Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: Розроблення інтелектуальної системи прогнозування успішності студентів

Виконав: студент _6_ курсу
групи__КН-61м
спеціальності 122"Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Мамай В.В
(прізвище та ініціали)

Керівник Процах Н.П.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Львів – 2021

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____КрошнийІ. М.
"____" _____20___ року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мамаю Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Розроблення інтелектуальної системи прогнозування успішності студентів*

керівник роботи *Процах Наталія Петрівна, доктор фізико-математичних наук, професор*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "31"_грудня_2020 року№ С-593

2. Термін подання студентом роботи 10 грудня 2021р.

3. Вихідні дані до роботи
Літературні джерела мережі інтернет; виданнята допоміжна література за тематикою роботи. Математичні моделі задач оптимізації та регресійний алгоритм машинного навчання.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1.Здійснити аналіз предметної області на тему "Застосування кластерного аналізу для прогнозування успішності студентів";
2.Здійснити попередній аналіз регресійного алгоритму машинного навчання для прогнозування;
3.Зібрати та проаналізувати дані успішності студентів;
4.Проаналізувати математичні моделі для розв'язку задач оптимізацій;
5.Перевірити математичні моделі на вхідні параметри та алгоритм регресії на тестовій вибірці даних, оцінити їх результативність та ефективність;
6.Реалізувати програмну систему з застосуванням математичних моделей та алгоритму регресії.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

ER-діаграма бази даних, фізична модель бази даних

6. Дата видачі завдання 18 грудня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	1. Дослідження актуальності проблеми мотивації та чинників, що впливають на успішність студентів 2. Огляд продуктів - аналогів для здійснення прогнозування	21.12.20-05.02.21	Виконано
2	1. Дослідження математичних моделей для розв'язку задач оптимізацій 2. Аналіз регресійних алгоритмів машинного навчання для прогнозування	12.02.21-05.05.21	Виконано
3	1. Використання Microsoft Solver Foundation для вирішення завдань лінійного програмування 2. Формулювання задачі прогнозування та її розв'язок за допомогою алгоритму регресії FastTree	10.05.21-17.07.21	Виконано
4	1. Вибір програмних технологій для розроблення архітектури системи 2. Програмна реалізація інтелектуальної системи прогнозування успішності студентів	26.07.21-03.11.21	Виконано
5	Оформлення пояснювальної записки до розробленої системи	04.11.21-07.12.21	Виконано
6	Здача пояснювальної записки на кафедру	10.12.21	Виконано

Студент _____ Мамай В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Процах Н.П.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

У даній роботі досліджено та реалізовано програмне забезпечення для інтелектуального прогнозування успішності студентів за допомогою таких моделей, як семестрові оцінки студентів і модель оцінок екзаменаційного і поточного контролю, а також застосовано алгоритм машинного навчання FastTree (швидкі дерева), який забезпечує високу точність прогнозування для різних завдань. Завдяки даним реалізаціям користувач, вводячи необхідні вхідні дані, отримує в результаті прогнозовані значення семестрових оцінок, екзаменаційних оцінок і поточного контролю, а також прогнозований середній рейтинговий бал. Результати прогнозу представлені у графічному та текстовому відображеннях. Також досліджено та порівняно ефективність використання цих моделей та алгоритму для прогнозування успішності студентів.

Графічна частина програмного забезпечення розроблена з використанням веб-фреймворку ASP.NET Core Blazor, що дозволяє користувачу взаємодіяти з програмним забезпеченням.

Для прогнозування використовуються такі бібліотеки, як ML.NET (для реалізації алгоритму FastTree) та Microsoft Solver Foundation (для реалізації математичних моделей).

Для розроблення серверної частини програмного забезпечення застосовано такі технології, як Entity Framework Core та ADO.NET для здійснення запитів до реляційної бази даних MS SQL Server, а також технологію LINQ для їх оброблення.

Дана робота містить 55 сторінок пояснювальної записки, 26 рисунків, 3 таблиці, 2 додатка, 21 джерело.

Ключові слова:C#, ASP.NET Core Blazor, FastTree, ML.NET, Microsoft Solver Foundation, Entity Framework Core, LINQ, ADO.NET.

ABSTRACT

In this work investigated and implemented software for intelligent prediction of student performance using models such as semester student grades and the model of exam and current control grades, as well as applied machine learning algorithm FastTree, which provides high prediction accuracy for various tasks . Thanks to these implementations, the user, entering the necessary input data, receives the predicted values of semester grades, exam grades and current control, as well as the predicted average rating score. The results of the forecast are presented in graphical and textual representations. The efficiency of using these models and algorithm for predicting student success is also investigated and compared.

The graphical part of the software is developed using the ASP.NET Core Blazor web framework, which allows the user to interact with the software.

Libraries such as ML.NET (for the implementation of the FastTree algorithm) and the Microsoft Solver Foundation (for the implementation of mathematical models) are used for forecasting.

Technologies such as Entity Framework Core and ADO.NET were used to develop the server part of the software to query the MS SQL Server relational database, as well as LINQ technology to process them.

The thesis contains 55 pages of explanatory note, 26 figures, 3 tables, 2 appendices, 21 used literary sources.

Keywords:C#, ASP.NET Core Blazor, FastTree, ML.NET, Microsoft Solver Foundation, Entity Framework Core, LINQ, ADO.NET.

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити інтелектуальну систему прогнозування успішності студентів з застосуванням математичних моделей та алгоритму машинного навчання, а саме:

1. Здійснити аналіз предметної області на тему “Застосування кластерного аналізу для прогнозування успішності студентів”;
2. Здійснити попередній аналіз регресійного алгоритму машинного навчання для прогнозування;
3. Зібрати та проаналізувати дані успішності студентів;
4. Проаналізувати математичні моделі для розв’язку задач оптимізації;
5. Перевірити математичні моделі на вхідні параметри та алгоритм регресії на тестовій вибірці даних, оцінити їх результативність та ефективність;
6. Реалізувати програмну систему з застосуванням математичних моделей та алгоритму регресії.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ - Програмний засіб

ВНЗ - Вищий навчальний заклад

ЗНО - Зовнішнє Незалежне Оцінювання

ВСТУП

Вища освіта є одним із основних джерел знань для студентів, завдяки якій студент, покладаючись на свої здібності, зусилля та мотивацію, стає кваліфікованим фахівцем у своїй галузі. Студент, який провчився в закладі лише один семестр, вже на цьому етапі може зрозуміти, що буде далі і що він буде вивчати в майбутньому. Розробка інтелектуальної системи для прогнозування успішності студентів передбачає визначення певних моделей навчання студентів, визначення того, що мотивує студентів вчитися краще, на що слід приділяти більше уваги, та які зовнішні фактори тим чи іншим чином впливають на успішність студента, і завдяки даній системі студент отримає певну інформацію по тому, як покращити свої результати навчання.

Актуальність проблеми. Вибір спеціальності та навчання за нею у наш час є важливим етапом у житті для кожного студента, але після певного періоду навчання на конкретній спеціальності студент може зрозуміти, що це не те, чим він хоче займатися у своєму житті. Також студент може й не усвідомлювати, що навколо нього є багато інших можливостей та галузей навчання, які можуть не тільки доповнити його, але й допомогти пізнати та зрозуміти свої бажання. Тому ця інтелектуальна система має на меті спрогнозувати майбутні успіхи студента у професійній сфері на основі досягнень студентів по цій спеціальності, а також на основі його особистих побажань.

Предмет дослідження – прогнозування успішності студентів з застосуванням методології машинного навчання і математичних бібліотек.

Об’єкт дослідження – актуальність та ефективність використання математичних рішень та методології машинного навчання для здійснення прогнозування успішності студентів.

Мета роботи полягає у розробленні веб-додатку, завдяки якому користувач буде легко здійснювати прогнозування успішності студентів та аналізувати отримані результати. Для досягнення поставленої мети передбачено розв’язання таких завдань:

- Виконання аналізу математичних бібліотек та алгоритму машинного навчання, що застосовуються для прогнозування;
- Виконання програмного дослідження прогнозування успішності студентів та аналіз ефективність даного прогнозування;
- Здійснення програмної реалізації системи та розроблення користувацького інтерфейсу;
- Здійснення графічного представлення отриманих результатів;
- Аналіз отриманих результатів та висновки.

Новизна роботи. Новизна даної роботи полягає в тому, що вона забезпечує простий і зручний інтерфейс для складних операцій прогнозування, не вимагає знань у сфері машинного навчання, а також спрощує сприйняття та візуалізацію прогнозованих значень.

Практичне значення одержаних результатів. Розроблено веб-додаток для здійснення прогнозування успішності студентів, отримано результати прогнозування у вигляді оцінок, а також досліджено ефективність використання математичних бібліотек та рішень для здійснення прогнозування успішності.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Актуальність проблеми мотивації та чинників, що впливають на успішність студентів

Багато досліджень в області мотивації до навчання показують, що мотивація у студентів з часом знижується в процесі навчання, особливо під час переходу від одного етапу навчання до іншого.

Пояснення досліджень полягає в тому, що не всі студенти будуть вивчати весь навчальний матеріал до дисциплін у силу зниження мотивації. Також на студента впливають внутрішні та зовнішні чинники, які або покращують, або погіршують його успішність. У такому випадку студенти починають більше приділяти увагу на простому виконанні завдань без засвоєння знань та навичок, які в першу чергу необхідно розвивати під час їх виконання. Студенти, що є більш вмотивованими, частіше за інших будуть навчатись краще; вони дізнаються більше, докладуть більше зусиль і наполегливості, а також будуть більш залучені до виконання різноманітних завдань.

Аналіз робіт у галузі моделювання дозволяє виділити окремі принципи побудови моделі та уточнити відповідну послідовність їх розробки:

1. Насамперед необхідно визначити проблему, для того що її взяти за основу для моделювання. Адже якщо проблема є незрозумілою, визначити модель буде вкрай складно.
2. Для того, щоб здійснити побудову моделі, необхідно для неї сформувати конкретні завдання та визначити цілі прогнозування. Відповідно до даних цілей збирають і оброблять інформацію, що належить даним завданням. Необхідно також переконатись в достовірності і повноті вхідної інформації, адже це є обов’язковою умовою для побудови моделей, що мають наукове обґрунтування;
3. Також додає складності те, що в даному випадку єдиного шаблону для створення таких моделей не існує, а тому кожна модель базується на тих поставлених завданнях, які вона повинна розв’язати.
4. Розроблення ефективних алгоритмів і вибір вхідних даних є основним

етапом моделювання і для розв'язання відповідних завдань ;

5. Розв'язання та реалізація практичних завдань є заключним етапом моделювання прогнозу.

1.2. Огляд продуктів - аналогів для здійснення прогнозування

Даний приклад є цілком точним представленням математичної моделі прогнозування : розрахунку конкурсного балу для вступу.

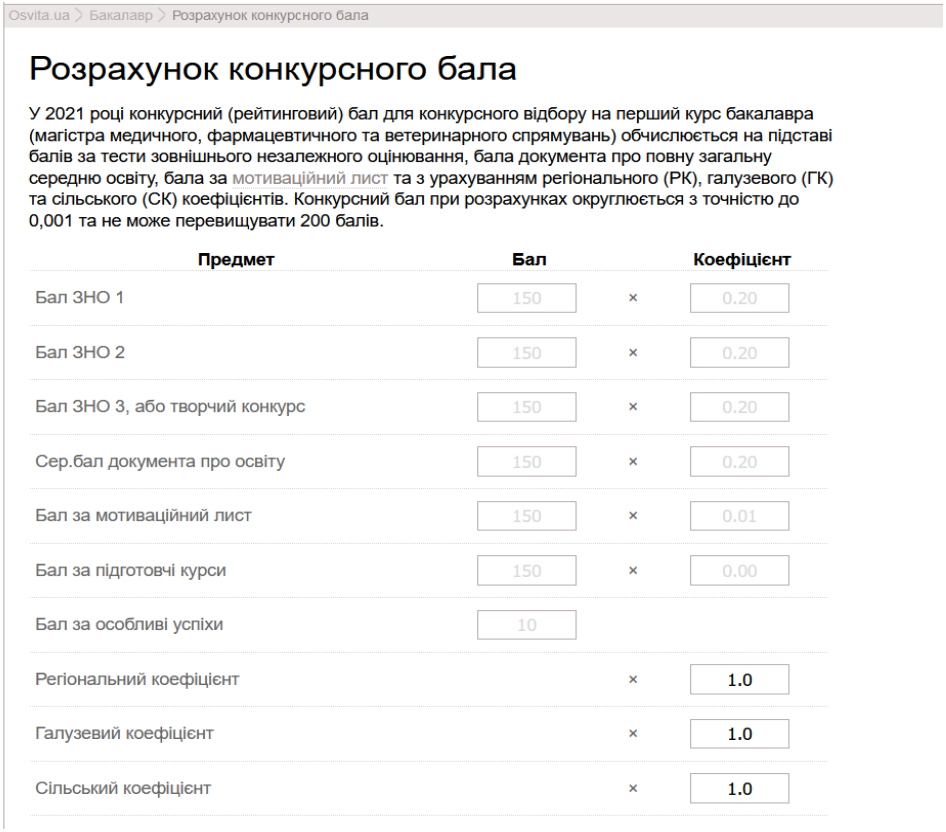

Рис. 1.1. Програмний продукт розрахунку конкурсного базу Зовнішнього Незалежного Оцінювання

У даному додатку користувач заповнює необхідні поля даними про результати складених вступних іспитів, а також поля з додатковими балами. Також користувач вказує коефіцієнти, що впливають на прогнозований результат.

Згідно отриманого результату користувач дізнається, чи буде достатньо йому набраних балів, щоб поступити на бажану спеціальність. Також відносно

нього він може знаходити інші спеціальності, що потребують саме такий прохідний бал.

Також продуктом аналогом прогнозування є продукт для визначення цін на квартири:

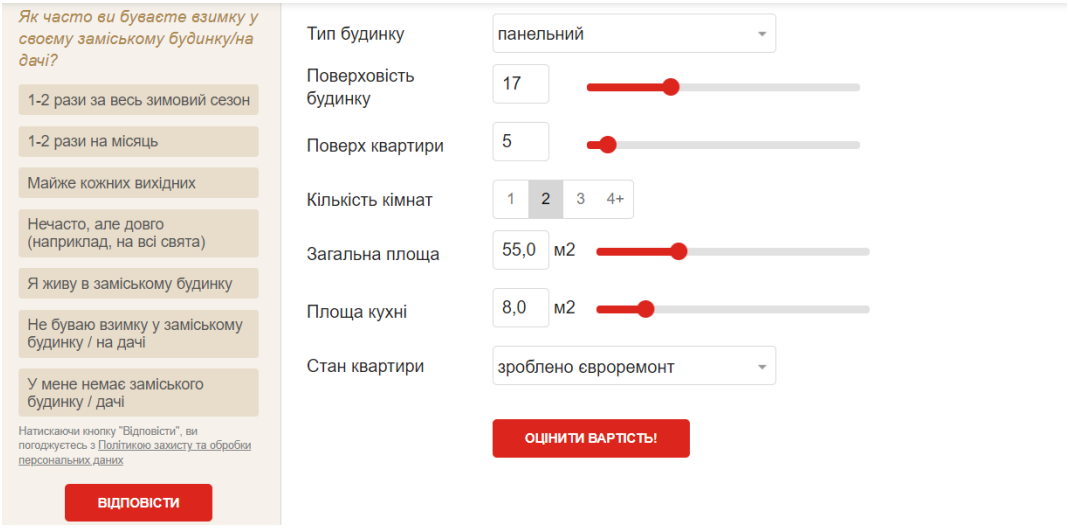

Рис. 1.2. Програмний продукт прогнозування цін на квартири

Даний сервіс аналітичного центру дозволяє приблизно розрахувати найімовірнішу зміну цін на квартири на найближчі місяці та є хорошим прикладом з застосуванням алгоритму машинного навчання. Залежність усіх цінових показників від часу описується лише кривими, якими легко побудувати тренд (похідну). А за напрямком цієї тенденції можна в лінійному наближенні отримати прогноз вартості тієї чи іншої квартири у найближчі місяці. Через високу інертність ринок нерухомості прогнозування у більшості випадків виконується з хорошою точністю. Наприклад, якщо зараз ціна квартири зростає на 3% на місяць, то в найближчі місяці вона навряд чи зміниться. За цей час темпи зростання цін можуть або трохи знизитися, або трохи підрости. Але на ринку нерухомості, на відміну від інших ринків, не буває такого, щоб в одному місяці ціни зростали, а в іншому - спадали.

1.3.Застосування інформаційних технологій для прогнозування успішності

Автоматизоване прогнозування – це техніка, яка набула поширення в декількох областях та секторах, включаючи освіту, медицину, біологію, політику та фінанси. Ця поширеність настійно пояснюється недавніми досягненнями в техніках машинного навчання.

Хоча підходи, застосовані системами прогнозування, можуть відрізнятися, вони всі дотримуються одного і того ж поняття – необхідно робити обґрунтований здогад про значення параметра, спостерігаючи, які змінні впливають на цей параметр і як вони впливали на нього в минулому. В ідеалі пояснення того, що ця здогадка не є випадковою, а обґрунтовується використанням історичних даних про змінні та їх співвідношення із параметром у процесі прогнозування.

Обсяг даних, необхідний для хорошого прогнозування, залежить від того, наскільки складним є параметр, який передбачається. Тобто це залежить від того, скільки змінних впливає на значення параметра. Насправді значна кількість параметрів, які слід передбачити, є досить складними, і для побудови гідних систем прогнозування зазвичай потрібні великі обсяги даних.

Наявність даних не обов'язково гарантує, що прогнозування буде працювати добре. Моделювання параметра, який передбачається шляхом ідентифікації змінних та зважування їх впливу, є складним завданням, важливим для реалізації успішної системи прогнозування. У вищій освіті такі системи використовуються для прогнозування середнього та остаточного балів студентів на різних рівнях. Своєчасне прогнозування балів полегшує раннє виявлення студентів, які можуть потребувати спеціального нагляду, і можуть бути використані для надання студентам зворотного зв'язку.

Прикладом прогнозування успішності студентів є наукова робота по розроблюваній програмній системі “Онлайн-система для мотивації студентів”.

У даній системі описано, що в системі використовується онлайн система, яка включає введення студентом поточного розкладу. Ця система генерує такі

початкові ставки для студента, щоб студент зумів досягнути власних перших балів за поточний розклад. Згодом студент і онлайн система «узгоджують» бали та ставки та встановлюють остаточні результати, і якщо студент досягає їх у кінці семестру, то студент отримує певну грошову винагороду по відношенню до зборів.

Підсумовуючи інформацію, вказана вище, можна вважати, що реалізація задач прогнозування успішності буде корисною для студентів. Прогнозовані дані зможуть надати студенту необхідну інформацію по тому, що студенту потрібно покращити в своєму процесі навчання, щоб досягти кращого результату та підвищити власну мотивацію. Прогнозування власного рейтингу надає студенту інформацію, як можна розподіляти зусилля в навчанні для отримання бажаного поточного та підсумкового рейтингу.

