром, коли студенти мають мало оцінок. Вищі значення Lambda штрафують модель за надмірну підгонку під випадкові викиди, що дозволяє отримати репрезентативний прогноз навіть при неповних вхідних даних.

```
var options = new MatrixFactorizationTrainer.Options {
    MatrixColumnIndexColumnName = "StudentIdEncoded",
    MatrixRowIndexColumnName = "CourseIdEncoded",
    LabelColumnName = "Label",
    NumberOfIterations = 10,
    ApproximationRank = 34,
    Lambda = 0.76
};
```

Рис. 3.7 – Визначення параметрів матричної факторизації

Значення параметрів NumberOfIterations, ApproximationRank та Lambda визначаються шляхом підбору комбінації значення, при яких значення RMSE на тестовій вибірці буде найменше [189]. Саме автоматизований розрахунок RMSE у програмному коді виступає моделлю оцінки похибки, що дозволяє контролювати точність системи перед видачею рекомендацій. Алгоритм такого

розрахунку оптимальних параметрів для матричної факторизації продемонстровано в Додатку Д.

Наступний крок це, власне, саме створення і тренування рекомендаційної моделі за допомогою алгоритму матричної факторизації. Тут використовується ланцюжок трансформерів і тренерів, які застосовуються до даних послідовно. В даному випадку, ланцюжок складається з кількох етапів перетворення даних, й тренера (MatrixFactorization(options)) для матричної факторизації (рис. 3.8).

В програмі використовується метод MapValueToKey, який перетворює стовпці з категоріальними значеннями в числові ключі. Робиться це через те, що алгоритм працює з індексами (цілі числа), а отже необхідно перетворити всі вхідні дані у числові значення. І якщо для значень властивостей ідентифікаторів здобувача та дисципліни все просто, так як вони мають числовий тип даних, то для назви дисципліни, яка має рядковий тип даних, необхідно використати метод FeaturizeText, який перетворює текстові дані в числові ознаки.

Наступний етап, це застосування створеного ланцюжку трансформерів та тренера до тренувального набору даних, внаслідок чого буде створено модель, яка може робити передбачення на нових даних.

```
EstimatorChain<MatrixFactorizationPredictionTransformer> pipeline =
  mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "StudentIdEncoded", inputColumnName:
  "StudentId")
            .Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "CourseIdEncoded",
              inputColumnName: "CourseId"))
            .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnName: "CourseTypeFeaturized",
              inputColumnName: "CourseType"))
            .Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));
TransformerChain<MatrixFactorizationPredictionTransformer> model = pipeline.Fit(trainSet);
IDataView predictions = model.Transform(testSet);
```

Рис. 3.8 – Визначення параметрів матричної факторизації

Після створення моделі, яка здатна давати передбачення, є необхідність в оцінюванні точності цих передбачень. Це досягається за рахунок оцінки кореня середньоквадратичної помилки, що розраховується в методі Evaluate (рис. 3.9)

```
RegressionMetrics metrics = mlContext.Regression.Evaluate(predictions, labelColumnName: "Label",
  scoreColumnName: "Score");
Console.WriteLine($"Root Mean Squared Error: {metrics.RootMeanSquaredError}");
```

Рис. 3.9 – Розрахунок кореня середньоквадратичної помилки

Варто зазначити, що система оцінює значення кореня середньоквадратичної помилки під час кожної ітерації циклу навчання, а отримане значення із metrics.RootMeanSquaredError, є середнім значенням для всього процесу (рис. 3.10).

```
iter       tr_rmse          obj
   0       26.2862   1.2533e+07
   1       14.2162   4.7278e+06
   2       11.2625   3.5456e+06
   3       10.6184   3.3295e+06
   4       10.2070   3.2125e+06
   5        9.8653   3.1205e+06
   6        9.5180   3.0269e+06
   7        9.1909   2.9471e+06
   8        8.8774   2.8770e+06
   9        8.5862   2.8050e+06
Root Mean Squared Error: 10.620811711100599
```

Рис. 3.10 – Приклад розрахунку кореня середньоквадратичної помилки

Наступним етапом є створення об’єкта, який дозволяє робити індивідуальні передбачення для окремих записів на основі натренованої моделі (рис. 3.11). Саме цей об’єкт дозволяє робити передбачення для окремого здобувача, на основі його характеристик, таких як інформація про здобувача та його взаємозв’язок із дисципліною, та отримувати рекомендації щодо дисциплін, які можуть бути найбільш відповідними до його уподобань та можливостей.

Приймає в себе тип вхідних даних (у цьому випадку це StudentDiscipline), які подаються до моделі для передбачення, тип вихідних даних (CourseRecommendation), який очікується отримати від моделі після передбачення, а також приймає в якості параметра натреновану модель, яка використовується для майбутніх передбачень.

```
PredictionEngine<StudentDiscipline, CourseRecommendation> predictionEngine =
    mlContext.Model.CreatePredictionEngine<StudentDiscipline, CourseRecommendation>(model);
```

Рис. 3.11 – Створення об’єкта для передбачень

Після того як рекомендаційну систему повністю налаштовано та вона готова до надання передбачень, програма витягує необхідні дані із бази даних (рис. 3.12). Це дані, які містять інформацію про доступні до вибору факультетські або університетські дисципліни, які здобувач ще не проходив. Таким чином, внаслідок цих декількох операцій по роботі з даними, система отримує список дисциплін, для яких потрібно спробувати дати передбачення по ймовірністю успішного проходження дисципліни.

```
List<Discipline> facultyDisciplines = _studentDesciplineRepository.GetFutureStudentFacultyDisciplines
  (actualStudentId);
List<Discipline> notSelectedfacultyDisciplines = GetNotPassedNearDisciplines(facultyDisciplines,
  actualStudentId, MandatoryEnum.FacultyOptional);
List<Discipline> universityDisciplines = _studentDesciplineRepository.GetFutureStudentUniversityDisciplines
  (actualStudentId);
List<Discipline> notSelecteduniversityDisciplines = GetNotPassedNearDisciplines(universityDisciplines,
  actualStudentId, MandatoryEnum.UniversityOptional);
List<Discipline> notSelectedDisciplines = [.. notSelectedfacultyDisciplines, ..
  notSelecteduniversityDisciplines];
```

Рис. 3.12 – Обробка списку дисциплін, необхідного для надання рекомендацій

Фінальним кроком, який стосується безпосередньо надання рекомендації, є спроба передбачити оцінку здобувача з певної дисципліни. Для цього системі потрібен унікальний ідентифікатор здобувача, список дисциплін, та об’єкт, який здатний давати передбачення (рис. 3.13). Кожна з дисциплін буде опрацьована в цьому алгоритмі, адже наперед не зрозуміло, яка з них може бути найвідповіднішою до можливостей та уподобань здобувача.

Головним в цій операції є метод Predict, який приймає вхідний зразок даних і використовує натреновану модель для отримання певного передбачення. Для цього випадку він буде передбачати рекомендації для здобувача, на основі його взаємодії з дисциплінаами, які були використані для навчання моделі.

На завершальній стадії своєї роботи, алгоритм віддає відсортований в порядку спадання список рекомендованих дисциплін, де на першій позиції знаходиться дисципліна із найбільшою ймовірністю успішного проходження, а на останній – той, який має найменшу ймовірність на проходження, або той,

для якого не вистачило вхідних даних для генерації передбачення.

```
var courses = (from notSelectedDiscipline in notSelectedDisciplines
               select (notSelectedDiscipline.Id, notSelectedDiscipline.DisciplineName));
var dict = new Dictionary<int, float>();
foreach (var (courseId, courseType) in courses) {
    var input = new StudentDiscipline {
        StudentId = studentId,
        CourseId = courseId,
        CourseType = courseType
    };
    var prediction = predictionEngine.Predict(input);
    Console.WriteLine($"Predicted preference for student {studentId} for course {courseId} ({courseType}):
      {prediction.Score}");
    dict[courseId] = float.IsNaN(prediction.Score) ? 0 : prediction.Score;
}
return dict.OrderByDescending(x => x.Value);
```

Рис. 3.13 – Генерація передбачення для конкретного здобувача

Фінальним етапом роботи системи є конвертація отриманих даних із рекомендаційної системи для відповідності типу даних у всій системі та маркування дисциплін, якщо здобувачем вони вже були обрані (рис. 3.14).

```
List<SelectedDiscipline> disciplinesToSelect = GetDisciplinesToSelect(predictionsForNewSubjects,
  notSelectedDisciplines);
MarkSelectedDesciplines(actualStudentId, disciplinesToSelect);
return disciplinesToSelect;
```

Рис. 3.14 – Обробка списку дисциплін, необхідного для надання рекомендацій

Позначення вже обраних дисциплін необхідне і такі дані запам’ятовуються в системі через те, що здобувач має можливість здійснювати перервибір бажаних дисциплін скільки завгодно разів. Це є однією із численних переваг автоматизації даного процесу, так як раніше здобувач мав можливість зробити свій вибір один єдиний раз.

**Висновки до третього розділу**

У розділі вирішено задачу вибору методів та програмної реалізації рекомендаційної системи.

Основні наукові та практичні результати розділу полягають у наступному:

1. Проведено порівняльний аналіз методів фільтрації за змістом (Content-based), колаборативної фільтрації (Collaborative Filtering) та гібридних підходів та обґрунтовано, що для задачі побудови індивідуального навчального плану найбільш ефективним є метод колаборативної фільтрації, оскільки він дозволяє врахувати колективний досвід здобувачів та прогнозувати успішність на основі історичних даних схожих користувачів, уникаючи проблеми «бульбашки фільтрів».

2. Розроблено метод подолання типових проблем рекомендаційних систем в освітньому середовищі, а саме систематизовано виклики («холодний старт», розрідженість даних, масштабування) та запропоновано архітектурні рішення для їх нівелювання: для вирішення проблеми масштабування впроваджено обмеження вибірки даних (вікно в 3 роки) та фільтрацію за спеціальністю. Для проблеми «холодного старту» запропоновано гібридний механізм із використанням анкетування першокурсників. Забезпечено захист від атак на систему шляхом валідації вхідних даних на етапі збору.

3. Створено програмний модуль рекомендаційної системи з використанням мови програмування C# та платформи ML.NET. Зокрема, реалізовано архітектуру з низькою зв’язністю через механізм Dependency Injection, виділивши логіку ML у окремий простір імен. Також імплементовано алгоритм матричної факторизації (MatrixFactorizationTrainer) з налаштуванням гіперпараметрів: кількість ітерацій (NumberOfIterations), ранг апроксимації (ApproximationRank) та коефіцієнт регуляризації (Lambda).

4. Створено діючий прототип системи, в якому розроблено конвеєр обробки даних (Pipeline), який включає етапи: завантаження даних (LoadFromEnumerable), трансформація категоріальних ознак у числові ключі (MapValueToKey), тренування моделі та генерація прогнозів (Predict). Для оцінки якості роботи прототипу інтегровано метрику RMSE (Root Mean Squared Error), що дозволяє в автоматичному режимі контролювати точність передбачень оцінок.

5. Забезпечено інтеграцію з інформаційною системою університету.

Розроблений модуль підтримує роботу з реальними даними про успішність здобувачів, забезпечуючи формування списку рекомендованих дисциплін у форматі CourseRecommendation, що підтверджує практичну цінність роботи та можливість її впровадження в реальний освітній процес.

# РОЗДІЛ 4
# ПРОТОТИП РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ФОРМУВАННЯ ІНДИВІДУАЛЬНОЇ ОСВІТНЬОЇ СИСТЕМИ

## 4.1 Аналіз функціоналу рекомендаційної системи для формування індивідуальної освітньої системи на основі 3-рівневої розподіленої архітектури

Згідно з розробленою логікою функціонування системи, процес формування індивідуальної освітньої траєкторії було докорінно трансформовано та перенесено безпосередньо в інтегроване середовище електронного кабінету здобувача освіти. На відміну від застарілого підходу, що базувався на використанні розрізнених Google-форм, які вимагали ручної обробки результатів та не мали прямого зв’язку з базою даних університету, запропоноване рішення забезпечує автоматизацію та цілісність даних.

Новий підхід надає здобувачам гнучкість у прийнятті рішень: вони отримують можливість редагувати свій вибір необмежену кількість разів протягом встановленого періоду кампанії вибору, що дозволяє уникнути помилок, характерних для одноразового анкетування. З адміністративної точки зору, система забезпечує централізований збір та збереження даних, що дозволяє деканатам отримувати актуальну статистику в режимі реального часу без необхідності консолідації зовнішніх файлів.

Крім того, розширені можливості інтерфейсу дозволяють наситити форму вибору додатковим контекстом: окрім назв дисциплін, система може відображати силабуси, відеопрезентації курсів, профілі викладачів та пререквізити. Найважливішою архітектурною перевагою є можливість безшовної інтеграції стороннього функціоналу, зокрема модулів інтелектуальної підтримки та рекомендаційних алгоритмів, впровадження яких було технічно неможливим у межах обмеженого інструментарію Google-форм.

Процес вибору дисциплін відбувається наступним чином:

1. Система визначає чи є для цього здобувача доступний вибір

дисциплін і якщо так, то показує йому таку опцію в меню доступних дій.

2. При відкритті сторінки вибору дисциплін система показує здобувачеві список ключових слів, які є характерними для його фаху. Такі дані поки що не враховуються алгоритмами рекомендаційної системи і зберігаються для того, щоб провести залежність вибраних ключових слів і обраних дисциплін здобувачами. Це може слугувати додатковим фактором покращення алгоритмів передбачення в майбутньому.

3. Система просить вказати інформацію щодо поточного місця роботи, бажаної посади\роботи й хобі. Такі чинники також можуть впливати на вибір здобувачів, так як залученість (або бажання бути залученим) до тієї чи іншої сфери також може мати свою вагу при виборі дисциплін. Такі дані також зберігаються для подальшого покращення і на даний момент не беруться в розрахунок

4. Система генерує список дисциплін, які здобувач має можливість обрати. Такий вибір може поділений на факультетські дисципліни або загальноуніверситетські та бути розмежований на два навчальні семестри.

5. Система за допомогою графічного інтерфейсу та текстових показників дає здобувачу інформацію про те, які дисципліни рекомендуються йому до вибору найбільшою мірою. Якщо ж система не знайшла достатньої кількості даних для надання рекомендацій, то вся вищезгадана додаткова інформація буде прихована, а дисципліни будуть показані в довільному порядку.

6. Після збереження результатів вибору здобувачем, якщо він отримував рекомендацію дисциплін з боку системи, для нього буде виведено додаткову сторінку, на які здобувач за бажанням може залишити свій відгук щодо роботи системи.

7. Здобувач може повторно перейти на сторінку вибору дисциплін й змінити свій вибір. В такому випадку вибрати теги і залишити коментар відносно ефективності системи він зможе тільки в тому випадку, якщо не робив цього раніше.

Даний приклад процесу дій описано з боку системи на діаграмі послідовності вибору дисциплін (рис. 4.1).

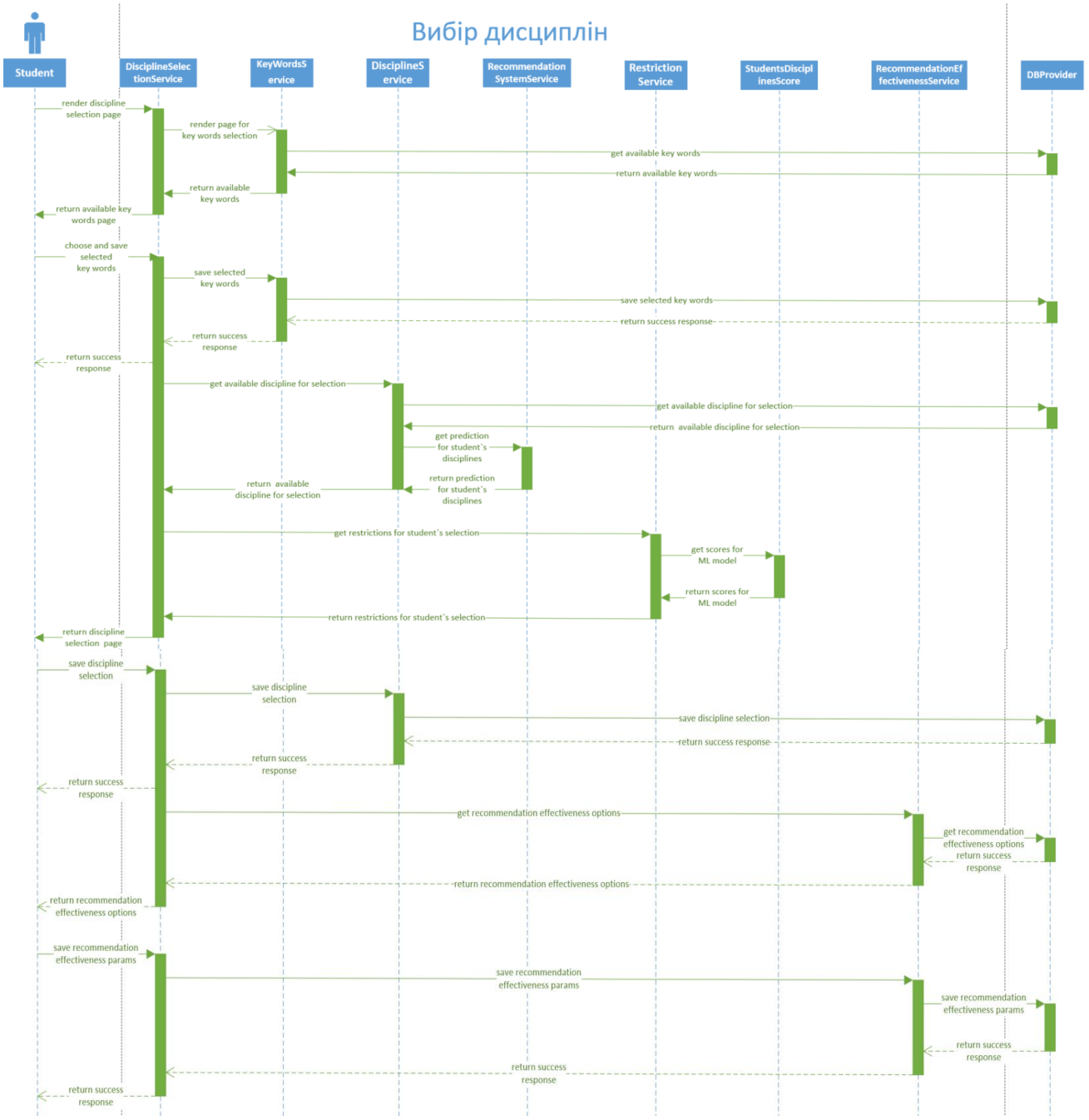


Рис. 4.1 – Діаграма послідовності процесу вибору дисциплін та отримання рекомендації щодо них

В процесі ж роботи над дослідженням система розширелась і тепер, окрім трьох вже існуючих сервісів, існують додаткові два сервіси (Особистий кабінет здобувача освіти та Електронний розклад) та чат-бот (рис. 4.2). Розглянемо детальніше:

1) Особистий кабінет здобувача освіти – це програмний модуль, який було спеціально розроблено для нашого дослідження, а також який підтримує процеси підтримки навчання та комунікації для здобувача.

2) Електронний розклад – власна розробка університету, для цифровізації розкладу занять в закладі. Містить інформацію про час та місце проведення занять для конкретної групи. Так як це окремий сервіс і розробка його ведеться паралельно до особистого кабінету здобувача освіти, то наша система в себе виводить на окрему сторінку вбудовану версію даного розкладу. Таке рішення дозволяє здобувачам передивлятись інформацію про розклад навчання не переходячи на інші домени. Окрім того, система індифікуючи належність до того чи іншого факультету, може більш точніше пропонувати йому розклад для перегляду.

3) Чат-бот першокурсника – чат-бот, написаний на популярній платформі Telegram, та який надає основну інформацію про університет, умови вступу та всю іншу корисну інформацію, яка необхідна для першокурсника. Розміщений на одному й тому ж сервісі, що і особистий кабінет здобувача освіти.

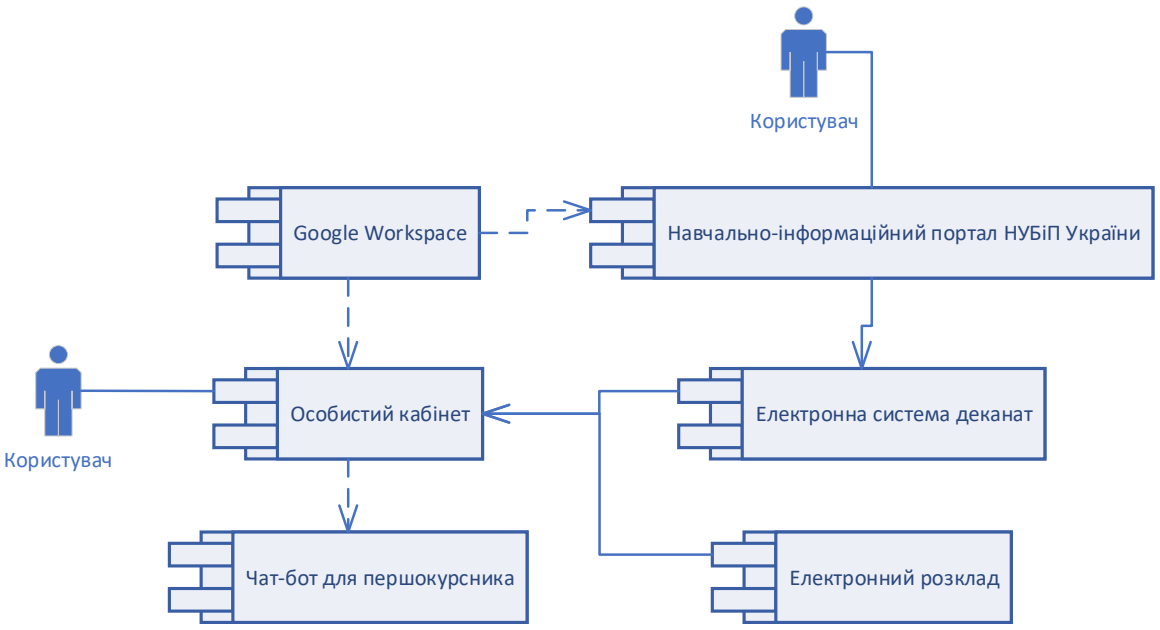


Рис. 4.2 – Архітектура взаємодії компонентів екосистеми цифрових сервісів університету

На етапі розробки таких рекомендаційних систем постає чимало важливих питань, серед яких одним із ключових виступає вибір типу платформи та архітектури. Залежно від того, чи буде система реалізована у вигляді мобільного, консольного чи десктопного застосунку, або ж це буде реалізація у вигляді вебсайту, а також як буде побудована її архітектура, можуть змінюватися вимоги до функціональності, масштабованості та зручності використання. Платформа та архітектура системи повинні як відповідати потребам кінцевого користувача, так і забезпечувати належну продуктивність разом з безпекою при обробці великої кількості даних.

Сучасні рекомендаційні системи потребують ефективного і зручного підходу до проєктування та розробки, особливо якщо брати до уваги контекст вибору дисциплін для здобувачів вищої освіти. Основним завданням тут виступає створення такого програмного продукту, який буде не лише функціональним, а й зручним для користувачів, враховуючи їх технічні можливості та потреби. Проте, вибір найбільш оптимальної платформи та способу побудови архітектури системи є складною задачею, оскільки кожен тип має свої переваги та недоліки в залежності від вимог користувачів, технічних чи фінансових можливостей, компетентностей фахівців або ж інших унікальних факторів, які характерні для тої чи іншої задачі. Неправильний вибір платформи або ж архітектури може призвести до проблем з масштабованістю, оновленням, створенням нового функціоналу, тестуванням, швидкодією, зручністю використання або ж взагалі довготривалості життя на ринку та спроможності конкурувати із аналогами.

Таким чином, вибір правильної платформи та архітектури для системи рекомендацій є складним багатофакторною задачею, яка потребує детального аналізу. Вивчення різних підходів та порівняння їх переваг чи недоліків є важливим кроком до розробки ефективної системи для здобувачів вищих навчальних закладів. Тому нашою метою є розробка рекомендацій для вибору оптимальної платформи та підходу до побудови для розробки системи рекомендацій для вибору дисциплін для здобувачів закладів вищої освіти. На

основі аналізу вимог до функціональності системи, зручності користування та технічних аспектів ми розглянемо ключові фактори, що визначають ефективність вибору кожної з платформ та архітектур.

