МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ОЛІЙНИК КОСТЯНТИН АНАТОЛІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р. техн. наук, професор
____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2025 р.

ЦИФРОВА ПЛАТФОРМА ЗБОРУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ РОЗВИТКУ УНІВЕРСИТЕТІВ УКРАЇНИ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Петро НІКОЛЮК, професор кафедри
інформаційних технологій ,
д-р фіз.-мат. наук, професор
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2025

АНОТАЦІЯ

Олійник К.А. Цифрова платформа збору даних для прогнозування розвитку університетів України. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2025.

У роботі розглянуто актуальні проблеми цифровізації українських університетів та визначено основні методи прогнозування розвитку вищих навчальних закладів в умовах сучасних технологічних змін. Автором розроблено цифрову платформу, що забезпечує ефективний збір, збереження та аналіз даних за допомогою алгоритмів машинного навчання, що дозволяє здійснювати прогнозування результатів розвитку університетів. Вивчено застосування новітніх інструментів для обробки великих даних та виявлення тенденцій у розвитку вищої освіти на основі статистичних та аналітичних методів. Створена система сприятиме покращенню процесів прийняття управлінських рішень і оптимізації ресурсів закладів вищої освіти на різних етапах їх розвитку. Особливу увагу приділено інтеграції платформи з існуючими освітніми системами та її здатності до масштабування на рівні національної системи вищої освіти України.

Ключові слова: цифровізація, університет, прогнозування, платформа, дані, аналітика.

87 с., 7 табл., 33 рис., 2 дод., 26 джерел

ABSTRACT

Oliynyk K.A. Digital data collection platform for forecasting the development of Ukrainian universities. Specialty 122 "Computer Science", educational program "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia 2025.

The paper considers the current problems of digitalization of Ukrainian universities and identifies the main methods for forecasting the development of higher education institutions in the context of modern technological changes. The author has developed a digital platform that provides effective collection, storage and analysis of data using machine learning algorithms, which allows predicting the results of university development. The application of the latest tools for processing big data and identifying trends in the development of higher education based on statistical and analytical methods has been studied. The created system will contribute to improving the processes of making managerial decisions and optimizing the resources of higher education institutions at different stages of their development. Particular attention is paid to the integration of the platform with existing educational systems and its ability to scale at the level of the national higher education system of Ukraine.

Keywords: digitalization, university, forecasting, platform, data, analytics.

87 p., 7 tables, 33 figures, 2 appendix, 26 sources

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Інтерфейс програмування застосунків

ARIMA - Автокорегресивна інтегрована модель ковзного середнього

CRM - Управління взаємовідносинами з клієнтами

ECTS - Європейська система переведення та накопичення кредитів

ERP - Система управління підприємством

EHEA - Європейська зона вищої освіти

GUI - Графічний інтерфейс користувача

LMS - Система управління навчанням

LSTM - Довготривала короткочасна пам’ять

XGBoost - Extreme Gradient Boosting

UML - Уніфікована мова моделювання

БД - База даних

ЄДЕБО - Єдина державна електронна база з питань освіти

ЗВО - Заклад вищої освіти

ВСТУП

У сучасних умовах розвитку вищої освіти України спостерігається низка системних проблем, що впливають на ефективність управління закладами вищої освіти. Серед них зниження кількості студентів, недостатнє фінансування, відтік кваліфікованих кадрів та низький рівень цифрової зрілості більшості університетів. Водночас цифровізація та розвиток інформаційних технологій відкривають нові можливості для підвищення ефективності управління, аналітики та стратегічного планування. У цьому контексті створення єдиної цифрової платформи збору та прогнозної аналітики освітніх даних є актуальною науково-практичною задачею, яка може забезпечити оптимізацію ресурсів, підвищення якості освітніх послуг і сталий розвиток університетів на національному рівні.

Об’єктом дослідження є процес цифрової трансформації та управління даними у вищих навчальних закладах України.

Предметом дослідження є методи, моделі та технології збору, обробки та прогнозування даних, що забезпечують ефективне стратегічне управління розвитком університетів.

Мета дослідження полягає у розробці концепції та реалізації цифрової платформи збору та прогнозної аналітики даних університетів України для підтримки прийняття обґрунтованих управлінських рішень.

Для досягнення поставленої мети були сформульовані наступні завдання дослідження:

1. Проаналізувати сучасний стан цифровізації та управління даними у вищих навчальних закладах України, виявити існуючі проблеми та потреби.
2. Дослідити методи збору, обробки та прогнозування даних, включаючи класичні статистичні підходи та сучасні алгоритми машинного навчання.
3. Розробити концепцію цифрової платформи збору та аналізу даних про університети з інтеграцією розрізнених систем та уніфікацією форматів інформації.

4. Реалізувати модулі платформи для збору, обробки, візуалізації та прогнозування даних.

5. Провести тестування платформи та оцінити її ефективність у контексті стратегічного управління закладами вищої освіти.

Методи дослідження включали: аналітичний метод для вивчення наукових джерел та сучасних практик управління університетами; порівняльний аналіз для визначення сильних і слабких сторін існуючих систем цифрового обліку та прогнозування; методи машинного навчання PROPheT, LSTM, Random Forest та XGBoost для прогнозування ключових показників розвитку університетів.

Наукова новизна роботи полягає у комплексному підході до створення цифрової платформи для збору, обробки та прогнозної аналітики даних університетів України, який поєднує алгоритми машинного навчання та аналітичні інструменти. Вперше запропоновано інтегровану архітектуру системи з можливістю прогнозування ключових показників діяльності закладів вищої освіти та оцінки ризиків управлінських рішень.

Практичне значення роботи полягає у можливості впровадження розробленої платформи у державні та регіональні системи управління вищою освітою, що забезпечить:

- централізоване зберігання та уніфікацію даних;
- автоматизацію збору та верифікації інформації;
- ефективне використання аналітики для стратегічного планування та прийняття управлінських рішень
- прогнозування потреб у студентських контингентах та викладацьких кадрах.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Стан та тенденції розвитку вищої освіти в Україні

1.1.1 Історичні передумови та формування системи вищої освіти

У XIX столітті розвиток університетської освіти в Україні відбувався в межах імперської системи управління. Створення Харківського (1804 р.), Київського (1834 р.) та Одеського (1865 р.) університетів започаткувало становлення класичної моделі вищої школи. Освітній процес базувався на європейських стандартах, проте був підпорядкований імперській централізації. У період Української Народної Республіки (1917-1921 рр.) вперше були спроби створення незалежної системи освіти з національним змістом. З утвердженням радянської влади система вищої освіти набула жорсткої централізованої форми, орієнтованої на підготовку фахівців для промисловості, науки та оборони [1].

Після здобуття Україною незалежності у 1991 році розпочалася реформація системи вищої освіти відповідно до європейських стандартів. Ключовим кроком стало приєднання України до Болонського процесу у 2005 році, що забезпечило інтеграцію у Європейський простір вищої освіти (ЕНЕА) [1]. Це сприяло запровадженню трирівневої структури навчання (бакалавр - магістр - доктор філософії), кредитно-модульної системи (ECTS) та механізмів забезпечення якості освіти. Водночас, починаючи з 2010-х років, розвиток інформаційних технологій і цифрових платформ поступово почав впливати на зміст, методи і форми освітнього процесу. З’явилися перші системи дистанційного навчання, електронні бібліотеки, відкриті освітні ресурси. Ці процеси заклали основу для майбутньої цифрової трансформації університетів, що сьогодні стає одним із ключових напрямів реформування вищої освіти України.

1.1.2 Сучасний стан університетів та основні виклики

На сучасному етапі система вищої освіти України характеризується значною кількістю закладів, розгалуженою структурою та водночас глибокими викликами, пов’язаними з демографічними, економічними, воєнними та технологічними змінами [2]. Кількість студентів за останні десять років зменшилася більш ніж на третину, що зумовлено демографічним спадом, міграційними процесами та зниженням престижу деяких спеціальностей. Серед ключових викликів, з якими стикаються українські університети, можна виділити:

- застарілу матеріально-технічну базу, що ускладнює впровадження сучасних ІТ-рішень;
- нестачу фінансування та недостатню диверсифікацію джерел доходів університетів;
- відтік наукових кадрів і студентів за кордон;
- низький рівень цифрової зрілості у більшості ЗВО;
- відсутність системної аналітики та моніторингу показників розвитку університетів.

В умовах воєнного стану 2022-2025 років освітня система продемонструвала високу стійкість, проте водночас постала потреба у нових цифрових рішеннях для управління навчальним процесом, моніторингу якості освіти та прогнозування розвитку ЗВО в умовах нестабільності. Багато університетів вимушено перейшли на змішану або дистанційну форму навчання, що стимулювало активне впровадження LMS-систем (Moodle, Canvas, Google Classroom), проте значна частина даних про освітні результати, кадровий потенціал, наукову діяльність залишилася розпорошеною і неінтегрованою. Таким чином, сучасний стан вищої освіти в Україні характеризується не лише викликами, а й можливостями для розвитку - передусім через створення єдиних цифрових платформ збору, аналізу та прогнозування освітніх даних. Такі рішення можуть стати ключовим елементом стратегічного управління у сфері вищої освіти.

1.1.3 Тенденції цифровізації освітнього простору

Цифровізація вищої освіти - це системний процес трансформації традиційних підходів до навчання, управління та наукової діяльності за допомогою інформаційно-комунікаційних технологій [3]. Вона передбачає створення цифрових освітніх екосистем, у яких дані, ресурси та сервіси інтегровані в єдиному інформаційному середовищі.

Багато ЗВО переходять до електронного документообігу, впроваджують CRM-системи для роботи з абітурієнтами, ERP-рішення для управління ресурсами, а також аналітичні панелі (дашборди) для моніторингу освітніх показників. Застосування LMS-систем, онлайн-курсів, відкритих освітніх платформ (Prometheus, Coursera, EdX) забезпечує гнучкість навчання і дозволяє збирати великі обсяги освітніх даних (learning analytics). На основі цих даних університети можуть аналізувати успішність студентів, активність викладачів, ефективність освітніх програм.