Висновки до розділу

1. У даному розділі розглянуто та досліджено проблемні аспекти покращення успішності студентів, насамперед проблему мотивації та впливу внутрішніх та зовнішніх чинників. Розроблено план реалізації моделювання, що дозволить розв’язати дану проблему.
2. Досліджено реалізацію продуктів – аналогів із застосуванням математичної моделі та алгоритму машинного навчання. Дані продукти – аналоги наглядно показують, яким чином працює прогнозування, яку проблемну область вони вирішують. Дані продукти є орієнтирами для побудови власного моделювання для вирішення проблеми прогнозування успішності студентів.
3. Досліджено інформаційне та технічне забезпечення даних продуктів та яким чином вони реалізовують математичні моделі та алгоритми машинного навчання, з урахуванням особливостей у кожній реалізації.
4. Отримано необхідну інформацію, яким чином можна досягти покращення успішності студентів.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Інформаційне забезпечення для здійснення прогнозування успішності студента на основі математичних моделей

Для здійснення прогнозування на основі математичних моделей буде використана бібліотека Microsoft Solver Foundation. Це бібліотека для математичного програмування, моделювання та оптимізації. За допомогою математичного моделювання вирішуються завдання прийняття рішень. Solver Foundation пропонує високоякісні інструменти для оптимізації їх рішень, дозволяючи вирішувати моделі в додатках навіть розробникам, які не є експертами в математичному моделюванні.

На сьогоднішні Solver Foundation володіє наступними ключовими можливостями:

- Моделювання і рішення сценаріїв за допомогою обмежень, цілей і даних;
- Розробка на мові Optimization Modeling Language (OML), імперативно в C #, функціонально в F # або будь-якою іншою мовою .NET;
- Вбудовані вирішувачі завдань для найбільш поширених типів моделей;
- Інтеграція з популярними вирішувачами: Gurobi, Ziena Knitro, Frontline Solver Platform SDK, Mosek, FICO Xpress, LINDO і lp_solve;
- Інтеграція з популярними інструментами Microsoft Office Excel і SharePoint для створення і рішення моделей.

Microsoft Solver Foundation призначений для багатьох типів користувачів, включаючи аналітиків, менеджерів ризиків та розробників додатків, які працюють на критично важливих для бізнесу системах. Дана бібліотека призначена допомогти бізнесу приймати кращі рішення. Її можна використовувати його в багатьох областях застосування, включаючи:

- Оптимізація ланцюга постачання в режимі реального часу;
- Управління енергетичним профілем центру обробки даних;
- Максимізація прибутку від інтернет-реклами;
- Логістика планування великих конференцій;

- Аналіз ризиків інвестиційних портфелів;
- Графіка та машинне навчання;
- Дослідження операцій;
- Бізнес-планування;
- Моделювання ризиків;
- Оптимізація рішень.

2.2. Використання Microsoft Solver Foundation для вирішення завдань лінійного програмування

Лінійне програмування використовується в багатьох реальних розрахунках: бізнес, економіка, транспорт, енергетика, телекомунікації, виробництво тощо. Головна мета: прийняти оптимальні рішення та заощадити ресурси, такі як гроші, час та матеріали.

Лінійне програмування –це конкретний випадок математичного програмування або оптимізації. Існує лінійна функція, яку потрібно максимізувати або звести до мінімуму. Також на дану функцію накладаються деякі обмеження та невід'ємні змінні.

Практично це відбувається наступним чином:

Рис.2.1. Область знаходження оптимального рішення

- Ми маємо функцію, яку необхідно максимізувати:

$$f(x, y) = c_1 x_1 + c_2 x_2 .$$

- Також задаються обмеження, що виглядають таким чином:

$$a_1 x_1 + b_1 x_2 \le d_1,$$

$$a_2 x_1 + b_2 x_2 \le d_2 .$$

- А також невід’ємні змінні:

$$x_1 \geq 0,$$

$$x_2 \geq 0.$$

На основі цієї інформації ми маємо змогу знайти максимальне значення. На графіку над сірою зоною знаходиться область. На лініях, які обмежують цю область, є точки, де розташовані максимальні значення.

Дана бібліотека є аналогом процедури “Пошук рішення” в програмі Excel та надає такий же функціонал. Для того, щоб виконати прогнозування, задаються рішення (Desicions) та межі допустимих прогнозованих значень (Constraints). До рішення також додається “ціль” (Goal) пошуку рішення, що являє собою задачу оптимізації і складається з виразів, що називаються термами (Terms). Як вже було згадано, задача оптимізації є багатоектремальною, тому ми можемо задати, до якого екстремуму буде прямувати наша функція – до мінімуму чи до максимуму.

Загальна структура програми пошуку оптимального рішення виглядає таким чином:

1) Створення контексту та математичної моделі, яка буде застосовуватись у задачі оптимізації:

```
SolverContext context = SolverContext.GetContext();
Model = context.CreateModel();
```

2) Створення рішень, тобто що нам необхідно знайти і що потрібно нам оптимізувати:

```
Decision x = new Decision(Domain.RealNonnegative, "firstDecision");
Decision y = new Decision(Domain.RealNonnegative, "secondDecision");
model.AddDecisions(x, y);
```

3) Додавання обмежень для пошуку оптимального рішення:

```
model.AddConstraints("someConstraint",
2 * x + 3 * y <= 1500,
2 * x + y <= 1000);
```

4) Додавання невід’ємних обмежень для отримання позитивних результатів:

```
model.AddConstraints("nonnegative",
x >= 0,
y >= 0);
```

5) Додавання цілі оптимізації (тобто що потрібно максимізувати):

```
model.AddGoal("cost", GoalKind.Maximize,50 * x + 40 * y);
```

6) Виклик функції для виконання операції оптимізації рішення:

```
Solution = context.Solve(new SimplexDirective());
```

7) Отримання результатів оптимізації у вигляді звіту та їх відображення:

```
Report = solution.GetReport();
Console.WriteLine("result: " + solution.Goals.First().ToDouble());
Console.WriteLine("x: {0}, y: {1}", x.ToDouble(), y.ToDouble());
Console.Write("{0}", report);
Console.ReadLine();
```

Звіт по задачі оптимізації виглядатиме наступним чином:

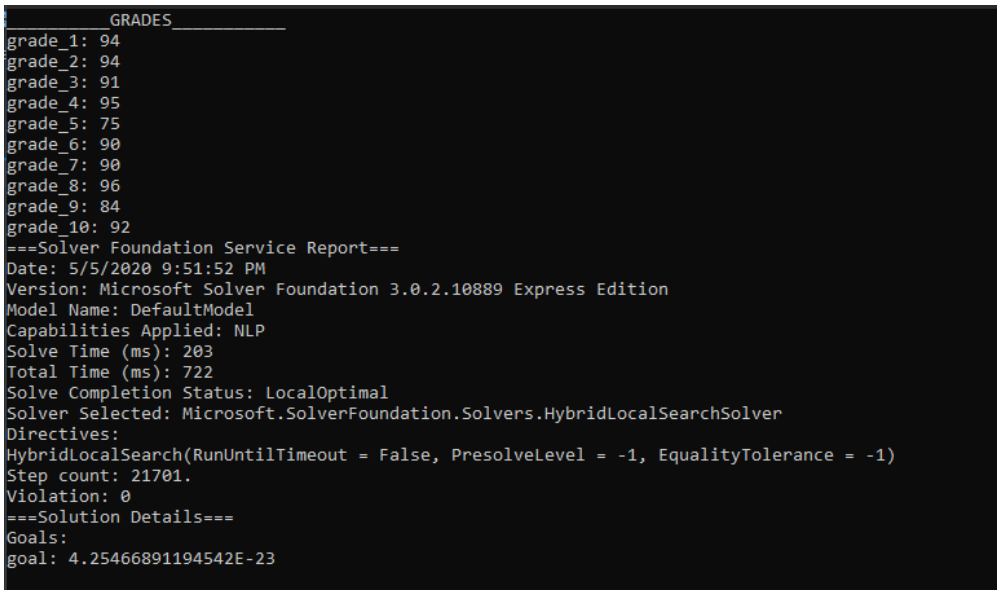

Рис.2.2. Звіт з результатами

У звіті вказано перелік прогнозованих оцінок з рейтингом в 90 балів, при заданій кількості дисциплін – 10, а також задано межі оцінок: від 50 до 97. У нижній частині звіту вказано деталі самого обчислення: час, за який виконувалось прогнозування, прогнозування конкретної моделі, ціль, тощо.

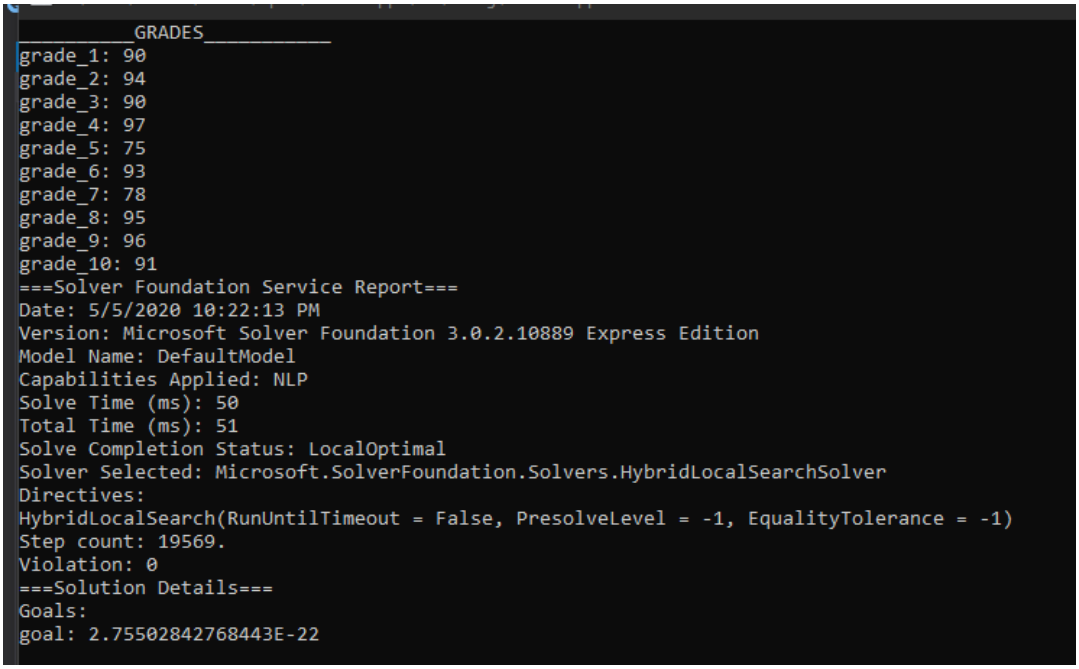

Рис.2.3. Звіт з результатами зі зміненими значеннями обмежень

У наступному звіті вже наведено приклад моделі прогнозування семестрових оцінок, де звужено межі оцінок: від 60 до 97 балів, при такому ж самому заданому рейтингу і кількості дисциплін. Тобто, міняючи ціль (в нашому випадку це загальний рейтинговий бал), а також, міняючи діапазон бажаних отриманих оцінок, математична бібліотека буде видавати різні результати.

Отже, за результатами нашого прогнозування, можна стверджувати, що бібліотека Microsoft Solver Foundation добре виконує свою поставлену задачу. Звичайно, що дана бібліотека містить ряд обмежень, які означають те, що прогнозування успішності студентів не здійснюється на основі попередньої історії даних про навчання – для даного прогнозування вже необхідно застосовувати більш серйозніші інструменти машинного навчання, охоплювати більше даних для здійснення більш точного прогнозування. Але функціоналу даної математичної бібліотеки цілком достатньо для того, щоб студент міг передбачити, які оцінки він повинен отримати, щоб досягнути бажаного результату.

2.3. Формулювання задачі прогнозування та її розв’язок за допомогою алгоритму регресії FastTree

Більшість задач машинного навчання розв’язують, використовуючи технології мови програмування Python та необхідні для цього бібліотеки (Python 3; Jupyter; SKlearn, NumPy, matplotlib), хоча й існують інші технології для здійснення прогнозування. Такою альтернативою є бібліотека ML.NET, яка застосовується для реалізації системи, використовуючи можливості .NET. За допомогою ML.NET можна отримувати прогнози наступних типів:

- Класифікація / категоризація;
- Регресія / прогнозування безперервних значень;
- Виявлення аномалій;
- Тимчасові ряди / послідовності;
- Класифікація зображень.

Алгоритм лінійної регресії - це варіант алгоритму дерева рішень, який допомагає обчислити лінійний зв’язок між залежною та незалежною змінною, а потім використовувати цей зв’язок у прогнозуванні. Відношення приймає форму лінійної формули, яка представляє ряд даних.

При підготовці даних для використання моделі лінійної регресії необхідно враховувати вимоги конкретних алгоритмів. Слід звернути увагу на кількість необхідних даних і спосіб їх використання. До даного типу моделі застосовуються наступні вимоги:

- Ключовий стовпець - кожна модель повинна містити числовий або текстовий стовпець, який однозначно ідентифікує кожен запис. Ключі компонентів заборонені.
- Прогнозований стовпець - необхідний принаймні один стовпець прогнозу. У модель можна включити кілька передбачуваних атрибутів, але вони повинні мати безперервні числові типи даних.
- Вхідні стовпці. Вхідні стовпці повинні мати безперервні числові дані; крім того, вони повинні мати відповідний тип даних.

План виконання задачі прогнозування за допомогою алгоритму регресії виглядає таким чином:

- Здійснення підготовки та аналізу даних;
- Завантаження та перетворення даних;
- Вибір алгоритму навчання та навчання моделі;
- Проведення оцінки моделі;
- Використання моделі для прогнозування.

Для початку роботи необхідно мати набір даних, що будуть застосовані для навчання моделі. Такі дані можуть зберігатись у різному вигляді : текстовому, табличному чи у базі даних. У випадку реалізації даної системи дані зберігаються у базі даних.

Дані повинні бути максимально повними та відображати свою суть – чим їх більше, тим краще – в такому випадку модель матиме змогу покращуватись на основі великої вибірки даних. Спотворені дані повинні бути доповненими, адже такі дані можуть погано вплинути на навчання моделі.

Після того, як проведено аналіз над колонками даних, визначено, що є ознаками (незалежними змінними), а що є міткою (залежною змінною), здійснюємо підключення бази даних до програмної системи:

```
var dbSource = new DatabaseSource(SqlClientFactory.Instance,
DBHelper.GetDbConnection(), STUDENTS_EXCELLENCE_SELECT);
```

Та здійснюємо вибірку даних :

```
Context = new MLContext();
var loader = Context.Data.CreateDatabaseLoader<StudentExcellenceML>();
Console.WriteLine("Loading data from database...");
var data = loader.Load(dbSource);
```

Після завантаження даних, нам необхідно їх поділити на дві вибірки : тестову та навчальну:

```
var set = Context.Data.TrainTestSplit(data, testFraction: 0.2);
var trainingData = set.TrainSet;
var testData = set.TestSet;
```

Необхідно також визначити, що буде прогнозуватись (тобто що є міткою). У програмній системі прогнозування успішності студентів прогнозованим значенням є оцінка:

```
Console.WriteLine("Preparing training operations...");
                var pipeline = Context.Transforms
                        .CopyColumns(outputColumnName: "Label", inputColumnName: "Grade")
```

Далі необхідно зазначити незалежні змінні, на основі яких здійснюється прогнозування. Якщо дані є нечисловими, або іншого типу даних (є категоріальними), їх необхідно всіх привести до єдиного типу даних:

```
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName: "StudyTimeF",
inputColumnName: "StudyTime"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName: "HealthStatusF",
inputColumnName: "HealthStatus"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName: "GoOutWithFriendsF",
inputColumnName: "GoOutWithFriends"))
```

Останній крок на етапі підготовки даних полягає в об'єднанні всіх стовпців ознак у стовпці Features.За замовчуванням алгоритм навчання обробляє лише ознаки, що у стовпці Features:

```
.Append(Context.Transforms.Concatenate("Features", "GoOutWithFriendsF", "HealthStatusF",
"StudyTimeF", "TravelTimeF", "GenderF", "CohabitationStatusF", "MotherEducationF",
"FatherEducationF", "HasExtraCoursesF", "HasExtraActivitiesF", "HasRomanticRelationshipsF",
"WorkdayAlcoholConsumptionF", "WeekendAlcoholConsumptionF", "FreeTimeF", "AbsencesF"))
```

А також обрати алгоритм машинного навчання:

```
.Append(Context.Regression.Trainers.FastTree());
```

Наступним етапом є навчання моделі. Навчання моделі здійснюється на навчальній вибірці:

```
       var model = pipeline.Fit(trainingData);
```

Після того, як навчання моделі було завершено, її необхідно перевірити на точність прогнозування на тестовій вибірці та оцінити її ефективність:

```
       IDataView transformedTestData = model.Transform(testData);
       RegressionMetrics trainedModelMetrics =
Context.Regression.Evaluate(transformedTestData);
```

Здійснивши оцінку даної моделі прогнозування, отримано такі результати:

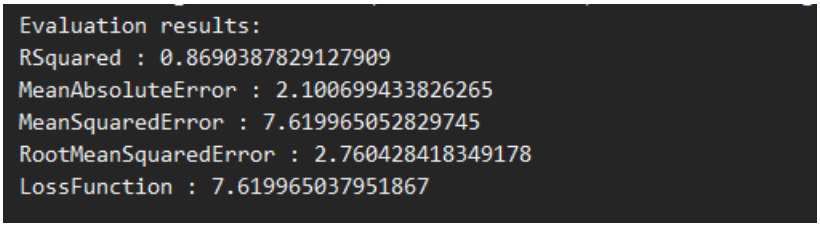

Рис.2.4. Зібрані метрики алгоритму машинного навчання

Дані метрики відображають, наскільки наша натренована модель точно здійснює прогноз. Беручи до уваги першу метрику (RSquared), можна зазначити, що модель доволі добре справляється з поставленою задачею. Чим ближче коефіцієнт знаходиться до 1, тим кращою є модель.

Висновки до розділу

1. У даному розділі детально описано та досліджено математичні моделі лінійного програмування та алгоритм лінійної регресії з застосуванням таких відповідних бібліотек, як Microsoft Solver Foundation та ML.NET. Розроблено план щодо реалізації даних підходів прогнозування успішності студентів.
2. Досліджено ефективність застосування математичних моделей у програмуванні, зібрано метрики, що наглядно показують результати прогнозування. Відносно цілі визначено точність даного прогнозування. Чим менше значення отриманої цілі, тим точнішим є прогноз на основі вхідних даних
3. Досліджено ефективність застосування алгоритму машинного навчання FastTree та зібрано метрики, що відображають ефективність натренованої моделі прогнозування. Загалом реалізація задачі прогнозування успішності студентів є виконаною та підтверджує свою працездатність.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Опис математичних моделей для розв'язку задач оптимізації

Основою математичних моделей розв'язування оптимізаційних задач є використання рейтингової системи для оцінювання успішності студентів українських ВНЗ. Система використовує критерії для оцінки поточної та семестрової успішності студентів.

У даній роботі описано дві моделі прогнозування: модель прогнозу оцінки за семестр та модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни.

Модель оцінки за семестр. Семестрова оцінка студента під час навчання може змінюватися в межах оцінок, визначених системою оцінювання. Ми вважаємо, що хорошим прогностичним орієнтиром для студента є досягнення бажаної оцінки за семестр.

Відповідна формула для обчислення оцінки:

$$1/n\sum_{i=1}^{n} S_i = R, \tag{3.1}$$

де n – к-сть дисциплін;

S_i –оцінка для і– тої дисципліни.

Використовуючи дану формулу, модель задачі прогнозування семестрової оцінки можна представити у вигляді задачі оптимізації, де необхідно знайти:

$$\min\Phi = (1/n\sum_{i=1}^{n} S_i - R_{con})^2, \tag{3.2}$$

де R_{con} – семестровий рейтинг у балах, який студент бажає отримати;

n – кількість дисциплін;

S_i –оцінка для і– тої дисципліни;

$s_{i,мін} \le s_i \le s_{i,мах}$ –обмеження на значення успішності в балах за дисципліною.