Всім відома платформа Netflix використовує доволі складну та масштабовану архітектуру для обробки даних і створення рекомендацій [190]. Так як їх корпорація є одним із лідерів на стрімінговому ринку, а відповідно їх платформа містить значну кількість функціоналу та працює із велими даними, то з метою розділити всю систему на маленькі незалежні сервіси, які обробляють специфічні задачі, Netflix використовує мікросервісну архітектуру.

Spotify – це ще один яскравий приклад гіганта в своїй ніші, який використовує рекомендаційну систему для підбору найбільш відповідної музики для користувача. Він використовує численні інструменти для аналізу, хмарні технології для зберігання та багаторівневу архітектуру. Побудована їх система як для вебсайтів так і для мобільних пристроїв, тому це чудовий приклад, коли система здатна задовільняти потреби в пошуку відповідних продуктів для користувачів різних платформ [191].

Платформа Youtube [192] націлена на те, щоб допомогти користувачам знаходити контент, який їм цікаво дивитися, та тим самим змушуючи проводити більше часу саме на їх платформі. Їх рекомендаційна система працює як через браузери в персональних комп’ютерах, так через мобільні застосунки, та навіть через телевізори, що є підкреслює широкий спектр застосування рекомендаційної системи на різних платформах.

У своїй статті [193] автор розглядає важливість не тільки створення алгоритмів рекомендаційної системи, а й про побудову правильної та вискокласної архітектури всього проекту, що може дозволити ефективно співпрацювати з різними командами та розробляти надійні системи, які відповідають вимогам реальних часу. В статті демонструється приклад такої реалізації на Node.js із допомогою React.

В своїй праці автори [194] описують створену ними систему та окремо представляють її архітектуру. Як можемо бачити, реалізацію системи було

вирішено створити орієнтуючись на вебплатформи, а основні потужності розрахунків рекомендаційної системи було вирішено розмістити окремо від основної логіки, яка міститься на сервері.

У своїй роботі [195] автор описує рекомендаційну систему формування команд виконавців з відповідними фаховими компетентностями та демонструє приклад реалізації програми на мові Java та на вебплатформі. Також він демонструє додатки, де наведені приклади роботи алгоритмів на мові Python та їх результат продемонстровано в консольному застосунку.

А інший молодий вчений, який досліджував тему «Нейромережні методи створення рекомендаційних систем для інтерактивного мистецтва з використанням доповненої реальності» [196] розглядає написання свого проєкту на мові Python із використанням технологій AR. Це яскравий приклад того, що обмеження списку платформ, де можна задіяти рекомендаційну систему, не існує, а все залежить тільки від навичок тієї чи іншої людини та креативності мислення.

Перед тим як почати розробку рекомендаційної системи для вибору дисциплін для здобувачів вищих навчальних закладів при формуванні індивідуального навчального плану, була необхідність у виборі правильного вектору створення системи, який полягав у вирішенні двох ключових питань: архітектурний підхід у побудові всієї системи та вибір платформи, яка б дала нам найбільше користі в плані ефективності, масштабованості та подальшому супроводі.

Розглянемо спочатку думки та дискусії, які точилися навколо вибору платформи. Даний вибір належало зробити серед чотирьох типів: вебсайт, десктопний застосунок, мобільна платформа або ж консольний застосунок.

Почнемо огляд із найбільш очевидних типів, як наприклад **Консольний застосунок**. Цей спосіб є найлегшим для програмної реалізації, адже не потребує такої кількості затрачених ресурсів, якби того потребували всі інші типи. Адже в нас немає необхідності в побудові користувацького інтерфейсу (не кажу чи про його адаптованість до різних платформ та продуманість з точки

зору користування). Тому таке рішення було б оптимальним, якби було занадто мало часу і нам потрібно було б зосередитися на алгоритмі. І хочі в світі індустрії інформаційних технологій завжди відчувається брак часу, розробка системі на такому типі платформи була б занадто великою жертвою на користь простоті, в той час, коли б знехтувалися фактори подальшого розвитку проекту, його доступності для широкого колу людей, зручність користування (адже, не слід забувати, що не всі користувачі вміють працювати із консольними застосунками та незручними меню, які, зазвичай, там приставлені). І фінальним фактором, який змушує дивитися в сторону інших платформ, є те, що зекономлений час все одно прийдеться витратити на допомогу простим користувачам в освоєнні такого роду програми. А це в далекій перспективі навпаки тільки збільшило б маштаб роботи над проектом.

Менш очевидним у розгляді є **Десктопний застосунок.** Це варіант, який має право на існування, що доводять численні реалізації різного роду функціональності саме на такого типу платформах. Це є особливо вигідним, якщо справа йде про систему, яка б вимагала великих обсягів даних і обчислень, що потребувало б високої продуктивності. Однак, якщо акцентувати увагу на нашу цільову аудиторію (здобувачів і робітників університету) і потребу в доступності системи, то десктопний застосунок виявляється менш зручним варіантом, аджее потребує попереднього завантаження та персонального комп’ютера. Здобувачі не всі ж мають комп’ютер та ще менша частку із них може користуватися ним при відключеннях електроенергії. Окремий головний біль, який вже більше стосується розробників такого типу застосукнів, це розробка системи під різні операційні системи, адже необхідно надати функціонал для користувачів Windows, MacOS та Linux. Окрім того, подальший супровід та оновлення функціоналу в цих системах є складнішим, ніж на тому ж вебсайті.

Інший доволі дискусійний варіант – це **Мобільний застосунок.** Цей тип платформи завойовує з року в рік все більший сегмент ринку [197] і зараз вже важко собі уявити буденного користувача без смартфону, особливо, якщо брати

в розрахунок проактивну молодь таку як здобувачі освіти. Це зумовлено зручністю користування даним девайсом, адже мобільний телефон завжди під рукою, і, за наявності стабільного інтернет з’єднання, його користувач може отримати швидкий доступ до будь-яких послуг, комунікації або інформації, яка може бути критично важливою в будь-який момент часу. Чимало компаній, які процвітають в своїй сфері та вже мають великі та багатофункціональні вебсайти, із завзятістю вкладають ресурси в розвиток мобільних застосунків для розширення коло потенційних покупців. І, якщо і необхідність мобільного застосунка для університету виглядає необхідною опцією, то такий варіант не виглядає оптимальним на початковому етапі створення системи, адже розробка такого роду вимагає значних витрат часу і ресурсів на підтримку версій для різних операційних систем. Крім того, для необхідності користуватися певного роду функціоналом, користувач повинен буде завантажити застосунок, в той час, коли у варіанті із вебсайтом потрібно просто перейти по певному посиланню. Також, не слід систему розробляти опираючись тільки на функціонал рекомендаційної системи, адже якщо є змога створити систему, яка надаватиме різні послуги для здобувачів освіти та працівників університету, то потрібно обрати плаформу, яка б могла дати можливість в подальшому підключати додаткові модулі функціональності. Така функціональність може потребувати, приміром, великий іконний інтерфейс, що важко реалізувати в мобільному застосунку.

І, нарешті, **Вебсайт** – це платформа, яка є найбільш універсальною для сучасних систем, так як доступ до вебплатформи можна отримати з будь-якого пристрою, який має підключення до Інтернету та який не потребує необхідності в додатковій установці застосунку на персональний пристрій. Такий тип платформи забезпечує легкість оновлень та інтеграцій, адже всі зміни можна впроваджувати на сервері, без необхідності оновлення клієнтських застосунків. Також важливим фактором виступає можливість інтеграції із вже існуючими системами, що можна використати для аутентифікації, збору даних необхідних для аналізу і т.д. З вищерозглянутого можна зробити висновок, що для випадку,

коли потрібно розробити рекомендаційну систему вибору дисциплін для здобувача (а в подальшому й інший функціонал), то найкращим варіантом є розробка вебсайту (рис. 4.3).

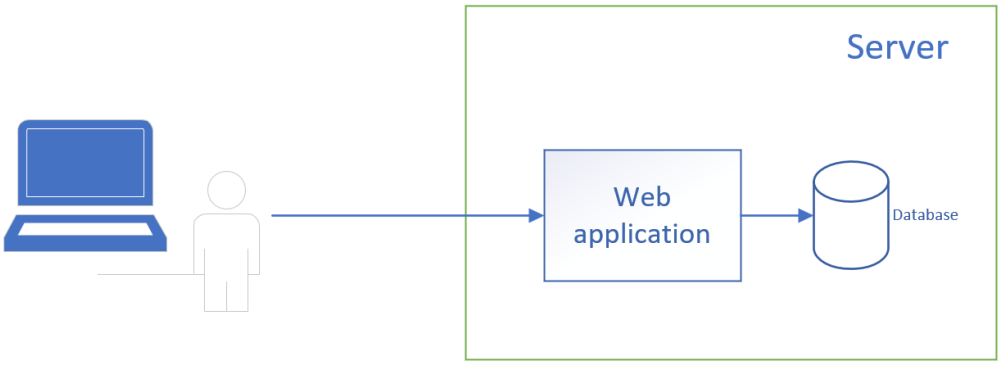


Рис. 4.3 – Вебсервісна архітектура системи

Іншою важливою проблематикою було обрання типу архітектури, на основі принципів якої буде реалізовано нашу систему. Найбільшу увагу привернули такі рішення, як трирівнева архітектура та мікросервісна. І хоча для реалізації рекомендаційних систем також можуть використовуватися подієво-орієнтована архітектура та Big Data архітектура, вони нам не підходять, так як перша – це архітектурний стиль програмування, який використовує події для ініціювання та обміну даними між окремими службами [198] і це не дуже підходить під вимоги нашої задачі, а другий – будується для ефективного збору, зберігання, обробки та аналізу величезних обсягів даних [199], а в нашої системи будуть не такі великі обсяги даних.

Найбільш очевидним є рішення обрати трирівневу архітектуру. Такий спосіб реалізації вже є класичним у наш час та часто зустрічається в системах в тому чи іншому форматі. Включаючи три рівні (клієнтський інтерфейс, бізнес-логіка та зберігання даних) [200], він дозволяє чітко розмежовувати функціональність і забезпечувати високу продуктивність. Такий вибір є класичним, коли йдеться про створення стабільної та масштабованої системи, де кожен із рівнів може розвиватися, оновлюватися та розширюватися в незалежності від інших.

Проте, трирівнева модель не є достатньо гнучкою для інтеграції із іншими системами, а тому нам прийшлось би «зашивати» рекомендаційну систему в середину сайту, що в подальшому могло б дати негативні наслідки. Це б викликало в майбутньому проблеми із супроводом та тестуванням, адже, щоб перевірити чи скористатися функціоналом рекомендаційної системи, довелося б запускати весь сайт, а не його окремий модуль. А сама рекомендаційна система є галуззю, яка доволі сильно відрізняється від ніші розробки вебсайтів, то може потребувати спеціаліста із іншими компетентностями, який може бути фахівцем у своїй справі, але не в аспекті розробки програмного забезпечення для сайтів.

Саме тому для підключення рекомендаційної системи до вебсайту було вибрано мікросервісну архітектуру. Така система проектується для більшої масштабованості або у випадку, коли потрібні високі вимоги до продуктивності [201]. Це дозволить ізольовано розгортати та масштабувати програмний модуль, який відповідає за алгоритми всередині, та дозволить відкинути необхідність запускати весь сайт. І хоча мікросервісна архітектура вимагає більш складного управління та моніторингу, так як створює додаткові труднощі в підтримці та інтеграції мікросервісів, особливо, коли йдеться про високі вимоги до продуктивності, такий крок виглядає найбільш виправданим на початку створення системи, адже початковий алгоритм рекомендаційної системи буде відносно простим та буде збільшуватися разом із збільшенням потреб до системи.

Приклад реалізації верхнерівневої комбінованої архітектури, яка складається з трирівневої архітектури для самого сайту та мікросервісної архітектури для підключення рекомендаційної системи можна побачити на рисунку 4.4.

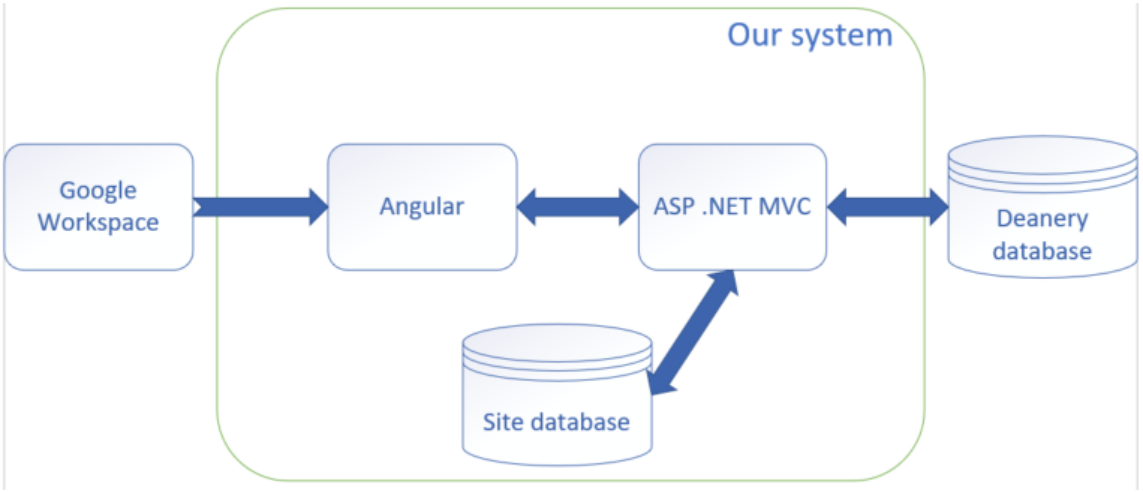


Рис. 4.4 – Верхнерівнева архітектура вебсайту

В клієнтському рівні ми в першу чергу вирішили акцентувати увагу на модулі, який надасть можливість обирати для здобувачів дисципліни (StudentSelectedDisciplinesComponent) та отримувати рекомендації для цих дисциплін через відповідні сервіси. Для реалізації клієнтського рівня було обрано фреймворк Angular, який в свою чергу також надає можливість розділити даний рівень на три інші підрівні, що також надає свої плюси в плані підтримки, оновлення та маштабованості (рис. 4.5).

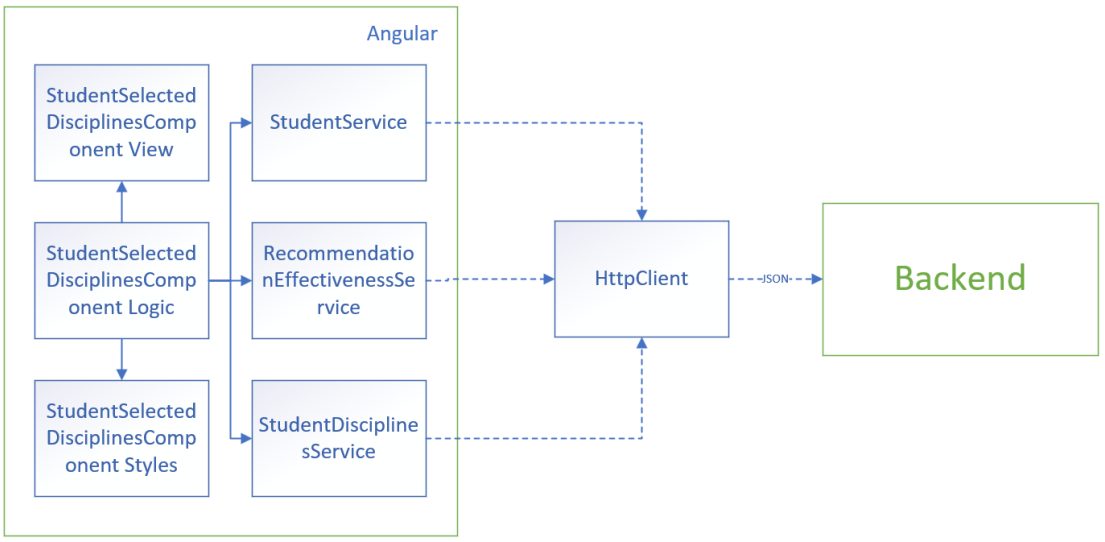


Рис. 4.5 – Клієнтський рівень вебсайту

Рівень бізнес-логіки – це серверна частина, було реалізовано на ASP.NET Core, що дозволяє розмітити потужності сайта на серверах із різними типом операційної системи, а також який дозволяє розділити даний рівень на інші три

підрівні, які спілкуються із клієнтом (StudentDisciplineController), містять моделі даних та сервіси (StudentService і т.д.), які ці моделі даних беруть із бази даних (рис. 4.6).

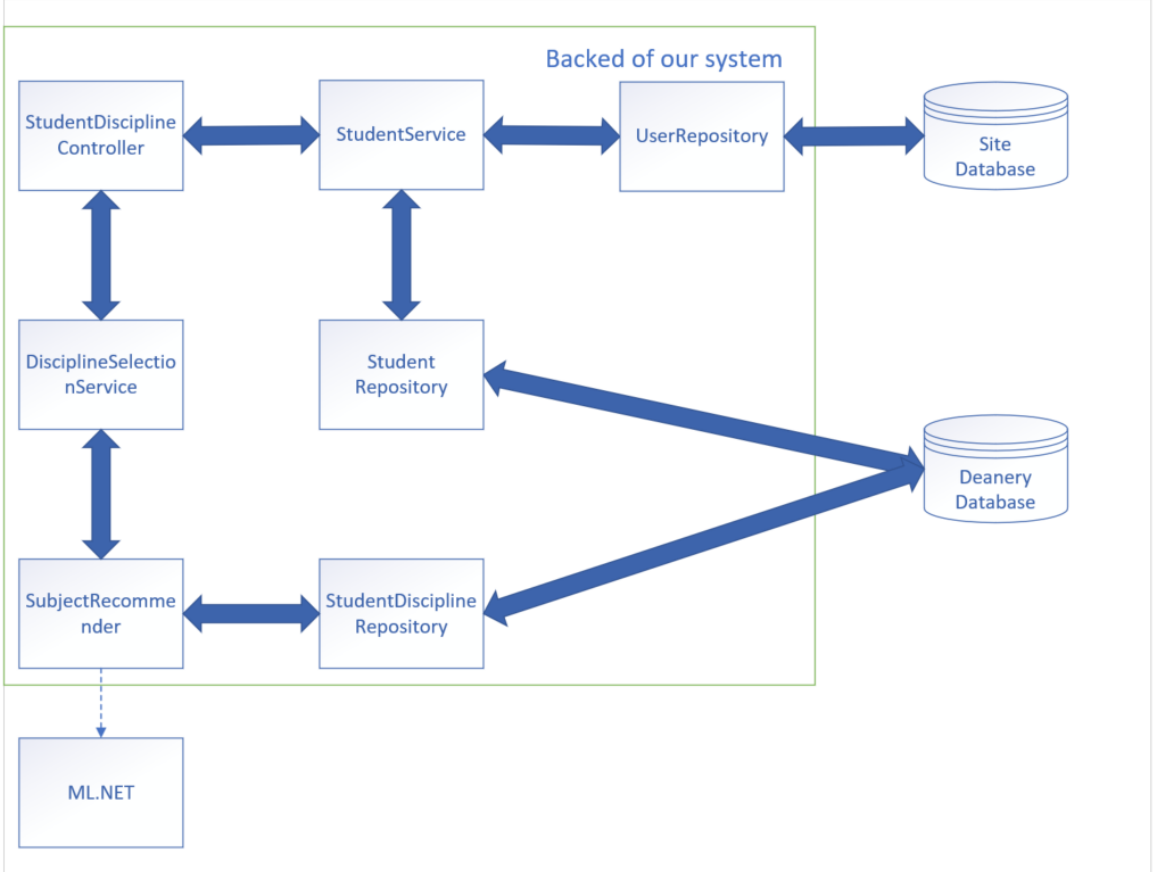


Рис. 4.6 – Рівень бізнес-логіки вебсайту

### *4.1.1 Аналіз та проєктування бази даних у контексті розробленої системи*

Розробляючи вебсервіс для комунікації здобувача та закладу вищої освіти, ми стикнулися із питаннями щодо нюансів реалізації бази даних для нашого проєкту. Серед численних версій систем управління базами даних, серед яких MSSQL, MySQL, PostgreSQL, Oracle Database і т.д., нам потрібно було обрати ту, яка найкраще підходила б до наших умов. Головні вимоги до бази даних були реляційність (бази даних, де таблиці поєднані між собою за допомогою зв’язків, що допомагає структувати архітектуру бази даних, спосіб усунення надлишкових даних та забезпечення їх більш ефективного зберігання

та обробки [202]), безкоштовне використання та спорідненість платформ. Під дані вимоги підходять PostgreSQL та MySQL, але фактор того, що існуюча система деканату написана на MySQL, у висновку, зіграв вирішальну роль на користь останньої. Дана система управління є широкопоширена із відкритим кодом система управління базами даних, яка має власний інтерфейс та може бути встановлена на більшість операційних систем [203].

Схема бази даних особистого кабінету представлена на рисунку 4.7.

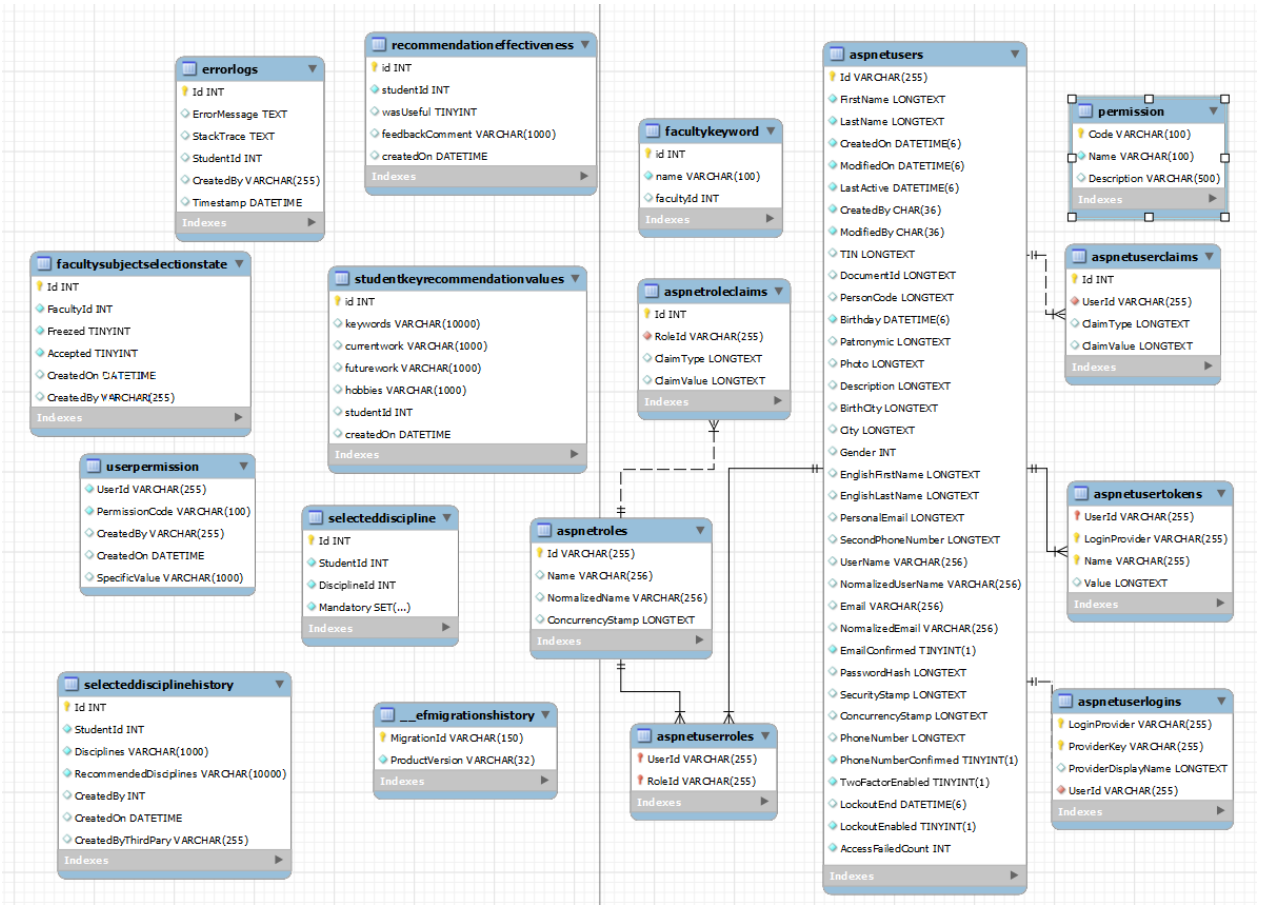


Рис. 4.7 – Схема бази даних особистого кабінету здобувача

Розглянемо детальніше подану схему. На перший погляд основним недоліком видається відсутність зв’язків між окремими таблицями. Проте, якщо врахувати, що всі таблиці, необхідні для внутрішнього функціонування системи, є взаємопов’язаними, а не пов’язаними залишаються лише ті, що спираються на сутності, дані яких імпортуються із зовнішніх джерел, ситуація постає значно логічнішою. База даних особистого кабінету здобувача природно орієнтується на сутності, пов’язані із здобувачем освіти, його індивідуальними

характеристиками та навчальною інформацією, при цьому значною мірою залежить від зовнішніх даних. Зберігання таких відомостей усередині системи призвело б до дублювання, надмірності та потенційної неузгодженості даних.