Використання інструментів Business Intelligence (Power BI, Grafana, Tableau) поступово стає необхідністю для ухвалення управлінських рішень. Збір і аналіз освітніх даних дозволяє прогнозувати динаміку вступу, оцінювати навантаження викладачів, оптимізувати фінансові процеси. Технології Big Data, Machine Learning та AI відкривають можливість створення адаптивних систем навчання та прогнозування розвитку університетів. В Україні лише починають формуватися перші пілотні проєкти у цій сфері, проте їх потенціал є надзвичайно високим.

Важливою тенденцією є поява відкритих освітніх даних (Open Data) та інтеграція ЗВО з державними платформами - зокрема, Єдиною державною електронною базою з питань освіти (ЄДЕБО), “Дія.Освіта”, Національним репозиторієм академічних текстів тощо. На рис.1.1 відображено платформу ЄДЕБО.

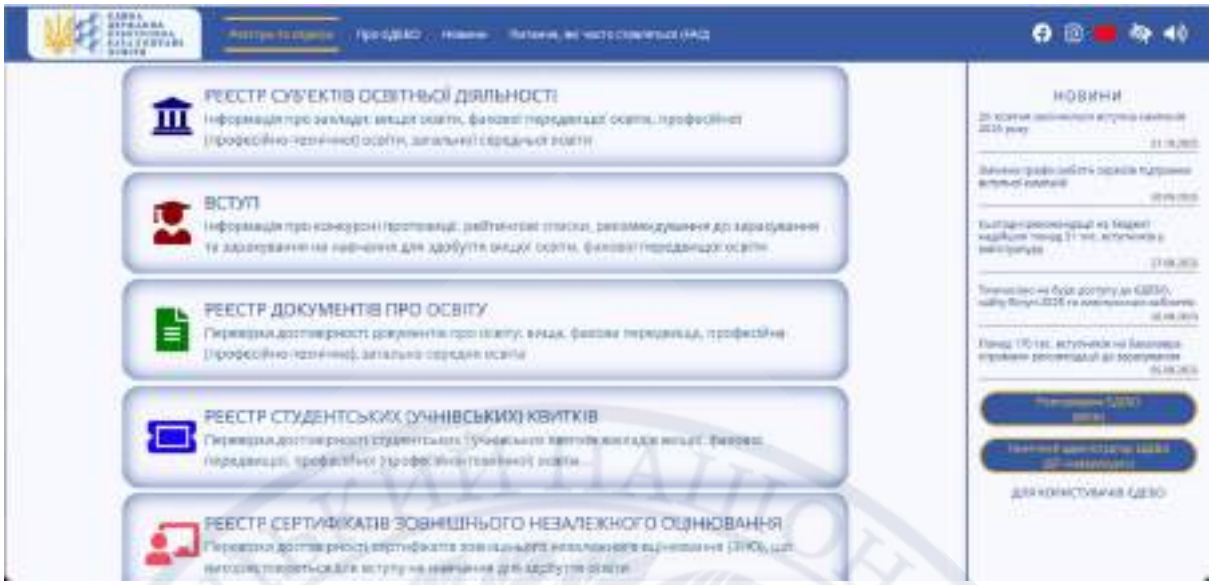

Рисунок 1.1 - ЄДЕБО [4]

Отже, цифровізація освіти - це не лише технологічне оновлення, а й зміна парадигми управління знаннями. Вона вимагає створення гнучких цифрових платформ збору даних, які дозволяють комплексно аналізувати розвиток університетів, визначати закономірності й формувати науково обґрунтовані прогнози.

1.1.4 Проблеми збору, обробки та аналізу даних у закладів освіти

Попри значний прогрес у напрямі цифровізації, більшість українських університетів стикаються з низкою проблем, пов’язаних із відсутністю уніфікованих систем збору та аналітики освітніх даних. Нині інформація про студентів, наукову діяльність, кадровий потенціал і матеріально-технічну базу зберігається у різних форматах - у локальних базах даних, електронних таблицях або навіть паперових документах.

Університети використовують різні програмні засоби (1С, Moodle, Google Workspace, локальні CRM), які не інтегровані між собою. Це призводить до дублювання даних, втрати актуальності та ускладнює створення комплексної аналітичної звітності. Дані часто зберігаються у несумісних форматах (CSV, Excel, PDF), що унеможливлює централізований аналіз. Немає єдиних протоколів обміну освітньою

інформацією між ЗВО та державними структурами. Більшість університетів не мають розвинених систем бізнес-аналітики. Звіти формуються вручну, що потребує значних часових і кадрових ресурсів. Через це керівники ЗВО приймають рішення, спираючись на неповні або застарілі дані.

Відсутність контролю за верифікацією інформації призводить до помилок, неточностей та суперечностей у базах даних. Це унеможливлює побудову надійних прогнозних моделей розвитку університетів. Викладачі та адміністративні працівники часто не мають навичок роботи з сучасними аналітичними інструментами (BI-системами, базами даних, хмарними сервісами). Це сповільнює впровадження цифрових інновацій. Нині в Україні немає єдиного середовища, що акумулювало б освітні дані з університетів, дозволяло аналізувати динаміку розвитку, прогнозувати кадрові потреби та ефективність освітньої політики. Вирішення цих проблем потребує створення єдиної цифрової платформи збору та аналітики даних про розвиток університетів України, яка б об’єднала інформацію з різних джерел, забезпечила її достовірність і дозволила здійснювати прогнозування на основі сучасних технологій штучного інтелекту. Саме така система може стати інструментом підтримки стратегічних рішень у сфері освіти, сприяти підвищенню якості управління та розвитку освітньої аналітики на національному рівні.

1.2 Використання інформаційних технологій у сфері освіти

1.2.1 Еволюція IT-рішень для управління університетами

Еволюція IT-рішень у сфері управління університетами відбувалася поетапно - від базових електронних реєстрів і локальних баз даних до комплексних корпоративних інформаційних систем [5]. На початкових етапах розвитку інформаційних технологій (1990-2000 рр.) основна увага приділялася автоматизації адміністративних процесів: веденню електронних журналів, обліку контингенту студентів, формуванню розкладів занять тощо [5]. Подальший розвиток (2000-2010 рр.)

характеризувався впровадженням інтегрованих систем управління освітніми закладами, які охоплювали не лише адміністративну, а й навчально-методичну діяльність. З’явилися перші університетські портали, що забезпечували комунікацію між студентами, викладачами та адміністрацією [5]. У сучасний період (з 2010-х рр.) університети активно впроваджують корпоративні системи класу ERP (Enterprise Resource Planning), які дозволяють об’єднати фінансові, кадрові, академічні та дослідницькі підсистеми в єдине інформаційне середовище. Впровадження хмарних технологій та мобільних застосунків створює умови для гнучкого управління ресурсами та прийняття рішень на основі даних у режимі реального часу.

1.2.2 Освітні аналітичні системи

Одним із ключових напрямів сучасної цифровізації освіти є використання аналітичних систем, що базуються на обробці великих обсягів освітніх даних (Big Data) та принципах бізнес-аналітики (Business Intelligence). Освітня аналітика (Learning Analytics) передбачає збір, обробку та інтерпретацію даних про діяльність студентів і викладачів з метою покращення результатів навчання та оптимізації освітніх стратегій. Такі системи дозволяють здійснювати прогнозування академічної успішності, виявляти ризики відрахування студентів, аналізувати ефективність педагогічних методів і програм підготовки. На рівні управління університетом BI-рішення забезпечують підтримку прийняття рішень, моніторинг ключових показників ефективності (KPI) та стратегічне планування розвитку закладу. Використання аналітики у сфері освіти сприяє переходу від інтуїтивного до доказового управління (evidence-based management), де рішення ухвалюються на основі об’єктивних даних.

1.2.3 Практика впровадження систем управління освітою

Системи управління освітнім процесом (Learning Management Systems, LMS) стали невід'ємним елементом інфраструктури сучасних університетів. Вони забезпечують організацію дистанційного та змішаного навчання, контроль за виконанням навчальних планів, збереження навчальних матеріалів та комунікацію між усіма учасниками освітнього процесу. Серед найпоширеніших LMS-рішень у закладах вищої освіти - Moodle, Canvas, Blackboard, Google Classroom. На рис.1.2 відображено платформу Moodle та вклад в цифровізацію освітнього процесу.