Оскільки семестрова оцінка розраховується з поточних і контрольних семестрових значень, наступним завданням є передбачити значення цих оцінок.

Модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни

Формула розрахунку семестрової оцінки для кожної дисципліни має вигляд:

$$S_i = P_i + E_i, \tag{3.3}$$

де P_i, E_i – оцінки поточного контролю та оцінки іспитів для кожної

i– тої семестрової дисципліни відповідно.

Використовуючи цю формулу , модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисциплін подаємо у вигляді задачі оптимізації, де необхідно знайти:

$$\min\Phi = \sum_{i=1}^{n} \left(P_i + E_i - S_i\right)^2, \tag{3.4}$$

де n – к-сть навчальних дисциплін;

S_i – прогнозоване значення семестрової оцінки для кожної дисципліни, що розраховане за попередньою моделлю;

P_i, E_i – оцінки поточного контролю та оцінки іспитів для кожної i – тої семестрової дисципліни відповідно.

$$b_{i,\text{мін}} \le b_i \le b_{i,\text{мах}}, \tag{3.5}$$

– обмеження на значення оцінок для поточного контролю та оцінки іспитів для кожної дисципліни відповідно.

Обчислюються прогнозні значення оцінок поточного контролю та оцінок іспитів для кожної i – тої семестрової дисципліни. Вони є основними покажчиками для покращення успішного навчання студента за кожною дисципліною.

Дані моделі на основі задач оптимізацій використані для однакового розподілу оцінок. Результати моделювання на основі описаних моделей показують, які оцінки студенту необхідно отримати, щоби досягти бажаного середнього рейтингового балу.

3.2.Аналіз регресійних алгоритмів машинного навчання для прогнозування

Для розв’язання подібних задач найчастіше застосовують алгоритми машинного навчання. Маючи натреновану модель на тестовому наборі даних, алгоритм на основі заданих значень може з великою точністю видати результат. Загалом алгоритми машинного навчання застосовуються для знаходження певної закономірності, відхилень, здійснення класифікації даних за певними ознаками. Для розроблення інтелектуальної системи прогнозування успішності студентів необхідно буде розв’язати задачу регресійного аналізу.

У задачі регресійного аналізу прогнозоване значення не обирається з статично заданої кількості даних, над якими модель тренувалась, адже результат даного машинного навчання – функція, задаючи якій вхідні параметри, можна знайти довільний прогноз (або, як називають, прогнозовану мітку).

У випадку прогнозування успішності студента, даною міткою буде оцінка з навчальної дисципліни. Вхідними параметрами є фактори, що впливають чи не впливають на навчання студент. Таких факторів є безліч, і завдяки розв’язанню поставленої задачі буде можливість дізнатись, які фактори є основними для досягнення академічних здобутків, а які фактори практично на них не впливають.

Також застосування алгоритмів регресії є такими:

- Прогнозування чи прогнозний аналіз. Одним із важливих застосувань регресії є прогнозування чи прогнозний аналіз. Наприклад, ми можемо прогнозувати ВВП, ціни на нафту або, простіше кажучи, кількісні дані, що змінюються з часом.
- Оптимізація – ми можемо оптимізувати бізнес-процеси за допомогою регресії. Наприклад, менеджер магазину може створити статистичну модель, щоб зрозуміти час приходу покупців.
- Виправлення помилок — У бізнесі прийняття правильного рішення є не менш важливим, ніж оптимізація бізнес-процесу. Регресія допоможе нам прийняти правильне рішення, а також виправити вже виконане рішення.

- Економіка - це інструмент, що найбільш використовується в економіці. Ми можемо використовувати регресію для прогнозування попиту, пропозиції, споживання, інвестицій у запаси тощо.
- Фінанси - фінансова компанія завжди зацікавлена в мінімізації портфеля ризиків і хоче знати фактори, що впливають на клієнтів. Все це можна передбачити за допомогою регресійної моделі.

У даній роботі застосовуються такі дані для прогнозування успішності студента:

- Стать студента (є ознакою);
- Вік – (є ознакою);
- Місце проживання студента – чи проживає студент разом з батьками чи окремо від батьків (є ознакою);
- Освіта у батьків (є ознакою);
- Час добирання до ВНЗ (є ознакою)
- Час, присвячений навчанню (є ознакою). Є одним з головних факторів, що впливає на успішність студента;
- Додаткові активності (хобі) – чи у студента є додаткові захоплення від навчання (є ознакою);
- Додаткові курси – чи студент вивчає додаткові дисципліни або займаються позаплановим навчанням (є ознакою). Є одним з головних факторів, що впливає на успішність студента;
- Особисте життя – чи студент на сьогоднішній день перебуває у романтичних відносинах (є ознакою);
- Відносини між родичами (є ознакою);
- Вільний час – час, вільний від навчання (є ознакою). Є одним з головних факторів, що впливає на успішність студента;
- Час, який студент виділяє для прогулянки з друзями(є ознакою);
- Стан здоров’я(є ознакою). Є одним з головних факторів, що впливає на успішність студента;
- Середній рейтинговий бал – є міткою. На основі даних про оцінки і

факторів, що впливають на її, здійснюється прогнозування.

Одним з таких алгоритмів, який застосований у даній, є алгоритм FastTree. Це ефективна реалізація алгоритму підвищення градієнта MART. Підвищення градієнта – це техніка машинного навчання для задач регресії. Вона будує кожне дерево регресії поетапно, використовуючи попередньо визначену функцію втрат для вимірювання помилки для кожного кроку та виправляє її на наступному. Дана модель прогнозування є ансамблем слабкіших моделей прогнозування. У задачах регресії даний алгоритм будує серію таких дерев поетапно, а потім вибирає оптимальне дерево за допомогою довільної диференційованої функції втрат.

Переваги застосування даного алгоритму:

- Швидкий – і до навчання, і до прогнозування;
- Легко налаштовується;
- Не чутливий до масштабу - Функції можуть бути поєднанням категорійних і безперервних даних;
- Хороша продуктивність - тренування на залишках дає дуже хорошу точність
- Багато доступного програмного забезпечення - алгоритм дуже часто використовуються, а також існує багато добре підтримуваного та перевіреного програмного забезпечення.

Недоліки:

- Чутливий до перенавчання та шумів - завжди необхідно слідкувати за навчанням моделі, щоб уникнути даного недоліку, але сучасні бібліотеки програмного забезпечення мають інструменти, щоб цього уникнути.

3.3.Аналіз метрик за результатами прогнозування алгоритмом регресії

Для виміру точності виконання алгоритму регресії використовуються наступні метрики:

R - квадрат - це статистичний показник, який представляє частку дисперсії залежної змінної, яка пояснюється незалежною змінною або змінними в

регресійній моделі. Тоді як кореляція пояснює силу зв'язку між незалежною та залежною змінною, R-квадрат пояснює, якою мірою дисперсія однієї змінної пояснює дисперсію другої змінної. Отже, якщо R2 моделі дорівнює 0,50, то приблизно половину спостережуваної варіації можна пояснити вхідними даними моделі. Даний показник розраховується за такою формулою:

$$R^2 = 1 - \frac{\text{Невизначена варіація}}{\text{Загальна варіація}} \quad (3.6)$$

Фактичний розрахунок R-квадрата вимагає кількох кроків. Це включає взяття точок даних (спостереження) залежних і незалежних змінних і знаходження лінії, яка найкраще підходить, часто з регресійної моделі. Звідти ви можете обчислити прогнозовані значення, відняти фактичні значення та звести результати в квадрат. Це дає список помилок у квадраті, який потім підсумовується і дорівнює непоясненій дисперсії.

Щоб обчислити загальну дисперсію, ви повинні відняти середнє фактичне значення з кожного з фактичних значень, звести результати в квадрат і підсумувати їх. Звідти розділіть першу суму помилок (пояснена дисперсія) на другу суму (загальну дисперсію), відніміть результат від одиниці, і ви отримаєте R-квадрат.

Середня абсолютна помилка визначає абсолютні втрати моделі та визначається за наступною формулою**:**

$$\text{L1} = \frac{1}{m}\sum_{i=1}^{m} |y\text{i} - \hat{y}\text{i}| \quad (3.7)$$

де m - кількість екземплярів у тестовому наборі,

$\hat{y}$i - передбачені мітки для кожного екземпляра,

yi - правильні мітки кожного екземпляра.

Втрата L1 є невід’ємною, зменшується метрикою. Менші значення вказують на кращу модель щодо цього показника.

Функція втрати - це середнє значення функції втрати, визначеної користувачем, обчислене для всіх екземплярів в тестовому наборі. Різниця між значеннями міток навчання та прогнозу, зробленого за допомогою моделі.

Середня квадратична помилка – визначає втрати моделі в квадраті та визначається за наступною формулою:

$$L2 = \frac{1}{m}\sum_{i=1}^{m} |yi - \hat{y}i|^2 \quad (3.8)$$

де m - кількість екземплярів у тестовому наборі,

$\hat{y}i$ - передбачені мітки для кожного екземпляра,

yi - правильні мітки кожного екземпляра.

Втрата L2 є невід'ємною, зменшується метрикою. Менші значення вказують на кращу модель щодо цього показника.

Коренева середня квадратична помилка - значення кореневої квадратної втрати яка є квадратним коренем втрати L2 (середньої квадратичної помилки).

Висновки до розділу

1. У даному розділі детально описано та досліджено математичні моделі лінійного програмування та алгоритм лінійної регресії. Представлено математичне забезпечення у вигляді метрик та формул, на основі якого здійснюється прогнозування.
2. Застосовано та розроблено математичні моделі для потреб здійснення прогнозування успішності студента. Представлено переваги і ефективність застосування даних моделей.
3. Здійснено аналіз алгоритмів машинного навчання у області регресії та обрано один для реалізованої системи. Досліджено алгоритм машинного навчання FastTree, представлено його переваги та недоліки, і також ефективність його застосування, базуючись на метриках.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Архітектура та компоненти системи прогнозування успішності студентів

Програмний засіб базуватиметься на основі трирівневої архітектури яка поділяється на такі рівні:

1. Data Access Layer – рівень доступу до бази даних, один із рівнів архітектури. Для цього розроблені такі CRUD операції як додавання, видалення, оновлення та вибірка даних.
2. Business Logic Layer – рівень бізнес – логіки, що є середнім рівнем між рівнем доступу до бази даних та рівнем представлення. На даному рівні обробляються дані, що приходять зі сторони клієнта, у такий вигляд, який є відомим для бази даних. Також на даному рівні здійснюється додаткова логіка та перевірки над даними.
3. Presentation Layer– рівень представлення, що представляє собою графічну частину програмного забезпечення, з якою користувач здійснює взаємодію

Даний програмний засіб складається з трьох основних модулів:

1. Графічного інтерфейсу, завдяки якому користувач здійснює прогнозування успішності студента;
2. Модулю, що відповідає за реалізацію математичних моделей, а саме : моделі оцінки за семестр та модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни;
3. Модулю, що відповідає за прогнозування успішності студента на основі алгоритму машинного навчання.

ER-діаграма бази даних виглядає таким чином (рис.4.1):

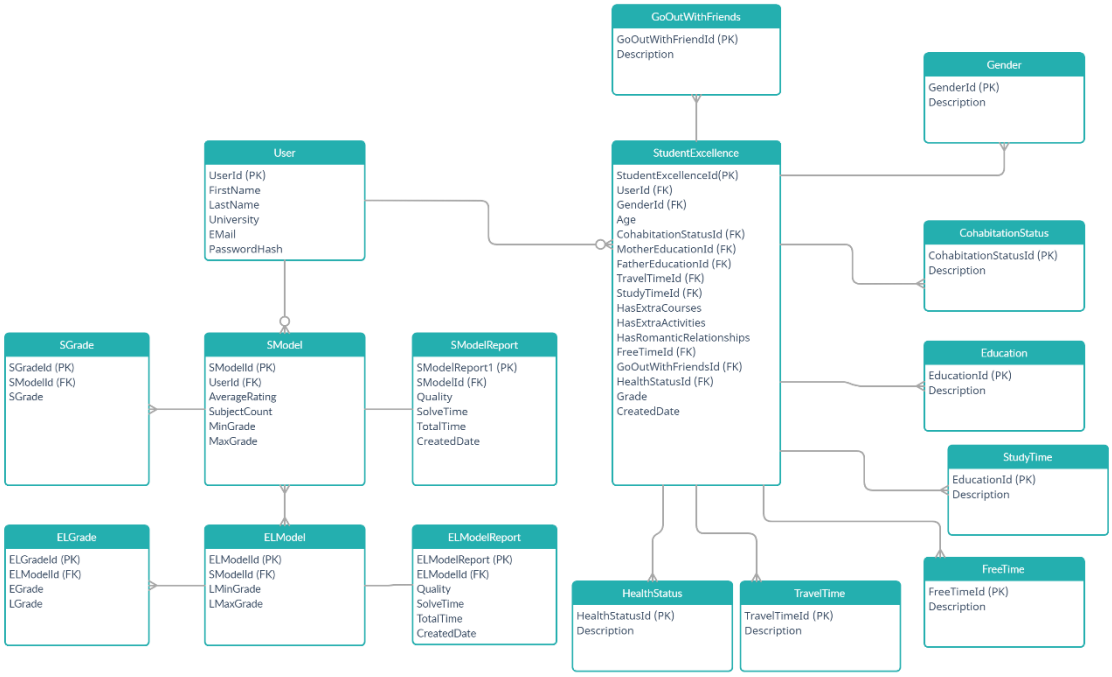

Рис. 4.1. ER- діаграма програмного засобу

Опис сутностей бази даних наведено нижче:

1. User –сутність користувача програмного засобу:
   - UserId(PK)– унікальний ідентифікатор користувача (автоінкремент);
   - FirstName – ім’я;
   - LastName – прізвище;
   - University – місце навчання;
   - EMail – пошта користувача (логін);
   - PasswordHash – захешований пароль користувача;
2. SModel – сутність моделі оцінки за семестр. В даній сутності знаходяться введені вхідні дані для моделі оцінки за семестр:
   - SModelID (PK) – унікальний ідентифікатор моделі;
   - UserId (PK) – унікальний ідентифікатор користувача;
   - AverageRating – середній бал користувача;
   - SubjectsCount – кількість семестрових дисциплін;
   - MinGrade –мінімальне значення семестрової оцінки;

- MaxGrade – максимальне значення семестрової оцінки.

3. SGrade – сутність, що містить результати прогнозу на основі моделі оцінки за семестр:
   - SGradeID (PK) – унікальне ідентифікатор оцінки;
   - SModelID (FK) – унікальний ідентифікатор моделі оцінки за семестр;
   - Grade – прогнозована оцінка за семестр.
4. SModelReport – сутність, що містить детальну інформацію про ефективність прогнозування на основі моделі оцінки за семестр:
   - SModelReportID (PK) – унікальний ідентифікатор метрики на основі моделі оцінки за семестр;
   - SModelID (FK) – унікальний ідентифікатор моделі;
   - Quality – якість прогнозування;
   - SolveTime – час знаходження оптимального рішення;
   - TotalTime – загальний час знаходження оптимального рішення;
   - CreatedDate – дата створення запису.
5. ExamAndLabModel – сутність моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни:
   - ELModelID (PK) – унікальний ідентифікатор моделі;
   - SemesterModelID (FK) – унікальний ідентифікатор моделі оцінки за семестр;
   - LMinGrade – мінімальне значення оцінки поточного контролю;
   - LMaxGrade – максимальне значення оцінки поточного контролю;
   - EMinGrade – мінімальне значення оцінки іспиту;
   - EMaxGrade – максимальне значення оцінки іспиту.
6. ELGrade – сутність що містить результати прогнозу моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни:
   - ELGradeID (PK) – унікальний ідентифікатор сутності;
   - ELModelID (FK) – унікальний ідентифікатор моделі;

- EGrade – прогнозований бал оцінки поточного контролю;
- LGrade – прогнозований бал іспиту.

7. ELReport – сутність, що містить детальну інформацію про ефективність прогнозування на основі моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни:
   - ExamAndLabModelReportID (PK) – унікальний ідентифікатор метрики на основі моделі;
   - ExamAndLabModelID (FK) – унікальний ідентифікатор моделі;
   - PrognozedQuality – якість прогнозування;
   - SolveTime – час знаходження оптимального рішення;
   - TotalTime – загальний час знаходження оптимального рішення;
   - CreatedDate – дата створення запису.
8. StudentExcellence – сутність, що відображає історію успішності студентів та фактори, що впливають на неї:
   - StudentExcellenceId(PK) – унікальний ідентифікатор запису;
   - UserId (FK) – унікальний ідентифікатор користувача;
   - GenderId (FK) – стать користувача;
   - Age – вік користувача;
   - CohabitationStatusId (FK) – статус проживання користувача;
   - MotherEducationId (FK) – освіта матері
   - FatherEducationId (FK) – освіта батька;
   - TravelTimeId (FK) – час добирання до ВНЗ;
   - StudyTimeId (FK) – час, що користувач приділяє навчанню;
   - HasExtraCourses – чи користувач має додаткові курси поза навчанням;
   - HasExtraActivities – чи користувач має додаткові активності поза навчанням;
   - HasRomanticRelationships – чи знаходиться користувач у романтичних стосунках
   - FreeTimeId (FK) – вільний час користувача від навчання;

- GoOutWithFriendsId (FK) – час, що користувач приділяє для прогулянок з друзями;
- HealthStatusId (FK) – стан здоров’я користувача;
- Grade – середній рейтинговий бал користувача;
- CreatedDate – дата створення запису;

9. Gender– сутність статі користувача:
   - GenderId (PK) – унікальний ідентифікатор запису;
   - Description – стать користувача;
10. CohabitationStatus – сутність статусу проживання користувача:
   - CohabitationStatusId (PK) – унікальний ідентифікатор запису;
   - Description – статус проживання користувача;
11. Education (FK) – сутність градацій рівнів освіти:
   - EducationId (PK) – унікальний ідентифікатор запису;
   - Description – рівень освіти;
12. TravelTime – сутність часу добирання студента до ВНЗ;
   - TravelTimeId (PK) – унікальний ідентифікатор запису;
   - Description – опис запису;
13. StudyTime – сутність часу, що користувач приділяє навчанню:
   - StudyTimeId (PK) – унікальний ідентифікатор запису;
   - Description – опис запису;
14. FreeTime – вільний час користувача від навчання:
   - FreeTimeId (PK) – унікальний ідентифікатор запису;
   - Description – опис запису;
15. GoOutWithFriendsId – сутність часу, що користувач приділяє для прогулянок з друзями:
   - GoOutWithFriendsId (PK) – унікальний ідентифікатор запису;
   - Description – опис запису;
16. HealthStatusId– сутність стану здоров’я користувача:
   - HealthStatusId (PK) – унікальний ідентифікатор запису;
   - Description – опис запису;

На основі ER-діаграми розроблено фізичну діаграму бази даних (рис.4.2):