Критично важливо забезпечувати доступ до актуальної інформації з системи деканату, оскільки як здобувачі, так і викладачі повинні отримувати лише достовірні та оновлені дані. Альтернативний підхід, що передбачав би періодичне оновлення інформації всередині нашої бази, вимагав би додаткової логіки, створював би зайве навантаження на сервери та ускладнював би супровід системи. З огляду на ці фактори було прийнято рішення реалізувати архітектуру бази даних саме у такому вигляді.

Варто також зазначити, що представлена схема не є представленням всієї архітектури бази даних, так як наразі в системі існують також інші таблиці, наприклад для заповнення здобувачем індивідуальної інформації або таблиці для проведення процесу опитування. Крім того, процес розробки особистого кабінету здобувача не зупиняється і зараз, а тому архітектура постійно удосконалюється, маштабується та видозмінюється.

Також варто розглянути існуючу базу даних (рис. 4.8), яка є основним джерелом даних про здобувача, його індивідуальної та навчальної інформації і т.д.

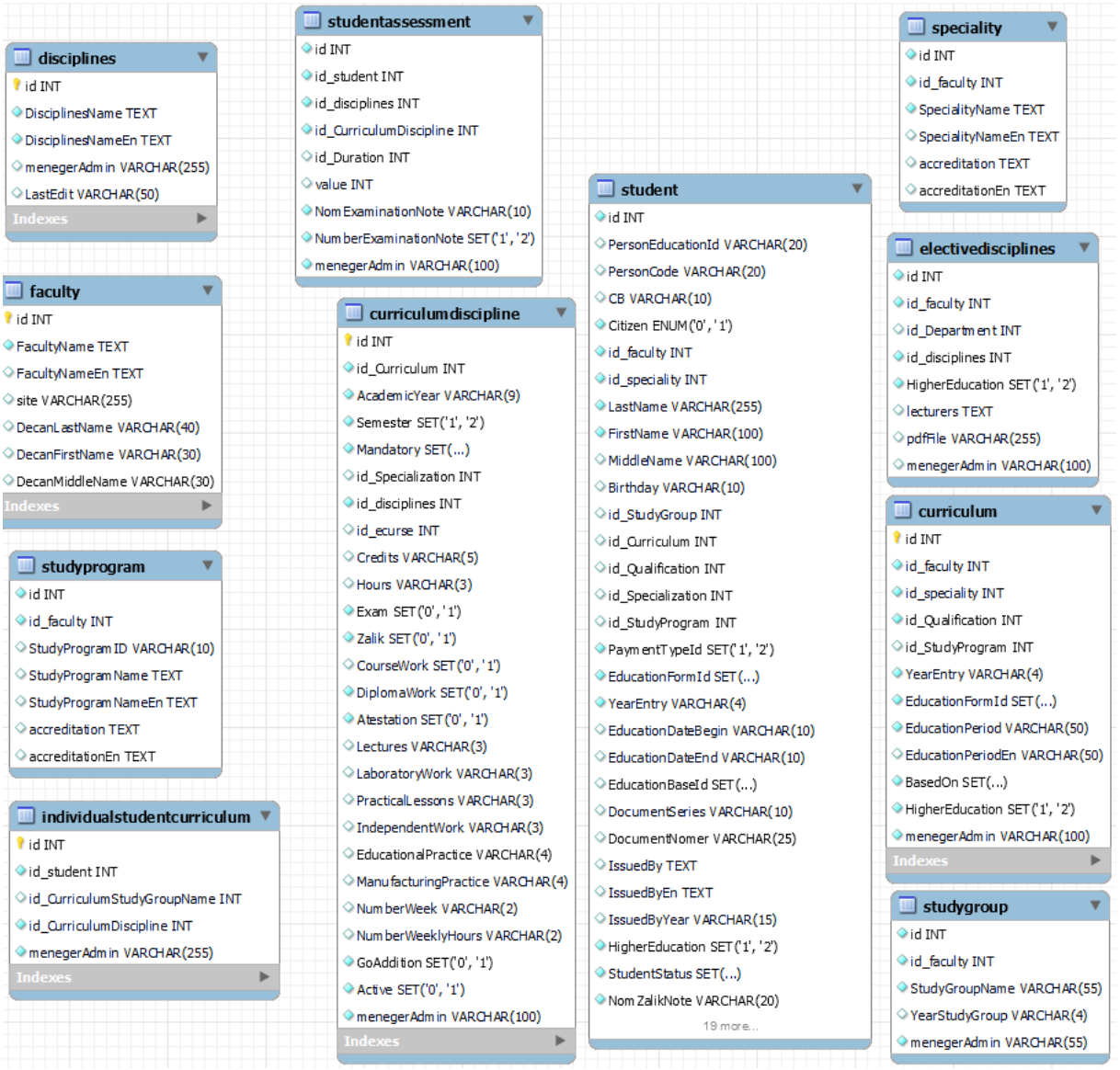


Рис. 4.8 – Схема бази даних електронного сервісу «Деканат»

Дана бази даних містить структури для зберігання інформації про здобувача, приналежність здобувача до певного навчального періоду, навчальної групи, факультету, навчальної програми, дані про всі дисципліни, за вибором та вже вибрані, а також оцінки здобувачів. Дана база даних не є приведена до третьої нормальної форми та немає чітких зв’язків між таблицями.

### *4.1.2 Опис серверної частини та обґрунтування вибору технологій для реалізації системи*

Технологія, яка обирається для серверної частини, є важливим аспектом

розробки системи, адже вона здатна вплинути на вибір технологій інших частин проєкту. Це буде зумовлено спорідністю екосистем, ефективністю поєднання, а також умовірністю того, що з відносною легкістю знайдеться розробник, який розуміється на всіх обраних технологіях. Не малу також роль відіграє сфера, для якої розробляється програмне забезпечення, адже, приміром, не має сенсу спиратися на функціональну мову програмування, яка характеризується своєю швидкодією, якщо необхідно реалізувати велику легкомаштабовану об’єктно-орієнтовану систему.

Тож, сформуємо вимоги до технології для реалізації особистого кабінету здобувача:

1) Технологія має бути такою, що підтримує масштабованість та зміну існуючого функціоналу. Архітектура нашої системи (яка багато в чому залежить від обраної технології) має бути гнучкою, стабільною та легко розширювальною. Також вона повинна підтримувати модульність, щоб в розробників була змога легше тестувати функціонал та задля зменшення вірогідності допущення помилки в неочікуваних місцях.

2) Продуктивність та надійність з великим навантаженням – це також важлива вимога до проєкту, адже системі належить працювати з великою кількістю даних різного роду та швидко виконувати різного роду операції або генерації звітів. Технології, що здатні забезпечити продуктивність високого рівня, за правило, є оптимізованими для обробки даних в асинхронних запитах, що в свою чергу забезпечує масштабованість і змогу працювати з великими об’ємами даних у реальному часі. Окрім того, надійність системи гарантує, що, навіть, при великих навантаженнях, система буде залишатися доступною та не втратить дані, що особливо важливо для університетських платформ, які зберігають критично важливу інформацію.

3) Спорідненість із офісним програмним забезпеченням – важливий аспект, так як університети часто використовують Microsoft Word, Excel та інші подібні інструменти в повсякденній діяльності для обробки даних, створення документів, складання звітів, довідок та іншої адміністративної документації.

Такі програми давно продовжують тримати лідируючі місця для багатьох освітніх закладів завдяки своїй зручності, сумісності з іншими системами та широкому функціоналу.

Тепер розглянемо технології, які можуть нам підійти:

1) Spring Boot – це фреймворк, який є частиною Spring Framework та спрощує створення вебзастосунків на Java, надаючи швидкий доступ до інформації та зменшуючи час налаштування. Дозволяє легко запускати додатки, завдяки використанню вбудованих серверів і модулів Spring Framework, Spring Boot [204]. Чудово підходить для задач, де потрібн маштабованість та стабільність роботи системи. Застосунки написані на даній технології можуть працювати на різних платформах.

2) Node.js – середовище для виконання JavaScript, що дозволяє виконувати код цієї мови програмування на серверній частині. Такий підхід дозволяє поєднувати одну мову програмування для серверної та клієнтської частини, що знижує поріг для необхідністю користування даною технологію та спрощує розробку. Node.js підходить для створення застосунків, що вимагають обробки великих файлів і великих навантажень на мережу завдяки своїй подієвій, неблокуючій і асинхронній архітектурі. [205]

3) Django – це фреймворк для розробки вебзастосунків на Python. З основних його переваг: велика кількість вбудованих функцій, інтегрована модель для аутентифікації та авторизації користувачів, обмеження запитів, взаємодія з реляційними базами даних та можливість версіонування. [206]

4) ASP.NET Core – це фреймворк, який є частиною великої екосистеми .NET Core та дозволяє створювати кросплатформенні вебзастосунки, які можуть працювати на різних операційних системах (Windows, Linux або ж Mac OS). Побудований з набору незалежних один від одного компонентів та підтримує Dependency Injection, що робить його дуже гнучким і розширюваним для різних проектів. Надає вбудовані шаблони для таки популярних фреймворків як Angular та React. [207]

Зважаючи на виділений перелік технологій, а також на описані нами вище

вимоги, можемо зробити висновок, що найкраще для базової технології нашого проєкту підійде саме ASP.NET Core. Даний фреймворк є пріоритетнішим для нас тому що:

1) підтримує кросплатформенність, а отже в нас з'являється можливість обрати сервер із різними операційними системами;

2) підтримка Dependency Injection, а отже в нас є можливість легкої інтеграції різних програмних модулів в роботу нашої системи;

3) об'єктно-орієнтований підхід, що дозволяє розробити маштабовані та гнучкі застосунки, сповідуючи сучасні тенденції програмування;

4) підтримує багатопоточність та асинхронність, що дозволяє швидко оброблювати великі масиви даних;

5) має велику кількість документації (як офіційної так і від простих користувачів) та прикладів вирішення різного роду помилок;

6) є частиною екосистеми Microsoft, а отже найкраще інтегрується із офісними застосунками, які також є розробкою цієї компанії;

7) фреймворк .NET також надає змогу створити іншого роду застосунки чи то для мобільних платформ, чи для ігрової індустрії, або ж десктопні рішення і т.д. У зв'язку з цим, коли особистий кабінет здобувача освіти достатньо розшириться та виникне потреба в мобільному застосунку для доступу до вже існуючого функціоналу, то ми зможемо використати, приміром, технологію .NET MAUI (кросплатформенний фреймворк для створення мобільного застосунку, в якому один і той же код буде запускатися для платформ Android, iOS, macOS та Windows [208]) та з легкістю інтегруватися у вже існуючу логіку та процеси обробки даних.

### *4.1.3 Інфраструктура сервера: вибір технологій та обґрунтування рішень*

Для того, щоб наша функціональність була доступна для широкого загалу, необхідно розмістити її на сервері та дозволити звертатися за

допомогою доменного імені. Серед переліку операційних систем для серверів є дві найвідоміші: Windows та Linux. Варто зазначити, що також є сервери, які працюють на macOS, але вони є дороговартісні та більше не підтримуються компанією Apple. Тому детально зупинимося на перевагах та недоліках двох інших операційних систем, в чому нам допоміг автор статті [209]. Так, Linux можна відзначити наступними характеристиками:

1) Відкритий доступ до вихідного коду, що дає можливість користуватися операційною системою безкоштовно;

2) Гнучкість в налаштуваннях – це характерна риса Linux, так як система надає можливість під специфічні потреби, включаючи вимкнення непотрібних функцій для покращення продуктивності чи зниження споживання ресурсів.

3) Має здатність легко масштабуватися від малих пристроїв до величезних дата-центрів, що є важливим для проєктів, коли ті потребують гнучкості в масштабуванні своїх ресурсів.

4) Linux працює стабільно і рідко вимагає перезавантаження. Сервери на із такою операційною системою можуть часто працювати місяцями, або ж роками без серйозних проблем.

5) З точки зору безпеки Linux – це основна операційна система для серверів через потужну систему прав доступу, системі управліннями пакетів, налаштування фаєрволів і т.д.

Переваги Windows напроти Linux, це:

1) Зрозумілий і дружелюбний інтерфейс, який є інтуїтивно зрозумілим і зручним навіть для тих, хто не має технічного досвіду, що є основною причиною того, чому Windows є настільки популярною серед звичайних користувачів і в корпоративному середовищі.

2) Простота встановлення – процес установки, і налаштування програм є досить зручними, навіть, для користувачів без технічних освіти.

3) Простота у використанні – ідеально підходить тому, хто не хоче глибоко занурюватися в налаштування або не має бажання працювати з

командним рядком.

4) Рівень захисту – останніми роками аспект, який було значно покращено в даній операційній системі. І хоча Windows не славиться таким же рівним захисту як і Linux, вони впевнено крокують на шляху до цього.

Тож, можемо зробити висновок, що у випадках, коли є хороший бюджет для купівлі чи оренди серверного обладнаня, коли є недосвідчені користувачі операційної системи Linux та адресного рядка або ж коли необхідне широке використання продуктів Microsoft на серверній частині, то необхідно звернути увагу саме на Windows. Робити ж вибір на користь Linux можна, коли є обмежений бюджет, є кваліфіковані працівники для роботи із такою системою, важливим фактором є надійність та швидкодія.

Враховубчи всі вищеописані фактори, можемо зробити, що для нашого проєкту, який є незалежним від операційної системи та над яким займаються розробники із достатнім рівним досвіду, але який немає достатнього фінансування, необхідно обрати саме операційну систему Linux. І хоча Windows і надає кращу сумісність із технологіями Microsoft, проте, як показують результати досліджень, однай та ж програма, яка написана на ASP.NET Core, може показувати на Linux значно кращий результат обробки великих масивів даних, ніж на Windows. Тому за хостинг для системи було обрано саме сервер, на якому встановлено операційну систему Linux.

До слова, даний сервер вирішено розмістити в інфраструктурі разом із генератором для постійної роботи сайту, а згодом планується перемістити в дата-центр із постійним електроживленням. Виділений для системи сервер містить 2 гігабайти оперативної пам’яті, що є достатнім для роботи рекомендаційної системи та функціональності всього проєкту загалом.

### *4.1.4 Користувацький інтерфейс: дизайн, функціональність та вибір технологій*

Користувацький інтерфейс – це ледь не найважливіша частина всієї системи. І хоча більшість логічних операцій відбувається на серверній частині

сайту, саме фронтенд справляє основне враження на користувача. Саме тому, приступаючи до роботи над цією користувацьким інтерфейсом системи, потрібно зважити всі кроки по реалізації даної частини.

Для створення користувацького інтерфейсу особистого кабінету здобувача освіти ми могли обрати серед наступних технологій:

1) JavaScript – мова програмування, яка є основою для написання сайтів та лежить в основі всіх інших бібліотек та фреймворків. [210]

2) jQuery – бібліотека, яка написана на JavaScript та містить зручний функціонал для управління елементами розмітки сторінки та їх стилями. Головний дивіз даної технології: «Пиши менше – роби більше». Проєкт із відкритим кодом. [211]

3) KnockoutJS – JS бібліотека, яка надає можливість двосторонньо зв’язувати код JS та елементи розмітки, що дозволяє автоматично змінювати значення в розмітці або в скріптовій логіці. [212]

4) Angular – це фреймворк, який на основі використання мови програмування Typescript, дає будувати вебзастосунки, які працюють як єдина вебсторінка. Також надає багато корисних функцій, серед яких оброблення помилок, двостороннє зв’язування даних, маршрутизація і т.д. Даний фреймворк є таким, що має відкритий код, а постійно вдосконалюється та розширюється світовою спільнотою розробників. [213]

5) React – бібліотека JavaScript, яка була створена розробника компанії Facebook для внутрішніх потреб. Згодом, її код зробили доступним і вона отримала широку популярність. Для цієї бібліотеки існує багато доповнень, а сама вона є доволі малою в об’ємах пам’яті. Дана бібліотека є однією із найпопулярніших для розробки проєктів. [214]

6) Vue.js – це фреймворк, написаний на JavaScript, який забезпечує високу продуктивність, легку інтеграцію та є простим для використання. [215]

Для вибору технології ми опиралися не тільки на необхідність реалізації рекомендаційної системи, а всієї системи в цілому. Даний проєкт повиннен містити функціональність академічної успішності здобувача, індивідуального

плану навчання, перегляд розкладу, керування портфоліо, опитування, вибір дисциплін і т.д. Також система повинна містити надійну систему маршрутизації та аутинтифікації, бути легкорозширювальною та масштабованою. Саме тому вибір було зроблено на користь Angular, так як він найкраще підходить розробку великих систем, де потребується багацький користувацький інтерфейс, різного роду валідації та інтерактивна взаємодія із користувачем. [216]

Angular вимагає певних спецефічних вмінь від розробника, адже застосунки, які написані на даному фреймворку, потребують ретельнішого підходу до стилю побудови архітектури, але забезпечує проєкт працюючою, легкорозширюючою архітектурою [217]. Проте це може виступати перевагою, у випадку коли потрібно розробити великий проєкт, так як фреймворк сам диктує правила, як розробник зобов’язаний робити правильну архітектуру. Також, деякі дослідження [218], демонструють, що в багатьох випадках Angular має більшу швидкодію обробки інформації та кращу ефективність роботи в Google Chrome (найпопулярніший вебпереглядач сучасності згідно праці [219]).

Приклад користувацького інтерфейсу системи для вибору дисциплін наведено на рисунку 4.9.

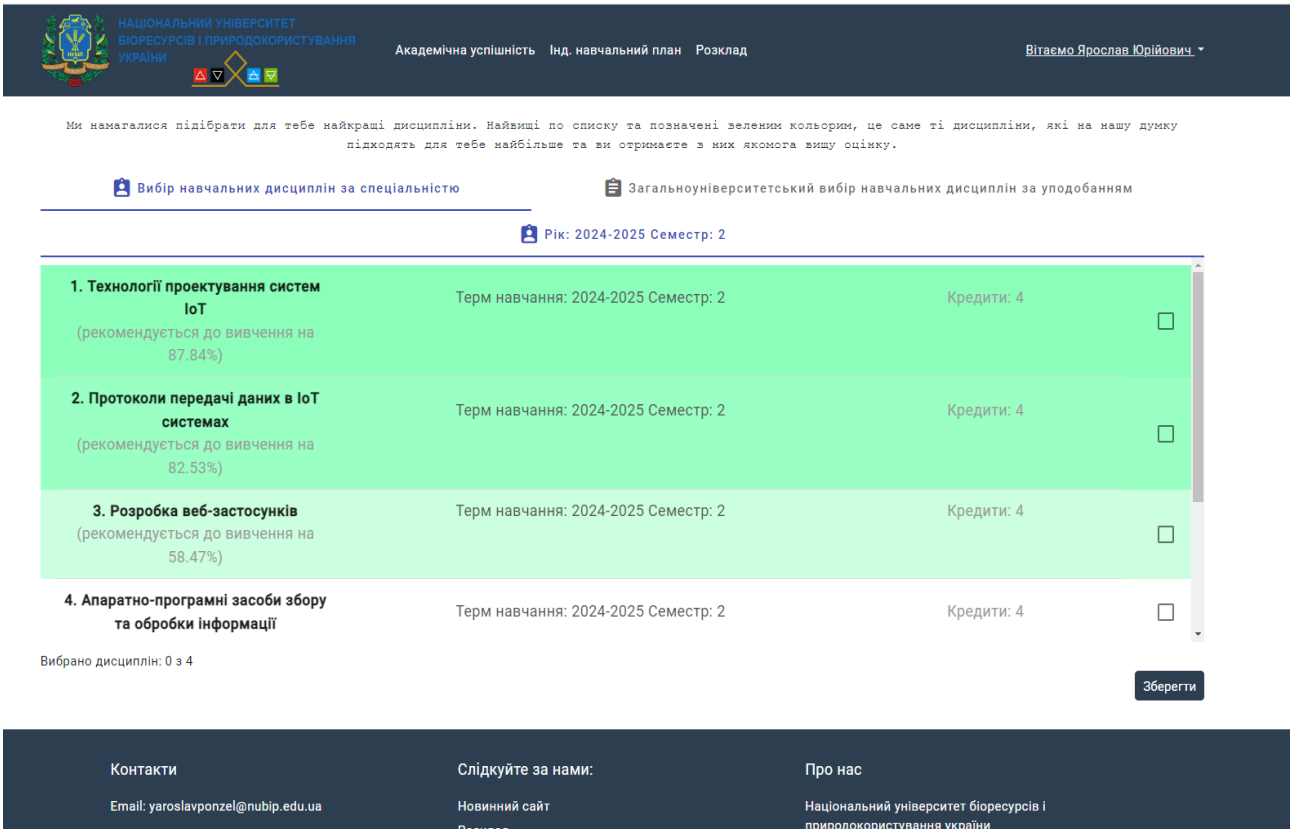


Рис. 4.9 – Користувацький інтерфейс системи

Використання архітектури односторінкового застосунку дозволило реалізувати перехід між етапами (опитування, вибір, зворотний зв’язок) миттєво, без необхідності повного перезавантаження сторінки, що значно покращує досвід користувача.

Зокрема, застосування можливостей Angular забезпечило наступні переваги при реалізації алгоритму:

1. Використання структурних директив дозволяє системі миттєво реагувати на наявність рекомендацій, тим самим автоматично приховувати або відображати блоки з додатковою інформацією та сортувати список дисциплін без додаткових запитів до сервера.

2. Реалізація збору даних про ключові слова, роботу та хобі виконана за допомогою Reactive Forms, що забезпечує гнучку валідацію введених даних та зручне їх збереження для подальшого аналізу.

3. Компонентний підхід дозволив розділити інтерфейс на незалежні блоки (картка дисципліни, форма опитування, модальне вікно відгуку), що

спрощує підтримку коду та масштабування системи в майбутньому.

## 4.2 Аналіз результативності та тестування розробленої системи

### *4.2.1 Аналіз точності прогнозів рекомендаційної системи*

З метою аналізу вибору дисциплін було створено спеціальну структуру в базі даних, яка враховувала вибір здобувача та рекомендації, які згенерувала йому рекомендаційна система, на основі актуальних на той момент даних. Після завершення процесу вибору дисциплін, в даній структурі містились записи 3939 записів (рис. 4.10).

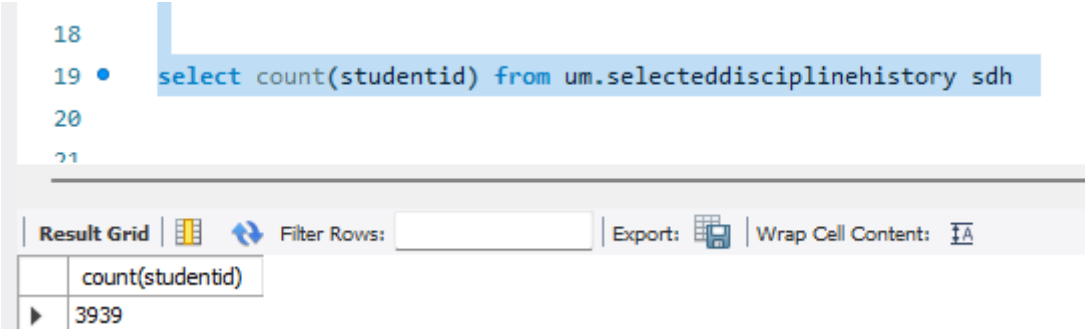


Рис. 4.10 – Кількість записів в журналі історії вибору дисциплін

Ці записи містять історію вибору або перевибору дисциплін здобувачами, а тому можуть містити повтори для здобувачів, так як останнім було надано можливість змінювати свій вибір в будь-яку мить (одна із переваг електронної системи над старими реалізаціями).

Всього архів вибору дисциплін містить записи для 1875 унікальних здобувачів (рис. 4.11).