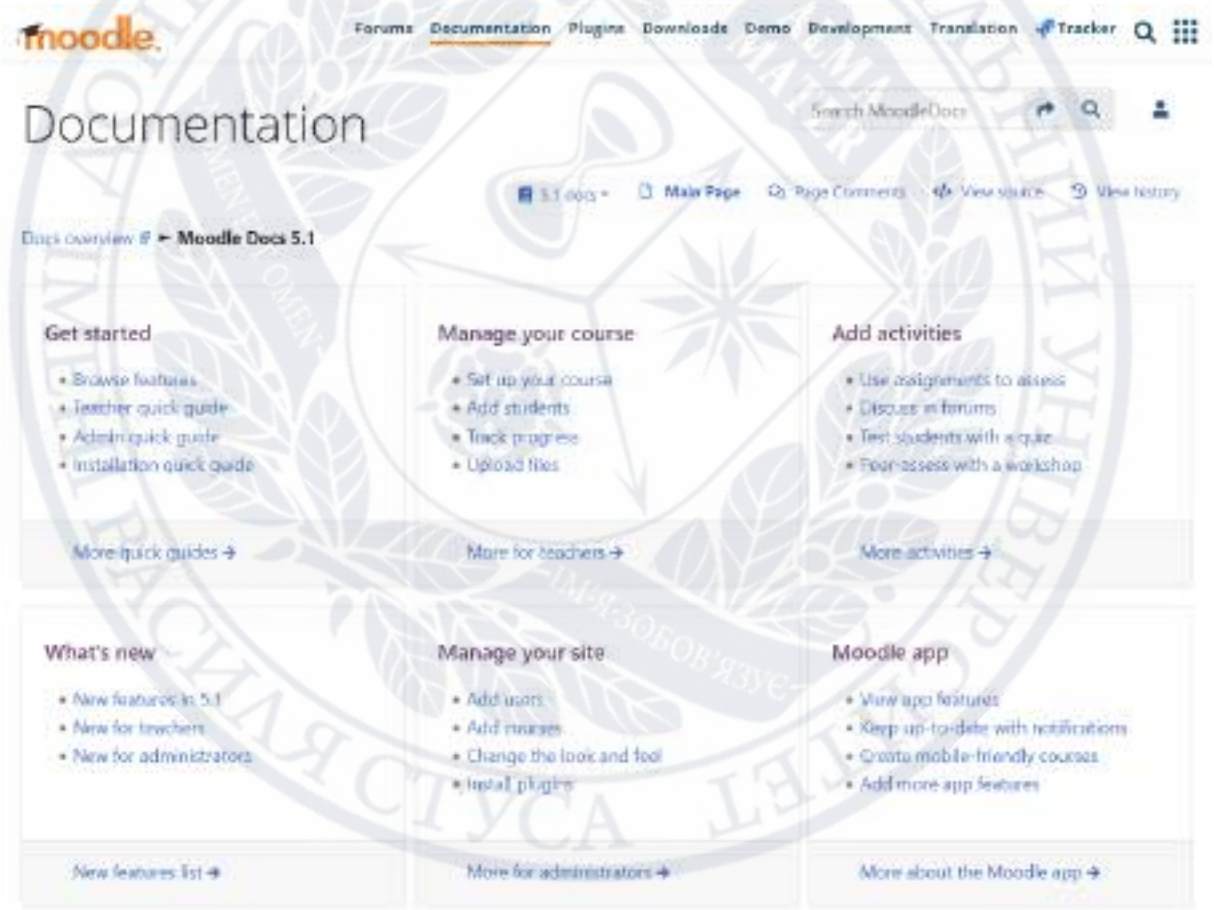

Рисунок 1.2 - Moodle [6]

Впровадження ERP-систем у вищій школі дозволяє інтегрувати навчальну, фінансову, кадрову, наукову та адміністративну діяльність у єдину інформаційну екосистему. Це підвищує прозорість управління,

сприяє ефективному розподілу ресурсів і забезпечує контроль за якістю освітніх послуг. Додатково, CRM-системи (Customer Relationship Management) використовуються для управління взаємовідносинами з абітурієнтами, студентами, випускниками та партнерами [7]. Вони допомагають університетам формувати позитивний імідж, підтримувати комунікацію з цільовими аудиторіями та підвищувати рівень залученості стейкхолдерів. На рис.1.3 відображено платформу Google Classroom.

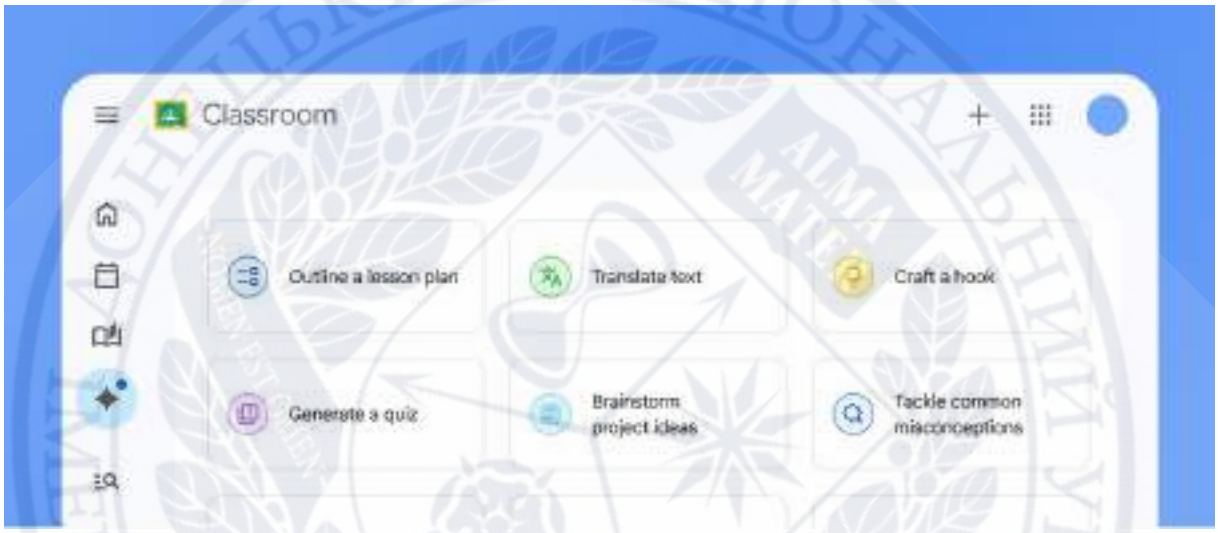

Рисунок 1.3 - Google Classroom [8]

Таким чином, інтеграція LMS, ERP та CRM у єдину цифрову платформу університету створює підґрунтя для формування «розумного університету» (Smart University), який базується на принципах цифрової трансформації, відкритих даних і аналітичного управління.

1.3 Необхідність створення цифрової платформи збору даних

В умовах глобальної цифрової трансформації суспільства питання ефективного управління освітньою системою набуває особливого значення. Для ухвалення обґрунтованих стратегічних рішень керівним органам освіти, аналітичним центрам та університетам необхідно мати оперативний доступ до достовірних, повних і структурованих даних. Проте на сьогодні в Україні

відсутній єдиний інструмент, який би забезпечував централізований збір, аналіз і прогнозування розвитку закладів вищої освіти (ЗВО). Така ситуація зумовлює потребу у створенні цифрової платформи збору даних для прогнозування розвитку університетів України, яка забезпечить інтеграцію інформаційних потоків, стандартизацію форматів даних і можливість побудови прогнозних моделей розвитку освітньої системи. Подібні платформи успішно функціонують у країнах Європейського Союзу та США, де вони є основою для аналітики вищої освіти, формування державної політики та оптимізації освітніх стратегій.

1.3.1 Проблеми традиційної аналітики у вищій освіті

Традиційна система аналітики у вищих навчальних закладах України базується на ручному зборі, обробці та аналізі даних. Основним джерелом інформації виступають локальні бази, Excel-таблиці, текстові звіти та форми, що передаються до Міністерства освіти і науки України або внутрішніх підрозділів університетів. Такий підхід має низку системних недоліків, які істотно обмежують ефективність управлінських процесів. У більшості університетів дані про студентів, викладачів, фінансові показники, наукові публікації та матеріально-технічні ресурси зберігаються у різних інформаційних системах, які не мають між собою зв’язку.

Через різноманітність форматів і методів збереження інформації (CSV, XML, JSON, Excel, локальні БД) університети не можуть обмінюватися інформацією ефективно. Відсутність узгоджених метаданих і форматів спричиняє помилки при агрегації інформації на національному рівні. Аналітичні звіти часто формуються вручну, що потребує часу та людських ресурсів. Навіть незначні зміни у показниках або структурах даних призводять до необхідності повторного збору і перевірки всієї інформації. Затримки у зборі інформації, помилки введення та дублювання записів призводять до втрати достовірності. В результаті управлінські рішення ухвалюються на основі застарілих або неточних відомостей.

Традиційні інструменти (таблиці, локальні звіти) не дозволяють будувати складні аналітичні моделі, виявляти тенденції або прогнозувати результати. Відсутність інтегрованих систем Business Intelligence позбавляє університети можливості проводити сценарне планування розвитку.

Процеси збору, перевірки та узгодження даних здебільшого виконуються вручну, що робить систему вразливою до людського фактору та унеможливлює оперативне оновлення інформації. Внаслідок цього університети не мають повноцінного аналітичного інструментарію для стратегічного управління [9]. Державні органи, у свою чергу, не можуть своєчасно оцінювати ефективність освітньої політики, прогнозувати кадрові потреби чи економічну доцільність фінансування окремих спеціальностей. Таким чином, традиційна модель збору та обробки даних є неефективною в умовах сучасного інформаційного суспільства і потребує глибокої цифрової модернізації.

1.3.2 Прогнозна аналітика у стратегічному розвитку університетів

Одним із ключових напрямів розвитку цифрової платформи збору даних є впровадження прогнозної аналітики (predictive analytics), що дозволяє не лише описувати поточний стан системи, а й передбачати її майбутній розвиток. Завдяки застосуванню статистичних моделей, алгоритмів машинного навчання та методів штучного інтелекту можна створювати науково обґрунтовані прогнози у сфері вищої освіти.

Моделі машинного навчання можуть враховувати демографічні показники, результати ЗНО/НМТ, тенденції на ринку праці та вподобання абітурієнтів для прогнозування кількості вступників за роками. Аналітика успішності студентів, працевлаштування випускників та попиту на спеціальності дозволяє формувати рекомендації щодо оновлення навчальних програм та акредитації спеціальностей. Збір і аналіз даних про викладання, результати студентів, наукову активність і публікації дає змогу формувати рейтинги викладачів і прогнозувати потреби у кадрах. Завдяки

комплексному аналізу доходів, видатків, грантової діяльності та контингенту студентів можна передбачати ризики фінансової нестабільності та пропонувати варіанти оптимізації бюджету.

Прогнозні моделі допомагають виявляти ризики відтоку студентів, низької якості навчання або неефективного використання ресурсів на ранніх етапах, що дозволяє вчасно реагувати. Платформа може надавати інструменти для сценарного аналізу - оцінки наслідків змін у фінансуванні, демографічній ситуації, нормативній базі чи політиці університету. Стратегічне значення прогнозної аналітики полягає у переході від реактивного до проактивного управління вищою освітою. Замість того щоб реагувати на проблеми постфактум, університети й державні структури можуть завчасно ідентифікувати потенційні виклики та формувати превентивні рішення.

Впровадження таких механізмів вимагає:

- наявності достовірних, структурованих даних;
- високопродуктивної цифрової платформи збору та зберігання інформації;
- методологічної бази для побудови моделей прогнозування;
- підготовки фахівців з освітньої аналітики та data science.