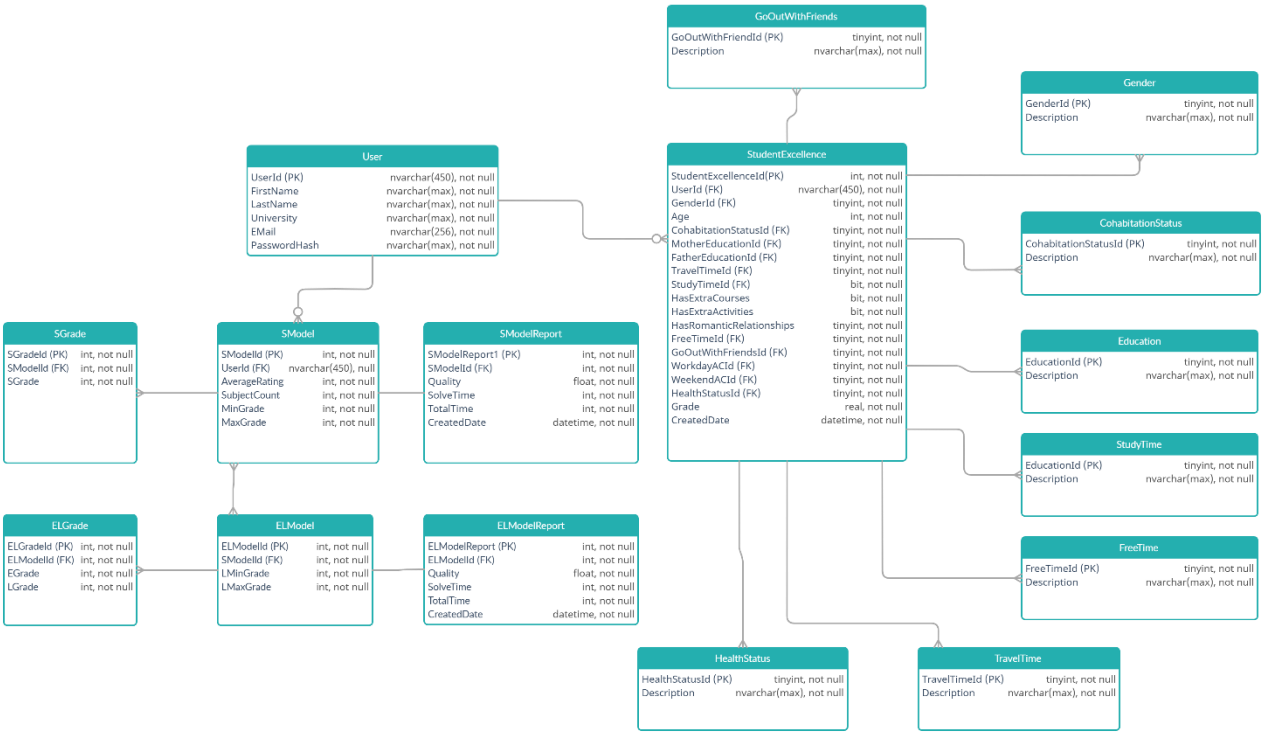

Рис. 4.2. Фізична діаграма бази даних

Програмна система складається з 5-ти логічних частин :

1) **StudentsPredictor.DAL** – рівень взаємодії програмної системи з базою даних;
2) **StudentsPredictor.BLL** – рівень бізнес – логіки. Взаємодіє з графічним представленням та рівнем, що відповідає за взаємодію до бази даних. Також взаємодіє з модулем, що відповідає за прогнозування на основі математичних моделей і модулем, що відповідає за прогнозування на основі алгоритму машинного навчання FastTree;
3) **StudentsPredictor.WEB** – рівень графічного представлення програмного засобу;
4) **ModelsPredictor** – модуль системи, що відповідає за реалізацію двох математичних моделей : моделі оцінки за семестр та модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни;
5) **MLNetIntegration** - модуль системи, що відповідає за реалізацію алгоритму машинного навчання FastTree;

4.2.Особливості роботи системи прогнозування успішності студента

Головну сторінку програмного засобу представлено на рис. 4.3.

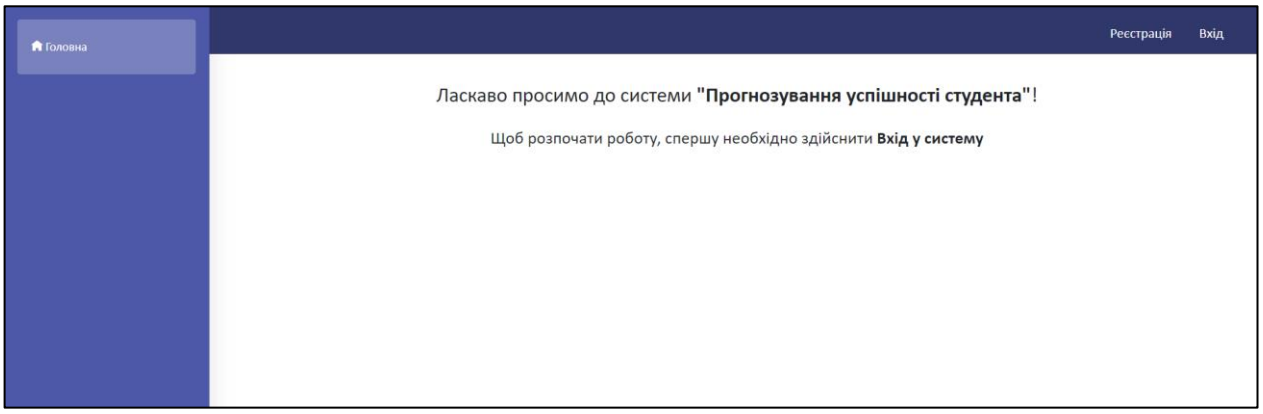

Рис. 4.3. Головна сторінка програмного засобу

Щоб увійти у систему, у неї потрібно спочатку авторизуватись, обравши пункт меню “Вхід”. Після того користувач перейде до сторінки авторизації:

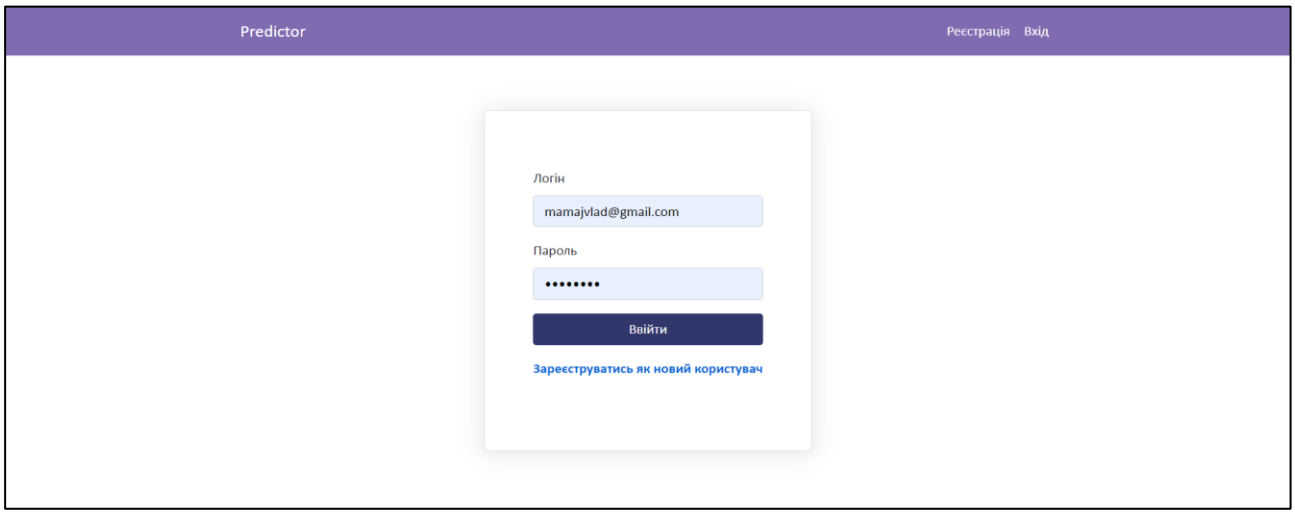

Рис. 4.4. Сторінка авторизації

Ввівши логін та пароль, авторизований користувач повертається до головної сторінки програмного засобу. Верхнє меню змінюється на “Аккаунт” та “Вихід” відповідно. Також появляється меню “Прогнозування” на боковому меню.

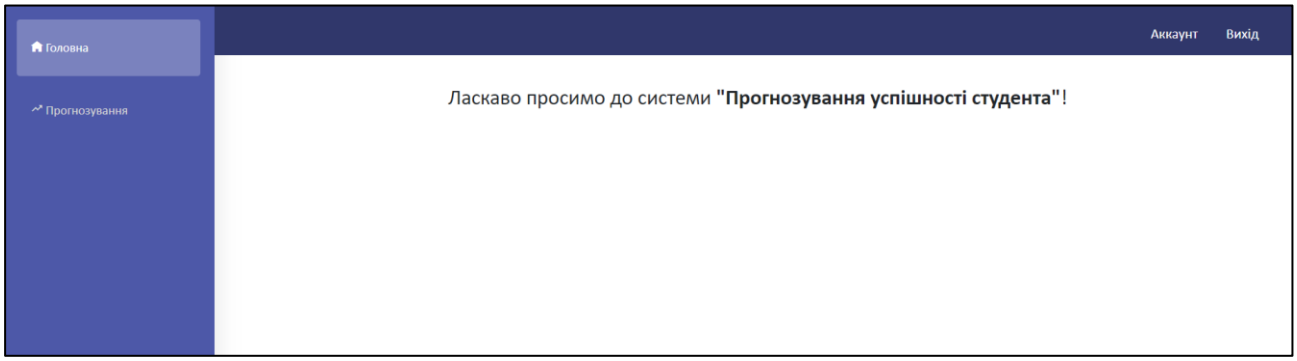

Рис. 4.5. Головна сторінка програмного засобу авторизованого користувача

Щоб розпочати роботу над прогнозуванням успішності студента, необхідно обрати пункт меню “Прогнозування”. Відкриється підменю, в якому доступні дві опції:

- Моделі прогнозування - прогнозування на основі математичних моделей;
- Фактори успішності навчання - прогнозування на основі алгоритму машинного навчання.

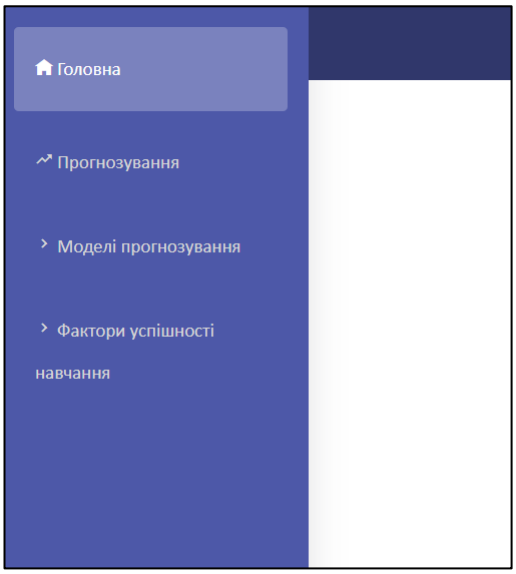

Рис. 4.6. Підпункти меню “Прогнозування”

Якщо обрати пункт меню “Моделі прогнозування”, то йому відкриється наступна сторінка:

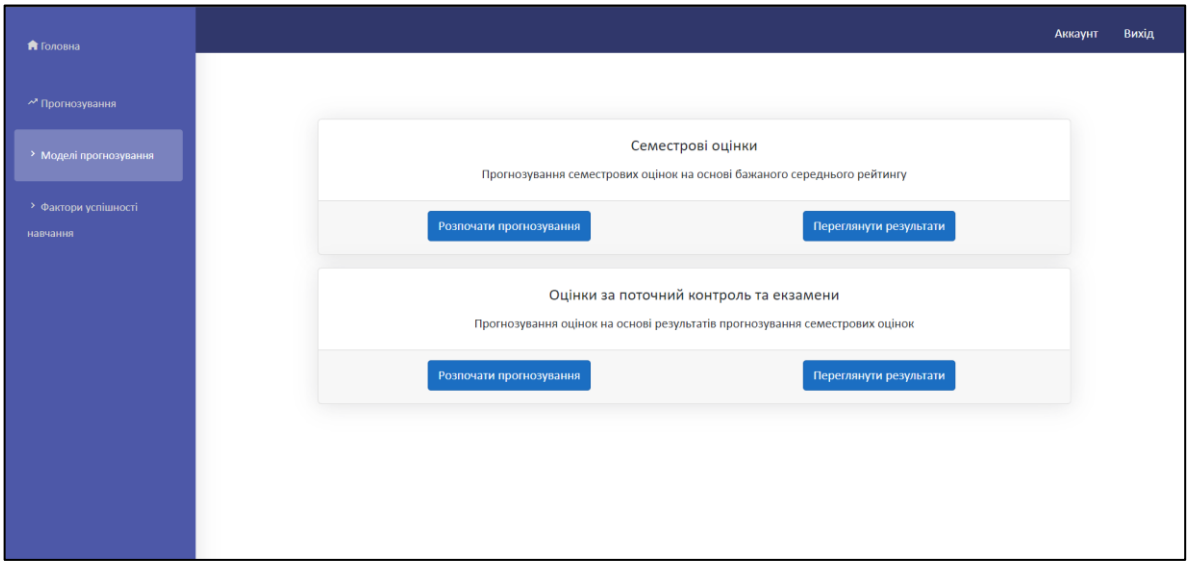

Рис. 4.7. Сторінка прогнозування на основі математичних моделей

На даній сторінці відображено:

- Секція “Семестрові оцінки” – секція, в якій користувач зможе здійснити прогноз оцінок за семестр на основі вхідних даних моделі оцінок за семестр. Для цього йому необхідно натиснути на кнопку “Розпочати прогнозування”. Кнопка “Переглянути результати” відповідає за перегляд історії прогнозувань користувача на основі цієї моделі.
- Секція “Оцінки за поточний контролю та екзамени” – секція, в якій користувач зможе здійснити прогноз оцінок за семестр на основі моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни, ввівши необхідні вхідні дані. Для цього йому необхідно натиснути на кнопку “Розпочати прогнозування”. Кнопка “Переглянути результати” відповідає за перегляд історії прогнозувань користувача на основі цієї моделі.

Здійснення прогнозування на основі моделі оцінок за семестр:

Для того, щоб здійснити прогнозування на основі моделі оцінок за семестр, користувачу необхідно натиснути на кнопку “Розпочати прогнозування” у секції “Семестрові” оцінки.

На наступній сторінці буде відображено представлення цієї моделі, де користувачу буде необхідно зробити наступне:

- Заповнити вхідні дані для моделі, а саме – середній рейтинговий бал,

кількість дисциплін та межі прогнозних значень оцінок;

- Щоб розпочати процес прогнозування, необхідно буде натиснути на кнопку “Отримати результат”;

Після цього користувач отримає результати у табличному в графічному представленні. Справа від секції вхідних даних будуть відображатись результати прогнозування, а внизу сторінки – ті ж самі результати у вигляді діаграми:

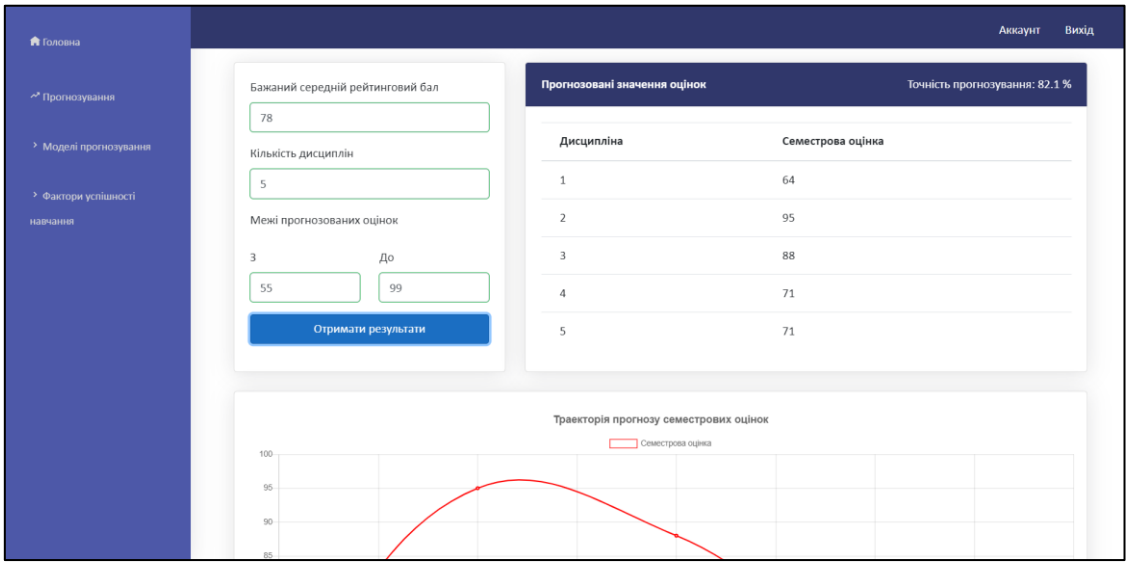

Рис. 4.8. Сторінка моделі оцінки за семестр та результати прогнозування (табличне представлення)

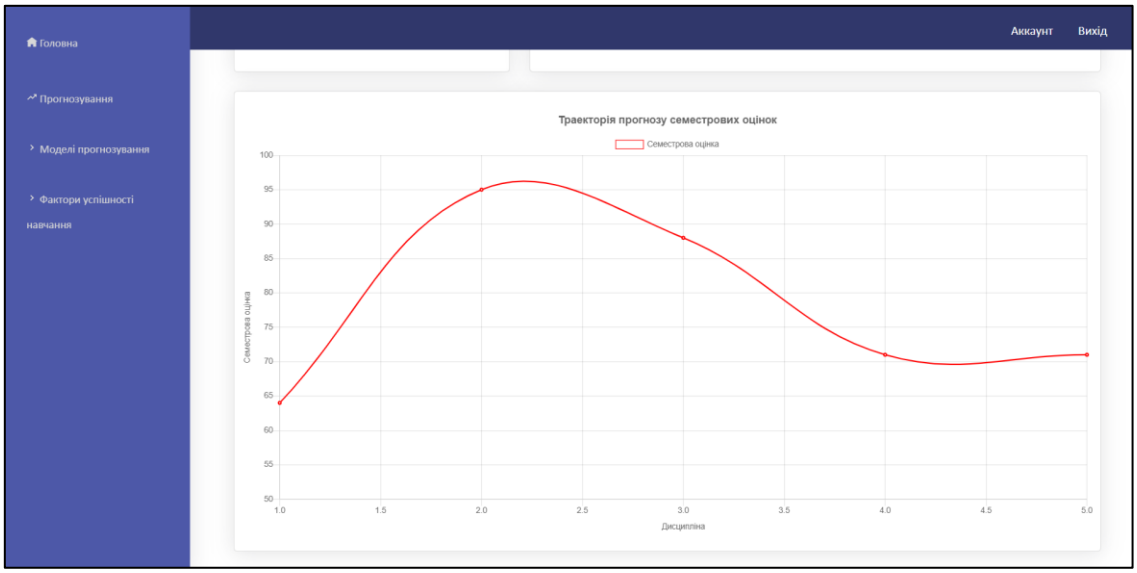

Рис.4.9. Сторінка моделі оцінки за семестр та результати прогнозування (графічне представлення)

Для перегляду результатів прогнозування, користувачу необхідно натиснути на кнопку “Переглянути результати” у секції “Семестрові оцінки”. Користувачу відкриється історія прогнозувань на основі даної моделі:

Рис.4.10. Сторінка історії прогнозувань на основі моделі оцінок за семестр

Для перегляду деталей прогнозування, необхідно натиснути на відповідному записі кнопку “Деталі”. Користувачу відкриється наступна сторінка:

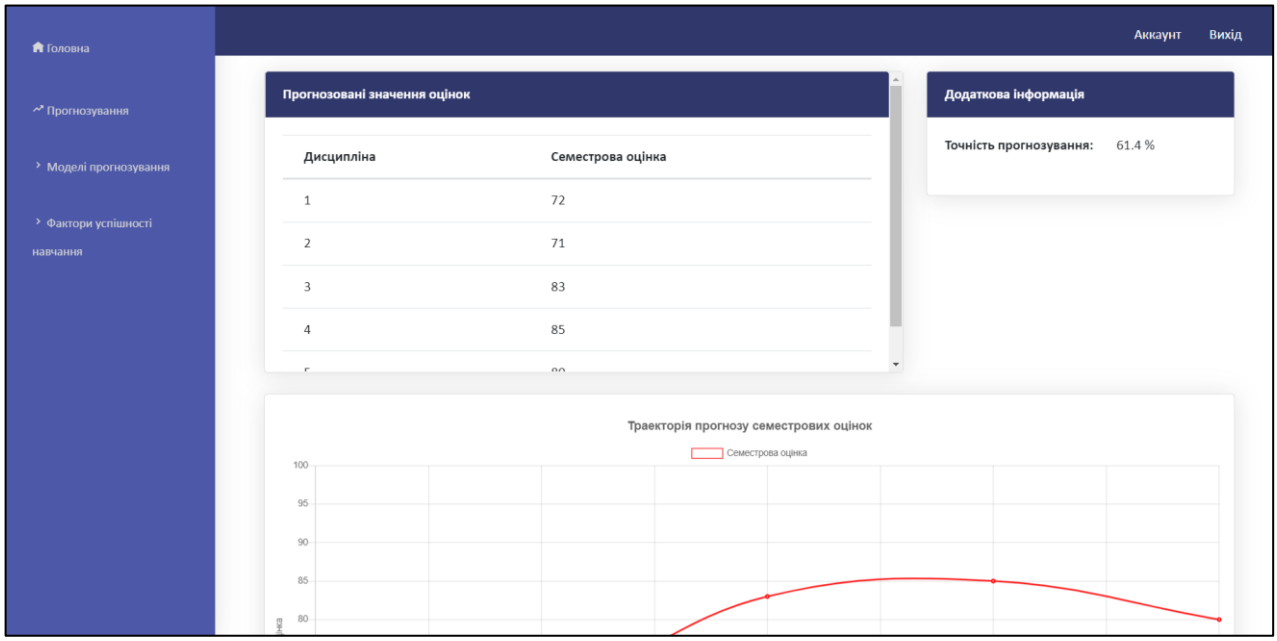

Рис.4.11. Деталі запису історії прогнозувань на основі моделі оцінок за семестр

Сторінка з результатами відображає деталі прогнозування у табличному і графічному представленнях, так само, як і під час здійснення прогнозування.