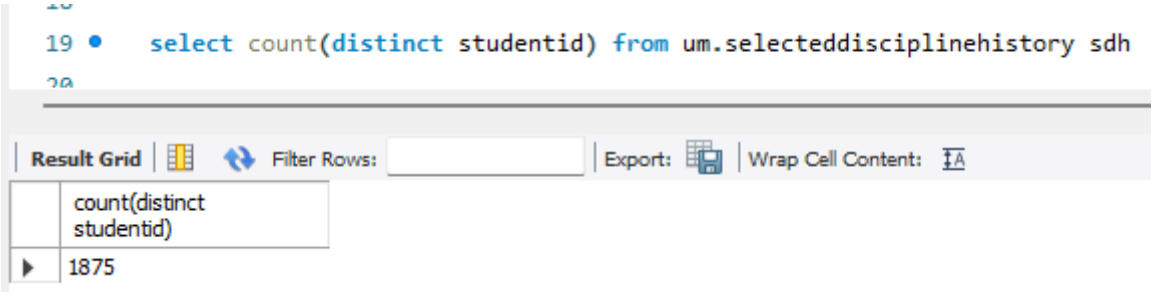


Рис. 4.11 – Кількість унікальних записів в журналі історії вибору дисциплін

Однак, не всіх здобувачів можна брати для розрахунку ефективності рекомендаційної системи, так як замість здобувача вибір міг здійснити завідуючий відділенням або інша компетентна особа (у випадку, якщо здобувач навіть не з'явився для будь-якого вибору). Тому із 1875 здобувачів маємо 1459 здобувачів, які самостійно зробили вибір (рис. 4.12).

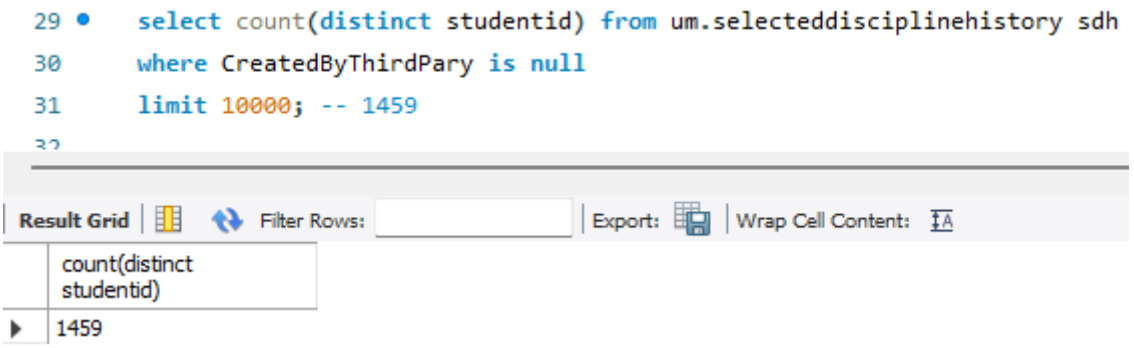


Рис. 4.12 – Кількість записів в журналі історії вибору дисциплін, де здобувач сам здійснював вибір

Для оцінки ефективності роботи рекомендаційної системи застосовано підхід, що базується на визначенні ступеня збігу між множиною рекомендованих дисциплін та фактичним вибором здобувача. Використання даної метрики є стандартизованим методом верифікації моделей машинного навчання згідно з [220] для обчислень. Він виражений формулою математичною формулою:

$$Accuracy_{recommendation} = \frac{\sum_{(\forall u,i/r(u,i)=1)} 1 - |p(u,i) - P(u,i)|}{R} \quad (4.1)$$

де:

- $r(u,i)$ – це бінарна функція, що вказує, чи був елемент $i$ рекомендований користувачу $u$. Якщо елемент був рекомендований, то $r(u,i) = 1$, а якщо ні – $r(u,i) = 0$.
- $p(u,i)$ – фактичний вибір користувача для елемента $i$.
- $P(u,i)$ – ймовірність того, що користувач $u$ зробить свій вибір на користь елементу $i$.
- $1 - |p(u,i) - P(u,i)|$ – вимірювання корисності для кожної пари

користувач-елемент. Абсолютна різниця між фактичною вибором та прогнозованим вибором вимірює, наскільки точним є передбачення системи. Якщо різниця нульова, значить, прогноз був точним, якщо ж завелика, то навпаки.

- $(\forall u, i / r(u, i) = 1)$ – сума, яка підсумовує всі значення $1 - |p(u, i) - P(u, i)|$ для всіх елементів, які були рекомендовані користувачу $u$ (тобто у випадку $r(u, i) = 1$), що дозволяє нам врахувати всі рекомендовані елементи, а також те, як добре ці елементи співвідносяться з реальним вибором користувачів.
- $R = \sum_{u,i} r(u, i)$ – загальна кількість рекомендацій, які система надала користувачу. Це свого роду нормалізатор, який надає змогу обчислити середнє значення точності на всіх рекомендаціях.

Для того, щоб прокалькулювати дані точності наданих рекомендацій за допомогою вищезгаданою формулою, було написано спеціальну програму (Додаток Е). Для цього ми використаємо таблицю в базі даних SelectedDisciplineHistory, в якій використаємо дані з полів StudentId (числове значення, типу: «53251»), Disciplines (рядкове значення, в якого шаблон «ідентифікатор дисципліни (тип дисципліни)», типу «53252 (факультетська), 53254 (університетська), ...»), RecommendedDisciplines (рядкове значення, в якого шаблон «ідентифікатор дисципліни (тип дисципліни): (прогнозована оцінка)», типу «53252 (факультетська): (80), 53254 (університетська): (60), ...»),).

В даній програмі було вирішено брати для порівняння точності прогнозу системи таку ж кількість прогнозованих елементів, яку користувач мав обрати. Ці елементи мають бути такими, які містять найбільшу потенційну оцінку з рекомендованої дисципліни. Тобто, якщо користувач має обрати три дисципліни із десяти можливих, то він отримає рекомендацію щодо всіх 10 дисциплін. Однак, для вимірювання ефективності нам потрібно взяти 3, які мають найбільшу потенційну оцінку по дисципліні. Таким чином ми дізнаємося, чи обрав здобувач дисципліни із найвищим прогнозованим

результатом, чи по якійсь причині зробив свій вибір на користь іншого дисципліни. Наприклад, користувач обирає дисципліни із вектору {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, та обирає три дисципліни {0, 2, 5}, а наша рекомендаційна система надала рекомендацію дисциплін вектором {1, 2, 6, 8, 4 ,5, 0, 3, 9, 7}, то для калькуляцій ми будемо брати прогноз системи {1, 2, 6}, який складається тільки із перших трьох елементів.

Для наочної демонстрації роботи алгоритму оцінки точності в таблиці 4.1 наведено фрагмент матриці порівняння для одного випадкового здобувача. Стовпці таблиці відповідають унікальним ідентифікаторам дисциплін, доступних для вибору. У даній матриці відображено результат бінарної класифікації, де перетин множини рекомендованих системою дисциплін та множини фактично обраних здобувачем дисциплін визначає успішність роботи алгоритму.

*Таблиця 4.1*

**Приклад матриці співвідношення рекомендованих та фактично вибраних дисциплін**

| **Рекомендовані / вибрані** | **Дисципліна 0** | **Дисципліна 2** | **Дисципліна 5** |
|---|---|---|---|
| **Дисципліна 1** | 0 | 0 | 0 |
| **Дисципліна 2** | 0 | 1 | 0 |
| **Дисципліна 6** | 0 | 0 | 0 |

де:

1 – це правильно передбачений рекомендаційною системою елемент;

2 – це обрана здобувачем дисципліна, які не було передбачено рекомендаційною системою.

Саме кількість «одиниць» у цій матриці (кількість перетинів) використовується для розрахунку персональної точності рекомендацій.

Для нормальної функціональної роботи рекомендаційної системи необхідні початкові дані здобувача, які будуть давати інформацію про те, які дисципліни проходив здобувач та фінальні результати складання іспитів чи заліків по них. На жаль база даних містила такого роду дані не для всіх

здобувачів, які проходили вибір дисциплін. Це зумовлено тим, що не всі здобувачі навчались на попередньому освітньому рівні в нашому університеті або навчались занадто давно, а тому їх дані можуть бути неактуальними для розрахунку. Отож, здобувачів, для яких система дала передбачення (оцінка > 0) хоча б по якійсь дисципліні всього 604 (рис. 4.13).

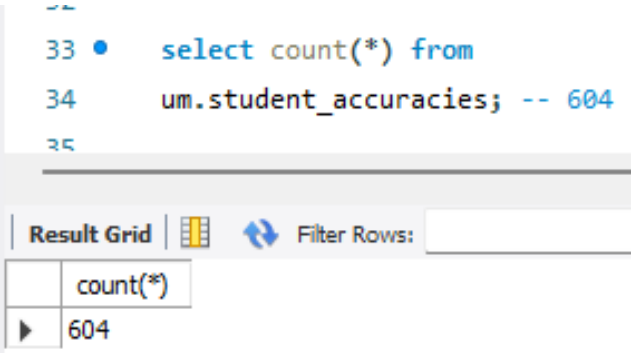


Рис. 4.13 – Кількість здобувачів, для яких система дала передбачення

Тоді отримаємо результат, що із 604 здобувачів для 386 здобувачів було вгадано хоча б одну дисципліну (рис. 4.14).

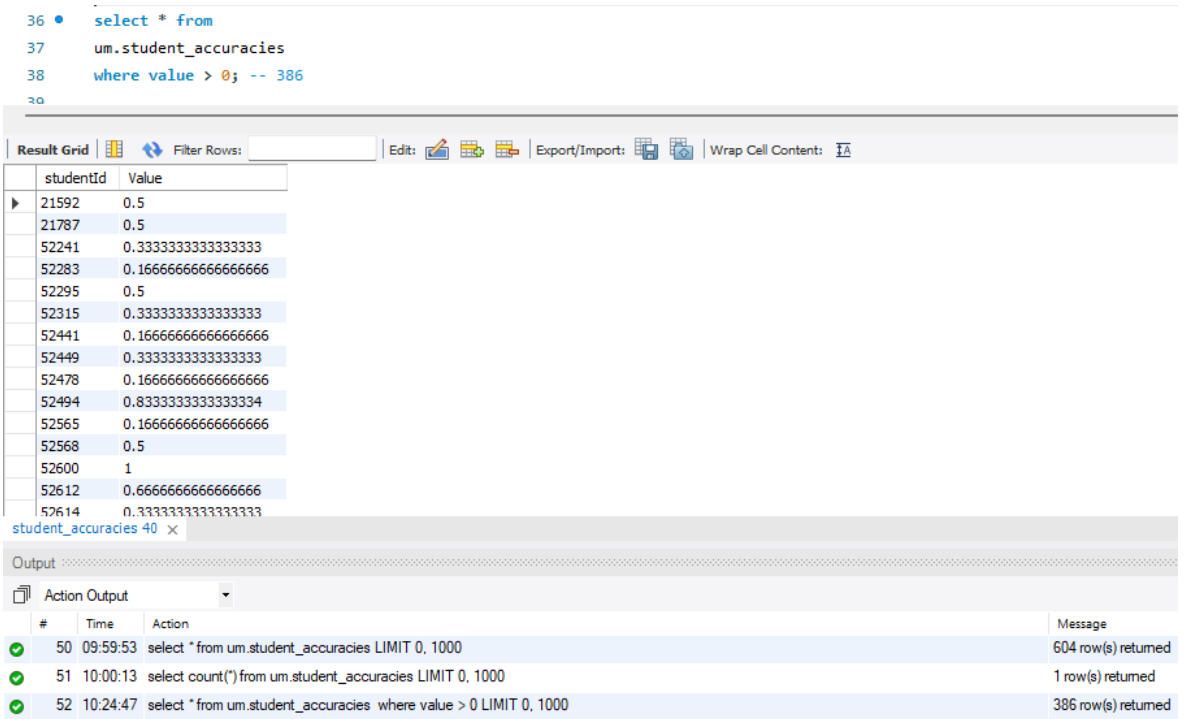


Рис. 4.14 – Кількість здобувачів, для яких було вгадано хоча б одну дисципліну

Вирахуємо середнє значення кількості вгаданих дисциплін для здобувачів

й отримаємо значення – 0.31152830337432985, що дорівнює 31.15 % відсотка вгаданих виборів (рис. 4.15).

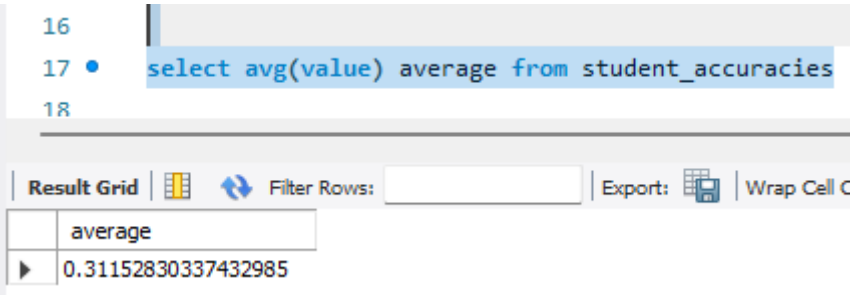


Рис. 4.15 – Результати обчислення середньої точності передбачень

Таке значення може бути розглянуте як середній рівень точності, що вказує на те, що система може допомогти здобувачам з вибором дисциплін, але є простір для покращення. Зокрема, можливе вдосконалення якості рекомендацій, щоб вони були більш релевантними для уподобань здобувача.

Також було розраховано точність передбачень відносно даних по факультетах, де значення склали від 0.81 до 0.57 (рис. 4.16).

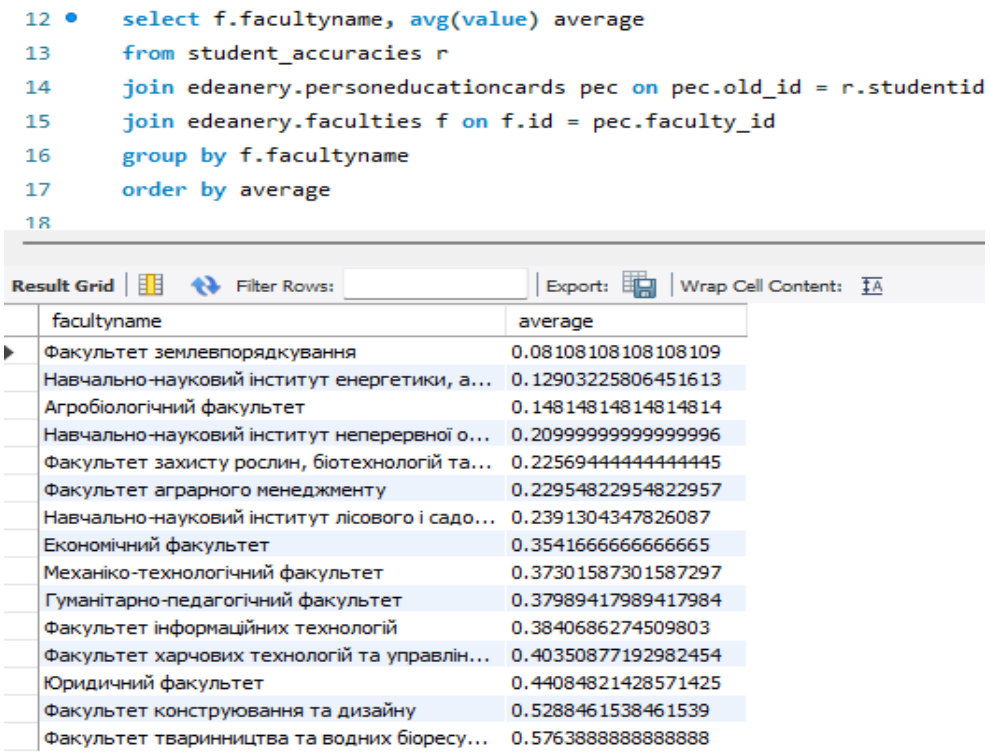


Рис. 4.16 – Результати обчислення точності передбачень відносно факультетів

Однією із ключовим мір ефективності роботи рекомендаційної системи є значення кореня середньої квадратичної помилки (RSME). Ця величина вимірює середню величину помилки між прогнозованими та фактичними значеннями. Чим менші значення RSME, тим вища точність передбачення моделі, оскільки ця величина є мірою відхилення між фактичними і прогнозованими значеннями, і її зменшення свідчить про покращення точності моделі.

В ході опрацювання 2543 здобувачів, було вирахувано середнє значення кореня середньої квадратичної помилки для всіх здобувачів (рис. 4.17). Значення RMSE в нашому випадку дорівнювало 15.06, що означає, що в середньому відхилення між прогнозами та реальними оцінками складає приблизно 15.06 балів. Оскільки оцінки варіюються від 0 до 100, максимальна можлива помилка для одного прогнозу – це 100 (якщо система передбачає оцінку 0, коли насправді вона має бути 100, або навпаки), то значить, що значення RSME для нашого алгоритму становить 15 % від загального результату.

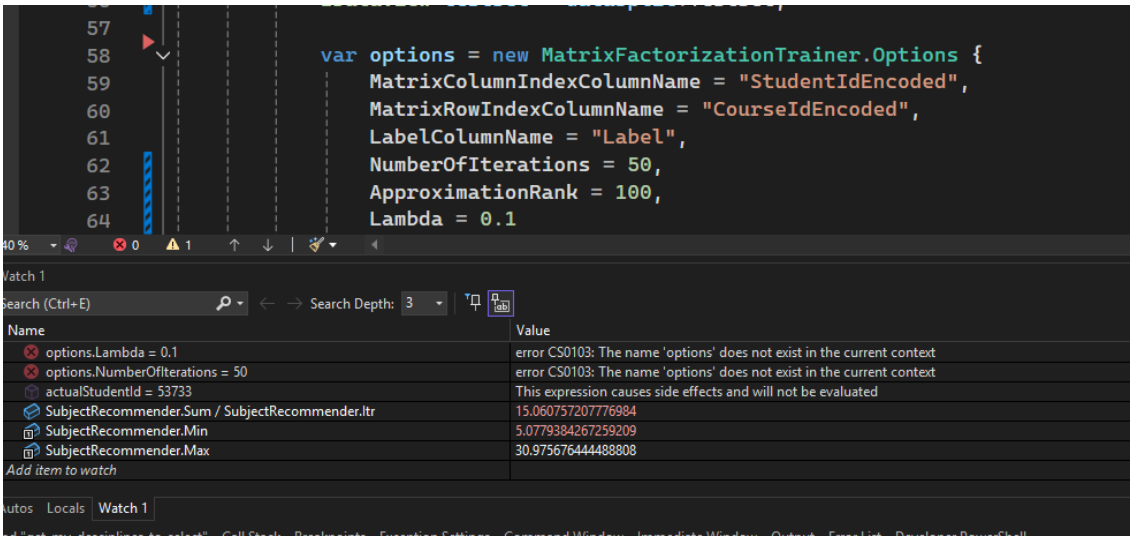


Рис. 4.17 – Результати обчислення RSME для рекомендаційної системи

Враховуючи те, що мета нашої системи – це допомогти у формуванні індивідуального шляху навчання, а не в його повній побудові, та наданні загальних рекомендації по дисциплінах, то помилка в 15 балів може бути

допустимою. Тобто, система може вірно порекомендувати курси, навіть якщо передбачена оцінка не є точною на 100 %. Для рекомендаційної системи це може бути достатньо, оскільки головне – це загальна тенденція до правильного вибору дисциплін.

Також, варто зазначити, що в ході опрацювання даних по RSME для 2543 здобувачів, було отримано мінімальні та максимальні значення цієї величини, які дорівнюють 5.08 та 30.98 відповідно. З однієї сторони, це хороший результат, якщо для деяких здобувачів алгоритм отримує помилки на рівні 5 балів, що свідчить про дуже точні передбачення оцінок. З іншої сторони, це невтішний результат, так як для деяких здобувачів помилка до 30 балів може свідчити про те, що їхні оцінки складно передбачити через різноманіття вподобань або, можливо, через відсутність достатніх даних для точного прогнозування.

В загальному, середнє, мінімальне та максимальне значення RSME є хорошими результатами для рекомендаційної системи, так як на основі цих даних в багатьох випадках система здатна надати якісну рекомендацію для здобувачів. Однак даний алгоритм потребує подальшого опрацювання та покращення, зокрема для здобувачів з великими помилками. Варто додатково оптимізувати модель або працювати з даними для цих випадків, щоб зменшити максимальні помилки і зробити систему ще більш точною.

### *4.2.2 Аналіз ефективності прогнозів рекомендаційної системи*

Однією із ключовим мір ефективності роботи рекомендаційної системи є значення кореня середньої квадратичної помилки (RSME). Ця величина вимірює середню величину помилки між прогнозованими та фактичними значеннями. Чим менші значення RSME, тим вища точність передбачення моделі, оскільки ця величина є мірою відхилення між фактичними і прогнозованими значеннями, і її зменшення свідчить про покращення точності моделі.

Однак, у процесі оцінювання точності рекомендаційної системи, яка

прогнозує підсумкові оцінки здобувачів з навчальних дисциплін у шкалі від 0 до 100, виникла проблема неповноти даних: частина здобувачів, для яких система сформувала прогноз, у підсумку не отримала жодної оцінки з дисципліни й із 386 здобувачів, для яких система вгадала принаймні одну дисципліну, тільки 277 здобувачів отримав оцінку (рис. 4.18).

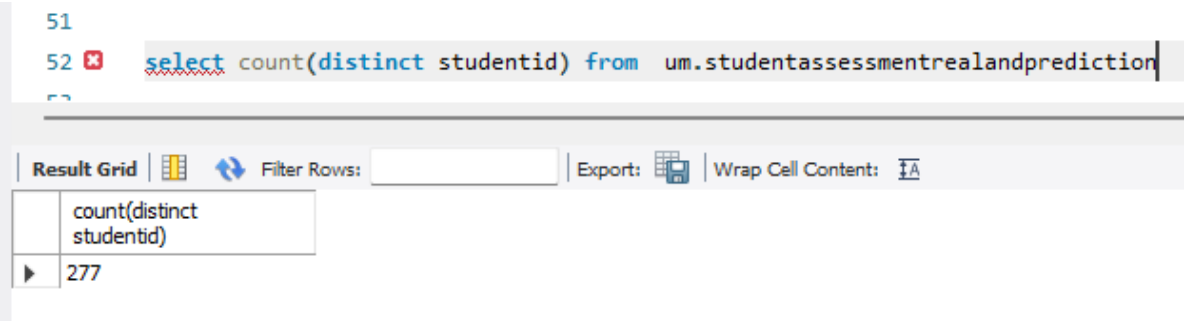


Рис. 4.18 – Результати обчислення RSME для рекомендаційної системи

Аналіз причини відсутності оцінки у здобувачів, що у більшій кількості випадків це пов’язано із наступними чинниками: відрахування, академічна заборгованість, невідвідування занять, дострокове припинення навчання або неявка на підсумковий контроль. Таким чином, відсутність фактичної оцінки не є коректним значенням цільової змінної для задачі регресії.

У зв’язку з цим, для забезпечення достовірності порівняння прогнозованих та реальних результатів, у фінальному розрахунку метрик точності (MAE, RMSE тощо) враховувались лише ті здобувачі, які отримали підсумкову оцінку з дисципліни.

Такий підхід дозволяє мінімізувати вплив неконтрольованих зовнішніх факторів та зосередитися на прямій оцінці ефективності моделі саме в межах її прогнозного призначення.

Також варто зазначити, що система деканату, із якої ми беремо дані для калькуляції рекомендації дисципліни для здобувачів містить дані тих здобувачів, які успішно склали дисципліни (отримали оцінки > 60), а дані про неуспішне складання дисциплін в ній відсутні (за винятком невеликих виключень). Тож наша рекомендаційна система спеціалізується на передбаченні точності оцінки  саме для здобувачів, які успішно склали іспит чи залік з тої чи

іншої дисципліни. Саме тому здобувачі без фактичного результату були вилучені з аналізу як такі, що не мають валідного значення цільової змінної.

В ході опрацювання показників здобувачів, було вирахувано середнє значення кореня середньої квадратичної помилки для всіх здобувачів. Значення RMSE в нашому випадку дорівнювало 13.367804748285831 (рис. 4.19), що означає, що в середньому відхилення між прогнозами та реальними оцінками складає приблизно 13.37 балів. Оскільки оцінки варіюються від 0 до 100, максимальна можлива помилка для одного прогнозу — це 100 (якщо система передбачає оцінку 0, коли насправді вона має бути 100, або навпаки), то значить, що значення RSME для нашого алгоритму становить 13 % від загального результату.