Отже, прогнозна аналітика є не лише інструментом дослідження, а й ключовим компонентом цифрової трансформації вищої освіти. Вона здатна підвищити якість стратегічного планування, забезпечити прозорість освітньої політики та створити науково обґрунтовану основу для сталого розвитку університетів України.

1.4 Методи та моделі прогнозування

Прогнозування розвитку університетів є ключовим елементом цифрової платформи, оскільки дозволяє не лише описувати поточний стан закладів, а й передбачати майбутні тенденції у навчальному процесі, кадровій політиці, фінансуванні та науковій діяльності. Для цього

використовуються як класичні статистичні підходи, так і сучасні алгоритми машинного навчання, що дозволяють поєднати наукову обґрунтованість із високою точністю прогнозів.

1.4.1 Класичні статистичні методи

Класичні статистичні методи прогнозування базуються на математичних моделях, що дозволяють оцінювати залежності між показниками та прогнозувати їхні зміни на основі історичних даних. Вони залишаються ефективним інструментом для університетських аналітичних задач завдяки прозорості та простоті інтерпретації результатів.

Регресійний аналіз є основним методом для оцінки залежностей між ключовими показниками університетів та факторними змінними [10]. Лінійна регресія дозволяє визначати вплив факторів, таких як демографічна ситуація, кількість викладачів чи фінансові ресурси, на кількість студентів або успішність випускників. Багатовимірна та поліноміальна регресія застосовуються у разі складніших нелінійних взаємозв’язків, дозволяючи точніше моделювати складні системи вищої освіти.

Аналіз часових рядів використовується для прогнозування показників, що змінюються у часі, таких як контингент студентів, фінансування чи наукова активність. Методи ARIMA (Autoregressive Integrated Moving Average), експоненційного згладжування та скользячого середнього дозволяють виявляти тренди, сезонні коливання та циклічні закономірності, забезпечуючи основу для коротко- та середньострокових прогнозів [11].

Кластерний аналіз допомагає сегментувати університети за схожими характеристиками: розміром, фінансовою стабільністю, рівнем наукової діяльності чи цифровою зрілістю. Такий підхід дозволяє будувати групові прогнози та адаптовані стратегії розвитку для різних кластерів закладів вищої освіти. Класичні статистичні методи мають переваги у вигляді простоти застосування та можливості інтерпретації результатів для

керівників університетів та державних органів. Водночас, вони обмежені у випадках великої кількості змінних або складних нелінійних взаємозв'язків, що потребує застосування сучасними методами машинного навчання.

1.4.2 Методи машинного навчання

Методи машинного навчання дозволяють прогнозувати розвиток університетів з урахуванням великої кількості факторів та складних залежностей, які важко формалізувати класичними статистичними методами[12]. Наприклад, PROPheT є адитивною моделлю часових рядів, розробленою компанією Facebook (Meta) для прогнозування даних, що характеризуються вираженими трендами та сезонними коливаннями [13]. Модель передбачає розкладання спостережуваного ряду на три основні компоненти: тренд, сезонність та ефекти святкових днів, що дозволяє отримати інтерпретовані прогнози навіть на основі неповних або нерівномірних даних. PROPheT відзначається здатністю автоматично визначати довгострокові тенденції та циклічні коливання, що робить її зручною для аналізу динаміки кількості студентів, попиту на спеціальності та інших ключових показників вищих навчальних закладів [13].

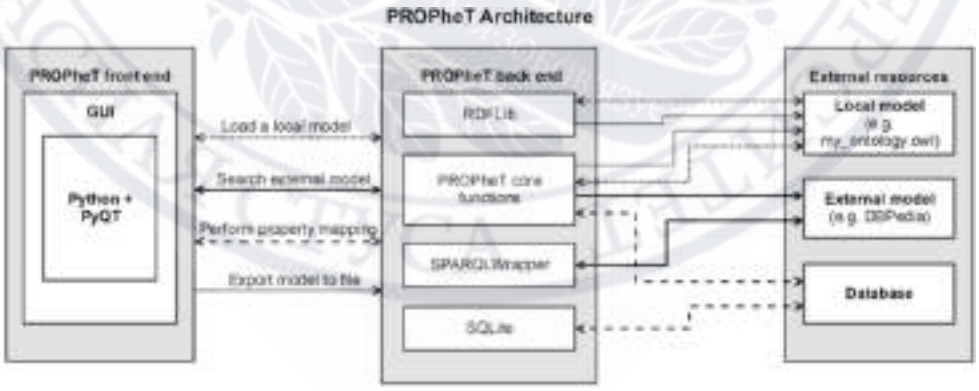

Рисунок 1.4 - Архітектура PROPheT [14]

Модель дозволяє враховувати додаткові фактори, такі як державні свята або зміни у вступній кампанії, та забезпечує наочне відокремлення впливу кожного компоненту на загальний прогноз. Завдяки своїй прозорості

та гнучкості PROPheT широко застосовується у практиці стратегічного планування, фінансового та кадрового менеджменту закладів вищої освіти. Незважаючи на це, її ефективність обмежується випадками, коли дані характеризуються складними нелінійними взаємозв'язками між багатьма зовнішніми змінними, що краще моделюється методами машинного навчання, такими як Random Forest або XGBoost.

Random Forest є ансамблевим методом машинного навчання, що базується на побудові множини дерев рішень і агрегації їх результатів для отримання точного прогнозу [15]. Кожне дерево в моделі навчається на випадковій підмножині даних із випадково обраними підмножинами ознак, що забезпечує зменшення переобучення та підвищує стійкість моделі до шуму в даних [15]. Такий підхід дозволяє ефективно моделювати складні нелінійні залежності між показниками діяльності університетів та зовнішніми факторами, що впливають на динаміку контингенту студентів, фінансування або попиту на спеціальності.

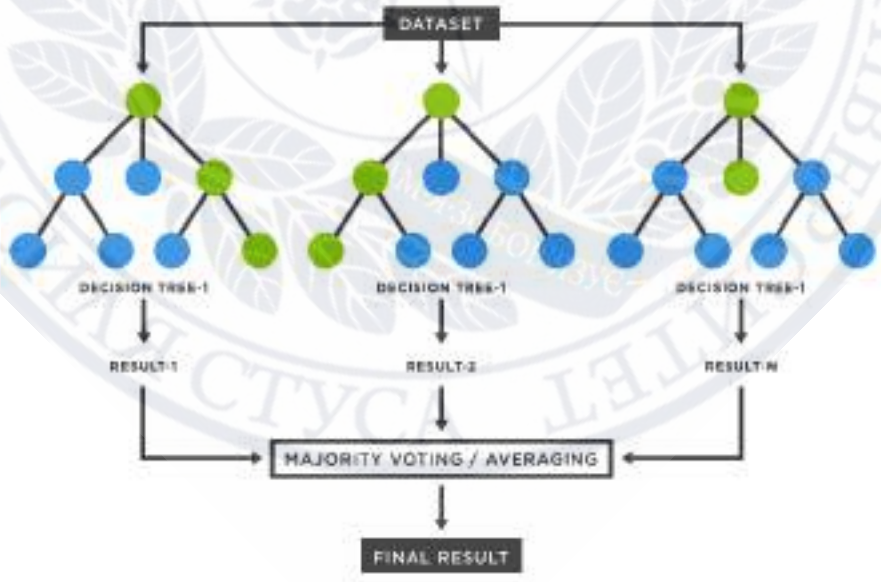

Рисунок 1.5 - Архітектура Random Forest [15]

Random Forest відзначається високою точністю прогнозів навіть при великій кількості змінних та наявності взаємопов'язаних факторів, що

робить його придатним для інтеграції даних із розрізнених джерел вищих навчальних закладів. Незважаючи на свою ефективність, Random Forest має обмеження у передбаченні часових рядів із вираженими сезонними або довгостроковими трендами, де більш придатними можуть бути рекурентні нейронні мережі або моделі типу PROPheT. У контексті роботи Random Forest забезпечує науково обґрунтовану підтримку прийняття рішень, дозволяючи прогнозувати ключові показники та моделювати сценарії розвитку системи.

В свою чергу XGBoost є сучасним алгоритмом градієнтного бустингу, який реалізує ефективне комбінування слабких моделей у сильну за рахунок послідовного навчання дерев рішень на залишках попередніх моделей [16]. Він вирізняється високою швидкістю обчислень, здатністю працювати з великими обсягами даних та забезпечує високу точність прогнозів навіть у складних нелінійних системах.

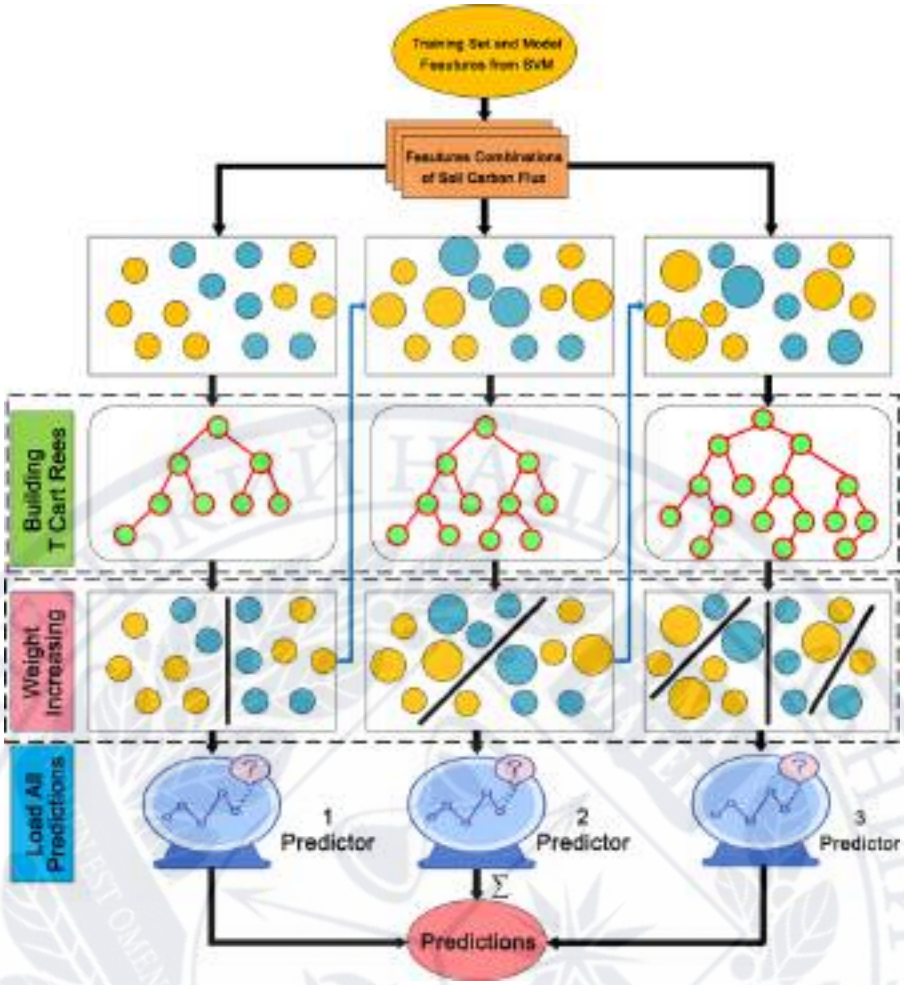

Рисунок 1.6 - Архітектура XGBoost [16]