Здійснення прогнозування на основі моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни:

Для того, щоб здійснити прогнозування на основі моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни, користувачу необхідно натиснути на кнопку “Розпочати прогнозування” у секції “Оцінки за поточний контроль та екзамени”. На наступній сторінці буде відображено представлення цієї моделі, де користувачу буде необхідно зробити наступне:

- Заповнити вхідні дані для моделі, а саме – обрати результати моделі оцінок за семестр, а так вказати межі прогнозованих оцінок – поточного контролю та іспиту;
- Щоб розпочати процес прогнозування, необхідно буде натиснути на кнопку “Отримати результати”;

Після цього користувач отримає результати у табличному в графічному представленні. Справа від секції вхідних даних будуть відображатись результати прогнозування, а внизу сторінки – ті ж самі результати у вигляді діаграми:

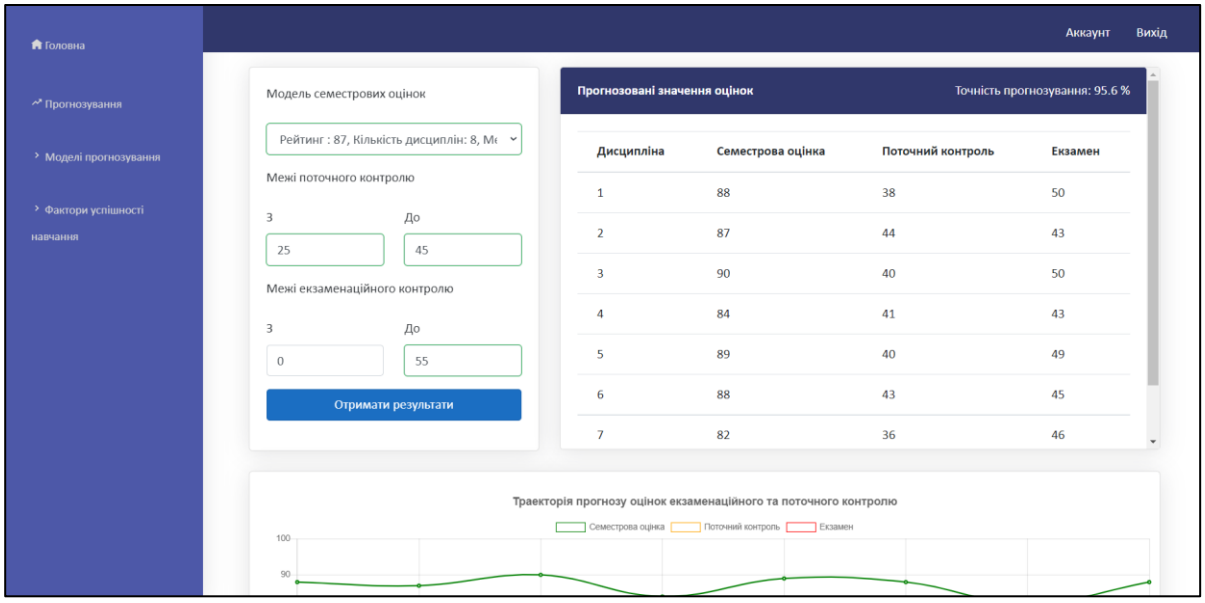

Рис. 4.12. Сторінка моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни та результати прогнозування (табличне представлення)

Рис.4.13. Сторінка моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни та результати прогнозування (графічне представлення)

Для перегляду результатів прогнозування, користувачу необхідно натиснути на кнопку “Переглянути результати” у секції “Оцінки за поточний контроль та екзамени ”. Користувачу відкриється історія прогнозувань на основі даної моделі:

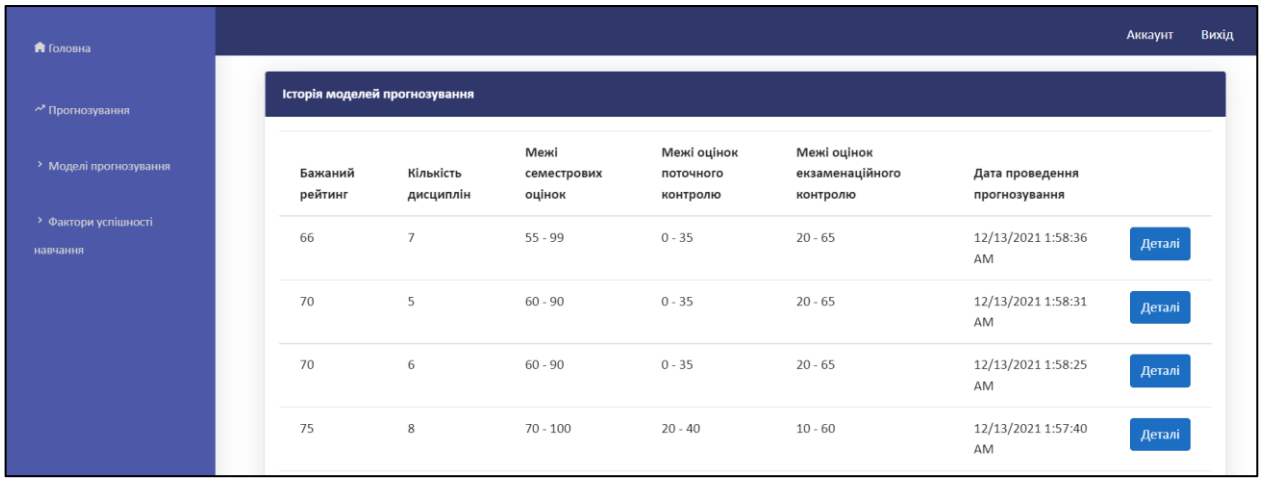

Рис.4.14. Сторінка історії прогнозувань на основі моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни

Для перегляду деталей прогнозування, необхідно натиснути на відповідному записі кнопку “Деталі”. Користувачу відкриється наступна сторінка:

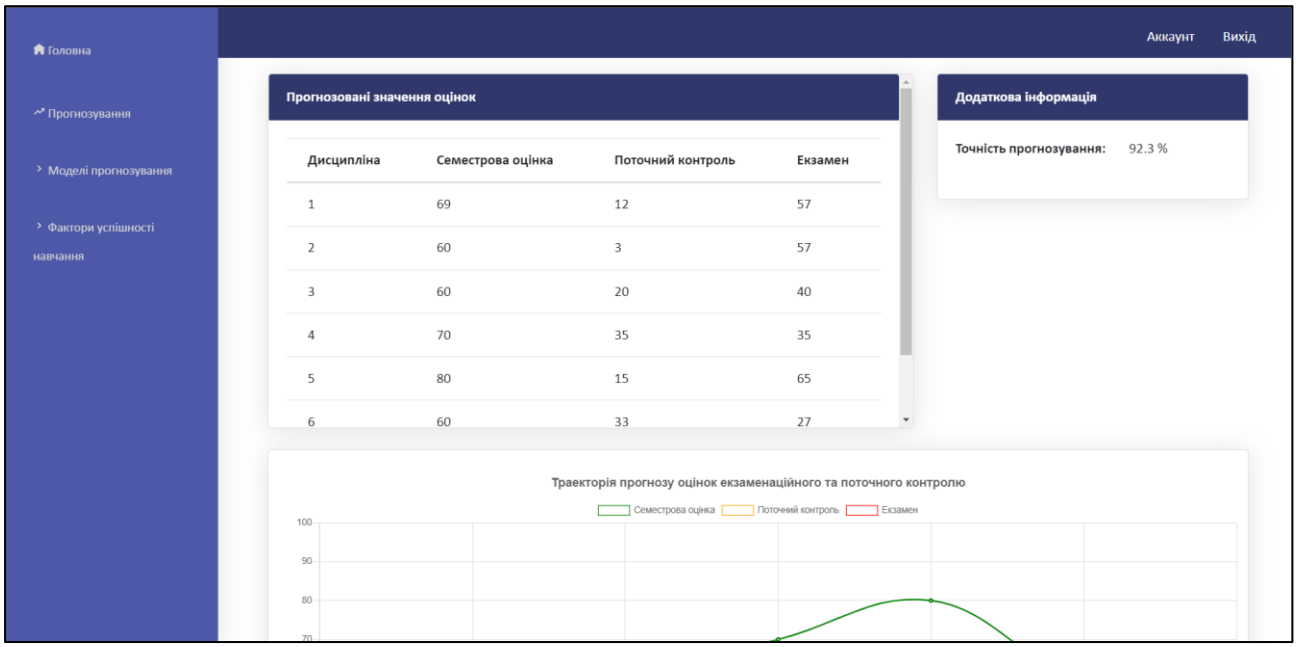

Рис.4.15. Деталі запису історії прогнозувань на основі моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни

Сторінка з результатами відображає деталі прогнозування у табличному і графічному представленнях, так само, як і під час здійснення прогнозування.

Здійснення прогнозування на основі алгоритму машинного навчання FastTree:

Для того, щоб здійснити прогнозування на основі алгоритму машинного навчання FastTree, користувачу необхідно обрати підпункт меню “Фактори успішності навчання” у меню “Прогнозування”. Користувачу відкриється наступна сторінка:

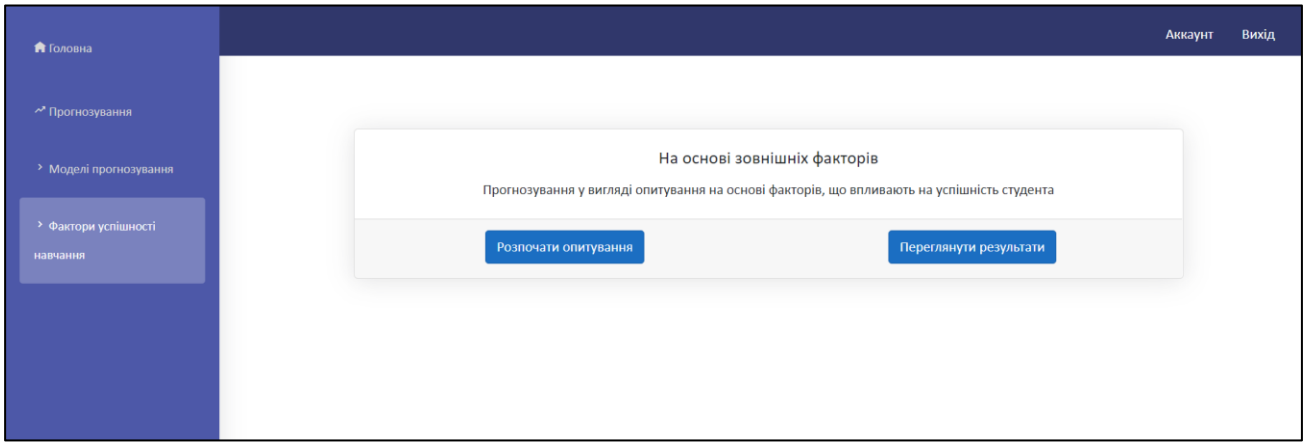

Рис. 4.16. Сторінка прогнозування на основі алгоритму машинного навчання FastTree

Далі користувачу необхідно натиснути на кнопку “Розпочати опитування”. Користувачу відкриється сторінка з опитуванням:

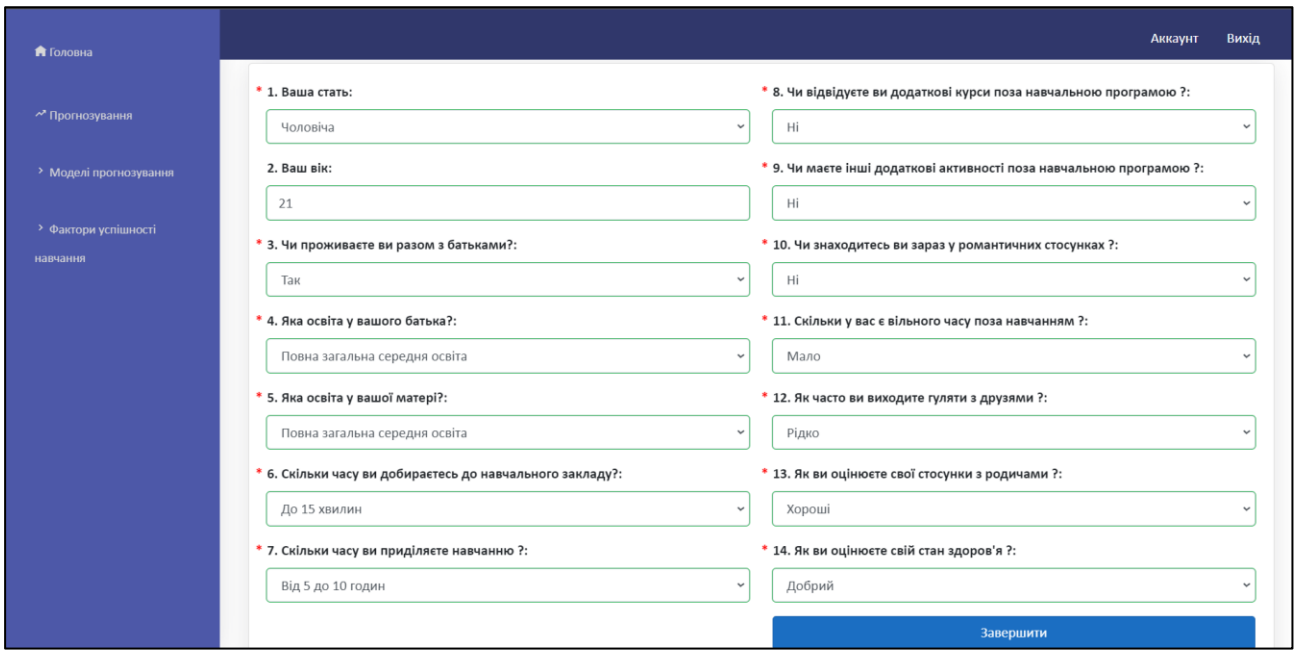

Рис. 4.17. Сторінка опитування

Користувачу пропонується перелік питань, на які повинен дати відповідь. Питання, що позначені червоною зірочкою, є обов’язковими до заповнення. Ці дані беруть участь у прогнозування як фактори, що впливають на успішність студента.

Після того, як користувач заповнив форму, йому необхідно натиснути кнопку “Завершити” щоб отримати результати прогнозування.