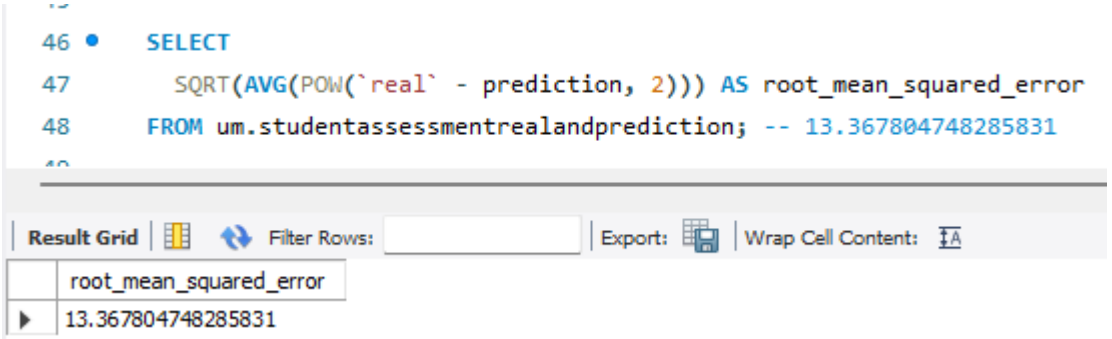


Рис. 4.19 – Результати обчислення RSME для рекомендаційної системи

Враховуючи те, що мета нашої системи – це допомогти у формуванні індивідуального шляху навчання, а не в його повній побудові, та наданні загальних рекомендації по дисциплінах, то помилка в 13 балів може бути допустимою. Тобто, система може вірно порекомендувати дисципліни, навіть якщо передбачена оцінка не є точною на 100 %. Для рекомендаційної системи це може бути достатньо, оскільки головне — це загальна тенденція до правильного вибору дисциплін.

Також було отримано дані кореня квадратичної помилки відносно факультету. Мінімальні та максимальні значення цих величин дорівнюють 6.14 та 24.09 відповідно (рис. 4.20). З однієї сторони, це хороший результат, якщо для деяких здобувачів алгоритм отримує помилки на рівні 6 балів, що свідчить