XGBoost надає можливість регуляризації та контролю складності моделей, що зменшує ризик переобучення та підвищує стабільність прогнозів при роботі з розрізненими джерелами даних [17]. Завдяки гнучкості алгоритму він ефективний як для короткострокових, так і для довгострокових прогнозів, дозволяючи здійснювати аналіз сценаріїв розвитку та оцінювати вплив окремих факторів на стратегічні показники діяльності університетів.

LSTM (Long Short-Term Memory) є різновидом рекурентних нейронних мереж, спеціально розробленим для моделювання послідовностей та часових рядів із довготривалими залежностями [18]. На відміну від класичних нейронних мереж, LSTM здатні зберігати інформацію про попередні стани на тривалий період і вчасно «забувати» нерелевантні

дані завдяки механізму спеціальних гейтів, що контролюють потік інформації.

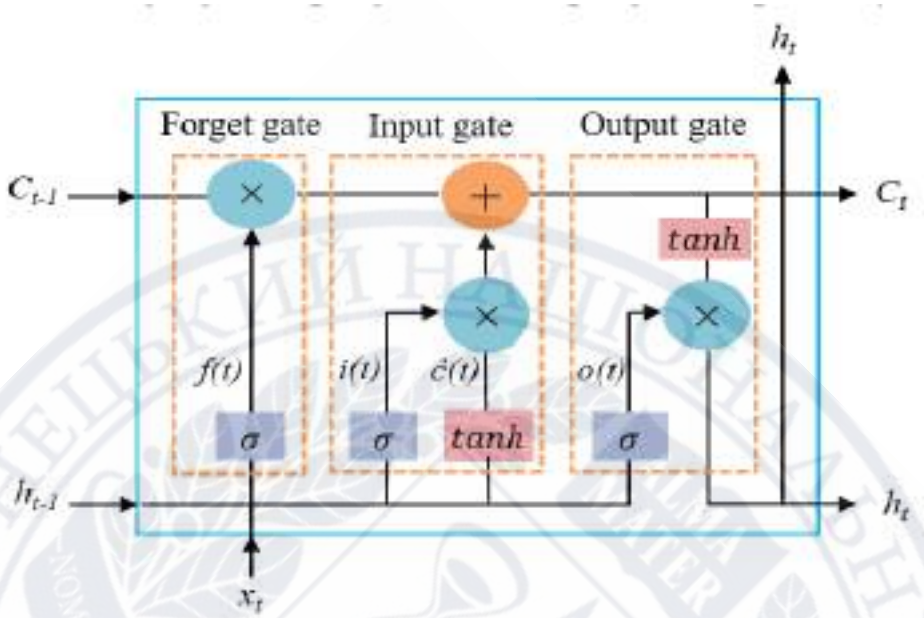

Рисунок 1.7 - Архітектура LSTM (Long Short-Term Memory) [19]

LSTM моделі дозволяють враховувати нелінійні взаємозв’язки між великою кількістю факторів, що традиційні статистичні методи врахувати не здатні [19]. Завдяки здатності адаптуватися до складних часових структур, LSTM забезпечують високу точність прогнозів у динамічних умовах освітньої системи та створюють можливості для сценарного планування та стратегічного управління університетами.

1.5 Висновки до розділу

У 1 розділі було проведено аналіз сучасного стану закладів вищої освіти України та встановлено, що їх розвиток характеризується низкою викликів: скороченням кількості студентів, недостатнім фінансуванням, відтоком науково-педагогічних кадрів і низьким рівнем цифрової зрілості більшості університетів. Разом із тим, в освітньому середовищі активно формуються передумови для цифрової трансформації. Відбувається впровадження систем електронного документообігу, LMS, CRM і ERP-рішень, що забезпечують автоматизацію управління навчальним процесом,

фінансами та ресурсами. Зростає роль аналітичних технологій - Business Intelligence, Big Data та Learning Analytics, які надають можливість об’єктивно оцінювати якість освітніх послуг, ефективність викладачів та результативність наукової діяльності.

Разом із тим, проведений аналіз показав наявність системних проблем у зборі, обробці та аналітиці освітніх даних. Інформація зберігається у розрізнених джерелах, не має уніфікованих форматів і часто формується вручну, що призводить до дублювання, втрати актуальності та обмежує можливості стратегічного планування. Відсутність централізованого сховища даних унеможливлює створення комплексних моделей розвитку ЗВО та своєчасне ухвалення управлінських рішень.

З огляду на це доведено доцільність створення цифрової платформи збору, обробки та аналітики даних про діяльність закладів вищої освіти України, яка повинна забезпечити:

- інтеграцію розрізнених інформаційних систем університетів і державних реєстрів, автоматизацію процесів збору, верифікації та оновлення інформації та впровадження прогнозної аналітики на основі штучного інтелекту та машинного навчання.

У межах дослідження було розглянуто моделі прогнозування, зокрема, проаналізовано класичні статистичні методи прогнозування (лінійна регресія, методи ковзного середнього, ARIMA) та сучасні інтелектуальні моделі, які було обрано для реалізації, а саме: Prophet, LSTM, XGBoost і Random Forest. Дані моделі забезпечують високу точність прогнозів та дозволяють аналізувати складні часові ряди й багатофакторні залежності у сфері вищої освіти.

РОЗДІЛ 2 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Вибір технологій для реалізації цифрової платформи

2.1.1 Огляд мови програмування Python

Python є однією з найпопулярніших мов програмування сучасності, яка широко використовується в наукових дослідженнях, аналітиці даних, веброзробці, автоматизації процесів, штучному інтелекті та машинному навчанні [20]. Її універсальність, простота синтаксису й потужна екосистема бібліотек зробили Python стандартом де-факто у багатьох сферах цифрової трансформації, включно з освітою та управлінням даними. Високий рівень абстракції дозволяє зосередитись на логіці алгоритмів, а не на технічних деталях реалізації, що особливо важливо для дослідницьких проєктів та систем збору й обробки даних. Для проєкту розроблення цифрової платформи збору даних для прогнозування розвитку університетів України, Python є оптимальним вибором завдяки своїй здатності швидко створювати прототипи, легко інтегруватися з базами даних і вебсервісами, а також забезпечувати гнучкість у реалізації аналітичних алгоритмів [20].

Python має одну з найпотужніших екосистем інструментів для обробки, аналізу та візуалізації великих обсягів інформації. Такі бібліотеки, як NumPy, Pandas, SciPy, Scikit-learn, TensorFlow і Matplotlib, формують основу для сучасних аналітичних систем [20]. Вони дозволяють ефективно працювати з багатовимірними масивами даних, виконувати статистичні операції, будувати моделі машинного навчання та створювати динамічні візуалізації.

Для задач прогнозування розвитку університетів Python надає інструменти для [21]:

- статистичного аналізу освітніх показників (наприклад, через Pandas і StatsModels);
- побудови моделей прогнозування на основі методів регресії,

кластеризації та нейронних мереж (Scikit-learn, TensorFlow, PyTorch);

- оцінки точності прогнозів та моделювання сценаріїв розвитку освітніх закладів;

Ключовою перевагою Python у контексті аналітики є можливість створення повного циклу обробки даних в єдиному середовищі - від збору до побудови прогнозу та подання результатів у зрозумілій формі. Це суттєво скорочує час розроблення та знижує складність системи [21]. Одним із критичних завдань цифрової платформи є забезпечення інтеграції з різними зовнішніми джерелами - державними реєстрами, внутрішніми інформаційними системами університетів, відкритими освітніми базами даних. Python має широкий набір засобів для роботи з API, серед яких найпопулярнішою є бібліотека requests, що забезпечує просту взаємодію з RESTful API, підтримку автентифікації, передачу параметрів, обробку JSON-відповідей і керування сесіями. Завдяки цьому Python може автоматично збирати дані з вебресурсів, синхронізувати їх з локальними базами та виконувати регулярні оновлення.

Бібліотека json, що входить до стандартного пакету Python, дозволяє безпосередньо працювати з даними у форматі JSON - основному форматі обміну інформацією між сервісами [22]. Це спрощує процес обробки освітніх даних, які надходять з різних джерел, забезпечуючи їх уніфікацію та конвертацію у внутрішній формат платформи.

Таким чином, Python надає єдину програмну основу для інтеграції, збору, обробки та аналізу інформації, що робить його незамінним інструментом для створення аналітичних рішень у сфері управління освітою.