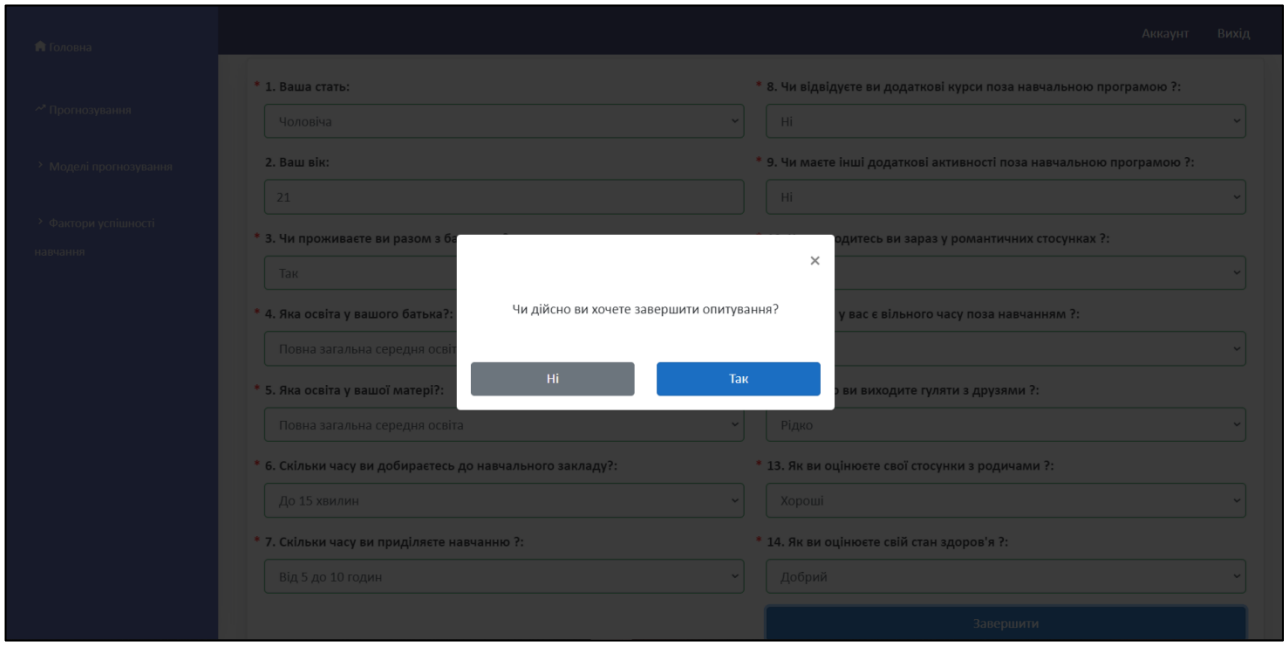

Рис. 4.18. Модальне вікно підтвердження завершення опитування

Для підтвердження завершення опитування користувач має натиснути на кнопку “Так”, або кнопку “Ні”, якщо користувач хоче повернутись до опитування. Далі система переведе користувача до результату прогнозування:

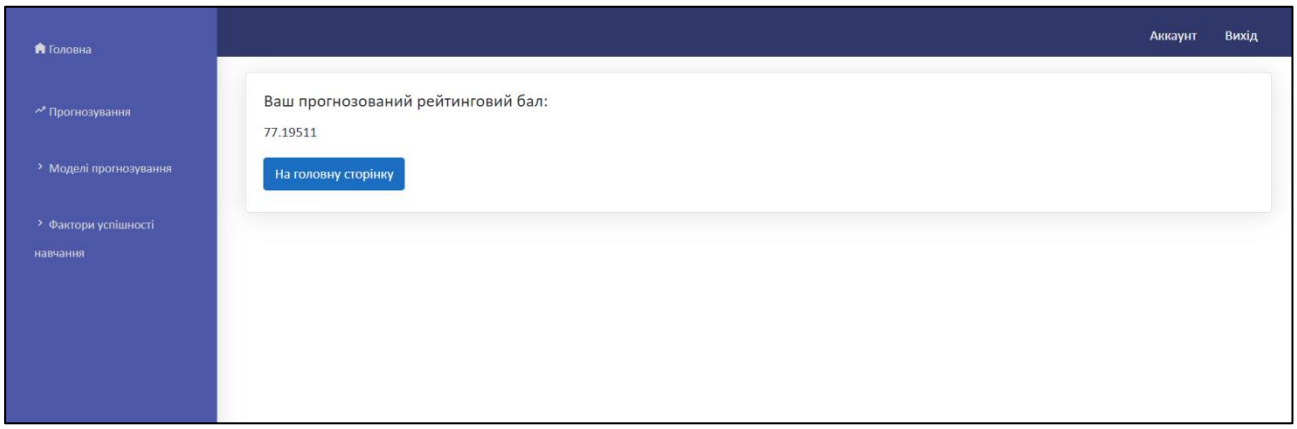

Рис. 4.19. Результат прогнозування на основі алгоритму машинного навчання

На даній сторінці користувачу відображається прогнозований середній бал на основі заповненого опитування користувачем.

Для перегляду історії прогнозувань користувачу необхідно перейти на початкову секцію опитування і натиснути на кнопку “Переглянути результати”.

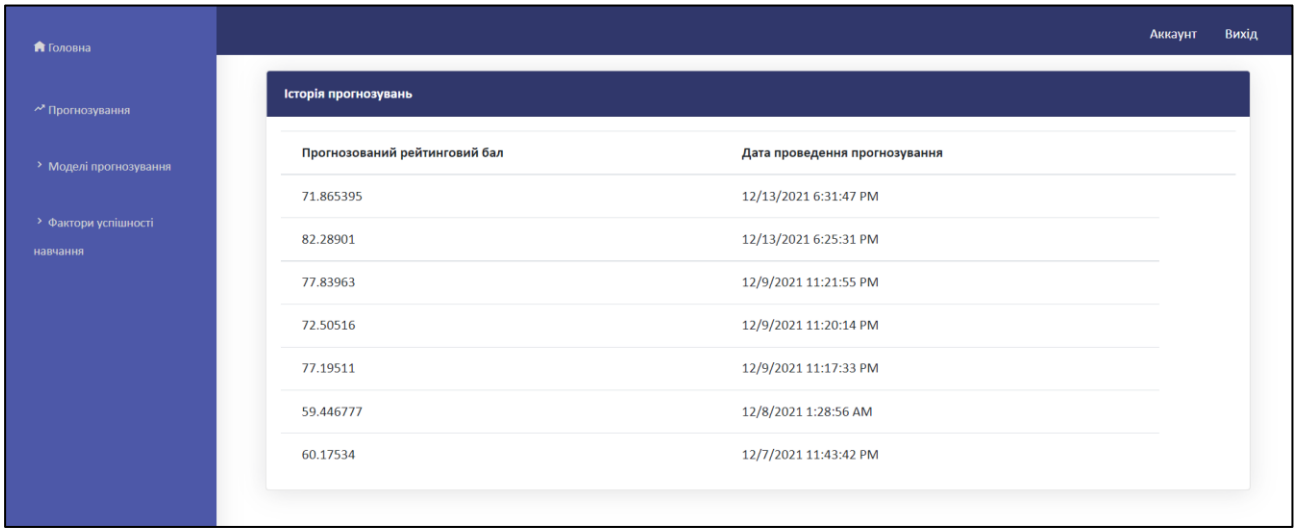

Рис. 4.20. Історія результатів прогнозування на основі алгоритму машинного навчання

Висновки до розділу

1. У даному розділі розроблено архітектуру системи та її складові. Спроектовано та реалізовано базу даних на основі ER-діаграми та фізичної діаграми, описано кожну її сутність. Описано логічні компоненти та модулі системи.
2. Реалізовано графічний інтерфейс користувача для роботи у системі, а також такий функціонал, як:
   - Авторизація та реєстрація користувача
   - Прогнозування на основі двох математичних моделей: моделі оцінки за семестр та модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни;
   - Прогнозування на основі алгоритму машинного навчання FastTree у вигляді соціального опитування;
   - Перегляд історії прогнозувань на основі математичних моделей та алгоритму машинного навчання FastTree.
3. Досліджено ефективність застосування математичних моделей та алгоритму машинного навчання FastTree та здійснено їх порівняння між собою на основі отриманих результатів та метрик.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1. Інформаційна карта проекту

Для оголошення проекту як стартап, необхідно на початку створити його інформаційну карту. Дана карта містить короткі відомості про проект Students Excellence (“Успішність студентів”), його вартість та тривалість розробки, тощо.

Короткі відомості про проект представлено у таблиці 5.1:

Таблиця 5.1. Інформаційна карта проекту

Назва номінації	Web application (Інтернет-застосунок)
Назва проекту	Students Excellence
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра інформаційних технологій, комп’ютерні науки
Прізвище, ім’я, по батькові автора	Мамай Владислав Віталійович
Цілі і задачі проекту	1. Програмна система повинна виконувати наступні функції: • Прогнозувати успішність студента на основі двох математичних моделей: моделі оцінки за семестр та модель прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни; • Прогнозувати успішність студента на основі алгоритму машинного навчання; • Мати функціонал перегляду історії прогнозувань на основі математичних моделей та алгоритму машинного навчання; 2. Система не повинна вимагати математичних знань та

	знань у машинному навчанні та штучному інтелекті. Програмний інтерфейс програми повинен бути інтуїтивним для користувача.
Короткий зміст проекту	Даний проект дозволяє передбачити майбутню успішність студента на основі оцінок, отриманих студентами за попередні семестри, а також на основі особистих побажань студента. Користувачу надано можливість самостійно задавати ту оцінку, яку він бажає отримати, а також надано можливість пройти опитування, щоб визначити, наскільки зовнішні фактори впливають на його успішність в навчанні.
Терміни виконання проекту	3 місяці
Бюджет проекту	75000 грн

5.2.Стратегії розвитку проекту

У даному пункті деталізовано декілька стратегій, що дозволяють зрозуміти, яку цільову аудиторію необхідно залучити, щоб проект мав змогу розвиватись та ставати впізнаваним.

В першу чергу, необхідно здійснити соціальне опитування серед студентів та абітурієнтів, чи цікаво б їм було отримати інструмент задля прогнозування власної успішності. Цільова аудиторія студентів є дуже великою, і в більшості студенти завжди відкриті до чогось нового та цікавого. Саме прогнозування майбутньої успішності у вигляді оцінок ніколи не залишалось студентів ніколи не залишалось поза увагою, адже всім студентам, без виключення, цікавить, що саме він зможе отримати в кінці семестру, і таким чином студент зможе розпланувати собі свій час для досягнення поставленої цілі.

Ще одним підходом для залучення цільової аудиторії є проведення онлайн\оффлайн лекції, на якій лектор (у даному випадку, розробник), розкаже загалом про штучний інтелект і машинне навчання, а саме – що це за технології і чому на сьогоднішній день вони є передовими технологіям, а також на прикладі показати, як програмна система виконує поставлені задачі з використанням даних технологій.

Після того, як програмна система стане більш відомою серед студентів, її можна буде почати пропонувати для використання університетами, перед тим удосконаливши систему на основі побажань студентів. Університет є рушійною силою в плані просування програмного засобу. Більшість університетів мають власний сайт, на якому вони розміщують всю необхідну інформацію, що прямо чи опосередковано стосується університету. Відповідно, тут починає працювати ефект реклами, а сама програмна система буде представлена у вигляді новинки\стартапу у розділі новинок технологій. Таким чином програмна система стане впізнаваною не тільки в межах одного університету, адже в університеті постійно відбуваються конференції, у яких можна взяти участь і представити власний продукт.

Якщо програмний продукт виправдає очікування людей, можна буде просувати цей продукт поза межами університетів, а саме – у соцмережах, більш серйозних наукових виданнях, які, скорше всього, самі забажають, щоб цей продукт просували саме в них. На даному етапі з реклами ще й можна отримати бонуси.

Отже, дотримуючись вищезазначених стратегій, можна отримати не лише задоволених користувачів програмного засобу, а ще й вдосконалити його, залучити велику аудиторію людей поза межами одної сфери діяльності, і отримати прибуток з реклами.

5.3. Розробка маркетингової моделі

Досягнення та зростання прибутку є основною ціллю маркетингу. Для того, щоб визначити, що саме допоможе отримати якнайбільше прибутку, необхідно виділити ключові переваги програмного продукту:

Таблиця 5.2. Визначення ключових переваг продукту

№ п/п	Потреба	Вигода проекту	Ключові переваги перед конкурентами
1	Простий та інтуїтивний інтерфейс	Зрозумілий та легкий інтерфейс, що не вимагає технічних знань	Легкість застосування програмної системи, кросплатформність
2	Прогнозування	Можливість прогнозування на основі математичних моделей та з застосуванням технологій штучного інтелекту та машинного навчання	Відсутність аналогів програмного забезпечення на ринку
3	Статистика	Аналіз історії прогнозувань для отримання точних прогнозувань	Отримання у текстовому і графічному представленнях результатів прогнозування, що зрозумілі користувачу

На даному етапі дохід від реклами буде основним рушієм в подальшій розробці програмного продукту (стартапу). Легкість застосування програмної системи, відсутність аналогів програмного забезпечення на ринку, представлення результатів прогнозування у вигляді, зрозумілому користувачу – це три основі ключові переваги даної системи, що дозволять залучити необхідну аудиторію людей, встановити взаємозв’язки з клієнтами та отримати прибуток.

5.4. Елементи фінансової моделі стартапу

Однією з переваг розроблення стартапу – є використання фінансової моделі, що дозволяє оцінити витрати на розроблення продукту. У таблиці 5.3 представлено основні пункти цієї моделі.

Таблиця 5.3. Складові фінансової моделі стартапу

№ п/п	Складова моделі	Витрати	Кількість / Тривалість	Сума
1	Онлайн - реклама	Від 4000 грн.	1 од.	4000 грн.
2	Хостинг продукту	5000 грн./рік	1 од.	5000 грн.
3	Витрати на розробку продукту	150 грн./год	3 місяці	75000 грн.

Сумарно реалізація програмного продукту коштуватиме 84000 грн.

ВИСНОВКИ

Під час розроблення інтелектуальної системи прогнозування успішності студента було здійснено аналіз математичних бібліотек та методик для прогнозування успішності студентів, а також аналіз алгоритмів машинного навчання, що дозволяють здійснювати прогнозування. На основі даних досліджень розроблено систему для прогнозування успішності студентів у вигляді веб-додатку та реалізовано основний функціонал системи, необхідний для її роботи: моделі оцінки за семестр, моделі прогнозу поточного контролю та оцінки іспитів для кожної семестрової дисципліни та алгоритму машинного навчання FastTree. Також розроблено відображення прогнозованих значень студента у табличному та графічному представленнях, реалізовано функціонал історії прогнозувань для користувача. Додатково проведено тестування даної системи. Загалом розроблено зручний користувацький інтерфейс, що надає змогу легко здійснювати поставлені задачі перед собою – дана система не вимагає додаткових технічних знань про здійснення прогнозування.

Розробивши інтелектуальну систему прогнозування успішності студентів з використанням технологій машинного навчання, а також оцінивши ефективність застосування алгоритму машинного навчання, досліджено та здійснено порівняння алгоритму машинного навчання, що розв’язує задачу регресійного аналізу, з математичними розв’язками, що розв’язують задачі оптимізації. Дослідження показали, що дві технології прогнозування показали достатні результати, але виконують різні задачі – алгоритм машинного навчання більш реально здійснює прогнозування успішності студента, враховуючи фактори, що на неї впливають, в той час як математичні розв’язки беруть до уваги лише успішність студента, не враховуючи їх.

СПИСОК ЛІТЕРАТУРИ

1. Федорчук Є. Н. / Прогнозування оцінювання успішності студентів за допомогою кластерного аналізу// Є.Н. Федорчук, Ю.Є. Федорчук – Modern scientific challenges and trends: a collection scientific works of the International scientific conference (20th June, 2018) – Warsaw: Sp. z o. o. "iScience", 2018. – 130 p 11– 15 ст.
2. Стаття “Прогнозування результатів підсумкового іспиту студентів з їх курсової діяльності” [Електронний ресурс]. – https://www.researchgate.net/publication/283212351_Predicting_Students'_Final_Exam_Scores_from_their_Course_Activities.
3. Вирішення проблем оптимізації за допомогою Microsoft Solver Foundation[Електронний ресурс]. – https://www.codeproject.com/Articles/1183168/Solving-optimization-problems-with-Microsoft-Solve.
4. Розрахунок конкурсного балу[Електронний ресурс]. – https://osvita.ua/consultations/konkurs-ball/.
5. Документація по застосуванню технології ASP.NET Core Blazor для розроблення веб-додатків [Електронний ресурс]. – https://docs.microsoft.com/ru– ru/aspnet/core/blazor/?view=aspnetcore– 3.1.
6. Використання Microsoft Solver Foundation для вирішення завдань лінійного програмування [Електронний ресурс]. –https://dzone.com/articles/using–microsoft– solver.
7. Прогнозування за допомогою навченої моделі [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-to-guides/machine-learning-model-predictions-ml-net.
8. Побудова моделі машинного навчання за допомогою SQL Server, ML.NET і C# [Електронний ресурс]. - https://dev.to/icebeam7/building-a-machine-learning-model-with-sql-server-ml-net-and-c-374a.
9. Прогноз вартості квартир в режимі онлайн [Електронний ресурс]. - https://www.irn.ru/.

10. Знайомство з розширеними деревами рішень [Електронний ресурс]. - https://indico.fnal.gov/event/15356/contributions/31377/attachments/19671/24560/DecisionTrees.pdf.
11. Метрики алгоритму машинного навчання FastTree [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.ml.data.regressionmetrics?view=ml-dotnet.
12. Регресійні алгоритми – огляд [Електронний ресурс]. - https://coderlessons.com/tutorials/python-technologies/uznaite-mashinnoe-obuchenie-s-python/regressionnye-algoritmy-obzor.
13. Прогнозування цін за допомогою регресії з ML.NET [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/dotnet/machine-learning/tutorials/predict-prices.
14. Алгоритм лінійної регресії [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/analysis-services/data-mining/microsoft-linear-regression-algorithm?view=asallproducts-allversions.
15. Машинне навчання з ML.NET – Посібник з дерев рішень [Електронний ресурс]. - https://rubikscode.net/2021/02/22/machine-learning-with-ml-net-guide-to-decision-trees/.
16. Що таке ML.NET та принципи роботи цієї системи [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work.
17. Прогнозування балів учня за допомогою лінійної регресії [Електронний ресурс]. - https://www.kaggle.com/dpringlewood/student-score-prediction-using-linear-regression
18. 3-шарова архітектура в ASP.Net [Електронний ресурс]. - https://www.c-sharpcorner.com/UploadFile/cb1429/3-layer-architecture-in-Asp-Net/.
19. Початок роботи з EF Core у веб-програмі MVC ASP.NET [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/aspnet/core/data/ef-mvc/intro?view=aspnetcore-6.0.
20. Вирішення задач оптимізації в Excel [Електронний ресурс]. - https://exceltable.com/vozmojnosti-excel/poisk-resheniya-v-excel.
21. Мамай В., Процах Н.П. Розроблення інтелектуальної системи прогнозування

успішності студентів // Комп’ютерне моделювання та інформаційні технології: матеріали третьої науково-практичної конференції студентів, аспірантів та молодих вчених (Львів, 14-16 жовтня 2021 р.). – Львів: кафедра інформаційних технологій НЛТУ України, 2021. – С. 19-22

ДОДАТКИ

ДОДАТОК А. Лістинг програмного коду моделі оцінки іспитів для кожної семестрової дисципліни