про дуже точні передбачення оцінок. З іншої сторони, це невтішний результат, так як для деяких здобувачів помилка до 24 балів може свідчити про те, що їхні оцінки складно передбачити через різноманіття вподобань або, можливо, через відсутність достатніх даних для точного прогнозування.

```
54    select f.facultyname, SQRT(AVG(POW(`real` - prediction, 2))) rmse
55    from um.studentassessmentrealandprediction r
56    join edeanery.personeducationcards pec on pec.old_id = r.studentid
57    join edeanery.faculties f on f.id = pec.faculty_id
58    group by f.facultyname
59    order by rmse
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| facultyname | rmse |
|---|---|
| Факультет тваринництва та водних біоресу... | 6.144109999999998 |
| Факультет захисту рослин, біотехнологій та... | 7.227231694499432 |
| Факультет аграрного менеджменту | 7.553300654972541 |
| Економічний факультет | 8.249226987573508 |
| Юридичний факультет | 10.039091516928087 |
| Гуманітарно-педагогічний факультет | 11.16146272446875 |
| Факультет харчових технологій та управлін... | 11.641005396848062 |
| Факультет землевпорядкування | 14.577223608900084 |
| Навчально-науковий інститут лісового і садо... | 14.75415610945678 |
| Факультет конструювання та дизайну | 14.879359574489943 |
| Факультет інформаційних технологій | 15.1844368434112 |
| Механіко-технологічний факультет | 15.980209091658363 |
| Навчально-науковий інститут неперервної о... | 20.76663671929956 |
| Навчально-науковий інститут енергетики, а... | 21.369648156998082 |
| Агробіологічний факультет | 24.09848630899181 |

Рис. 4.20 – Результати обчислення RSME відносно факультетів

В загальному, середнє, мінімальне та максимальне значення RSME є хорошими результатами для рекомендаційної системи, так як на основі цих даних в багатьох випадках система здатна надати якісну рекомендацію для здобувачів. Однак даний алгоритм потребує подальшого опрацювання та покращення, зокрема для здобувачів з великими помилками. Варто додатково оптимізувати модель або працювати з даними для цих випадків, щоб зменшити максимальні помилки і зробити систему ще більш точною.

### *4.2.3 Аналіз опитування здобувачів з метою вивчення потреб та уподобань*

Також в рамках процесу вибору дисциплін здобувачі, які отримали рекомендації від системи, були опитані щодо задоволеності системи. Результати опитування наведено за наступним онлайн посиланням https://docs.google.com/spreadsheets/d/1f1d8aTcXvmQ29LCoLK2HK3MSGxycOW6g98nnk95PzEU/edit?usp=sharing. В розрахунок взято всі відгуки, в яких користувачі залишили коментар щодо роботи системи. Згідно з отриманими даними, 163 із 220 здобувачів в цілому задоволені системою, що становить 74 % опитаних. Водночас 26 % здобувачів відзначили, що не задоволені рекомендацією (поставили оцінку 0), що вказує на можливі недоліки моделі, які потребують коригування.

Якщо проаналізувати відгуки користувачів, то їх умовно можна поділити на:

- позитивні – коментарі, в яких зазначається ефективність, зручність та відповідність запропонованих дисциплін їхнім інтересам («Дисципліни були запропоновані влучно, згідно моїх відповідей в опитуванні», «Вона була дуже ефективною та логічною»);
- змішані або частково позитивні – коментарі, в яких відзначались користь системи, але також було висловлено певні побажання по функціоналу. («Так, доволі вдалий підбір, але багато дисциплін дуже цікавих чомусь пропустило. Добавте, будь ласка, у вибір вподобань більше тем пов’язаних…», «Вибір має орієнтуватися не лише на уподобаннях щодо знань, а й на практичному викладанні – тобто я надам перевагу посередньому дисципліни, але який подають цікаво»);
- негативні відгуки – коментарі, де вказано недоліки системи та піддано критиці неточність її передбачень. («У підборі дисциплін за спеціальністю ефективність була низька, дисципліни не відповідали моїм інтересам», «Я розумію, що університет біоресурсів і природокористування, але аграрна

політика не цікавить, дякую.»);

- зауваження щодо інтерфейсу або іншого функціоналу – коментарі, в яких міститься відгук конкретно не про саму рекомендаційну систему, а про користувацький інтерфейс, недостатність інформації. («Можна зробити таблиці скролл-областю висотою в сторінку і кнопку зберегти внизу сторінки, бо перемотувати таке собі сто дисциплін до низу.», «Потрібно додати корисні посилання до всіх дисциплін, а не лише на їх частину»);
- відгуки без чіткої змістовності – коментарі, які по своєму змістовому наповненню, що не дає змоги чітко оцінити досвід користувача. («так», «+») тощо.

Отож, можна зробити висновок, що в цілому система вибору дисциплін уже демонструє високу базову ефективність і значний потенціал для покращення процесу вибору дисциплін. Однак, для підвищення її точності, користувацької зручності та загальної задоволеності здобувачів доцільно продовжити роботу над удосконаленням алгоритмів персоналізації, розширенням функціональних можливостей інтерфейсу та врахуванням додаткових чинників, що впливають на вибір дисциплін. Реалізація цих покращень дозволить зробити рекомендаційну систему більш гнучким та ефективним інструментом підтримки здобувачів у процесі формування власного навчального плану.

### 4.3 Пропозиції щодо покращення функціональності та ефективності системи

Поглиблення персоналізації та розширення обсягу даних, які враховуються під час формування рекомендацій є одним із ключових напрямів вдосконалення рекомендаційної системи вибору дисциплін. На основі зібраних відгуків здобувачів і сучасних підходів до побудови рекомендаційних систем можна виділити кілька конкретних векторів розвитку системи, що надасть можливість у майбутньому значно підвищити її точність, відповідність та задоволеність користувачів.

По-перше, доцільним є включення у розрахунок ключових слів, які здобувачі самостійно обирають під час роботи із системою. Кожен здобувач має індивідуальні інтереси, які можуть відображатися у фіксованому переліку обраних ним понять. Надання можливості вводити власні ключові слова дозволить системі краще розпізнавати специфічні інтереси користувача або більш гнучко формувати профіль користувача для надання рекомендації. Такий підхід зробить рекомендації більш персоналізованими та точними, особливо для здобувачів зі складними або нестандартними вподобаннями та інтересами.

По-друге, значно покращити релевантність рекомендацій допоможе урахування контекстної інформації про здобувача, зокрема:

- його поточну роботу (якщо здобувач уже працює за фахом чи в суміжній галузі);
- очікувану майбутню кар’єру або цільову професію;
- особисті захоплення та хобі.
- тощо

Ці дані нададуть змогу формувати пропозиції дисциплін не лише за академічними інтересами, а й із врахуванням професійних перспектив і особистісних уподобань здобувача. Наприклад, здобувач, який працює у сфері маркетингу, міг би отримати рекомендації щодо дисциплін з психології споживача або цифрової аналітики, навіть якщо його основна освітня програма належить до іншої галузі. Аналогічно, хобі, такі як дизайн, фотографія чи спорт, можуть сигналізувати про потенційно цікаві міждисциплінарні дисципліни.

Третій напрямок удосконалення пов’язаний із урахуванням персоналії викладача, який веде дисципліну. Для значної частини здобувачів стиль викладання, харизма викладача, його наукові досягнення або репутація відіграють критично важливу роль у виборі дисципліни. А тому вдалим рішенням буде:

- зберігати і обробляти інформацію про викладачів, зокрема їхню наукову спеціалізацію, досвід, викладацький стиль;

- використовувати рейтинги викладачів, сформовані на основі відгуків здобувачів;
- тощо.

Крім того, важливо впровадити можливість змінювати або підлаштовувати під себе принципів роботи самої рекомендаційної системи. На сучасному етапі розвитку технологій користувачі очікують мати контроль над тим, як саме формуються для них рекомендації. Система має надати можливість здобувачу обрати один або кілька підходів для генерації рекомендацій, наприклад:

- орієнтація на викладача – пріоритет надається дисциплінам, які читають конкретні викладачі;
- орієнтація на власні оцінки або ж академічні досягнення (система враховує успішність здобувача у суміжних дисциплінах для прогнозування його успіху у запропонованих дисциплінаах);
- змішаний підхід, коли рекомендації формуються на основі декількох критеріїв, з можливістю встановлення вагових коефіцієнтів для кожного з них (наприклад, 40 % інтереси, 30 % викладач, 30 % майбутня професія).

Такий механізм надасть здобувачу гнучкий інструмент для формування індивідуального навчального плану, що максимально відповідає його академічним планам, професійним цілям та особистим уподобанням.

Варто також зазначити, що на основі відгуків користувачів про роботу систему стає очевидним також той факт, що чималій кількості здобувачів не було зрозуміло на основі чого було зроблено розрахунок рекомендації системою. («Не дуже ефективна, не розумію чому мені видало наприклад іміджелогію... Але ок.», «Я не обирав податки як такі, що мене цікавлять. Чому податкове законодавство на 1 місці (я не проти, але це дивно, враховуючи відмічені терми...», «незрозумілий підбір рекомендацій для загальноуніверситетських», «Так, доволі вдалий підбір, але багато дисциплін дуже цікавих чомусь пропустило. Добавте, будь ласка, у вибір вподобань більше тем пов’язани...»). Тож є також запит здобувачів про покращення

розуміння роботи алгоритмів рекомендацій, що є важливим фактором у рекомендаційних системах та де на допомогу приходить пояснювальний штучний інтелект – класична проблема «чорної скриньки», притаманна складним алгоритмам машинного навчання: користувачі отримують результат, але не розуміють логіки його формування, що викликає недовіру навіть до релевантних пропозицій.

Для вирішення цього протиріччя та нівелювання скептицизму здобувачів необхідна інтеграція в систему методів пояснювального штучного інтелекту. Оскільки використана модель матричної факторизації є складною для прямої інтерпретації, найбільш ефективним підходом вбачається застосування методів післяобробки, таких як LIME або SHAP. Це дозволить супроводжувати кожну рекомендацію зрозумілим поясненням, що забезпечить прозорість прийняття рішень та підвищить рівень довіри до системи в освітньому середовищі.

**Висновки до четвертого розділу**

У четвертому розділі вирішено задачу створення прототипу системи, експериментальної перевірки його ефективності та визначення напрямків розвитку.

Основні наукові та практичні результати розділу:

1. Обґрунтовано та реалізовано архітектуру системи й на основі аналізу вимог до масштабованості та навантаження обрано мікросервісну архітектуру для рекомендаційного модуля, яка інтегрована в загальну трирівневу архітектуру веб-платформи. Для серверної частини обрано ASP.NET Core (продуктивність, кросплатформеність), для клієнтської – Angular (модульність, SPA), для бази даних – MySQL (сумісність з існуючою системою електронного деканату). Розгортання здійснено на ОС Linux, що забезпечує стабільність та економічну ефективність. Це дозволило створити єдине інформаційне середовище (Особистий кабінет, Електронний розклад, Чат-бот), яке підтримує здобувача на всіх етапах навчання.

2. Здійснено експериментальну перевірку ефективності системи, а

саме проведено аналіз точності прогнозів на історичних даних (вибірка 604 здобувачів для precision та 2543 для RMSE). Показник RMSE – середнє значення склало 13.37 (на шкалі 0-100), що становить ~13 % похибки. Встановлено, що для задачі рекомендаційного характеру такий рівень точності є прийнятним, оскільки дозволяє виявити загальну тенденцію успішності. Система коректно передбачила входження дисципліни у топ вибору для 31.15 % випадків, що підтверджує працездатність алгоритму колаборативної фільтрації в умовах реального університету.

3. Проведено валідацію через опитування користувачів, за результатами якого, 163 із 220 респондентів (74 % користувачів) висловили задоволення роботою системи (позитивні та змішані відгуки). Це підтверджує практичну цінність розробки. Виявлено кореляцію між негативними відгуками та відсутністю розуміння логіки рекомендацій («чому мені це радять?»), що актуалізує потребу в прозорості алгоритмів.

4. Запропоновано шляхи підвищення точності через врахування контекстних даних: ключових слів, поточної зайнятості, хобі та профілю викладача, що дозволить перейти від чистої колаборативної фільтрації до гібридної моделі, що покращить персоналізацію.

5. На основі аналізу «проблеми чорної скриньки» та відгуків здобувачів запропоновано інтеграцію методів пояснювального штучного інтелекту (XAI). Визначено, що використання інтерпретованих моделей або методів післяобробки (LIME, SHAP) дозволить підвищити довіру до системи, забезпечуючи прозорість рішень, що є критично важливим для освітньої сфери.

# ВИСНОВКИ

У дисертаційній роботі вирішено актуальне науково-прикладне завдання розробки та вдосконалення методів та моделей обробки даних, а також синтез рекомендацій для побудови індивідуального навчального плану здобувачів вищої освіти на основі алгоритмів машинного навчання.

Основні наукові та практичні результати роботи полягають у наступному:

1. Проаналізовано стан проблеми та моделі рекомендаційних систем та встановлено, що існуючі підходи до вибору дисциплін часто не враховують індивідуальні здібності здобувачів освіти та їх попередній академічний досвід. Обґрунтовано, що використання методів машинного навчання, зокрема колаборативної фільтрації, є найбільш перспективним для автоматизації цього процесу в умовах інформаційного освітнього середовища.

2. Розроблено математичні моделі збору й обробки даних, а також підходи до фільтрації та пошуку даних у межах рекомендаційної системи, побудовано логічні моделі сутностей освітнього процесу, а також формалізовано задачу вибору дисциплін як задачу лінійного програмування з обмеженнями. Вперше обґрунтовано модель рекомендаційної системи, яка базується на гібридному врахуванні академічної успішності, анкетних даних та зворотного зв’язку, що забезпечує комплексний підхід до персоналізації.

3. Обґрунтовано новий підхід до прогнозування вибору навчальних дисциплін на основі методів аналізу великих даних та машинного навчання, зокрема, метод прогнозування успішності на основі матричної факторизації шляхом введення регуляризації за нормою Фробеніуса та мінімізації функції втрат (RMSE), що дозволило адаптувати алгоритми машинного навчання до специфіки розріджених даних успішності здобувачів освіти та вирішити проблему масштабування при збільшенні кількості користувачів.

4. Створено та програмно реалізовано прототип рекомендаційної системи у вигляді веб-сервісу, побудованого на основі трирівневої архітектури (клієнт – сервер – база даних). Реалізації рівня представлення (Angular), рівня бізнес-логіки (ASP.NET Core) та рівня даних (MySQL) дозволили забезпечити

модульність системи, чітке розмежування функціоналу та високу продуктивність обробки запитів. Удосконалено методи взаємодії між інформаційними сервісами університету шляхом оптимізації API та розширення можливостей обміну даними, що покращило узгодженість та продуктивність системи. Інтелектуальне ядро системи на базі ML.NET інтегровано в архітектуру як окремий компонент, що взаємодіє з існуючою базою даних деканату в режимі реального часу.

5. Результати тестування на реальних даних (2543 здобувачі) показали середню похибку прогнозування (RMSE) на рівні 13.37 балів та точність визначення пріоритетних дисциплін 31.15 %. Високий рівень задоволеності користувачів (74 %), отриманий за результатами опитування, підтверджує ефективність запропонованих рішень та доцільність їх впровадження в освітній процес.

6. Запропоновано впровадження концепції пояснюваного штучного інтелекту для забезпечення прозорості формування рекомендацій та підвищення довіри користувачів. Також обґрунтовано необхідність інтеграції контекстних даних, зокрема кар’єрних цілей здобувачів та профілів викладачів, як ключового етапу переходу до повноцінної гібридної рекомендаційної моделі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Верховна Рада України прийняла Закон щодо розвитку індивідуальних освітніх траєкторій та вдосконалення освітнього процесу у вищій освіті / Прес-служба Апарату Верховної Ради України. 23.04.2024. URL: https://www.rada.gov.ua/news/razom/248731.html (дата звернення: 15.11.2025).

2. Венгер С. А., Марченко А. О. Інтелектуальна модель формування індивідуальної освітньої траєкторії здобувача на навчальній платформі. Сучасні інформаційні технології у сфері безпеки та оборони. 2024. № 1 (49). С. 160–170. URL: https://doi.org/10.33099/2311-7249/2024-49-1-160-170.

3. Шевчук Г. Й. Індивідуальна освітня траєкторія студента: суть і ключові аспекти організації. URL: https://ps.journal.kspu.edu/index.php /ps/article/view/4437/3922 (дата звернення: 15.11.2025).

4. Про вищу освіту : Закон України. Відомості Верховної Ради (ВВР). 2014. № 37-38. ст.2004. URL: https://zakon.rada.gov.ua/laws/show/1556-18#Text (дата звернення: 15.11.2025).

5. Klikh L., Zazymko O., Rudyk Ya. Exploring the formation of individual educational paths for bachelor's and master's students at NULES of Ukraine. Humanities Studios: Pedagogy, Psychology, Philosophy. 2023. Vol. 11(3). P. 52–60. URL: https://doi.org/10.31548/hspedagog14(3).2023.52-60.

6. Хто зібрав найбільше першокурсників: топ-10 університетів України. Dengi.ua. 04.09.2025. URL: https://dengi.ua/ua/career/9756135-khto-zibrav-najbilshe-pershokursnikiv-top-10-universitetiv-ukrayini (дата звернення: 15.11.2025).

7. Hlazunova O., Ponzel Y. Technologies and algorithms for the implementation of the recommendation system for creating an individual study plan for a higher education student. CEUR Workshop Proceedings. 2024. Vol. 3771. P. 110–117. URL: https://ceur-ws.org/Vol-3771/paper03.pdf (дата звернення: 15.11.2025).

8. Harvard Mobile App. Harvard University. URL: https://www.harvard.edu/community/ (дата звернення: 15.11.2025).

9. UCSF Mobile. Google Play. URL: https://play.google.com/store/apps/details?id=edu.ucsf.cls (дата звернення: 15.11.2025).

10. NAUgo. Built For Students – By Students! Northern Arizona University. URL: https://in.nau.edu/naugo/ (дата звернення: 15.11.2025).

11. CSUN Mobile App. California State University, Northridge. URL: https://www.csun.edu/it/software-services/services/csun-mobile-app (дата звернення: 15.11.2025).

12. Oiler Mobile. Google Play. URL: https://play.google.com/store/apps/details?id=com.blackboard.android.central.findlay (дата звернення: 15.11.2025).

13. THE UA Mobile APP. The University of Akron. URL: https://www.uakron.edu/mobile/ (дата звернення: 15.11.2025).

14. UH Go. Designed with your needs in mind. University of Houston. URL: https://www.uh.edu/go/ (дата звернення: 15.11.2025).

15. CSUSM Mobile App. California State University San Marcos. URL: https://www.csusm.edu/iits/services/mobile-app/index.html (дата звернення: 15.11.2025).

16. Posner M. S. Mobile App Developer: What New Features Do You Want. Cypress College. 09.04.2014. URL: https://www.cypresscollege.edu/2014/04/09/mobile-app-developer-what-new-features-do-you-want/ (дата звернення: 15.11.2025).

17. CIU Mobile. Google Play. URL: https://play.google.com/store/apps/details?id=tr.edu.ciu.ciumobile (дата звернення: 15.11.2025).

18. RU Mobile App. Radford University. URL: https://www.radford.edu/information-technology-services/index.html (дата звернення: 15.11.2025).

19. UCF Mobile. University of Central Florida. URL: https://ucfmobile.ucf.edu/ (дата звернення: 15.11.2025).

20. UNCG Mobile. The University of North Carolina at Greensboro. URL: https://www.uncg.edu/uncg-mobile/ (дата звернення: 15.11.2025).

21. Qatar University Mobile. Qatar University. URL: https://www.qu.edu.qa/ (дата звернення: 15.11.2025).

22. Mobile Apps for the Windsor Campus Community. University of Windsor. URL: https://www.uwindsor.ca/372824/mobile-apps-campus-community (дата звернення: 15.11.2025).

23. Istanbul Medipol University Mobile App. Medipol University. URL: https://www.medipol.edu.tr/en/ogrenci/our-mobile-application (дата звернення: 15.11.2025).

24. Teno - leading mobile app for school-administration. Aptoide. URL: https://teno.ua.aptoide.com/app (дата звернення: 15.11.2025).

25. School Management Mobile App. SchoolPlus. URL: https://www.schoolplusapp.com/ (дата звернення: 15.11.2025).

26. School Parent App – Simplify Operations and Boost School Success. HelloParent. URL: https://www.helloparent.com/ (дата звернення: 15.11.2025).

27. Як Дія, тільки в УжНУ: студенти ФМЦТ розробили мобільний застосунок UzhNU Information System. Медіацентр УжНУ. 26.04.2022. URL: https://mediacenter.uzhnu.edu.ua/news/iak-diia-tilky-v-uzhnu-studenty-fmtst-rozrobyly-mobilnyj-zastosunok-uzhnu-information-system/2022-04-26-51299 (дата звернення: 15.11.2025).

28. Університет у смартфоні: презентували додаток "KNU online". Укрінформ. 09.07.2020. URL: https://www.ukrinform.ua/rubric-technology/3060294-universitet-u-smartfoni-prezentuvali-dodatok-knu-online.html (дата звернення: 15.11.2025).

29. ЧДТУ – другий університет в Україні, що матиме власний мобільний застосунок для студентів. ЧДТУ. 14.07.2021. URL: https://chdtu.edu.ua/news/item/16441-chdtu-druhyi-universytet-v-ukraini-shcho-matyme-vlasnyi-mobilnyi-zastosunok-dlia-studentiv (дата звернення: 15.11.2025).

30. Твій університет в кишені: Розклад занять та вакансії для

студентів. Univera. URL: https://univera.app/uk/students (дата звернення: 15.11.2025).

31. Univera в парі з платформою Дія. Instagram. URL: https://www.instagram.com/diia.gov.ua/p/CzwFwRtNAMA/?img_index=2 (дата звернення: 15.11.2025).

32. Як додати студентський квиток у застосунок Дія? Дія. URL: https://paperless.diia.gov.ua/instruction/yak-dodati-studentskii-kvitok-u-zastosunok-diya (дата звернення: 15.11.2025).

33. Дія.Освіта для всіх. Дія.Освіта. URL: https://osvita.diia.gov.ua/ (дата звернення: 15.11.2025).

34. Мрія змінює українську освіту. Мрія. URL: https://mriia.gov.ua/app (дата звернення: 15.11.2025).

35. “UniSched” – універсальний помічник для викладачів та здобувачів освіти. Блог ЛНУ. URL: https://university-blog.lnu.edu.ua/story-16/ (дата звернення: 15.11.2025).

36. Застосунок «Мій ДонДУУ». KitApp. URL: https://kitapp.pro/uk/portfolio/mij-donduu/ (дата звернення: 15.11.2025).

37. AR Book. URL: https://arbook.info/ (дата звернення: 15.11.2025).

38. Turing A. M. Computing machinery and intelligence. Mind. 1950. Vol. 49. P. 433–460. URL: https://courses.cs.umbc.edu/471/papers/turing.pdf (дата звернення: 15.11.2025).

39. Погореленко А. К. Штучний інтелект: сутність, аналіз застосування, перспективи розвитку. URL: https://ej.journal.kspu.edu/index.php/ej/article/view/405/401 (дата звернення: 15.11.2025).

40. Баранов О. А. Визначення терміну “штучний інтелект”. Інформація і право. 2023. № 1(44). URL: https://doi.org/10.37750/2616-6798.2023.1(44).287537.

41. Петрів С. Ю. Штучний інтелект: переваги та недоліки. Штучний інтелект у правовій практиці: межі та можливості : Збірник тез Всеукраїнського круглого столу (15 березня 2024 року). URL:

https://dspace.lvduvs.edu.ua/bitstream/1234567890/7424/1/15_03_2024.pdf#page=141 (дата звернення: 15.11.2025).

42. Бриль І. В. Штучний інтелект в реаліях сучасності. Proceedings of the XVII International scientific and practical conference “System analysis and intelligent system for management” (Ankara, Turkey, May 02-05, 2023). URL: https://books.google.com.ua/books?id=UOG9EAAAQBAJ&pg=PA66 (дата звернення: 15.11.2025).

43. Колесніков А., Карапетян О. Штучний інтелект: переваги та загрози використання. Ефективна економіка. 2023. № 8. URL: https://www.nayka.com.ua/index.php/ee/article/view/1991 (дата звернення: 15.11.2025).

44. Каткова Т. Г. Штучний інтелект в україні: правові аспекти. Право і суспільство. 2020. № 6. URL: https://doi.org/10.32842/2078-3736/2020.6.1.8.

45. Buchholz K. The Companies With the Most AI Patents. Statista. 26.01.2023. URL: https://www.statista.com/chart/18211/companies-with-the-most-ai-patents/ (дата звернення: 15.11.2025).

46. Костенко О. В. Аналіз національних стратегій розвитку штучного інтелекту. Інформація і право. 2022. № 2(41). URL: https://doi.org/10.37750/2616-6798.2022.2(41).270365.

47. Бойові завдання в ЗСУ виконуватимуть дрони зі штучним інтелектом. Мілітарний. URL: https://militarnyi.com/uk/news/bojovi-zavdannya-v-zsu-vykonuvatymut-drony-zi-shtuchnym-intelektom/ (дата звернення: 15.11.2025).

48. DOD Announces Establishment of Generative AI Task Force. U.S. Department of Defense. URL: https://www.war.gov/News/Releases/Release/Article/3489803/dod-announces-establishment-of-generative-ai-task-force/ (дата звернення: 15.11.2025).

49. Хаустова В. Є., Решетняк О. І., Хаустов М. М., Зінченко В. А. Напрямки розвитку технологій штучного інтелекту в забезпеченні обороноздатності країни. Бізнес Інформ. 2022. № 3. С. 17–26. URL: https://doi.org/10.32983/2222-4459-2022-3-17-26.

50. Artificial Intelligence in Ukraine / The Expert Committee on the Development of Artificial Intelligence of Ukraine. 2021. URL: https://niss.gov.ua/sites/default/files/2021-10/ai_ukraine_v2.pdf (дата звернення: 15.11.2025).

51. Harvard designs AI sandbox that enables exploration, interaction without compromising security. The Harvard Gazette. URL: https://news.harvard.edu/gazette/story/newsplus/harvard-designs-ai-sandbox-that-enables-exploration-interaction-without-compromising-security/ (дата звернення: 15.11.2025).

52. What is machine learning (ML)? Google Cloud. URL: https://cloud.google.com/learn/what-is-machine-learning (дата звернення: 15.11.2025).

53. Zhou Z.-H., Liu S. Machine Learning. Springer, 2021. URL: https://doi.org/10.1007/978-981-15-1967-3.

54. Alpaydin E. Machine Learning, revised and updated edition. MIT Press, 2021. 280 p.

55. Bergmann D. What is machine learning? IBM. URL: https://www.ibm.com/think/topics/machine-learning (дата звернення: 15.11.2025).

56. Tufail S., Riggs H., Tariq M., Sarwat A. I. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. Electronics. 2023. Vol. 12(8). URL: https://doi.org/10.3390/electronics12081789.

57. Hardesty L. The history of Amazon's recommendation algorithm. Collaborative filtering and beyond. Amazon Science. 22.11.2019. URL: https://www.amazon.science/the-history-of-amazons-recommendation-algorithm (дата звернення: 15.11.2025).

58. Selimi D., Pireva Nuci K. The use of Recommender Systems in web technology and an in-depth analysis of Cold State problem. arXiv. 2020. URL: https://doi.org/10.48550/arXiv.2009.04780.

59. Amatriain X., Basilico J. Netflix Recommendations: Beyond the 5 stars

(Part 1). Netflix Technology Blog. 06.04.2012. URL: https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429 (дата звернення: 15.11.2025).

60. Amatriain X., Basilico J. Netflix Recommendations: Beyond the 5 stars (Part 2). Netflix Technology Blog. 20.06.2012. URL: https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5 (дата звернення: 15.11.2025).

61. Understanding recommendations on Spotify. Spotify. URL: https://www.spotify.com/ua-en/safetyandprivacy/understanding-recommendations (дата звернення: 15.11.2025).

62. Recommendations on YouTube. YouTube. URL: https://www.youtube.com/howyoutubeworks/recommendations/ (дата звернення: 15.11.2025).

63. Bhatia T. Recommendation Systems: Technology Review. 2018. URL: https://www.researchgate.net/publication/326539074_Recommendation_Systems_Technology_Review (дата звернення: 15.11.2025).

64. How TikTok recommends content. TikTok. URL: https://support.tiktok.com/en/using-tiktok/exploring-videos/how-tiktok-recommends-content (дата звернення: 15.11.2025).

65. Lada A., Wang M., Yan T. How Does News Feed Predict What You Want to See? Meta. 26.01.2021. URL: https://about.fb.com/news/2021/01/how-does-news-feed-predict-what-you-want-to-see/ (дата звернення: 15.11.2025).

66. Reddit’s Approach to Content Recommendations. Reddit. URL: https://support.reddithelp.com/hc/en-us/articles/23511859482388-Reddit-s-Approach-to-Content-Recommendations (дата звернення: 15.11.2025).

67. Twitter's Recommendation Algorithm. X Blog. 31.03.2023. URL: https://blog.x.com/engineering/en_us/topics/open-source/2023/twitter-recommendation-algorithm (дата звернення: 15.11.2025).

68. Brovman Y. M. eBay accelerates its AI-driven recommendation engine with Vertex AI. Google Cloud Blog. 08.03.2024. URL:

https://cloud.google.com/blog/products/ai-machine-learning/ebay-uses-vertex-ai-vector-search-for-recommendations (дата звернення: 15.11.2025).

69. Pets A., Levzhynskyi O. Under the Hood of the Grammarly Editor, Part Two: How Suggestions Work. Grammarly. 22.04.2022. URL: https://www.grammarly.com/blog/engineering/how-suggestions-work-grammarly-editor/ (дата звернення: 15.11.2025).

70. What is explainable AI? IBM. URL: https://www.ibm.com/think/topics/explainable-ai (дата звернення: 15.11.2025).

71. Turri V. What is Explainable AI? SEI Blog. 17.01.2022. URL: https://www.sei.cmu.edu/blog/what-is-explainable-ai/ (дата звернення: 15.11.2025).

72. Ali S., Abuhmed T., El-Sappagh S., et al. Explainable Artificial Intelligence (XAI): What we know and what is left to attain. Information Fusion. 2023. Vol. 99. URL: https://doi.org/10.1016/j.inffus.2023.101805.

73. Phillips P. J., Hahn C. A., Fontana P. C., Yates A. N., Greene K., Broniatowski D. A., Przybocki M. A. Four Principles of Explainable Artificial Intelligence. NIST. 2021. URL: https://doi.org/10.6028/NIST.IR.8312.

74. Holzinger A., Saranti A., Molnar C., Biecek P., Samek W. Explainable AI Methods - A Brief Overview. URL: https://iphome.hhi.de/samek/pdf/HolXXAI22b.pdf (дата звернення: 15.11.2025).

75. Caffo B. S., D'Asaro F. A., Garcez A., Raffinetti E. Editorial: Explainable artificial intelligence models and methods in finance and healthcare. Frontiers in Artificial Intelligence. 2022. Vol. 5. URL: https://doi.org/10.3389/frai.2022.970246.

76. Durán J. M. Dissecting scientific explanation in AI (sXAI): A case for medicine and healthcare. Artificial Intelligence. 2021. Vol. 297. URL: https://doi.org/10.1016/j.artint.2021.103498.

77. Approximating the Core of Cooperative Games. DeepMind. 01.03.2024. URL: https://deepmind.google/research/publications/approximating-the-core-of-cooperative-games/ (дата звернення: 15.11.2025).

78. Better Insights on Machine Learning Models with xAI in Analytics

Workbench 3.0. FICO. URL: https://community.fico.com/s/analytics-workbench-xai (дата звернення: 15.11.2025).

79. Proven AI for a Thriving Lending Ecosystem. Zest.ai. URL: https://www.zest.ai/ (дата звернення: 15.11.2025).

80. Khosravi H., Buckingham Shum S., Chen G., et al. Explainable Artificial Intelligence in education. Computers and Education: Artificial Intelligence. 2022. Vol. 3. URL: https://doi.org/10.1016/j.caeai.2022.100074.

81. Insights from the community workshop on explainable AI in education. European Commission. 07.11.2024. URL: https://education.ec.europa.eu/lv/news/insights-from-the-community-workshop-on-explainable-ai-in-education (дата звернення: 15.11.2025).

82. He R. PinSage: A new graph convolutional neural network for web-scale recommender systems. Medium. 15.08.2018. URL: https://medium.com/pinterest-engineering/pinsage-a-new-graph-convolutional-neural-network-for-web-scale-recommender-systems-88795a107f48 (дата звернення: 15.11.2025).

83. Food Discovery with Uber Eats: Recommending for the Marketplace. Uber Blog. 10.09.2018. URL: https://www.uber.com/en-UA/blog/uber-eats-recommending-marketplace/ (дата звернення: 15.11.2025).

84. Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations. Uber Blog. 04.12.2019. URL: https://www.uber.com/en-UA/blog/uber-eats-graph-learning/ (дата звернення: 15.11.2025).

85. Innovative Recommendation Applications Using Two Tower Embeddings at Uber. Uber Blog. 26.07.2023. URL: https://www.uber.com/en-UA/blog/innovative-recommendation-applications-using-two-tower-embeddings/ (дата звернення: 15.11.2025).

86. Yelp Recommendation software. Yelp. URL: https://trust.yelp.com/recommendation-software/ (дата звернення: 15.11.2025).

87. Fitbit software adds smarter, personalized guidance. Google Blog. 05.01.2017. URL: https://blog.google/products/fitbit/fitbit-software-adds-smarter-

personalized-guidance/ (дата звернення: 15.11.2025).

88. Chung K. Announcing Goodreads Personalized Recommendations. Goodreads. 15.09.2011. URL: https://www.goodreads.com/blog/show/303-announcing-goodreads-personalized-recommendations (дата звернення: 15.11.2025).

89. What to Watch FAQ. IMDb. 02.09.2025. URL: https://help.imdb.com/article/imdb/discover-watch/what-to-watch-faq/GPZ2RSPB3CPVL86Z (дата звернення: 15.11.2025).

90. Представляємо вам інтерактивного радника Steam. Steam. URL: https://store.steampowered.com/news/app/593110/view/1716373422378712840 (дата звернення: 15.11.2025).

91. Hall R. Personalized Recommendations at Etsy. CodeAsCraft. 17.11.2014. URL: https://www.etsy.com/codeascraft/personalized-recommendations-at-etsy (дата звернення: 15.11.2025).

92. Advanced machine learning helps Play Store users discover personalised apps. DeepMind. 18.11.2019. URL: https://deepmind.google/blog/advanced-machine-learning-helps-play-store-users-discover-personalised-apps/ (дата звернення: 15.11.2025).

93. Personalize your Google Play Store experience. Google Play Help. URL: https://support.google.com/googleplay/answer/13780792 (дата звернення: 15.11.2025).

94. Stream recommended songs from Apple Music on the web. Apple Music. URL: https://support.apple.com/en-mk/guide/music-web/apdm6e2944a5/web (дата звернення: 15.11.2025).

95. Wall S., Schellmann H. LinkedIn’s job-matching AI was biased. The company’s solution? More AI. MIT Technology Review. 23.06.2021. URL: https://www.technologyreview.com/2021/06/23/1026825/linkedin-ai-bias-ziprecruiter-monster-artificial-intelligence/ (дата звернення: 15.11.2025).

96. Rao J., Borchers C., Lin J. Coursera-REC: Explainable MOOCs Course Recommendation using RAG-facilitated LLMs. 2024. URL:

https://doi.org/10.35542/osf.io/dnf7r.

97. De Medio C., Limongelli C., Sciarrone F., Temperini M. MoodleREC: A recommendation system for creating courses using the moodle elearning platform. Computers & Education. 2020. Vol. 149. URL: https://doi.org/10.1016/j.chb.2019.106168.

98. Olawin R. My time at Quizlet as a Data Science, Machine Learning Intern. Medium. 06.10.2022. URL: https://medium.com/tech-quizlet/my-time-at-quizlet-as-a-data-science-machine-learning-intern-1a8f90de8fe0 (дата звернення: 15.11.2025).

99. Poluriezov D. Від концепції до реалізації: як побудувати ефективну рекомендаційну систему на основі машинного навчання за два дні. DOU. 12.02.2024. URL: https://dou.ua/forums/topic/47504/ (дата звернення: 15.11.2025).

100. Recommender systems for teachers. Prometheus-X. URL: https://prometheus-x.org/uc-recommender-systems-for-teachers/ (дата звернення: 15.11.2025).

101. Bodnar L., Shulakova K., Gryzun L. Algorithmic support of the web service recommendation system for learning foreign languages. Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies. 2021. № 2 (6). P. 100–106. URL: https://doi.org/10.20998/2079-0023.2021.02.16.

102. Fedushko S., Ustyianovych T., Syerov Y. Intelligent Academic Specialties Selection in Higher Education for Ukrainian Entrants: A Recommendation System. Journal of Intelligence. 2022. Vol. 10(2). URL: https://doi.org/10.3390/jintelligence10020032.

103. Chornous G., Lem T. Developing hybrid recommendation systems: Ukrainian dimension. Access to science, business, innovation in digital economy. 2022. Vol. 3(2). P. 89–106. URL: https://doi.org/10.46656/access.2022.3.2(1).

104. Sundar M. R., Shreya G., Jawahar T. Course Recommendation System. International Journal of Computer Applications. 2020. Vol. 175(29). URL: https://www.ijcaonline.org/archives/volume175/number29/sundari-2020-ijca-

920823.pdf (дата звернення: 15.11.2025).

105. Жежерун О. П., Яремко С. А. Побудова рекомендаційних систем на основі онтологій. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/0b66951c-94d4-4294-86bb-f4aec2636274/content (дата звернення: 15.11.2025).

106. Паулін О. М., Комлева Г. О., Улізко Г. В. Рекомендаційна система для допомоги у вивченні музичних творів. Технології та техніка. 2020. № 5. URL: https://doi.org/10.32838/2663-5941/2020.5/16.

107. Castells P., Jannach D. Recommender Systems: A Primer. arXiv. 2023. URL: https://doi.org/10.48550/arXiv.2302.02579.

108. Jannach D., Pu P., Ricci F., Zanker M. Recommender Systems: Past, Present, Future. AI Magazine. 2021. Vol. 42(3). URL: https://doi.org/10.1609/aimag.v42i3.18139.

109. Клопов І. О., Стеценко М. К., Бобро Д. В. Методи рекомендаційних систем. URL: https://api.dspace.wunu.edu.ua/api/core/bitstreams/eda0903f-c388-4637-9b2c-aeea22cf71fa/content (дата звернення: 15.11.2025).

110. Zhang Q., Lu J., Zhang G. Q. Recommender Systems in E-learning. Journal of Smart Environment and Green Computing. 2021. Vol. 1. P. 76–89. URL: http://dx.doi.org/10.20517/jsegc.2020.06.

111. Ali M., Yousaf M., Behlol G. Artificial Intelligence in Learning Management System: A Case Study of the Students of Mass Communication. Voyage Journal of Educational Studies. 2023. Vol. 3(2). P. 92–114. URL: https://doi.org/10.58622/vjes.v3i2.52.

112. Hashemi-Pour C., Kirvan P., Brush K. What is a learning management system (LMS)? TechTarget. 22.10.2024. URL: https://www.techtarget.com/searchcio/definition/learning-management-system (дата звернення: 15.11.2025).

113. Sharable Content Object Reference Model (SCORM®). Advanced Distributed Learning (ADL) Initiative. URL: https://www.adlnet.gov/past-projects/scorm/ (дата звернення: 15.11.2025).

114. Singh T. M., Reddy C. K. K., Murthy B. V. R., Nag A., Doss S. AI and Education: Bridging the Gap to Personalized, Efficient, and Accessible Learning. Internet of Behavior-Based Computational Intelligence for Smart Education Systems. IGI Global, 2024. URL: https://doi.org/10.4018/979-8-3693-8151-9.ch005.