2.1.2 Реалізація графічного інтерфейсу користувача

Графічний інтерфейс користувача є ключовим компонентом цифрової платформи, оскільки забезпечує зручний та інтуїтивно зрозумілий доступ до

функціоналу системи для адміністраторів, аналітиків та науковців. Для реалізації GUI в обраній платформі використовується мова програмування Python разом із стандартними бібліотеками Tkinter та ttk (Themed Tkinter), що дозволяє створювати стабільні, кросплатформенні та ефективні інтерфейси [23]. Tkinter є стандартним модулем Python для розробки графічних інтерфейсів. Основні переваги Tkinter полягають у [23]:

- доступності у стандартній бібліотеці Python без необхідності встановлення додаткових пакетів;
- кросплатформенності - програми працюють на Windows, Linux та macOS без модифікацій;
- простоті синтаксису та швидкості розробки базових елементів інтерфейсу;
- інтеграції з іншими модулями Python, що дозволяє органічно поєднувати GUI із аналітичними або мережевими компонентами платформи.

Tkinter забезпечує створення основних елементів інтерфейсу, таких як вікна, кнопки, поля введення, текстові віджети, меню та списки. Для побудови GUI платформи було використано структуру головного вікна, в яке інтегруються панелі навігації, області для відображення даних та панелі керування аналітичними процесами [23].

Бібліотека ttk (Themed Tkinter) є розширенням Tkinter, яке надає сучасні стилізовані віджети та можливість легко застосовувати теми оформлення (themes). Основні переваги ttk полягають у [23]:

- створенні сучасного, привабливого та інтуїтивно зрозумілого дизайну GUI;
- доступі до поліпшених віджетів (Treeview, Notebook, Combobox, Progressbar), які підтримують більш складні сценарії взаємодії з користувачем;
- можливості централізованого управління стилями елементів інтерфейсу.

У розробленій платформі Treeview використовується для відображення структурованих таблиць даних про університети (наприклад, кількість студентів, рейтинги, фінансові показники). Notebook дозволяє організувати вкладки для різних аналітичних модулів (збір даних, візуалізація, прогнозування), а Combobox - реалізувати фільтри та вибір категорій для динамічного відображення результатів.

Реалізація графічного інтерфейсу передбачає модульну архітектуру:

1. Головне вікно - контейнер для всіх елементів GUI, керує основною логікою взаємодії.
2. Панель навігації - меню та кнопки для перемикання між функціональними модулями платформи.
3. Область відображення даних - інтегрує таблиці, графіки та візуалізації, створені на основі бібліотек аналітики (Pandas, Matplotlib, Plotly).
4. Панель керування - елементи введення параметрів для запуску аналітичних моделей, оновлення даних та експорту результатів.

Таке розділення дозволяє забезпечити гнучкість GUI, спрощує тестування та подальше масштабування платформи.

Реалізація графічного інтерфейсу користувача на основі Tkinter та ttk дозволяє створити зручну, гнучку та інтуїтивно зрозумілу платформу для збору, обробки та прогнозної аналітики даних університетів України. Вона забезпечує легкий доступ до ключових функцій системи, інтегрується з аналітичними модулями на Python та дозволяє ефективно представляти результати у вигляді таблиць, графіків та дашбордів. Використання стандартних бібліотек Python робить інтерфейс надійним, кросплатформеним та зручним для подальшого розвитку цифрової платформи.

2.1.3 Архітектура програмного модуля

Програмний модуль цифрової платформи збору та аналізу даних побудований за принципом модульності, що дозволяє забезпечити гнучкість, масштабованість та простоту підтримки системи. Основні функціональні блоки включають InfoTabDefs, AnalizTabDefs та GetUniversityData, кожен з яких відповідає за окремий етап обробки інформації. InfoTabDefs відповідає за відображення та первинне сортування даних про університети. Цей компонент забезпечує інтерактивне представлення структурованої інформації у вигляді таблиць, дозволяючи користувачам швидко ознайомитися з ключовими показниками закладів. Водночас він формує основу для подальшої аналітики, надаючи доступ до даних у стандартизованому форматі.

AnalizTabDefs реалізує функції аналітичної обробки та візуалізації даних. У цьому модулі інтегруються аналітичні алгоритми, що виконують статистичні обчислення, побудову графіків, дашбордів та прогнозних моделей. Система дозволяє не лише оцінювати поточний стан університетів, а й здійснювати прогнозні оцінки розвитку освітніх програм, контингенту студентів та ресурсної бази. GetUniversityData відповідає за збір інформації з різних джерел, включаючи локальні бази даних, API державних реєстрів та відкриті освітні платформи. Модуль забезпечує уніфікацію даних, їх попередню обробку та передачу у відповідні компоненти для візуалізації та аналітики.

Взаємодія між компонентами відбувається у реальному часі. Дані, отримані через GetUniversityData, передаються в InfoTabDefs для первинного відображення і одночасно надходять до AnalizTabDefs для проведення аналітики та побудови прогнозних моделей. Така архітектура дозволяє забезпечити ефективний потік інформації, мінімізувати дублювання обчислень та підтримувати високий рівень узгодженості даних.

Модульна структура також створює умови для подальшого розширення системи. Додаткові аналітичні функції, нові джерела даних або

покращені методи візуалізації можна інтегрувати без кардинальної перебудови існуючих компонентів. Це забезпечує стійкість системи до зростання обсягів інформації та змін у вимогах користувачів, що є критично важливим для платформи, орієнтованої на стратегічне прогнозування розвитку університетів України.

2.2 Технології збору, аналізу та інтеграції освітніх даних

Для ефективного збору, обробки та аналізу даних у сфері освіти застосовуються сучасні технології інтеграції з відкритими джерелами та локальної попередньої обробки. Основним джерелом інформації є відкритий реєстр ЄДЕБО, який містить детальні дані про заклади освіти України, їхні спеціальності, студентів та випускників. Використання API цього реєстру дозволяє отримувати актуальні дані у структурованому форматі, що забезпечує достовірність та оперативність інформації.

2.2.1 Інтеграція з API зовнішніх джерел

Збір даних відбувається через спеціалізовані функції, які взаємодіють із API ЄДЕБО та World Bank Open Data. Серед даних ЄДЕБО:

- specialnosti - отримує інформацію про ліцензовані спеціальності, освітній рівень, регіон та конкретний заклад освіти;
- zakladi_osviti - повертає перелік закладів освіти певної категорії та регіону;
- university_info - надає детальну інформацію про конкретний заклад, включно зі структурою факультетів та спеціальностей;
- university_educators та university_prof_educators - формують дані про студентів та здобувачів професійно-технічної освіти за різними параметрами (дата, рівень освіти, код спеціальності, основа вступу);
- university_entrant та university_graduate - отримують статистику щодо вступників та випускників за рік, спеціальністю та формою навчання.

Параметри запитів включають освітній ступінь, рік вступу або випуску, регіональний код, код спеціальності, основу вступу та формат експорту (JSON, XML, XLSX). Такий підхід дозволяє динамічно формувати запити відповідно до аналітичних потреб та отримувати дані у машинозчитуваному форматі для подальшої обробки.

Дані використані з Word Bank [24] представлені в табл. 2.1.

Таблиця 2.1 - Використані дані з World Bank Open Data

Розшифрування індикатора	RESR API endpoint path (base endpoint: https://api.worldbank.org/v2/ country/UKR/indicator/)
Загальна чисельність населення	/SP.POP.TOTL?format=json
Приріст населення, % (річний)	/SP.POP.GROW?format=json
Частка населення віком 15–24 років (% від загальної чисельності)	/SP.POP.1524.TO.ZS?format=json
Очікувана тривалість життя при народженні	/SP.DYN.LE00.IN?format=json
Державні витрати на освіту (% від ВВП)	/SE.XPD.TOTL.GD.ZS?format=json
ВВП на душу населення (USD)	/NY.GDP.MKTP.CD?format=json
Рівень інфляції, CPI (% річний)	/FP.CPI.TOTL.ZG?format=json
Рівень безробіття (% робочої сили)	/SL.UEM.TOTL.ZS?format=json
Рівень безробіття 15–24 років (%)	/SL.UEM.1524.ZS?format=json

2.2.2 Локальна попередня обробка даних

Для підвищення надійності та доступності інформації всі отримані дані автоматично зберігаються у локальній папці saved_data. Збереження у форматі JSON забезпечує швидкий доступ до даних без необхідності повторних запитів до API, що особливо важливо при роботі з великими обсягами інформації або обмеженим інтернет-з'єднанням. Механізм роботи включає наступні етапи:

1. Збереження та структуризація - отримані дані автоматично записуються у файли із назвою, що містить параметри запиту (рік, спеціальність, освітній рівень, регіон).
2. Fallback до офлайн-файлів - у разі недоступності API система намагається знайти локальний файл із відповідними даними, що дозволяє проводити аналіз навіть у відсутності з'єднання.
3. Валідація та попереднє очищення - дані перевіряються на наявність ключових полів та правильність структури JSON.
4. Підготовка до інтеграції - локально збережені дані можна об'єднувати та обробляти для побудови статистичних та графічних аналітичних звітів (наприклад, розподіл студентів за спеціальностями, вступ/випуск, форми навчання).

Такий підхід дозволяє ефективно інтегрувати дані з різних джерел, уникати втрат інформації та забезпечує гнучкість у проведенні аналітичних досліджень у сфері освіти.

2.3 Висновки до розділу

В даному розділі було досліджено технології обрані для розробки цифрової платформи. Аналіз показав, що мова програмування Python є оптимальним інструментом для розробки цифрової платформи збору, обробки та прогнозної аналітики даних університетів України. Її переваги включають простоту синтаксису, потужну екосистему бібліотек для роботи

з даними та можливість інтеграції з вебсервісами й базами даних. Використання Python забезпечує швидке створення прототипів, гнучкість у реалізації аналітичних алгоритмів та повний цикл обробки інформації в єдиному середовищі. Модульна архітектура інтерфейсу забезпечує зручний доступ до функціоналу платформи та інтеграцію з аналітичними модулями. Візуалізація даних через таблиці, графіки та дашборди підвищує наочність результатів та полегшує процес прийняття рішень.