ExamAndLabsModelPrognozer.cs

```
using Microsoft.SolverFoundation.Services;
using ModelSolver.Models;
using ModelSolver.Repositories;
using System;
using System.Collections.Generic;

namespace ModelSolver.Helpers
{
    class ExamAndLabsModelPrognozer
    {
        public List<ExamAndLabsGrade> SolveModel(ExamAndLabsModelParam
modelParams, out ExamAndLabsModelReport modelreport)
        {
            SemesterModelRepository semester = new SemesterModelRepository();
            List<int> semesterGrades =
semester.GetSemesterGradesByModeld(modelParams.SemesterModelId);

            SolverContext prognozedContext = SolverContext.GetContext();
            prognozedContext.ClearModel();
            Model examLabsmodel = prognozedContext.CreateModel();

            List<Decision> LabsGrades = new List<Decision>();
            List<Decision> ExamGrades = new List<Decision>();

            examLabsmodel = CreateModelDesicions(examLabsmodel,
semesterGrades.Count, out LabsGrades, out ExamGrades);
            examLabsmodel = CreateLabsGradesConstraints(examLabsmodel,
modelParams, semesterGrades.Count, LabsGrades);
            examLabsmodel = CreateExamGradesConstraints(examLabsmodel,
modelParams, semesterGrades.Count, ExamGrades);

            List<ExamAndLabsGrade> prognozedGrades = new
List<ExamAndLabsGrade>();

            long prognozedTime = 0, prognozedTotalTime = 0;
            double totalGoal = 0;

            for (int i = 0; i < semesterGrades.Count; i++)
            {
                Goal examAndLabsGoal = examLabsmodel.AddGoal("goal_" + (i + 1),
GoalKind.Minimize, CreateModelGoal(semesterGrades[i], LabsGrades[i],
ExamGrades[i]));
                Solution ELModelSolution = prognozedContext.Solve(new
HybridLocalSearchDirective());
                Report solverReport = ELModelSolution.GetReport();

                prognozedTime += solverReport.SolveTime;
                prognozedTotalTime += solverReport.TotalTime;
                totalGoal += examAndLabsGoal.ToDouble();

                ExamAndLabsGrade grades = new ExamAndLabsGrade();
                grades.SemesterGrade = semesterGrades[i];
                grades.LabsGrade = (int)Math.Round(LabsGrades[i].ToDouble(), 0);
                grades.ExamGrade = (int)Math.Round(ExamGrades[i].ToDouble(), 0);
```

```
                prognozedGrades.Add(grades);
            }

            totalGoal = new Former().FormPrognozeQualityForELModel(totalGoal /=
semesterGrades.Count);
            modelreport = CreateModelReport(prognozedTime, prognozedTotalTime,
totalGoal);

            return prognozedGrades;
        }

        private Model CreateModelDesicions(Model, int semesterGradesCount, out
List<Decision> LabsGrades, out List<Decision> ExamGrades)
        {
            LabsGrades = new List<Decision>();
            ExamGrades = new List<Decision>();

            for (int i = 0; i < semesterGradesCount; i++)
            {
                Decision x = new Decision(Domain.RealNonnegative, "LGrade_" + (i
+ 1));
                Decision y = new Decision(Domain.RealNonnegative, "EGrade_" + (i
+ 1));
                LabsGrades.Add(x);
                ExamGrades.Add(y);
            }

            model.AddDecisions(LabsGrades.ToArray());
            model.AddDecisions(ExamGrades.ToArray());

            return model;
        }

        private Model CreateLabsGradesConstraints(Model, ExamAndLabsModelParam
modelParams, int semesterGradesCount, List<Decision> LabsGrades)
        {
            for (int i = 0; i < semesterGradesCount; i++)
            {
                model.AddConstraints("LGrade_c_" + (i + 1),
                LabsGrades[i] >= modelParams.LabsMinGrade,
                LabsGrades[i] <= modelParams.LabsMaxGrade);
            }

            return model;
        }

        private Model CreateExamGradesConstraints(Model, ExamAndLabsModelParam
modelParams, int semesterGradesCount, List<Decision> ExamGrades)
        {
            for (int i = 0; i < semesterGradesCount; i++)
            {
                model.AddConstraints("EGrade_c_" + (i + 1),
                ExamGrades[i] >= modelParams.ExamMinGrade,
                ExamGrades[i] <= modelParams.ExamMaxGrade);
            }

            return model;
        }

        private Term CreateModelGoal(Term semesterGrade, Decision labsGrade,
Decision examGrade)
        {
            //minF = pow((SUM(PK) + SUM(EK) - SUM(GRADES)),2)

            Term SEMESTER_GRADE = Model.Sum(semesterGrade);
```

```
            Term LABS_GRADE = Model.Sum(labsGrade);
            Term EXAM_GRADE = Model.Sum(examGrade);
            Term GOAL = Model.Power(Model.Difference(Model.Sum(LABS_GRADE,
EXAM_GRADE), SEMESTER_GRADE), 2);

            return GOAL;
        }

        private ExamAndLabsModelReport CreateModelReport(long prognozedTime,
long prognozedTotalTime, double totalGoal)
        {
            return new ExamAndLabsModelReport()
            {
                PrognozedQuality = totalGoal,
                PrognozedTime = prognozedTime,
                PrognozedTotalTime = prognozedTotalTime,
                DateOfPrognoze = DateTime.Now
            };
        }
    }
}
```

ExamAndLabsModelRepository.cs

```
using ModelSolver.Helpers;
using ModelSolver.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ModelSolver.Repositories
{
    public class ExamAndLabsModelRepository
    {
        private readonly string saveExamAndLabsModelQuery = "INSERT INTO
ExamAndLabsGradeModel (SemesterModelId, LabsMinGrade, LabsMaxGrade,
ExamMinGrade, ExamMaxGrade) " +

"VALUES(@SemesterModelId, @LabsMinGrade, @LabsMaxGrade, @ExamMinGrade,
@ExamMaxGrade) " +
                                                        "SELECT
CAST(SCOPE_IDENTITY() AS int)";

        private readonly string savePrognozedExamAndLabsGradesQuery = "INSERT
INTO ExamAndLabsGrade(ExamAndLabsModelId, ExamGrade, LabsGrade) " +

"VALUES(@ExamAndLabsModelId, @ExamGrade, @LabsGrade) ";

        private readonly string saveExamAndLabsModelReportQuery = "INSERT INTO
ExamAndLabsGradeModelReport(ExamAmdLabsModelId, PrognozedQuality,
PrognozedSolveTime, PrognozedTotalTime, DateOfPrognoze) " +
                                                                "
VALUES(@ExamAmdLabsModelId, @PrognozedQuality, @PrognozedSolveTime,
@PrognozedTotalTime, @DateOfPrognoze)";

        public void SaveExamAndLabsModelResults(ExamAndLabsModelParam
examAndLabsModel, List<ExamAndLabsGrade> prognozedGrades, ExamAndLabsModelReport
modelReport)
        {
            int prognozedModelId =
SavePrognozedExamAndLabsModel(examAndLabsModel);
```

```
            SavePrognozedExamAndLabsGrades(prognozedModelId, prognozedGrades);
            SaveExamAndLabsModelReport(prognozedModelId, modelReport);
        }
        private int SavePrognozedExamAndLabsModel(ExamAndLabsModelParam
examAndLabsModel)
        {
            using (SqlConnection connection = new SqlConnection(new
ModelSolverContext().SPRConnection))
            using (SqlCommand cmd = new SqlCommand(saveExamAndLabsModelQuery,
connection))
            {
                cmd.Parameters.AddWithValue("@SemesterModelId",
examAndLabsModel.SemesterModelId);
                cmd.Parameters.AddWithValue("@LabsMinGrade",
examAndLabsModel.LabsMinGrade);
                cmd.Parameters.AddWithValue("@LabsMaxGrade",
examAndLabsModel.LabsMaxGrade);
                cmd.Parameters.AddWithValue("@ExamMinGrade",
examAndLabsModel.ExamMinGrade);
                cmd.Parameters.AddWithValue("@ExamMaxGrade",
examAndLabsModel.ExamMaxGrade);

                connection.Open();
                return (int)cmd.ExecuteScalar();
            }
        }

        private void SavePrognozedExamAndLabsGrades(int prognozedModelId,
List<ExamAndLabsGrade> prognozedGrades)
        {
            using (SqlConnection connection = new SqlConnection(new
ModelSolverContext().SPRConnection))
            using (SqlCommand cmd = new
SqlCommand(savePrognozedExamAndLabsGradesQuery, connection))
            {
                connection.Open();

                foreach (var grade in prognozedGrades)
                {
                    cmd.Parameters.Clear();
                    cmd.Parameters.AddWithValue("@ExamAndLabsModelId",
prognozedModelId);
                    cmd.Parameters.AddWithValue("@ExamGrade", grade.ExamGrade);
                    cmd.Parameters.AddWithValue("@LabsGrade", grade.LabsGrade);

                    cmd.ExecuteNonQuery();
                }

                connection.Close();
            }
        }


        private void SaveExamAndLabsModelReport(int prognozedModelId,
ExamAndLabsModelReport modelReport)
        {
            using (SqlConnection connection = new SqlConnection(new
ModelSolverContext().SPRConnection))
            using (SqlCommand cmd = new
SqlCommand(saveExamAndLabsModelReportQuery, connection))
            {
                cmd.Parameters.AddWithValue("@ExamAmdLabsModelId",
prognozedModelId);
                cmd.Parameters.AddWithValue("@PrognozedQuality",
modelReport.PrognozedQuality);
```

```
                cmd.Parameters.AddWithValue("@PrognozedSolveTime",
modelReport.PrognozedTime);
                cmd.Parameters.AddWithValue("@PrognozedTotalTime",
modelReport.PrognozedTotalTime);
                cmd.Parameters.AddWithValue("@DateOfPrognoze",
modelReport.DateOfPrognoze);

                connection.Open();
                cmd.ExecuteNonQuery();
                connection.Close();
            }
        }


    }
}
```

ExamAndLabsGradesService.cs

```
using AutoMapper;
using StudentsPrognozer.BLL.DTOs;
using StudentsPrognozer.DAL.Interfaces;
using StudentsPrognozer.DAL.Models;
using System.Collections.Generic;
using System.Linq;

namespace StudentsPrognozer.BLL.Services
{
    public class ExamAndLabsGradesService
    {
        private ISemesterGradesRepository semesterGradesRepository;
        private IExamAndLabsGradesRepository examAndLabsGradesRepository;

        public ExamAndLabsGradesService(ISemesterGradesRepository
semesterGradesRepository, IExamAndLabsGradesRepository
examAndLabsGradesRepository)
        {
            this.semesterGradesRepository = semesterGradesRepository;
            this.examAndLabsGradesRepository = examAndLabsGradesRepository;
        }

        public List<ExamAndLabsGradesDTO> ShowPrognozedResults(string userId,
int semesterModelId)
        {
            var semesterGrades =
semesterGradesRepository.GetSemesterGradesByModelId(semesterModelId);
            var examAndLabsGrades =
examAndLabsGradesRepository.GetLastPrognozedExamAndLabsGrades(userId);

            IMapper mapper = new MapperConfiguration(cfg =>
            {
                cfg.CreateMap<ExamAndLabsGrade, ExamAndLabsGradesDTO>()
                 .ForMember(x => x.LabsGrade, m => m.MapFrom(y => y.LabsGrade))
                 .ForMember(x => x.ExamGrade, m => m.MapFrom(y => y.ExamGrade));

            }).CreateMapper();

            var prognozedGrades = new List<ExamAndLabsGradesDTO>();
            var semesterGradesList = new List<SemesterGrade>(semesterGrades);

            mapper.Map(examAndLabsGrades, prognozedGrades);

            for(int i = 0; i < prognozedGrades.Count; i++)
            {
```

```
                prognozedGrades[i].SemesterGrade = semesterGradesList[i].Grade;
            }

            return prognozedGrades;
        }

        public List<ExamAndLabsGradesDTO>
ShowSemesterExamAndLabsGradesByModelId(int examAndLabsModelId)
        {
            var examAndLabsGrades =
examAndLabsGradesRepository.GetExamAndLabsGradesByModelId(examAndLabsModelId);

            List<ExamAndLabsGrade> elGrades = examAndLabsGrades.ToList();
            int semesterModelId = 0;

            List<ExamAndLabsGradesDTO> prognozedGrades = new
List<ExamAndLabsGradesDTO>();

            if (elGrades != null)
            {
                semesterModelId =
elGrades[0].ExamAndLabsModel.SemesterModel.SemesterModelId;
                var semesterGrades =
semesterGradesRepository.GetSemesterGradesByModelId(semesterModelId);

                IMapper mapper = new MapperConfiguration(cfg =>
                {
                    cfg.CreateMap<ExamAndLabsGrade, ExamAndLabsGradesDTO>()
                .ForMember(x => x.LabsGrade, m => m.MapFrom(y => y.LabsGrade))
                .ForMember(x => x.ExamGrade, m => m.MapFrom(y => y.ExamGrade));

                }).CreateMapper();

                prognozedGrades = new List<ExamAndLabsGradesDTO>();
                var semesterGradesList = new
List<SemesterGrade>(semesterGrades);

                mapper.Map(examAndLabsGrades, prognozedGrades);

                for (int i = 0; i < prognozedGrades.Count; i++)
                {
                    prognozedGrades[i].SemesterGrade =
semesterGradesList[i].Grade;
                }
            }

            return prognozedGrades;
        }

    }
}
```

ExamAndLabsModel.razor

```
@page "/prognozer/ExamAndLabsModel"

@using Microsoft.AspNetCore.Http
@using Microsoft.AspNetCore.Identity
@using StudentsPrognozer.BLL.Services
@using StudentsPrognozer.BLL.DTOs
@using StudentsPrognozer.DAL.Models;
@using StudentsPrognozerWEB.Data;
@using ChartJs.Blazor.Charts
@using ChartJs.Blazor.ChartJS.Common
```

```
@using ChartJs.Blazor.ChartJS.Common.Properties
@using ChartJs.Blazor.ChartJS.Common.Enums
@using ChartJs.Blazor.ChartJS.LineChart
@using ChartJs.Blazor.ChartJS.Common.Axes
@using ChartJs.Blazor.ChartJS.Common.Axes.Ticks
@using ChartJs.Blazor.ChartJS.Common.Handlers
@using ChartJs.Blazor.Util

@inject SemesterModelService ModelService
@inject SemesterGradesService GradesService
@inject ExamAndLabsModelService ELModelService
@inject ExamAndLabsGradesService ELGradesService
@inject ExamAndLabsModelReportService ELModelReportService
@inject NavigationManager NavManager
@inject UserManager<UserInfo> UserManager

<div class="container-fluid" id="container-wrapper">
<div class="row">
<div class="col-lg-4">
<div class="card mb-4">
<div class="card-body">
<EditForm Model="@examAndLabsModel" OnValidSubmit="SendDataToServer">
<DataAnnotationsValidator />
<ValidationSummary />

<div class="form-group">
<label>Модель семестрових оцінок</label>
</div>

<div class="form-group">
<CustomInputSelect @bind-Value="semesterModel.SemesterModelId" class="form-
control">
                                @if (semesterModels != null)
                                {
                                    @foreach (var model in semesterModels)
                                    {
<option value="@model.SemesterModelId">@model.Description</option>
                                    }
                                }
</CustomInputSelect>
</div>

<div class="form-group">
<label>Межі поточного контролю</label>
</div>

<div class="row">
<div class="form-group col-md-6">
<label>З</label>
<InputNumber @bind-Value="examAndLabsModel.LabsMinGrade" class="form-control"
name="minGrade" />
</div>
<div class="form-group col-md-6">
<label>До</label>
<InputNumber @bind-Value="examAndLabsModel.LabsMaxGrade" class="form-control"
name="maxGrade" />
</div>
</div>

<div class="form-group">
<label>Межі екзаменаційного контролю</label>
</div>

<div class="row">
<div class="form-group col-md-6">
```

```
<label>З</label>
<InputNumber @bind-Value="examAndLabsModel.ExamMinGrade" class="form-control"
name="minGrade" />
</div>
<div class="form-group col-md-6">
<label>До</label>
<InputNumber @bind-Value="examAndLabsModel.ExamMaxGrade" class="form-control"
name="maxGrade" />
</div>
</div>
<div class="form-group">
<button type="submit" class="btn btn-primary btn-block">Отримати
результати</button>
</div>
</EditForm>
</div>
</div>
</div>
<div class="col-lg-8">
<div class="card card-fixed-exam-and-lab mb-4 my-custom-scrollbar"
style="height:540px">
<div class="card-header py-3 d-flex flex-row align-items-center justify-content-
between">
<div>
<h6 class="m-0 font-weight-bold text-center text-white">Прогнозовані значення
оцінок</h6>
</div>
                    @if (examAndLabsModelReport != null)
                    {
<div class="text-white">
                            Точність прогнозування:
@examAndLabsModelReport.PrognozedQuality %
</div>
                    }
</div>
<div class="card-body">
                    @if (examAndLabsGrades != null)
                    {
<div class="table-responsive">
<table class="table align-items-center table-flush">
<thead>
<tr>
<th>Дисципліна</th>
<th>Семестрова оцінка</th>
<th>Поточний контроль</th>
<th>Екзамен</th>
</tr>
</thead>
<tbody>
                                    @for (int index = 1; index <=
examAndLabsGrades.Count; index++)
                                    {
<tr>
<td>@index</td>
<td>@examAndLabsGrades[index - 1].SemesterGrade</td>
<td>@examAndLabsGrades[index - 1].LabsGrade</td>
<td>@examAndLabsGrades[index - 1].ExamGrade</td>
</tr>
                                    }
</tbody>
</table>
</div>
                    }
</div>
</div>
```

```
</div>

<div class="col-lg-12">
<div class="card mb-4">
<div class="card-body">
<ChartJsLineChart @ref="_lineChartJs" Config="@_lineConfig" Width="600"
Height="300" />
</div>
</div>
</div>
</div>
</div>

@code {
    [Inject]
    private IHttpContextAccessor HttpContextAccessor { get; set; }
    private UserInfo CurrentUser { get; set; }
    private IEnumerable<SemesterModelDTO> semesterModels;
    private List<ExamAndLabsGradesDTO> examAndLabsGrades;

    private ExamAndLabsModelDTO examAndLabsModel = new ExamAndLabsModelDTO();
    private ExamAndLabsModelReportDTO examAndLabsModelReport;
    private SemesterModelDTO semesterModel = new SemesterModelDTO();
    private LineConfig _lineConfig;
    private ChartJsLineChart _lineChartJs;
    private LineDataset<Point> GradeDataSet;

    protected override async Task OnInitializedAsync()
    {
        CurrentUser = await
UserManager.GetUserAsync(HttpContextAccessor.HttpContext.User);
        examAndLabsModel.UserGuid = CurrentUser.Id;
        semesterModels = await
ELModelService.ShowUserSemesterModels(CurrentUser.Id);
    }

    protected override void OnInitialized()
    {
        _lineConfig = new LineConfig
        {
            Options = new LineOptions
            {
                Responsive = true,
                Title = new OptionsTitle
                {
                    Display = true,
                    Text = "Траєкторія прогнозу оцінок екзаменаційного та
поточного контролю",
                    FontSize = 16
                },
                Legend = new Legend
                {
                    Position = Position.Top
                },
                Tooltips = new Tooltips
                {
                    Mode = InteractionMode.Nearest,
                    Intersect = false
                },
                Scales = new Scales
                {
                    yAxes = new List<CartesianAxis>
{
                        new LinearCartesianAxis
                        {
```

```
                            ScaleLabel = new ScaleLabel
                            {
                                LabelString = "Семестрова оцінка"
                            },
                            Ticks = new LinearCartesianTicks
                            {
                                SuggestedMin = 0,
                                SuggestedMax = 100
                            }
                        }
                    },
                    xAxes = new List<CartesianAxis>
{
                        new LinearCartesianAxis
                        {
                            ScaleLabel = new ScaleLabel
                            {
                                LabelString = "Дисципліна"
                            },
                            Ticks = new LinearCartesianTicks
                            {
                                SuggestedMin = 1
                            }
                        }
                    }
                },
                Hover = new LineOptionsHover
                {
                    Intersect = true,
                    Mode = InteractionMode.Y
                }
            }
        };
    }

    public void FormChartModel()
    {
        _lineConfig.Data.Datasets.Clear();

        GradeDataSet = new LineDataset<Point>
        {
            BackgroundColor =
ColorUtil.FromDrawingColor(System.Drawing.Color.White),
            BorderColor =
ColorUtil.FromDrawingColor(System.Drawing.Color.Green),
            Label = "Семестрова оцінка",
            Fill = false,
            BorderWidth = 2,
            PointRadius = 2,
            PointBorderWidth = 2,
            SteppedLine = SteppedLine.False,
            Hidden = false
        };

        GradeDataSet.AddRange(examAndLabsGrades.Select(p => new
Point(p.SubjectNumber, p.SemesterGrade)));
        _lineConfig.Data.Datasets.Add(GradeDataSet);

         GradeDataSet = new LineDataset<Point>
        {
            BackgroundColor =
ColorUtil.FromDrawingColor(System.Drawing.Color.White),
            BorderColor =
ColorUtil.FromDrawingColor(System.Drawing.Color.Orange),
            Label = "Поточний контроль",
```

```
            Fill = false,
            BorderWidth = 2,
            PointRadius = 2,
            PointBorderWidth = 2,
            SteppedLine = SteppedLine.False,
            Hidden = false
        };

        GradeDataSet.AddRange(examAndLabsGrades.Select(p => new
Point(p.SubjectNumber, p.LabsGrade)));
        _lineConfig.Data.Datasets.Add(GradeDataSet);

         GradeDataSet = new LineDataset<Point>
        {
            BackgroundColor =
ColorUtil.FromDrawingColor(System.Drawing.Color.White),
            BorderColor = ColorUtil.FromDrawingColor(System.Drawing.Color.Red),
            Label = "Екзамен",
            Fill = false,
            BorderWidth = 2,
            PointRadius = 2,
            PointBorderWidth = 2,
            SteppedLine = SteppedLine.False,
            Hidden = false
        };

        GradeDataSet.AddRange(examAndLabsGrades.Select(p => new
Point(p.SubjectNumber, p.ExamGrade)));
        _lineConfig.Data.Datasets.Add(GradeDataSet);
        _lineChartJs.Update();
    }

    public void SendDataToServer()
    {
        if (semesterModel.SemesterModelId == 0 && semesterModels != null) {
semesterModel = semesterModels.First(); }

        examAndLabsModel.SemesterModelId = semesterModel.SemesterModelId;
        ELModelService.PrognozeExamAndLabsGrades(examAndLabsModel);
        examAndLabsGrades = ELGradesService.ShowPrognozedResults(CurrentUser.Id,
semesterModel.SemesterModelId);
        examAndLabsModelReport =
ELModelReportService.ShowPrognozedExamAndLabsModelReport(CurrentUser.Id);

        int rangeCounter = 1;
        foreach (var examAndLabsGrade in examAndLabsGrades)
        {
            examAndLabsGrade.SubjectNumber = rangeCounter;
            rangeCounter++;
        }

        FormChartModel();
    }
}
```