115. Ситник О. Ю., Дубровський С. С. Особливості розвитку ринку інформаційних технологій в Україні. Економічні горизонти. 2022. № 3(21). С. 72–82. URL: https://doi.org/10.31499/2616-5236.3(21).2022.263688.

116. Стратегія розвитку вищої освіти в україні на 2021–2031 роки. МОН України. URL: https://mon.gov.ua/static-objects/mon/sites/1/rizne/2020/09/25/rozvitku-vishchoi-osviti-v-ukraini-02-10-2020.pdf (дата звернення: 15.11.2025).

117. Collaborative Filtering in Recommender System: An Overview. Medium. URL: https://medium.com/@evelyn.eve.9512/collaborative-filtering-in-recommender-system-an-overview-38dfa8462b61 (дата звернення: 15.11.2025).

118. Matrix factorization. Google Developers. URL: https://developers.google.com/machine-learning/recommendation/collaborative/matrix (дата звернення: 15.11.2025).

119. Approximated Matrix. ScienceDirect. URL: https://www.sciencedirect.com/topics/computer-science/approximated-matrix (дата звернення: 15.11.2025).

120. Bergmann D., Stryker C. What is a loss function? IBM. URL: https://www.ibm.com/think/topics/loss-function (дата звернення: 15.11.2025).

121. Mean-Squared-Error. ScienceDirect. URL: https://www.sciencedirect.com/topics/engineering/mean-squared-error (дата звернення: 15.11.2025).

122. Google Developers. URL: https://developers.google.com/machine-learning/crash-course/descending-into-ml/loss (дата звернення: 15.11.2025).

123. Supervised Learning. Stanford University. URL: https://cs229.stanford.edu/notes2022fall/main_notes.pdf (дата звернення: 15.11.2025).

124. Frobenius Norm. ScienceDirect. URL:

https://www.sciencedirect.com/topics/mathematics/frobenius-norm (дата звернення: 15.11.2025).

125. Яремчук Н. А., Редьога О. Ю., Сікоза О. М. Особливості арифметизації дискретних вербальних шкал. URL: https://ela.kpi.ua/server/api/core/bitstreams/8c59d2a8-4bc1-4caa-91df-39024f503020/content (дата звернення: 15.11.2025).

126. Herzog R., Köhne F., Kreis L., Schiela A. Frobenius-type norms and inner products of matrices and linear maps with applications to neural network training. arXiv. 2023. URL: https://arxiv.org/pdf/2311.15419 (дата звернення: 15.11.2025).

127. Chen H., Chen X., Elmasri M., Sun Q. Gradient descent in matrix factorization: Understanding large initialization. arXiv. 2023. URL: https://arxiv.org/pdf/2305.19206 (дата звернення: 15.11.2025).

128. Explicit Matrix Factorization: ALS, SGD, and All That Jazz. Medium. 16.03.2016. URL: https://medium.com/insight-data/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea (дата звернення: 15.11.2025).

129. Patel D., Patel F., Chauhan U. Recommendation Systems: Types, Applications, and Challenges. International Journal of Computing and Digital Systems. 2023. Vol. 13(1). P. 2210–142. URL: https://doi.org/10.12785/ijcds/130168.

130. Content-based Filtering. ScienceDirect. URL: https://www.sciencedirect.com/topics/computer-science/content-based-filtering (дата звернення: 15.11.2025).

131. Козенко О. В., Мазурець О. В., Молчанова М. О., Собко О. В. Використання метрик косинусної схожості та індексу жаккара для інтелектуального аналізу семантичної подібності текстових документів. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/ec2a6344-e543-48e1-8c2d-408bab1160b/content (дата звернення: 15.11.2025).

132. Zhou Z.-H. Learnware: On the Future of Machine Learning. Frontiers of Computer Science. 2016. URL: https://www.lamda.nju.edu.cn/publication/fcs16learnware.pdf (дата звернення: 15.11.2025).

133. Grzybowski A., Jin K., Wu H. Pros and Cons of Machine Learning. Ophthalmology Management. 2024. Vol. 28. P. 20–21, 38. URL: https://www.ophthalmologymanagement.com/issues/2024/januaryfebruary/pros-and-cons-of-machine-learning (дата звернення: 15.11.2025).

134. Mahmood S. Restrictions, Challenges and Opportunities for AI and ML. Ijist Jour. 2023. URL: https://www.researchgate.net/publication/379897214_Restrictions_Challenges_and_Opportunities_for_AI_and_ML (дата звернення: 15.11.2025).

135. Williamson S. M., Prybutok V. The Era of Artificial Intelligence Deception: Unraveling the Complexities of False Realities and Emerging Threats of Misinformation. Information. 2024. Vol. 15(6). URL: https://doi.org/10.3390/info15060299.

136. Mahesh B. Machine Learning Algorithms – A Review. International Journal of Science and Research. 2019. URL: https://www.researchgate.net/publication/344717762_Machine_Learning_Algorithms_-A_Review (дата звернення: 15.11.2025).

137. Rahaman M. J. A Comprehensive Review to Understand the Definitions, Advantages, Disadvantages and Applications of Machine Learning Algorithms. International Journal of Computer Applications. 2024. Vol. 186(31). P. 43–47. URL: https://doi.org/10.5120/ijca2024923868.

138. Kavlakoglu E. What are Naïve Bayes classifiers. IBM. URL: https://www.ibm.com/think/topics/naive-bayes (дата звернення: 15.11.2025).

139. Kavlakoglu E. What are support vector machines (SVMs)? IBM. URL: https://www.ibm.com/think/topics/support-vector-machine (дата звернення: 15.11.2025).

140. What is linear regression? IBM. URL: https://www.ibm.com/think/topics/linear-regression (дата звернення: 15.11.2025).

141. Lee F. What is logistic regression? IBM. URL: https://www.ibm.com/think/topics/logistic-regression (дата звернення: 15.11.2025).

142. What is k-means clustering? IBM. URL:

https://www.ibm.com/think/topics/k-means-clustering (дата звернення: 15.11.2025).

143. Sarker I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Computer Science. 2021. Vol. 2. Art. 160. URL: https://doi.org/10.1007/s42979-021-00592-x.

144. What is principal component analysis (PCA)? IBM. URL: https://www.ibm.com/think/topics/principal-component-analysis (дата звернення: 15.11.2025).

145. What is the Apriori algorithm? IBM. URL: https://www.ibm.com/think/topics/apriori-algorithm (дата звернення: 15.11.2025).

146. Wang Y., Huang S.-T. Training TSVM with the proper number of positive samples. Pattern Recognition Letters. 2005. URL: https://doi.org/10.1016/j.patrec.2005.03.034.

147. What is a generative model? IBM. URL: https://www.ibm.com/think/topics/generative-model (дата звернення: 15.11.2025).

148. What is generative AI? IBM Research. URL: https://research.ibm.com/blog/what-is-generative-AI (дата звернення: 15.11.2025).

149. Toledo Jr. T. A Gentle Introduction to Self-Learning With Python Implementation. Towards Data Science. 25.10.2022. URL: https://towardsdatascience.com/a-gentle-introduction-to-self-learning-5d6d40349f7c (дата звернення: 15.11.2025).

150. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. Second edition. A Bradford Book, 2015. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (дата звернення: 15.11.2025).

151. Yu J., Dai Y., Liu X., et al. Unleashing the Power of Multi-Task Learning: A Comprehensive Survey Spanning Traditional, Deep, and Pretrained Foundation Model Eras. arXiv. 2024. URL: https://arxiv.org/pdf/2404.18961 (дата звернення: 15.11.2025).

152. Mohammed A., Kora R. A comprehensive review on ensemble deep learning: Opportunities and challenges. Journal of King Saud University - Computer

and Information Sciences. 2023. Vol. 35(2). P. 757–774. URL: https://doi.org/10.1016/j.jksuci.2023.01.014.

153. Hardesty L. Explained: Neural networks. MIT News. 14.04.2017. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (дата звернення: 15.11.2025).

154. Aha D. W., Kibler D., Albert M. K. Instance-Based Learning Algorithms. Machine Learning. 1991. Vol. 6(1). P. 37–66. URL: https://doi.org/10.1023/A:1022689900470.

155. Injadat M. A. M. Summary of Challenges and ML Techniques. ResearchGate. URL: https://www.researchgate.net/figure/Summary-of-Challenges-and-ML-Techniques_fig6_348402802 (дата звернення: 15.11.2025).

156. Nehrey M., Hnot T. What is Data Filtering. Data Science Tools Application for Business Processes Modelling in Aviation. IGI Global, 2019. P. 15. URL: https://doi.org/10.4018/978-1-5225-7588-7.ch006.

157. Торубка Т. В., Пуйда В. Я. Метод видалення адитивних шумів на зображеннях літальних апаратів. 2013. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/nov/6700/22131-136.pdf (дата звернення: 15.11.2025).

158. Гавриленко С., Зозуля В. Дослідження методів виявлення аномалій на етапі попередньої обробки даних. Системи управління навігації та зв'язку. 2022. № 1(67). С. 52–56. URL: https://doi.org/10.26906/SUNZ.2022.1.052.

159. Mishchuk O., Tkachenko R. Методи оброблення та заповнення пропущених параметрів у даних екологічного моніторингу. Scientific Bulletin of UNFU. 2019. Vol. 29(6). URL: https://doi.org/10.15421/40290623.

160. Deineha O. Редукція термів лямбда-числення: оцінка прогнозивних здатностей LLM. Information technology and society. 2024. URL: https://doi.org/10.32689/maup.it.2024.1.7.

161. Лавренюк М. С., Новіков О. М. Огляд методів машинного навчання для класифікації великих обсягів супутникових даних. Системи обробки інформації. 2018. № 1. URL: https://doi.org/10.20535/SRIT.2308-8893.2018.1.04.

162. Дорошенко І., Книгніцька Т., Дереторська Т. Порівняння алгоритмів машинного навчання для прогнозування смертності від вірусу COVID-19. SWorldJournal. 2022. URL: https://doi.org/10.30888/2663-5712.2022-11-02-045.

163. Основні поняття кластеризації та постановка задачі. КНУ. URL: https://csc.knu.ua/media/study/asp/mod_probl_inf_tech_sys_analysis_ivohin/lecture/lec11.pdf (дата звернення: 15.11.2025).

164. Martin-Clemente R., Zarzoso V. LDA via L1-PCA of Whitened Data. IEEE Transactions on Signal Processing. 2019. URL: https://doi.org/10.1109/TSP.2019.2955860.

165. Grzechca D., Rybka P. Level Crossing Barrier Machine Faults and Anomaly Detection with the Use of Motor Current Waveform Analysis. Energies. 2021. Vol. 14(11). URL: https://doi.org/10.3390/en14113206.

166. Li L., Zhang Z., Zhang S. Hybrid Algorithm Based on Content and Collaborative Filtering in Recommendation System Optimization and Simulation. Scientific Programming. 2021. URL: https://doi.org/10.1155/2021/7427409.

167. Aberbach H., Jeghal A., My Abdelouahed S., Tairi H. E-learning Recommendation Systems: A Literature Review. Digital Technologies and Applications. 2022. P. 361–370. URL: https://doi.org/10.1007/978-3-031-01942-5_36.

168. Idakwo J., Agbogun J. B., Kolajo T. A Survey on Recommendation System Techniques. UMYU Scientifica. 2023. Vol. 2(2). P. 112–119. URL: https://doi.org/10.56919/usci.2322.012.

169. Ghazanfar M. A., Prugel-Bennett A. A Scalable, Accurate Hybrid Recommender System. 2010 Third International Conference on Knowledge Discovery and Data Mining. 2010. URL: https://doi.org/10.1109/WKDD.2010.117.

170. Meleshko Y. Методи оцінки якості роботи рекомендаційних систем. Системи управління навігації та зв’язку. 2018. № 5(51). С. 92–97. URL: https://doi.org/10.26906/SUNZ.2018.5.092.

171. Roy D., Dutta M. A systematic review and research perspective on

recommender systems. Journal of Big Data. 2022. Vol. 9. Art. 59. URL: https://doi.org/10.1186/s40537-022-00592-5.

172. Mohamed M. H., Khafagy M., Ibrahim M. H. Recommender Systems Challenges and Solutions Survey. 2019 International Conference on Innovative Trends in Computer Engineering (ITCE). 2019. URL: https://doi.org/10.1109/ITCE.2019.8646645.

173. Cutting V., Stephen N. A Review on using Python as a Preferred Programming Language for Beginners. International Research Journal of Engineering and Technology (IRJET). 2021. Vol. 08(08). URL: https://www.irjet.net/archives/V8/i8/IRJET-V8I8505.pdf (дата звернення: 15.11.2025).

174. Sahoo D., Nayak S. C., Paul N. R. Designed Framework for Advanced Intelligent Job Recommendation System. 2023 OITS International Conference on Information Technology (OCIT). 2023. URL: https://doi.org/10.1109/OCIT59427.2023.10430867.

175. Васильєв О. М. Програмування мовою Java. URL: https://books.google.com.ua/books?id=D1lcEAAAQBAJ (дата звернення: 15.11.2025).

176. Bagchi S. Performance and Quality Assessment of Similarity Measures in Collaborative Filtering Using Mahout. Procedia Computer Science. 2015. Vol. 50. P. 229–234. URL: https://doi.org/10.1016/j.procs.2015.04.055.

177. Naika A., Samant L. Correlation review of classification algorithm using data mining tool: WEKA, Rapidminer , Tanagra, Orange and Knime. International Conference on Computational Modeling and Security (CMS 2016). 2016.

178. Gibson A., Nicholson C., Patterson J., Warrick M. Deeplearning4j: Distributed, open-source deep learning for Java and Scala on Hadoop and Spark. 2016. URL: https://doi.org/10.6084/M9.FIGSHARE.3362644.V2.

179. Kirch-Prinz U., Prinz P. A Complete Guide to Programming in C++. Jones and Bartlett Publishers, Inc., 2002. URL:

https://www.idpoisson.fr/volkov/C++.pdf (дата звернення: 15.11.2025).

180. Chanda, Aggarwal R. K. Job Recommendation System Implementation in Python vs. C++. International Journal of Recent Technology and Engineering (IJRTE). 2019. Vol. 8(4). P. 2299–2302. URL: https://doi.org/10.35940/ijrte.D8132.118419.

181. Abadi M., Barham P., Chen J., Chen Z. TensorFlow: A system for large-scale machine learning. arXiv. 2016. URL: https://doi.org/10.48550/arXiv.1605.08695.

182. Hladyshev D., Brodskyi M., Lisnykh L. Technical and agricultural sciences in modern realities: problems, prospects and solutions. 2023. URL: https://books.google.com.ua/books?id=uOG9EAAAQBAJ (дата звернення: 15.11.2025).

183. Hahsler M. recommenderlab: An R Framework for Developing and Testing Recommendation Algorithms. 2022. URL: https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf (дата звернення: 15.11.2025).

184. Karim M. R. Machine Learning with Scala Quick Start Guide. 2019.

185. Oni S. Introduction To Recommendation system In Javascript. Becoming Human. 18.04.2018. URL: https://becominghuman.ai/introduction-to-recommendation-system-in-javascript-74209c7ff2f7 (дата звернення: 15.11.2025).

186. Hrnjica B., Mušić D., Softic S. Development of recommender systems using ML.NET. Conference: Development of recommender systems using ML.NET. 2019.

187. .NET dependency injection. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/core/extensions/dependency-injection (дата звернення: 15.11.2025).

188. MatrixFactorizationTrainer.Options Class. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.matrixfactorizationtrainer.options?view=ml-dotnet-preview (дата звернення: 15.11.2025).

189. Tutorial: Build a movie recommender using matrix factorization with ML.NET. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/tutorials/movie-recommendation (дата звернення: 15.11.2025).

190. Amatriain X., Basilico J. System Architectures for Personalization and Recommendation. Netflix Technology Blog. 27.03.2013. URL: https://netflixtechblog.com/system-architectures-for-personalization-and-recommendation-e081aa94b5d8 (дата звернення: 15.11.2025).

191. Karbhari V. Spotify Recommendation Platform. Medium. 23.06.2021. URL: https://medium.com/acing-ai/spotify-recommendation-platform-fdd58ed4f99e (дата звернення: 15.11.2025).

192. Goodrow C. On YouTube's recommendation system. YouTube Blog. 15.09.2021. URL: https://blog.youtube/inside-youtube/on-youtubes-recommendation-system/ (дата звернення: 15.11.2025).

193. Lin N. Architecture and Application of Interative Transformer-based Movie Recommendation System. Medium. 10.10.2023. URL: https://medium.com/@nelsonlin0321/architecture-of-react-interactive-movie-recommendation-system-based-on-behavior-sequence-1b1597d8981e (дата звернення: 15.11.2025).

194. Nurcahya A., Supriyanto S. Content-based recommender system architecture for similar e-commerce products. Jurnal Informatika. 2020. Vol. 14(3). P. 90. URL: https://doi.org/10.26555/jifo.v14i3.a18511.

195. Небесний Р. М. Рекомендаційна система формування команд виконавців з відповідними фаховими компетентностями. URL: https://m.tntu.edu.ua/storage/pages/00000969/dissertation_nebesny.pdf (дата звернення: 15.11.2025).

196. Кулягін А. І. Нейромережні методи створення рекомендаційних систем для інтерактивного мистецтва з використанням доповненої реальності. 2024. URL: https://scholar.google.com/citations?view_op=view_citation&hl=uk&user=aF8i0LQAAAAJ&citation_for_view=aF8i0LQAAAAJ:d1gkVwhDpl0C (дата звернення: 15.11.2025).

197. Small A. Mobile vs Desktop (vs Tablet) Activities: Consumer and Business Statistics in 2023. Martech Zone. 05.05.2023. URL: https://martech.zone/mobile-vs-tablet-vs-desktop-statistics/ (дата звернення: 15.11.2025).

198. What is an Event-Driven Architecture? AWS. URL: https://aws.amazon.com/ru/event-driven-architecture/ (дата звернення: 15.11.2025).

199. Yaseen H. K., Obaid A. M. Big Data: Definition, Architecture & Applications. JOIV International Journal on Informatics Visualization. 2020. Vol. 4(1). URL: https://doi.org/10.30630/joiv.4.1.292.

200. Бойко Н. І. Еволюція побудови архітектур інформаційних систем. перспективи розвитку «хмарної» архітектури. 2015. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2018/jun/12945/24-348-367.pdf (дата звернення: 15.11.2025).

201. Скляренко О. В., Савченко Я. О., Литвиненко Л. О., Сушинський О. Є. Архітектурні підходи до розробки масштабованих веб-застосунків. Кібербезпека: освіта, наука, техніка. 2024. № 4 (24). URL: https://www.csecurity.kubg.edu.ua/index.php/journal/article/view/613/485 (дата звернення: 15.11.2025).

202. What is a relational database? Google Cloud. URL: https://cloud.google.com/learn/what-is-a-relational-database (дата звернення: 15.11.2025).

203. Letkowski J. Doing database design with MySQL. 2015. URL: https://www.researchgate.net/publication/271910489_Doing_database_design_with_MySQL (дата звернення: 15.11.2025).

204. Ganesh M., Albert S. A. R., Swamidason I. T. J. An Analysis of the Significance of Spring Boot in The Market. 2022 International Conference on Inventive Computation Technologies (ICICT). 2022. URL: https://doi.org/10.1109/ICICT54344.2022.9850910.

205. Shah H., Soomro T. R. Node.js Challenges in Implementation. Global Journal of Computer Science and Technology. 2017. Vol. 17(2). P. 72–83.

206. Idris N., Foozy C. F. M. A Generic Review of Web Technology: DJango and Flask. International Journal of Advanced Science Computing and Engineering. 2021. Vol. 2(1). P. 34–40. URL: https://doi.org/10.30630/ijasce.2.3.29.

207. Пономарьов І. В., Кравець А. В. The methodology for developing web applications on the platform ASP.NET CORE. System technologies. 2020. Vol. 1(126). P. 111–117.

208. What is .NET MAUI? Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/maui/what-is-maui?view=net-maui-9.0 (дата звернення: 15.11.2025).

209. Awan M. T. Linux vs. Windows: A Comparison of Two Widely Used Platforms. Journal of Computer Science and Technology Studies. 2022. Vol. 4(1). P. 41–53. URL: https://doi.org/10.32996/jcsts.2022.4.1.4.

210. Skalka J., Kapusta J., Benko Ľ. JavaScript Fundamentals. 2021. URL: https://doi.org/10.17846/FPVaI-2021-19.

211. Freeman A., Sanderson S. jQuery. 2011. URL: https://doi.org/10.1007/978-1-4302-3405-0_20.

212. Ambler T., Cloud N. Knockout. JavaScript Frameworks for Modern Web Dev. 2015. P. 121–153. URL: https://doi.org/10.1007/978-1-4842-0662-1_7.

213. Sahani A. K., Singh P., Jeyamani V. Web Development Using Angular: A Case Study. Journal of Informatics Electrical and Electronics Engineering (JIEEE). 2020. Vol. 1(2). P. 1–7. URL: https://doi.org/10.54060/JIEEE/001.02.005.

214. Chen S., Thaduri U. R., Ballamudi V. K. R. Front-End Development in React: An Overview. Engineering International. 2019. Vol. 7(2). P. 117–126. URL: https://doi.org/10.18034/ei.v7i2.662.

215. Stasko S. Розробка веб-застосунку для моніторингу власних коштів monobank користувачів на основі VUE.JS та NODE.JS. 2024.

216. Emmanni P. S. Comparative Analysis of Angular, React, and Vue.js in Single Page Application Development. International Journal of Science and Research (IJSR). 2023. Vol. 12(6). URL: https://doi.org/10.21275/SR24401230015.

217. Dymora P., Mazurek M., Nycz M. Comparison of Angular, React, and

Vue Technologies in the Process of Creating Web Applications on the User Interface Side. Journal of Engineering Technology and Computer Science. 2023. URL: https://doi.org/10.15584/jetacomps.2023.4.21.

218. Skrzypiec S., Plechawska-Wójcik M. Comparative analysis of Angular and React development frameworks. Journal of Computer Sciences Institute. 2023. Vol. 28. P. 256–263. URL: https://doi.org/10.35784/jcsi.3724.

219. Al-Shehari T., Zhioua S. An empirical study of web browsers' resistance to traffic analysis and website fingerprinting attacks. Cluster Computing. 2018. Vol. 21(2). URL: https://doi.org/10.1007/s10586-018-2817-4.

220. Hernández del Olmo F., Gaudioso E. Evaluation of recommender systems: A new approach. Expert Systems with Applications. 2008. Vol. 35(3). P. 790–804. URL: https://doi.org/10.1016/j.eswa.2007.07.047.

# ДОДАТОК А

## Документ впровадження основних результатів дисертаційної роботи

**ЗАТВЕРДЖУЮ**

**Проректор з наукової роботи та інноваційної діяльності Національного університету біоресурсів і природокористування України, доктор сільськогосподарських наук, професор**

**_____ Оксана ТОНХА**

**«__» _____ 2025 р.**

М. П.

**А К Т**
**про впровадження у виробництво**
**результатів дисертації на здобуття ступеня доктора філософії**

Цим актом стверджується, що результати дисертації на тему: «Рекомендаційна система для побудови індивідуального плану навчання здобувача вищої освіти»,
(назва теми)

яку представлено на здобуття ступеня доктора філософії з галузі знань «Інформаційні технології» та спеціальності 122 «Комп'ютерні науки», виконаної Понзелем Ярославом Юрійовичем
(ПІБ здобувача)

впроваджено у Національному університеті біоресурсів і природокористування України.
(назва підприємства, де здійснювалось впровадження)

1. Вид впроваджуваних результатів Комп'ютерна програма
(методика, рекомендації, пропозиції, модель, експериментальні дані тощо)
«Модель персоналізованих рекомендацій у цифрових сервісах підтримки користувачів».

2. Новизна отриманих результатів Авторське право на твір № 138537
(патенти, авторські свідоцтва тощо)
.

3 Значущість отриманих результатів впровадження сучасних інформаційних
(економічний, соціальний, науково-технічний ефект)
технологій у систему вищої освіти; розробка й апробація нових алгоритмів рекомендаційного типу, орієнтованих на освітню сферу; створення основи для подальших досліджень у галузі інтелектуальних освітніх систем та цифрової трансформації університетів .

4. Зв'язок роботи з науковими програмами, планами, темами
(назва, № державної реєстрації)
НДР №: 110/11-пр-2020 «Створення моделі гібридного веб-орієнтованого середовища доставки навчального контенту в умовах відкритої університетської освіти».

Начальник науково-дослідної частини НУБіП України

Володимир Отченашко
(підпис) (Ім'я, ПРІЗВИЩЕ)
«6» жовтня ____ р.

Керівник інформаційно-обчислювального центру, де безпосередньо впроваджені результати дисертації

Віктор Теплюк
(підпис) (Ім'я, ПРІЗВИЩЕ)
«6» жовтня 2025 р.

Директор НДІ (Заступник декана з наукової роботи)

_____ Володимир Кравченко
(підпис) (Ім'я, ПРІЗВИЩЕ)

«6» лютого 2026 р.

Здобувач

_____ Ярослав Понзель
(підпис) (Ім'я, ПРІЗВИЩЕ)

«6» лютого 2026 р.

# ДОДАТОК Б

## Свідотцтво про реєстрацію авторського права на твір

УКРАЇНА

# СВІДОЦТВО

**про реєстрацію авторського права на твір**

**№ 138537**

**Комп'ютерна програма «Модель персоналізованих рекомендацій у цифрових сервісах підтримки користувачів»**

(вид, назва твору)

Автор (співавтори) **Понзель Ярослав Юрійович, Глазунова Олена Григорівна**

(прізвище, ім'я, по батькові (за наявності), псевдонім (за наявності))

Авторські майнові права належать повністю **Національний університет біоресурсів і природокористування України, вул. Героїв Оборони, 15, м. Київ, 03041**

(прізвище, ім'я, по батькові (за наявності) фізичної особи / найменування юридичної особи, адреса)

Дата реєстрації 11 серпня 2025 р.

**Виконувач обов'язків Директора Державної організації «Український національний офіс інтелектуальної власності та інновацій»**

**Ігор ПАРЕНЧУК**

УКРНОІВІ

# ДОДАТОК В

## Список опублікованих праць за темою дисертації

**Статті у наукових виданнях,**
**включених до міжнародних наукометричних баз даних**
**Web of Science Core Collection та/або Scopus**

1. Hlazunova O., **Ponzel Y.** Technologies and algorithms for the implementation of the recommendation system for creating an individual study plan for a higher education student. CEUR Workshop Proceedings. 2024. Vol. 3771. P. 110–117. (Понзелем Я. Ю. проведено літературний науковий пошук, розроблено концепцію та технічні вимоги до рекомендаційної системи, запропоновано та деталізовано алгоритми її функціонування, проведено експериментальні дослідження та проаналізовано отримані результати, підготовлено та оформлено публікацію. Глазуновою О. Г. визначено наукову новизну та теоретичні основи дослідження, надано наукове консультування щодо вибору напрямів та методів дослідження, проведено рецензування та загальне керівництво роботою над статтею).

**Статті у наукових виданнях,**
**включених до Переліку наукових фахових видань України**

2. **Понзель Я. Ю.** Архітектура вебсайту з інтегрованою системою для вибору предметів. Наука і техніка сьогодні. 2025. Вип. 1 (42). С. 1331–1343.

3. Глазунова О. Г., **Понзель Я. Ю.** Математична модель обробки даних з використанням комбінованих методів спільної фільтрації та матричної факторизації для рекомендаційних систем в освіті. Технічна інженерія. 2025. № 1 (95). С. 266–273. *(Глазуновою О. Г. сформульовано концептуальну ідею дослідження щодо створення математичної моделі для прогнозування результатів навчання та визначено методологічні підходи до масштабування системи, зокрема запропонувавши шляхи інтеграції контентної фільтрації та врахування додаткових характеристик предметів для підвищення якості рекомендацій. Понзелем Я. Ю. здійснено обґрунтування та вибір інструментарію реалізації, виконавши розроблення програмної складової на базі ML.NET, проведено безпосереднє налаштування процесу факторизації матриць, застосовано методи оптимізації через градієнтний спуск та*

*регуляризацію за нормою Фробеніуса, а також експериментально підтверджено ефективність моделі шляхом мінімізації середньоквадратичної помилки)*.

**Тези наукових доповідей**

4. **Понзель Я. Ю.**, Голуб Б. Л. Формування основного функціоналу для реалізації інформаційної системи в сфері комунікації студента та університету. Інформаційні технології: економіка, техніка, освіта ‘2022: XIII Міжнародна науково-практична конференція молодих вчених, м. Київ, 26–27 жовтня 2022 року: тези доповіді. Київ, 2022. С. 90–91. *(Понзелем Я. Ю. проведено збір та аналіз вимог користувачів, сформовано перелік основних функцій інформаційної системи комунікації, розроблено схеми взаємодії модулів та описано їхнє призначення. Голуб Б. Л. надано консультації щодо структурування та систематизації функціональних вимог, визначення теоретико-методологічних основ розробки інформаційної системи).*

5. **Понзель Я. Ю.,** Голуб Б. Л. Сутність системи комунікації студента та закладу вищої освіти. Збірник наукових праць за матеріалами Теоретичні та прикладні аспекти розробки комп’ютерних систем: V Всеукраїнська науково-практична конференція студентів і аспірантів, м. Київ, 26 квітня 2023 року: тези доповіді. Київ, 2023. С. 32–33. *(Понзелем Я. Ю. проведено аналіз існуючих комунікаційних процесів між студентом і ЗВО, визначено ключові проблеми та завдання для автоматизації, сформульовано сутність та функціональне призначення системи комунікації, підготовлено публікацію до друку. Голуб Б. Л. надано науково-методичні рекомендації щодо теоретичного обґрунтування концепції системи та визначення її місця в інформаційному просторі ЗВО).*

6. **Понзель Я. Ю.**, Глазунова О. Г. Підсистема платіжного контролю системи комунікації студента та закладу вищої освіти. Інформаційні технології: економіка, техніка, освіта: IV Міжнародна науково-практична конференція молодих вчених, м. Київ, 28–29 жовтня 2025 року: тези доповіді. Київ, 2025. URL: http://econference.nubip.edu.ua/index.php/itete/XIV/paper/view/3056. *(аналіз актуальності впровадження сучасних цифрових рішень в освітніх установах*

*проведено Понзелем Я. Ю., а також досліджено переваги та недоліки інтеграції повноцінної платіжної системи (Liqpay та ін.) для ЗВО, обґрунтовано необхідність реалізації саме системи платіжного контролю (а не миттєвої оплати), визначено її ключовий функціонал (статистика, зберігання квитанцій, облік стану оплати) для систематизації роботи фінансового відділу, а також підготовлено матеріал для публікації. Глазуновою О. Г. визначено науково-практичне значення дослідження в умовах обмежених фінансових можливостей ЗВО, надано методологічні рекомендації щодо порівняльного аналізу комерційних та освітніх моделей оплати, а також скориговано напрям дослідження у бік ефективної автоматизації внутрішніх фінансових процесів в університеті).*

**Свідоцтво про реєстрацію авторського права на твір**

1. 7. **Понзель Я. Ю.,** Глазунова О. Г. Модель персоналізованих рекомендацій у цифрових сервісах підтримки користувачів: свідоцтво про реєстрацію авторського права на твір № 138537. Комп’ютерна програма. Дата реєстрації 11 серпня 2025 р. *(Понзелем Я. Ю. виконано побудову математичної моделі та її програмну реалізацію, застосувавши підхід матричної факторизації засобами бібліотеки ML.NET, а також реалізовано алгоритми оптимізації та регуляризації моделі з використанням норми Фробеніуса для уникнення перенавчання, а також розроблено програмні модулі для налаштування гіперпараметрів і автоматизованого розрахунку метрик точності (RMSE), забезпечивши працездатність та валідацію комп’ютерної програми. Глазуновою О. Г. сформульовано загальну концепцію дослідження та поставлено задачу щодо створення системи персоналізованих рекомендацій для підвищення ефективності цифрових сервісів підтримки користувачів).*

# ДОДАТОК Б

## Свідотцтво про реєстрацію авторського права на твір

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Trainers;
using Microsoft.ML.Trainers.Recommender;
using Models;
using Models.Enums;
using Repositories;

public class StudentDiscipline
{
    public int StudentId { get; set; }
    public int CourseId { get; set; }
    public string CourseType { get; set; }
    public float Label { get; set; }
}

public class CourseRecommendation
{
    public float Label;
    public float Score;
}

namespace ML
{
    public class SubjectRecommender(IStudentDesciplineRepository studentDesciplineRepository) :
ISubjectRecommender
    {
        private readonly IStudentDesciplineRepository _studentDesciplineRepository =
studentDesciplineRepository;

        public List<SelectedDiscipline> Calculate(int actualStudentId) {
            List<StudentDiscipline> sameSpecialityDesciplinesScores =
_studentDesciplineRepository.GetSameSpecialityDesciplinesScores(actualStudentId);
            if (sameSpecialityDesciplinesScores.Count == 0)
                return [];
```

```
        var mlContext = new MLContext();
        IDataView trainingData = mlContext.Data.LoadFromEnumerable(sameSpecialityDesciplinesScores);
        DataOperationsCatalog.TrainTestData dataSplit = mlContext.Data.TrainTestSplit(trainingData,
testFraction: 0.2);
        IDataView trainSet = dataSplit.TrainSet;
        IDataView testSet = dataSplit.TestSet;

        var options = new MatrixFactorizationTrainer.Options {
           MatrixColumnIndexColumnName = "StudentIdEncoded",
           MatrixRowIndexColumnName = "CourseIdEncoded",
           LabelColumnName = "Label",
           NumberOfIterations = 10,
           ApproximationRank = 34,
           Lambda = 0.76
        };

        EstimatorChain<MatrixFactorizationPredictionTransformer> pipeline =
mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "StudentIdEncoded",
inputColumnName: "StudentId")
                .Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName:
"CourseIdEncoded", inputColumnName: "CourseId"))
                .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnName:
"CourseTypeFeaturized", inputColumnName: "CourseType"))
                .Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));
        TransformerChain<MatrixFactorizationPredictionTransformer> model = pipeline.Fit(trainSet);
        IDataView predictions = model.Transform(testSet);

        RegressionMetrics metrics = mlContext.Regression.Evaluate(predictions, labelColumnName:
"Label", scoreColumnName: "Score");
        Console.WriteLine($"Root Mean Squared Error: {metrics.RootMeanSquaredError}");

        PredictionEngine<StudentDiscipline, CourseRecommendation> predictionEngine =
           mlContext.Model.CreatePredictionEngine<StudentDiscipline, CourseRecommendation>(model);

        List<Discipline> notSelectedDisciplines =
_studentDesciplineRepository.GetNotSelectedDisciplines(actualStudentId);
```

```
        IOrderedEnumerable<KeyValuePair<int, float>> predictionsForNewSubjects
            = GetPredictionsForNewSubjects(predictionEngine, notSelectedDisciplines, actualStudentId);


        List<SelectedDiscipline> disciplinesToSelect = GetDisciplinesToSelect(predictionsForNewSubjects,
notSelectedDisciplines);
        MarkSelectedDesciplines(actualStudentId, disciplinesToSelect);
        return disciplinesToSelect;
    }


    private List<SelectedDiscipline> GetDisciplinesToSelect(IOrderedEnumerable<KeyValuePair<int,
float>> predictionsForNewSubjects,
            List<Discipline> studentDisciplines) {
        List<SelectedDiscipline> disciplinesToSelect = [];
        foreach (var predictionItem in predictionsForNewSubjects) {
            var selectedItem = studentDisciplines.FirstOrDefault(x => x.Id == predictionItem.Key);
            if (selectedItem == null)
                continue;
            var discipline = new SelectedDiscipline {
                AcademicYear = selectedItem.AcademicYear,
                Credits = selectedItem.Credits,
                DisciplineName = selectedItem.DisciplineName,
                LaboratoryWork = selectedItem.LaboratoryWork,
                Id = selectedItem.Id,
                Exam = selectedItem.Exam,
                Hours = selectedItem.Hours,
                Semester = selectedItem.Semester,
                IndependentWork = selectedItem.IndependentWork,
                Mandatory = selectedItem.Mandatory,
                Zalik = selectedItem.Zalik,
                PracticalLessons = selectedItem.PracticalLessons,
                PredictictedScore = predictionItem.Value,
                CurriculumDisciplineId = selectedItem.CurriculumDisciplineId,
                Lecturers = selectedItem.Lecturers,
                AttachmentUrl = selectedItem.AttachmentUrl,
            };
            disciplinesToSelect.Add(discipline);
```

```
            }
            return disciplinesToSelect;
        }

        private IOrderedEnumerable<KeyValuePair<int, float>> GetPredictionsForNewSubjects(
            PredictionEngine<StudentDiscipline, CourseRecommendation> predictionEngine,
            IEnumerable<Discipline> notSelectedDisciplines, int studentId) {
            var courses = (from notSelectedDiscipline in notSelectedDisciplines
                    select (notSelectedDiscipline.Id, notSelectedDiscipline.DisciplineName));
            var dict = new Dictionary<int, float>();
            foreach (var (courseId, courseType) in courses) {
                var input = new StudentDiscipline {
                    StudentId = studentId,
                    CourseId = courseId,
                    CourseType = courseType
                };
                var prediction = predictionEngine.Predict(input);
                Console.WriteLine($"Predicted preference for student {studentId} for course {courseId}
({courseType}): {prediction.Score}");
                dict[courseId] = float.IsNaN(prediction.Score) ? 0 : prediction.Score;
            }
            return dict.OrderByDescending(x => x.Value);
        }

        private void MarkSelectedDesciplines(int studentId, List<SelectedDiscipline> disciplinesToSelect) {
            var studentSelectedDesciplinesIds =
_studentDesciplineRepository.GetSelectedDisciplines(studentId);
            disciplinesToSelect.ForEach(x => x.Selected =
                x.Mandatory == (int)MandatoryEnum.FacultyOptional
                    ? studentSelectedDesciplinesIds.Contains(x.CurriculumDisciplineId)
                    : studentSelectedDesciplinesIds.Contains(x.Id));
        }
    }
}

using Models;
```

```
using Models.Enums;
using MySql.Data.MySqlClient;
using System.Data;
using System.Text;

namespace Repositories
{
   public class StudentDesciplineRepository : IStudentDesciplineRepository
   {
      public List<StudentDiscipline> GetSameSpecialityDesciplinesScores(int id) {
         return GetSameSpecialityDesciplinesScoresInner(id);
      }
      public List<Discipline> GetNotSelectedDisciplines(int studentId) {
         List<Discipline> facultyDisciplines = GetFutureStudentFacultyDisciplines(studentId);
         List<Discipline> notSelectedfacultyDisciplines = GetNotPassedNearDisciplines(facultyDisciplines,
studentId, MandatoryEnum.FacultyOptional);
         List<Discipline> universityDisciplines = GetFutureStudentUniversityDisciplines(studentId);
         List<Discipline> notSelecteduniversityDisciplines =
GetNotPassedNearDisciplines(universityDisciplines, studentId, MandatoryEnum.UniversityOptional);
         List<Discipline> notSelectedDisciplines = [.. notSelectedfacultyDisciplines, ..
notSelecteduniversityDisciplines];
         return notSelectedDisciplines;
      }
      private List<Discipline> GetNotPassedNearDisciplines(List<Discipline> studentDisciplines, int
studentId, MandatoryEnum mandatory) {
         return FilterNotPassedNearDisciplines(studentDisciplines, studentId, mandatory);
      }
```