Модульна структура цифрової платформи забезпечує ефективний потік інформації від збору до аналітики та прогнозування. Така архітектура дозволяє легко масштабувати систему, інтегрувати нові джерела даних і аналітичні модулі без кардинальної перебудови, що є критично важливим для платформи стратегічного прогнозування розвитку університетів. Використання API відкритих джерел, зокрема ЄДЕБО, дозволяє оперативно отримувати достовірні структуровані дані про заклади освіти, студентів та випускників. Локальна попередня обробка даних у форматі JSON підвищує надійність та доступність інформації, забезпечує можливість офлайн-аналізу та гнучко інтегрує дані з різних джерел для подальшого статистичного та графічного представлення.

РОЗДІЛ 3 ЕТАПИ РОЗРОБКИ ЦИФРОВОЇ ПЛАТФОРМИ ЗБОРУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ РОЗВИТКУ УНІВЕРСИТЕТІВ

3.1 Аналіз вимог до системи

Аналіз вимог до системи передбачає визначення ключових характеристик, необхідних для ефективного збору, обробки та аналізу освітніх даних. Система орієнтована на забезпечення аналітичних потреб Міністерства освіти і науки, університетів та дослідницьких центрів, з можливістю прогнозування чисельності студентів та візуалізації тенденцій.

3.1.1 Цільові користувачі

Система створюється для трьох основних категорій користувачів, кожна з яких має специфічні потреби у доступі до освітньої інформації та аналітичних інструментів. Міністерство освіти і науки потребує моніторингу та стратегічного планування. Університети використовують систему для управління власними студентськими показниками та оптимізації освітніх програм. У табл.3.1 зведено інформацію про цільових користувачів системи.

Таблиця 3.1 - Цільові користувачі системи

Цільова категорія	Основні потреби	Приклади використання
Міністерство освіти і науки	Моніторинг статистики, стратегічне планування, аналіз набору та випуску студентів	Аналіз популярності спеціальностей, прогноз чисельності студентів
Університети	Контроль власних показників, управління набором, оцінка форм навчання	Оптимізація освітніх програм, аналіз денних та заочних студентів
Аналітичні центри	Наукові дослідження, підготовка звітів, моделювання тенденцій	Прогнозування набору, побудова статистичних моделей

3.1.2 Функціональні та нефункціональні вимоги

Функціональні вимоги визначають можливості системи щодо збору, обробки та візуалізації даних, включно з прогнозуванням. Нефункціональні вимоги описують властивості системи, що забезпечують її стабільність, продуктивність та безпеку. У таблиці 3.2 відображено функціональні вимоги.

Таблиця 3.2 - Функціональні вимоги

Опис	Приклади реалізації	Пріоритет
Забезпечення збору, обробки, аналізу та візуалізації даних	Автоматичне отримання даних з API ЄДЕБО та World Bank Open Data; підтримка JSON, XML, XLSX; збереження даних локально; обробка даних про вступників, випускників, студентів; фільтрація за роками, спеціальностями, регіонами; побудова графіків, діаграм та прогнозних моделей	Високий
Прогнозування освітніх показників	Використання моделей ARIMA, Random Forest, Gradient Boosting для прогнозу набору студентів та випуску	Високий
Інтеграція з локальними та зовнішніми джерелами	Використання локальних JSON-файлів для резервного доступу; інтеграція з додатковими відкритими даними	Середній
Підтримка аналітичних запитів	Генерація звітів за спеціальностями, формами навчання, регіонами; порівняння університетів та факультетів	Високий
Користувацький інтерфейс для візуалізації	Відображення графіків, діаграм та прогнозів у веб-інтерфейсі	Середній

У таблиці 3.3 відображено нефункціональні вимоги.

Таблиця 3.3 - Нефункціональні вимоги

Опис	Приклади реалізації	Пріоритет
Продуктивність	Обробка великих масивів даних за прийнятний час; підтримка паралельних потоків	Високий
Надійність	Система повинна коректно працювати при відсутності підключення до Інтернету (використання локальних файлів); мінімізація помилок при запитах API	Високий
Масштабованість	Можливість додавання нових джерел даних, нових типів аналітики та прогнозних моделей	Середній
Безпека даних	Контроль доступу, захист локальних файлів та конфіденційних даних	Високий
Зручність використання	Інтуїтивно зрозумілий інтерфейс, можливість фільтрації та вибору параметрів запитів	Середній
Портативність	Сумісність з різними ОС та можливість запуску на локальному ПК або сервері	Низький
Підтримка формату даних	Робота з JSON, XML, XLSX для сумісності з зовнішніми системами та звітністю	Середній

3.1.3 Джерела даних для аналізу

Джерела даних визначають, звідки система отримує інформацію для аналізу та прогнозування. Основними джерелами є офіційні API ЄДЕБО, World Bank Open Data. API ЄДЕБО забезпечує доступ до офіційних даних про заклади освіти, що дозволяє автоматизувати процес збору освітньої інформації. World Bank Open Data надає міжнародні соціально-економічні показники, які використовуються для аналізу та побудови прогнозних моделей розвитку освіти.

Таблиця 3.4 - Джерела даних для аналізу

Джерело даних	Формат	Опис використання
API ЄДЕБО	JSON, XML, XLSX	Автоматичний збір даних про заклади освіти, студентів, вступників та випускників
Локальні файли	JSON	Збереження резервних копій запитів для офлайн-аналізу
World Bank Open Data	JSON	Міжнародні соціально-економічні показники для порівняльного та прогнозного аналізу
Додаткові відкриті дані	CSV, XLSX, JSON	Статистична та демографічна інформація, що інтегрується для комплексного аналізу з розрізнених ресурсів

3.2 Проєктування архітектури платформи

UML (Unified Modeling Language - уніфікована мова моделювання) - це стандартна графічна мова, яка використовується для візуалізації, специфікації, конструювання та документування програмних систем. UML допомагає розробникам зрозуміти структуру та поведінку системи, а також полегшує комунікацію між членами команди під час проєктування програмного забезпечення.

3.2.1 Діаграма використання

Діаграма використання - це один із базових типів поведінкових UML-діаграм, який показує взаємодію користувачів (акторів) із системою через певні сценарії використання (випадки використання) [25]. Вона дозволяє описати що робить система, не заглиблюючись у те, як саме це реалізовано, не навантажуючи користувача надмірними даними. Такі діаграми допомагають формалізувати основні функції системи та визначити її межі взаємодії з користувачами. На рис.3.1 відображено діаграму використання.

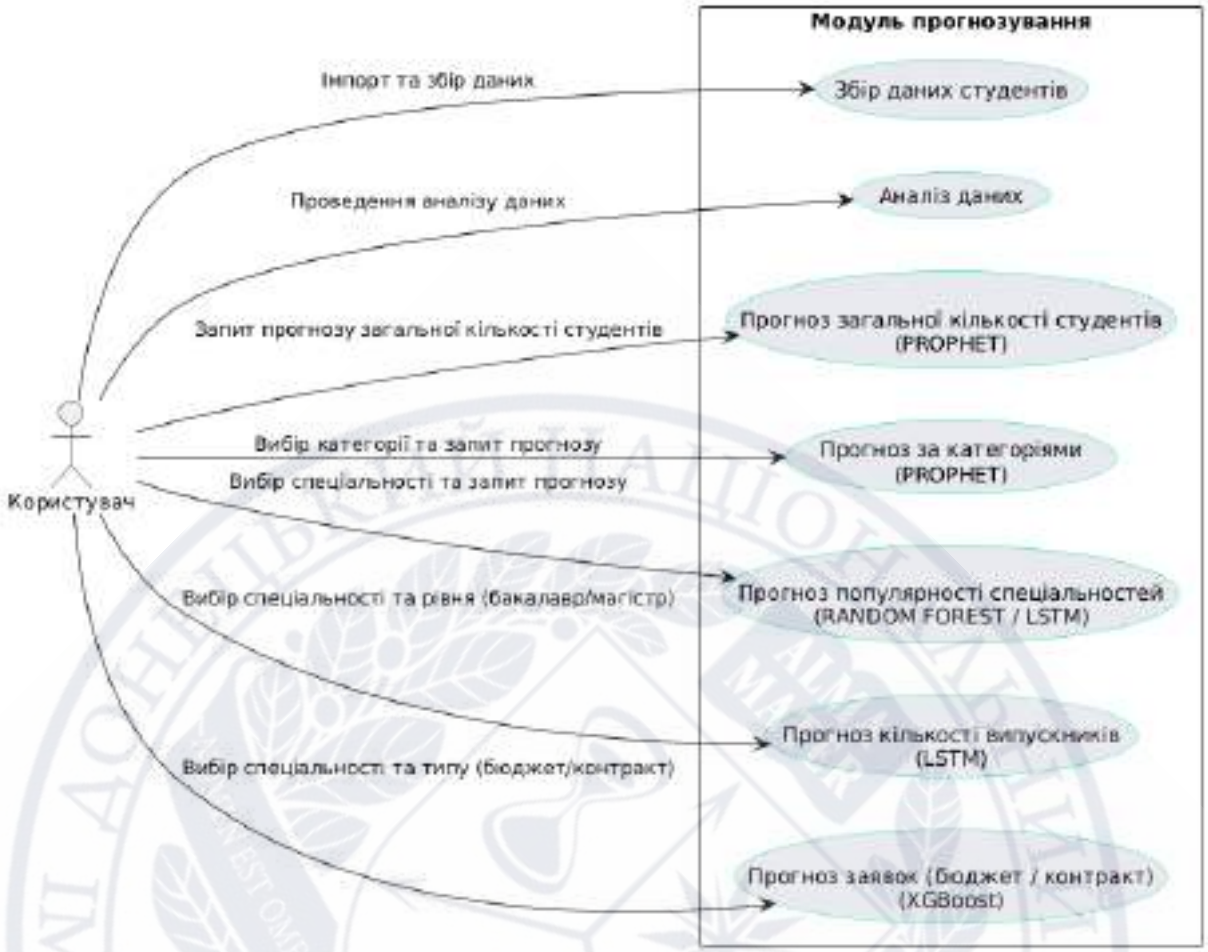

Рисунок 3.1 - Діаграма використання

Користувач може збирати студентські дані, аналізувати їх, будувати прогнози загальної кількості студентів та прогнозувати за окремими категоріями за допомогою модуля PROPheT. Для оцінки популярності спеціальностей доступні методи Random Forest та LSTM, що дозволяють враховувати історичні тенденції та складні залежності між даними. Аналогічно, для оцінки кількості випускників застосовується LSTM, який моделює довгострокові тренди з урахуванням історичних даних за спеціальностями та рівнями освіти (бакалавр, магістр).