ДОДАТОК Б. Лістинг програмного коду алгоритму машинного навчання FastTree

StudentsExcellencePredictorRepository.cs

```
using System;
using System.Data.SqlClient;
using System.Diagnostics;
using Microsoft.ML;
using Microsoft.ML.Data;
using MLNETIntegration.Configurations;
using MLNETIntegration.Interfaces;
using MLNETIntegration.Models;

namespace MLNETIntegration.Repositories
{
    public class StudentsExcellencePredictorRepository :
IStudentsExcellencePredictorRepository
    {
        private const string STUDENTS_EXCELLENCE_SELECT = @"SELECT

[GoOutWithFriendsPK] AS GoOutWithFriends,
[HealthStatusPK] AS HealthStatus,
[StudyTimePK] AS StudyTime,
[TravelTimePK] AS TravelTime,
[GenderPK] AS Gender,
[CohabitationStatusPK] AS CohabitationStatus,
[MotherEducationPK] AS MotherEducation,
[FatherEducationPK] AS FatherEducation,
[HasExtraCourses],
[HasExtraActivities],
[HasRomanticRelationships],
[Age],
[FamilyRelationshipsQualityPK],
[FreeTimePK] AS FreeTime,
[AbsencesPK] AS Absences,
[Grade]
FROM [StudentExcellence] Order By [Grade]";


        private MLContext Context { get; set; }

        public void TrainFactorBasedModel()
        {
            try
            {
                var dbSource = new DatabaseSource(SqlClientFactory.Instance,
DBHelper.GetDbConnection(), STUDENTS_EXCELLENCE_SELECT);

                Context = new MLContext();

                var loader =
Context.Data.CreateDatabaseLoader<StudentExcellenceML>();

                Console.WriteLine("Loading data from database...");
                var data = loader.Load(dbSource);

                var set = Context.Data.TrainTestSplit(data, testFraction: 0.2);
                var trainingData = set.TrainSet;
                var testData = set.TestSet;

                Console.WriteLine("Preparing training operations...");
```

```
                var pipeline = Context.Transforms
                        .CopyColumns(outputColumnName: "Label", inputColumnName:
"Grade")

.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"StudyTimeF", inputColumnName: "StudyTime"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"HealthStatusF", inputColumnName: "HealthStatus"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"GoOutWithFriendsF", inputColumnName: "GoOutWithFriends"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"TravelTimeF", inputColumnName: "TravelTime"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"GenderF", inputColumnName: "Gender"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"CohabitationStatusF", inputColumnName: "CohabitationStatus"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"MotherEducationF", inputColumnName: "MotherEducation"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"FatherEducationF", inputColumnName: "FatherEducation"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"HasExtraCoursesF", inputColumnName: "HasExtraCourses"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"HasExtraActivitiesF", inputColumnName: "HasExtraActivities"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"HasRomanticRelationshipsF", inputColumnName: "HasRomanticRelationships"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName: "Age",
inputColumnName: "Age"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"FamilyRelationshipsQualityF", inputColumnName: "FamilyRelationshipsQuality"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"FreeTimeF", inputColumnName: "FreeTime"))
.Append(Context.Transforms.Categorical.OneHotEncoding(outputColumnName:
"AbsencesF", inputColumnName: "Absences"))
.Append(Context.Transforms.Concatenate("Features", "GoOutWithFriendsF",
"HealthStatusF", "StudyTimeF", "TravelTimeF", "GenderF", "CohabitationStatusF",
"MotherEducationF", "FatherEducationF", "HasExtraCoursesF",
"HasExtraActivitiesF", "HasRomanticRelationshipsF","FreeTimeF", "AbsencesF"))
.Append(Context.Regression.Trainers.FastTree());

                var model = pipeline.Fit(trainingData);

                Context.Model.Save(model, data.Schema,
"FactorBasedPredictionModel.zip");

                // Measure trained model performance
                // Apply data prep transformer to test data
                IDataView transformedTestData = model.Transform(testData);

                // Extract model metrics and get RSquared
                RegressionMetrics trainedModelMetrics =
Context.Regression.Evaluate(transformedTestData);
                double rSquared = trainedModelMetrics.RSquared;

                Debug.WriteLine("Evaluation results:");
                Debug.WriteLine($"RSquared : " + trainedModelMetrics.RSquared);
                Debug.WriteLine($"MeanAbsoluteError : " +
trainedModelMetrics.MeanAbsoluteError);
                Debug.WriteLine($"MeanSquaredError : " +
trainedModelMetrics.MeanSquaredError);
                Debug.WriteLine($"RootMeanSquaredError : " +
trainedModelMetrics.RootMeanSquaredError);
                Debug.WriteLine($"LossFunction : " +
trainedModelMetrics.LossFunction);
            }
```

```
            catch(Exception ex)
            {
                Console.WriteLine(ex);
            }
        }

        public void PredictGrade(StudentExcellenceML studentExcellenceModel)
        {
            Context = new MLContext();

            var trainedModel =
Context.Model.Load("FactorBasedPredictionModel.zip", out var modelSchema);

            var predictionEngine =
Context.Model.CreatePredictionEngine<StudentExcellenceML,
StudentExcellencePrediction>(trainedModel);

            var prediction = predictionEngine.Predict(studentExcellenceModel);

            studentExcellenceModel.Grade = prediction.Grade;
        }
    }
}
```

StudentsExcellenceService.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using AutoMapper;
using MLNETIntegration.Interfaces;
using MLNETIntegration.Models;
using StudentsPrognozer.BLL.DTOs.FactorBased;
using StudentsPrognozer.DAL.Interfaces;
using StudentsPrognozer.DAL.Models;

namespace StudentsPrognozer.BLL.Services
{
    public class StudentsExcellenceService
    {
        private IStudentsExcellencePredictorRepository
studentsExcellencePredictorRepository;
        private IStudentsExcellenceRepository studentsExcellenceRepository;

        public StudentsExcellenceService(IStudentsExcellencePredictorRepository
studentsExcellencePredictorRepository, IStudentsExcellenceRepository
studentsExcellenceRepository)
        {
            this.studentsExcellencePredictorRepository =
studentsExcellencePredictorRepository;
            this.studentsExcellenceRepository = studentsExcellenceRepository;
        }

        public void PredictGrafeByFactors(string userId, StudentExcellenceDTO
studentExcellenceDTO)
        {
            var mapper = new MapperConfiguration(cfg =>
cfg.CreateMap<StudentExcellenceDTO, StudentExcellenceML>()
  .ForMember(dest => dest.Age, m => m.MapFrom(source => source.Age))
  .ForMember(dest => dest.CohabitationStatus, m => m.MapFrom(source =>
source.CohabitationStatus - 1))
  .ForMember(dest => dest.FamilyRelationshipsQuality, m => m.MapFrom(source =>
source.FamilyRelationshipsQuality))
  .ForMember(dest => dest.FatherEducation, m => m.MapFrom(source =>
source.FatherEducation - 1))
```

```
.ForMember(dest => dest.FreeTime, m => m.MapFrom(source => source.FreeTime))
.ForMember(dest => dest.Gender, m => m.MapFrom(source => source.Gender - 1))
.ForMember(dest => dest.GoOutWithFriends, m => m.MapFrom(source =>
source.GoOutWithFriends))
.ForMember(dest => dest.HasExtraActivities, m => m.MapFrom(source =>
source.HasExtraActivities))
.ForMember(dest => dest.HasExtraCourses, m => m.MapFrom(source =>
source.HasExtraCourses))
.ForMember(dest => dest.HasRomanticRelationships, m => m.MapFrom(source =>
source.HasRomanticRelationships))
.ForMember(dest => dest.HealthStatus, m => m.MapFrom(source =>
source.HealthStatus))
.ForMember(dest => dest.MotherEducation, m => m.MapFrom(source =>
source.MotherEducation - 1))
.ForMember(dest => dest.StudyTime, m => m.MapFrom(source => source.StudyTime))
.ForMember(dest => dest.TravelTime, m => m.MapFrom(source => source.TravelTime))
           ).CreateMapper();

            var studentExcellenceMl = mapper.Map<StudentExcellenceDTO,
StudentExcellenceML>(studentExcellenceDTO);
            studentsExcellencePredictorRepository.TrainFactorBasedModel();

studentsExcellencePredictorRepository.PredictGrade(studentExcellenceMl);

            mapper = new MapperConfiguration(cfg =>
cfg.CreateMap<StudentExcellenceML, StudentExcellence>()
.ForMember(dest => dest.Age, m => m.MapFrom(source => source.Age))
.ForMember(dest => dest.CohabitationStatusPk, m => m.MapFrom(source =>
Convert.ToByte(source.CohabitationStatus)))
.ForMember(dest => dest.FamilyRelationshipsQualityPk, m => m.MapFrom(source =>
source.FamilyRelationshipsQuality))
.ForMember(dest => dest.FatherEducationPk, m => m.MapFrom(source =>
source.FatherEducation))
.ForMember(dest => dest.FreeTimePk, m => m.MapFrom(source => source.FreeTime))
.ForMember(dest => dest.GenderPk, m => m.MapFrom(source => source.Gender))
.ForMember(dest => dest.GoOutWithFriendsPk, m => m.MapFrom(source =>
source.GoOutWithFriends))
.ForMember(dest => dest.HasExtraActivities, m => m.MapFrom(source =>
source.HasExtraActivities))
.ForMember(dest => dest.HasExtraCourses, m => m.MapFrom(source =>
source.HasExtraCourses))
.ForMember(dest => dest.HasRomanticRelationships, m => m.MapFrom(source =>
source.HasRomanticRelationships))
.ForMember(dest => dest.HealthStatusPk, m => m.MapFrom(source =>
source.HealthStatus))
.ForMember(dest => dest.MotherEducationPk, m => m.MapFrom(source =>
source.MotherEducation))
.ForMember(dest => dest.StudyTimePk, m => m.MapFrom(source => source.StudyTime))
.ForMember(dest => dest.TravelTimePk, m => m.MapFrom(source =>
source.TravelTime))
           ).CreateMapper();

            var studentExcellence = mapper.Map<StudentExcellenceML,
StudentExcellence>(studentExcellenceMl);
            studentExcellence.UserId = userId;
            studentExcellence.StudentExcellenceId = Guid.NewGuid();
            studentExcellence.DateCreated = DateTime.Now;

            try
            {

studentsExcellenceRepository.SaveUserInputData(studentExcellence);
            }
            catch (Exception ex)
            {
```

```
                Console.WriteLine(ex);
            }
        }

        public bool IsUserSurveyResultsAlreadyExist(string userId)
            =>
studentsExcellenceRepository.IsUserSurveyResultsAlreadyExist(userId);

        public float GetUserPredictedGrade(string userId)
            => studentsExcellenceRepository.GetUserPredictedGrade(userId);

        public Task<List<StudentExcellence>> GetUserPredictions(string userId)
          =>
Task.FromResult(studentsExcellenceRepository.GetUserPredictions(userId));
    }
}
```

FactorBasedSurvey.razor

```
@page "/prognozer/FactorBased/Survey"

@using Microsoft.AspNetCore.Http
@using Microsoft.AspNetCore.Identity
@using StudentsPrognozer.BLL.Services
@using StudentsPrognozer.BLL.DTOs
@using StudentsPrognozer.BLL.DTOs.FactorBased
@using StudentsPrognozer.DAL.Models;
@using StudentsPrognozer.BLL.Helpers;
@using StudentsPrognozer.BLL.Enums;
@using StudentsPrognozerWEB.Data;

@inject NavigationManager NavManager
@inject UserManager<UserInfo> UserManager
@inject StudentsExcellenceService StudentExcellenceService

<div class="d-flex justify-content-center">
<div class="card card-body">
<EditForm Model="@studentExcellenceModel" OnValidSubmit="PrognozeGrade">
<DataAnnotationsValidator />
<ValidationSummary />
<div class="row">
<div class="col-md-6">
<div class="form-group required">
<label class="font-weight-bold control-label">1. Ваша стать: </label>
<CustomInputSelect @bind-Value="studentExcellenceModel.Gender" class="form-
control">
                            @foreach (var item in EnumHelper.GetAll<Gender>())
                            {
<option value="@item.Key">@item.Value</option>
                            }
</CustomInputSelect>
</div>

<div class="form-group">
<label class="font-weight-bold control-label">2. Ваш вік: </label>
<InputNumber @bind-Value="studentExcellenceModel.Age" class="form-control" />
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">3. Чи проживаєте ви разом з
батьками?: </label>
<CustomInputSelect @bind-Value="studentExcellenceModel.CohabitationStatus"
class="form-control">
                            @foreach (var item in
EnumHelper.GetAll<CohabitationStatus>())
```

```
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">4. Яка освіта у вашого
батька?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.FatherEducation"
class="form-control">
                                @foreach (var item in
EnumHelper.GetAll<EducationDegree>())
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">5. Яка освіта у вашої
матері?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.MotherEducation"
class="form-control">
                                @foreach (var item in
EnumHelper.GetAll<EducationDegree>())
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">6. Скільки часу ви добираєтесь до
навчального закладу?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.TravelTime" class="form-
control">
                                @foreach (var item in
EnumHelper.GetAll<TravelTime>())
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">7. Скільки часу ви приділяєте
навчанню ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.StudyTime" class="form-
control">
                                @foreach (var item in
EnumHelper.GetAll<StudyTime>())
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>
</div>
<div class="col-md-6">

<div class="form-group required">
<label class="font-weight-bold control-label">8. Чи відвідуєте ви додаткові
курси поза навчальною програмою ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.HasExtraCourses"
class="form-control">
```

```
                            @foreach (var item in
EnumHelper.GetAll<ExtraCourses>())
                            {
<option value="@item.Key">@item.Value</option>
                            }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">9. Чи маєте інші додаткові
активності поза навчальною програмою ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.HasExtraActivities"
class="form-control">
                            @foreach (var item in
EnumHelper.GetAll<ExtraActivities>())
                            {
<option value="@item.Key">@item.Value</option>
                            }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">10. Чи знаходитесь ви зараз у
романтичних стосунках ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.HasRomanticRelationships"
class="form-control">
                            @foreach (var item in
EnumHelper.GetAll<RomanticRelationhips>())
                            {
<option value="@item.Key">@item.Value</option>
                            }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">11. Скільки у вас є вільного часу
поза навчанням ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.FreeTime" class="form-
control">
                            @foreach (var item in EnumHelper.GetAll<FreeTime>())
                            {
<option value="@item.Key">@item.Value</option>
                            }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">12. Як часто ви виходите гуляти з
друзями ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.GoOutWithFriends"
class="form-control">
                            @foreach (var item in
EnumHelper.GetAll<GoOutWithFriends>())
                            {
<option value="@item.Key">@item.Value</option>
                            }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">13. Як ви оцінюєте свої стосунки з
родичами ?:</label>
<CustomInputSelect @bind-
Value="studentExcellenceModel.FamilyRelationshipsQuality" class="form-control">
                            @foreach (var item in
```

```
EnumHelper.GetAll<FamilyRelationshipsQuality>())
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>

<div class="form-group required">
<label class="font-weight-bold control-label">14. Як ви оцінюєте свій стан
здоров'я ?:</label>
<CustomInputSelect @bind-Value="studentExcellenceModel.HealthStatus"
class="form-control">
                                @foreach (var item in
EnumHelper.GetAll<HealthStatus>())
                                {
<option value="@item.Key">@item.Value</option>
                                }
</CustomInputSelect>
</div>

<div class="form-group">
<button type="submit" class="btn btn-primary btn-block">Завершити</button>
</div>
</div>
</div>
</EditForm>
</div>
</div>

@if (NonValidDataDialogOpen)
{
<ModalDialogOk Text="Необхідно заповнити усі обов'язкові поля."
             OnClose="@OnNonValidDataDialogClose">
</ModalDialogOk>
}

@if (AreYouSureToSaveDataDialogOpen)
{
<ModalDialogOkCancel Text="Чи дійсно ви хочете завершити опитування?"
             OnClose="@OnAreYouSureToSaveDataDialogClose">
</ModalDialogOkCancel>
}


@code {
    [Inject]
    private IHttpContextAccessor HttpContextAccessor { get; set; }
    private UserInfo CurrentUser { get; set; }

    private StudentExcellenceDTO studentExcellenceModel = new
StudentExcellenceDTO();

    public bool NonValidDataDialogOpen { get; set; }

    public bool AreYouSureToSaveDataDialogOpen { get; set; }

    protected override async Task OnInitializedAsync()
    {
        CurrentUser = await
UserManager.GetUserAsync(HttpContextAccessor.HttpContext.User);
    }

    public void PrognozeGrade()
    {
        if (studentExcellenceModel.CohabitationStatus == 0 ||
```

```
            studentExcellenceModel.FatherEducation == 0 ||
            studentExcellenceModel.FreeTime == 0 ||
            studentExcellenceModel.Gender == 0 ||
            studentExcellenceModel.GoOutWithFriends == 0 ||
            studentExcellenceModel.HealthStatus == 0 ||
            studentExcellenceModel.StudyTime == 0 ||
            studentExcellenceModel.TravelTime == 0 ||
            studentExcellenceModel.MotherEducation == 0 ||
            studentExcellenceModel.FamilyRelationshipsQuality == 0)
        {
            OpenNonValidDataDialog();
        } else
        {
            OpenAreYouSureToSaveDataDialog();
        }
    }

    private async Task OnNonValidDataDialogClose(bool accepted)
    {
        NonValidDataDialogOpen = false;
        StateHasChanged();
    }

    private void OpenNonValidDataDialog()
    {
        NonValidDataDialogOpen = true;
        StateHasChanged();
    }

    private async Task OnAreYouSureToSaveDataDialogClose(bool accepted)
    {
        AreYouSureToSaveDataDialogOpen = false;
        StateHasChanged();

        if (accepted)
        {
            StudentExcellenceService.PredictGrafeByFactors(CurrentUser.Id,
studentExcellenceModel);
            NavManager.NavigateTo("/prognozer/FactorBased/SurveyResult");
        }
    }

    private void OpenAreYouSureToSaveDataDialog()
    {
        AreYouSureToSaveDataDialogOpen = true;
        StateHasChanged();
    }
}
```