# ДОДАТОК Д

## Код розрахунку оптимальних параметрів для матричної факторизації

```
var all = studentService.GetFacultiesStudentsByYearAndType(["1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20"], HigherEducationEnum.Master, "2024");
        int i = 0;
        foreach (var student in all) {
           _subjectRecommender.Calculate(student.Id);
           i++;
        }
        var myList = new List<StatisticData>();
        var groupedList = SubjectRecommender.StatisticList.GroupBy(x => x.StudentId);
        foreach (var item in groupedList) {
           var minValueItem = item.OrderBy(x => x.RMSE).First();
           myList.Add(minValueItem);
        }


public class SubjectRecommender(IStudentDesciplineRepository studentDesciplineRepository) :
ISubjectRecommender
{
   private readonly IStudentDesciplineRepository _studentDesciplineRepository =
studentDesciplineRepository;


   public static List<StatisticData> StatisticList = [];


   public void Calculate(int actualStudentId) {
      List<StudentDiscipline> sameSpecialityDesciplinesScores =
_studentDesciplineRepository.GetSameSpecialityDesciplinesScores(actualStudentId);
      if (sameSpecialityDesciplinesScores.Count == 0)
         return [];


      var mlContext = new MLContext();
      IDataView trainingData = mlContext.Data.LoadFromEnumerable(sameSpecialityDesciplinesScores);
      DataOperationsCatalog.TrainTestData dataSplit = mlContext.Data.TrainTestSplit(trainingData,
testFraction: 0.2);
      IDataView trainSet = dataSplit.TrainSet;
      IDataView testSet = dataSplit.TestSet;



      //first calculation
```

```
//int startIterations = 10, endIterations = 200, stepIterations = 20;
//int startRank = 5, endRank = 50, stepRank = 10;
//double startLambda = 0.01, endLambda = 1.0, stepLambda = 0.1;

//second calculation
//int startIterations = 5, endIterations = 15, stepIterations = 1;
//int startRank = 30, endRank = 40, stepRank = 1;
//double startLambda = 0.75, endLambda = 0.85, stepLambda = 0.01;

//third calculation
//int startIterations = 7, endIterations = 13, stepIterations = 1;
//int startRank = 30, endRank = 36, stepRank = 1;
//double startLambda = 0.70, endLambda = 0.85, stepLambda = 0.01;

//final calculation
int startIterations = 9, endIterations = 10, stepIterations = 1;
int startRank = 33, endRank = 34, stepRank = 1;
double startLambda = 0.75, endLambda = 0.77, stepLambda = 0.01;

var myList = new List<StatisticData>();
for (int iterations = startIterations; iterations <= endIterations; iterations += stepIterations) {
   for (int rank = startRank; rank <= endRank; rank += stepRank) {
      for (double lambda = startLambda; lambda <= endLambda; lambda += stepLambda) {
         var options = new MatrixFactorizationTrainer.Options {
            MatrixColumnIndexColumnName = "StudentIdEncoded",
            MatrixRowIndexColumnName = "CourseIdEncoded",
            LabelColumnName = "Label",
            NumberOfIterations = iterations,
            ApproximationRank = rank,
            Lambda = lambda
         };

         EstimatorChain<MatrixFactorizationPredictionTransformer> pipeline =
mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "StudentIdEncoded",
inputColumnName: "StudentId")
                  .Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName:
```

```
"CourseIdEncoded", inputColumnName: "CourseId"))
                    .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnName:
"CourseTypeFeaturized", inputColumnName: "CourseType"))
                    .Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));
            TransformerChain<MatrixFactorizationPredictionTransformer> model = pipeline.Fit(trainSet);
            IDataView predictions = model.Transform(testSet);


            var str = $"NumberOfIterations: {options.NumberOfIterations}, ApproximationRank:
{options.ApproximationRank}, Lambda: {options.Lambda}";
            RegressionMetrics metrics = mlContext.Regression.Evaluate(predictions, labelColumnName:
"Label", scoreColumnName: "Score");
            Console.WriteLine(str);
            Console.WriteLine($"Root Mean Squared Error: {metrics.RootMeanSquaredError}");
            Console.WriteLine("");
            myList.Add(new StatisticData {
               Iteration = iterations,
               Rank = rank,
               Lambda = lambda,
               RMSE = metrics.RootMeanSquaredError,
               StudentId = actualStudentId,
            });
            StatisticList.AddRange(myList);
         }
      }
   }
```

# ДОДАТОК Е

## Код розрахунку точності системи

```
using UMPirsonCalculation;
using UMPirsonCalculation.Models;

internal class Program
{
    private static void Main() {
        var respository = new Respository();
        List<StudentDisciplinesSelectionHistory> list = respository.StudentDisciplinesSelectionHistoryList();
        List<StudentDisciplinesSelection> allStudentDisciplinesSelection = [];
        foreach (var selectionHistory in list) {
            string[] selectedDisciplines = StringParser.ParseSelectedDisciplines(selectionHistory.Disciplines);
            Dictionary<string, double> recommendedDisciplines  =
StringParser.ParseRecommendedDisciplines(
                selectionHistory.RecommendedDisciplines, selectionHistory.Disciplines);
            if (recommendedDisciplines.Count <= 0) {
                continue;
            }

            var studentDisciplinesSelection = GetStudentDisciplinesSelectionModel(selectionHistory,
                selectedDisciplines, recommendedDisciplines);
            allStudentDisciplinesSelection.Add(studentDisciplinesSelection);
            var result = SelectionCalculator.Calc(studentDisciplinesSelection);
            respository.SaveStudentAccuracy(studentDisciplinesSelection.StudentId, result);
            SaveStudentDisciplinesScoreAndPrediction(selectionHistory, selectedDisciplines,
recommendedDisciplines, respository);
        }
    }

    private static void SaveStudentDisciplinesScoreAndPrediction(
            StudentDisciplinesSelectionHistory selectionHistory, string[] selectedDisciplines,
            Dictionary<string, double> recommendedDisciplines, Respository respository) {
        var topRecommendedDisciplines = recommendedDisciplines.OrderByDescending(pair => pair.Value)
            .Take(selectedDisciplines.Length)
            .ToDictionary(pair => pair.Key, pair => pair.Value);
        foreach (string selectedDiscipline in selectedDisciplines) {
```

```
            if (!topRecommendedDisciplines.TryGetValue(selectedDiscipline, out double predictionValue)) {
                continue;
            }
            int parsedSelectedDiscipline = int.Parse(selectedDiscipline);
            int realDegree = respository.GetRealDegree(selectionHistory.StudentId, parsedSelectedDiscipline);
            if (realDegree > 0) {
                respository.SaveStudentAssessmentRealAndPrediction(selectionHistory.StudentId, realDegree,
predictionValue);
                continue;
            }
            realDegree = respository.GetDisciplineRealDegree(parsedSelectedDiscipline,
selectionHistory.StudentId);
            if (realDegree > 0) {
                respository.SaveStudentAssessmentRealAndPrediction(selectionHistory.StudentId, realDegree,
predictionValue);
            }
        }
    }

    private static StudentDisciplinesSelection GetStudentDisciplinesSelectionModel(
            StudentDisciplinesSelectionHistory selectionHistory, string[] selectedDisciplines,
            Dictionary<string, double>  recommendedDisciplines) {
        var topRecommendedDisciplines = recommendedDisciplines.OrderByDescending(pair => pair.Value)
            .Take(selectedDisciplines.Length)
            .ToDictionary(pair => pair.Key, pair => pair.Value);
        var studentDisciplinesSelection = new StudentDisciplinesSelection {
            StudentId = selectionHistory.StudentId,
            DisciplineSelectionItems = []
        };
        foreach (var recommendedDiscipline in topRecommendedDisciplines) {
            var wasSelected = selectedDisciplines.Any(x => x == recommendedDiscipline.Key);
            studentDisciplinesSelection.DisciplineSelectionItems.Add(new DisciplineSelectionItem {
                DisciplineId = recommendedDiscipline.Key,
                Value = recommendedDiscipline.Value,
                Selected = wasSelected
            });
```

```
        }
        return studentDisciplinesSelection;
    }
}

namespace UMPirsonCalculation
{
    internal class StringParser
    {
        public static string[] ParseSelectedDisciplines(string input) {
            return input
                .Split(new[] { ", " }, StringSplitOptions.RemoveEmptyEntries)
                .Select(part => new string(part.TakeWhile(char.IsDigit).ToArray()))
                .Where(num => !string.IsNullOrEmpty(num))
                .ToArray();
        }

        public static Dictionary<string, double> ParseRecommendedDisciplines(string input, string
selectedDisciplines) {
            int facultyCount = selectedDisciplines.Split(',')
                .Count(part => part.Contains("FacultyOptional"));
            int universityCount = selectedDisciplines.Split(',')
                .Count(part => part.Contains("UniversityOptional"));
            int facultyRecommendationParsed = 0;
            int universityRecommendationParsed = 0;

            var dictionary = new Dictionary<string, double>();
            var entries = input.Split(new[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

            foreach (var entry in entries) {
                var parts = entry.Split(new[] { "(", "):", ")" }, StringSplitOptions.RemoveEmptyEntries);

                if (parts.Length >= 2) {
                    if (parts[1] == "FacultyOptional") {
                        facultyRecommendationParsed++;
```

```
                if (facultyRecommendationParsed > facultyCount) {
                    continue;
                }
            } else {
                universityRecommendationParsed++;
                if (universityRecommendationParsed > universityCount) {
                    continue;
                }
            }
            double grade = double.Parse(parts[3].Replace(',', '.'));
            if (grade <= 0) {
                continue;
            }
            string id = parts[0];
            dictionary[id] = grade;
        }
    }

    return dictionary;
  }
 }
}

using UMPirsonCalculation.Models;

namespace UMPirsonCalculation
{
  internal class SelectionCalculator
  {
    public static double Sum = 0;
    public static double Calc(StudentDisciplinesSelection model) {
      List<DisciplineSelectionItem> items = model.DisciplineSelectionItems;

      List<DisciplineSelectionItem> selectedItems = items.Where(x => x.Selected).ToList();
      double selectedItemsAverage = (double)selectedItems.Count / items.Count;
```

```
            Sum += selectedItemsAverage;
            Console.WriteLine($"Student {model.StudentId} has value: {selectedItemsAverage}");
            return selectedItemsAverage;
        }
    }
}

using MySql.Data.MySqlClient;
using UMPirsonCalculation.Models;

namespace UMPirsonCalculation
{
    internal class Respository
    {
        public bool SaveStudentAccuracy(int studentId, double value) {
            try {
                using var connection = new MySqlConnection("Server=localhost;User
ID=root;Password=0000;Database=UM");
                connection.Open();

                var query = @"
                INSERT INTO student_accuracies (StudentId, Value)
                VALUES (@studentId, @value);";

                using var command = new MySqlCommand(query, connection);
                command.Parameters.AddWithValue("@studentId", studentId);
                command.Parameters.AddWithValue("@value", value);

                int rowsAffected = command.ExecuteNonQuery();
                return rowsAffected > 0;
            } catch (Exception ex) {
                Console.WriteLine($"Error: {ex.Message}");
                return false;
            }
```

```
    }

    public bool SaveStudentAssessmentRealAndPrediction(int studentId, int real, double? prediction) {
        try {
            using var connection = new MySqlConnection("Server=localhost;User
ID=root;Password=0000;Database=UM");
            connection.Open();

            var query = @"
            INSERT INTO studentassessmentrealandprediction (StudentId, `real`, prediction)
            VALUES (@studentId, @real, @prediction);";

            using var command = new MySqlCommand(query, connection);
            command.Parameters.AddWithValue("@studentId", studentId);
            command.Parameters.AddWithValue("@real", real);
            command.Parameters.AddWithValue("@prediction", prediction);

            int rowsAffected = command.ExecuteNonQuery();
            return rowsAffected > 0;
        } catch (Exception ex) {
            Console.WriteLine($"Error: {ex.Message}");
            return false;
        }
    }

    public List<StudentDisciplinesSelectionHistory> StudentDisciplinesSelectionHistoryList() {
        using var connection = new MySqlConnection("Server=localhost;User
ID=root;Password=0000;Database=UM");
        connection.Open();
        List<StudentDisciplinesSelectionHistory> list = [];
        var query =
            @$"SELECT h.studentid, h.disciplines, h.RecommendedDisciplines
                FROM selectedDisciplinehistory h
                JOIN (
                    SELECT studentid, MAX(createdon) AS createdon
                    FROM selectedDisciplinehistory
```

```
                where createdby = studentid
                GROUP BY studentid
            ) AS last_posts
            ON h.studentid = last_posts.studentid
            AND h.createdon = last_posts.createdon
            WHERE h.disciplines <> ";";
        using var command = new MySqlCommand(query, connection);
        using var reader = command.ExecuteReader();
        while (reader.Read()) {
            list.Add(new StudentDisciplinesSelectionHistory() {
                StudentId = reader.GetInt32(0),
                Disciplines = reader.GetString(1),
                RecommendedDisciplines = reader.GetString(2),
            });
        }
        return list;
    }

    public int GetRealDegree(int student, int disciplineId) {
        using var connection = new MySqlConnection("Server=localhost;User
ID=root;Password=0000;Database=UM");
        connection.Open();
        List<StudentDisciplinesSelectionHistory> list = [];
        var query =
            @$"SELECT sa.Value
            FROM edeanery.studentassessment sa
            join edeanery.curriculumdiscipline cd on cd.id = sa.CurriculumDiscipline_id
            join edeanery.personeducationcards pec on pec.id = sa.personeducationcards_id
            where (cd.id_disciplines = @disciplineId or cd.old_id = @disciplineId) and pec.Old_id =
@student;";
        using var command = new MySqlCommand(query, connection);
        command.Parameters.AddWithValue("@student", student);
        command.Parameters.AddWithValue("@disciplineId", disciplineId);
        using var reader = command.ExecuteReader();
        int value = -1;
        while (reader.Read()) {
```

```
            value = reader.GetInt32(0);
        }
        return value;
    }

    public int GetSomeoneRealDegree(int disciplineId) {
        using var connection = new MySqlConnection("Server=localhost;User
ID=root;Password=0000;Database=UM");
        connection.Open();
        List<StudentDisciplinesSelectionHistory> list = [];
        var query =
            @$"SELECT sa.Value
            FROM edeanery.studentassessment sa
            join edeanery.curriculumdiscipline cd on cd.id = sa.CurriculumDiscipline_id
            join edeanery.personeducationcards pec on pec.id = sa.personeducationcards_id
            where (cd.id_disciplines = @disciplineId or cd.old_id = @disciplineId);";
        using var command = new MySqlCommand(query, connection);
        command.Parameters.AddWithValue("@disciplineId", disciplineId);
        using var reader = command.ExecuteReader();
        int value = -1;
        while (reader.Read()) {
            value = reader.GetInt32(0);
        }
        return value;
    }

    // in case when curreiculum does not much
    public int GetDisciplineRealDegree(int disciplineId, int studentId) {
        using var connection = new MySqlConnection("Server=localhost;User
ID=root;Password=0000;Database=edeanery");
        connection.Open();
        List<StudentDisciplinesSelectionHistory> list = [];
        var query =
            @$"select sa.value
            from StudentAssessment sa
            join CurriculumDiscipline cd on cd.id = sa.CurriculumDiscipline_id
```

```
            join disciplines d on d.id = cd.id_disciplines
            join personeducationcards pec on pec.id = sa.personeducationcards_id
            where (d.id = @disciplineId or d.id = (select id_disciplines from CurriculumDiscipline where
Old_id = @disciplineId)) and pec.Old_id = @studentId;";
        using var command = new MySqlCommand(query, connection);
        command.Parameters.AddWithValue("@disciplineId", disciplineId);
        command.Parameters.AddWithValue("@studentId", studentId);
        using var reader = command.ExecuteReader();
        int value = -1;
        while (reader.Read()) {
            value = reader.GetInt32(0);
        }
        return value;
    }
  }
}

namespace UMPirsonCalculation.Models
{
  internal class StudentDisciplinesSelectionHistory
  {
    public int StudentId { get; set; }
    public required string Disciplines { get; set; }
    public required string RecommendedDisciplines { get; set; }
  }
}

namespace UMPirsonCalculation.Models
{
  internal class StudentDisciplinesSelection
  {
    public int StudentId { get; set; }
    public required List<DisciplineSelectionItem> DisciplineSelectionItems { get; set; }
  }
}
```

```
namespace UMPirsonCalculation.Models{
  internal class DisciplineSelectionItem
    {
        public required string DisciplineId { get; set; }
        public double Value { get; set; }
        public bool Selected { get; set; }
    }
}
```