Система також підтримує прогнозування заявок на бюджет чи контракт, де аналітик обирає спеціальність і тип навчання. Для цього використовується XGBoost, який дозволяє врахувати складні взаємозв’язки між показниками попередніх років та побудувати точний прогноз для прийому студентів. Результати прогнозів та аналізів відображаються у

вигляді графіків та дашбордів, що дозволяє користувачу оцінювати тенденції та приймати управлінські рішення. Додатково передбачено модуль допомоги, який надає інструкції та пояснення щодо роботи платформи. Взаємозв'язки між функціональними блоками показують, що аналіз даних базується на зібраній інформації, а прогнози автоматично інтегруються з візуалізацією для наочного представлення результатів. Таким чином, вся взаємодія користувача централізована та логічно структурована, а додаток забезпечує комплексну підтримку у зборі, аналізі та прогнозуванні даних університетів.

3.2.2 Діаграма послідовності

Діаграма послідовності - це один із основних типів поведінкових UML-діаграм, який використовується для моделювання взаємодії між об'єктами у часі [25]. Вона показує, яким чином об'єкти обмінюються повідомленнями, щоб виконати певну функцію або сценарій системи. Вона дозволяє візуально оцінити послідовність операцій і встановити, як саме компоненти координують свою роботу для отримання кінцевого результату. На рис. 3.2 відображено діаграму послідовності.

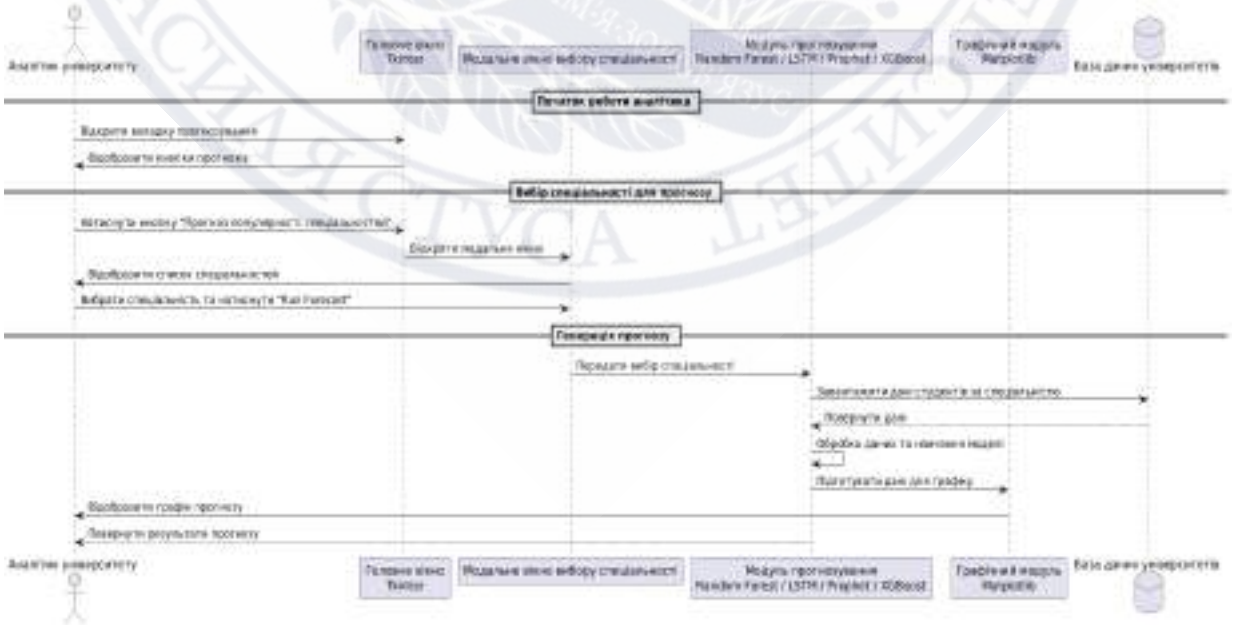

Рисунок 3.2 - Діаграма послідовності

Діаграма ілюструє взаємодію користувача з додатком від моменту запуску до отримання прогнозів. Спершу користувач відкриває додаток і через графічний інтерфейс ініціює завантаження даних з папки saved_data. Модуль завантаження даних обробляє всі JSON-файли, формує DataFrame та повертає його GUI.

Після цього користувач може обрати категорію для прогнозування або побудувати загальний прогноз. Коли він обирає категорію, GUI передає DataFrame і вибрану категорію модулю PROPheT. PROPheT підготовлює дані, групує їх за датами, навчає модель і генерує прогноз на кілька наступних років. Після цього прогнозні дані повертаються до GUI.

GUI передає ці дані до модуля візуалізації (Matplotlib), який створює графік і відображає його користувачу. Аналогічний процес відбувається для інших типів прогнозів Random Forest, LSTM або XGBoost, дані агрегуються, модель навчається на даних, генерується прогноз, і результат візуалізується на графіку. Такий підхід забезпечує узгоджену роботу всіх модулів та формує прозорий ланцюг обробки даних від вводу до візуалізації результатів.

3.2.3 Діаграма компонентів

Діаграма компонентів - це різновид структурної UML-діаграми, яка відображає фізичну структуру програмної системи, тобто те, з яких компонентів вона складається та як ці компоненти взаємодіють між собою [26]. Компонент у UML - це логічна частина системи, яка інкапсулює певний функціонал і може бути незалежно розроблена, протестована та повторно використана. Компоненти можуть представляти модулі, бібліотеки, сервіси, бази даних або інші частини програмного забезпечення. Використання діаграми компонентів дозволяє зрозуміти архітектуру системи на високому рівні та чітко визначити відповідальність кожного модуля. На рис.3.3 відображено діаграму компонентів.

Рисунок 3.3 - Діаграма компонентів

У верхній частині діаграми розташований актор “Користувач”, який взаємодіє з графічним інтерфейсом користувача (GUI). GUI розбито на вкладки: Data Collection, Data Analysis, Analytical Forecast та Help, кожна з яких відповідає за певний функціонал - введення та перегляд даних, аналітику, побудову прогнозів та отримання довідкової інформації.

GUI зв’язаний із шаром Data Layer, який включає компоненти JSON Loader та Data Preprocessing. JSON Loader відповідає за завантаження даних із збережених файлів, а Data Preprocessing - за обробку і підготовку даних для подальшого аналізу та прогнозування.

Оброблені дані передаються у Forecasting Module, де знаходяться модулі прогнозування: PROPheT Model та додаткові ML Models (XGBoost, LSTM, Random Forest). Результати прогнозів передаються у Visualization, де компонент Matplotlib Charts відповідає за побудову графіків. GUI взаємодіє з Visualization, відображаючи графіки прогнозів і аналітики користувачу у вкладці Analytical Forecast або Data Analysis. Таким чином, діаграма чітко показує, як дані проходять через систему від завантаження та обробки до прогнозування і візуалізації, а користувач отримує інтерактивний доступ до всіх функцій додатку.

3.3 Реалізація основних модулів

3.3.1 Модуль збору даних із відкритих джерел та попередньої обробки даних

Метод geographical_distribution_of_students() аналізує, як розподіляються студенти по регіонах України за певною спеціальністю, рівнем освіти та базою вступу.

Параметри (вводяться користувачем):

- дата (наприклад, 01.01.2024)
- освітній рівень (бакалавр, магістр тощо)
- основа вступу (повна середня освіта, фаховий молодший бакалавр тощо)
- код спеціальності (наприклад, 122)

Перетворює введені значення на коди (словники qf_map і eb_map). Викликає university_educators() для отримання статистики. Підраховує кількість студентів-бюджетників і контрактників у кожному регіоні. Будує горизонтальну гістограму порівняння бюджет/контракт. Як результат отримуємо графік розподілу студентів за регіонами + збереження у saved_diagrams/students_per_region.png.

Метод distribution_of_institutions_by_region() показує, скільки закладів освіти в кожному регіоні пропонують певну спеціальність.

Параметри:

- освітній рівень
- код спеціальності

Використовує функцію specialnosti() для отримання списку закладів. Підраховує кількість закладів у кожному регіоні. Будує горизонтальний стовпчиковий графік. Як результат отримуємо діаграму у відтінках зеленого, файл specialty_distribution.png.

Метод time_based_speciality_popularity() досліджує популярність спеціальності у часі (кількість студентів по роках).

Параметри:

- початкова та кінцева дати
- рівень освіти
- основа вступу
- код спеціальності

В методі є цикл по роках між датами. Для кожної дати викликає university_educators(). Підраховує загальну кількість студентів. Будує лінійний графік залежності кількості студентів від часу. Отримуємо динаміку кількості студентів за роками (PNG-файл popularity_over_time.png).

Функція entrant_graduate_trends() порівнює кількість вступників і випускників за вибрані роки, отримуючи дані через university_entrant() та university_graduate(), і відображає їх у круговій діаграмі з відсотками, зберігаючи результат у файлі entrant_vs_graduate.png.

Метод analysis_of_learning_forms() показує розподіл студентів за формами навчання (денна, заочна, вечірня, бюджет/контракт) за певну дату та спеціальність, використовуючи university_educators(), і будує кругову діаграму, що зберігається як learning_form_distribution.png.

Метод rozpodil_facultetiv() аналізує кількість факультетів у університетах України, отримуючи дані через zakladi_osviti() та university_info() у потоках, і будує стовпчикову діаграму faculty_distribution.png.

Метод university_profile_comparison() дозволяє порівняти два університети за спеціальностями або факультетами, використовуючи university_info(), і для спеціальностей будує подвійний стовпчиковий графік university_profile_comparison.png, а для факультетів показує спільні та унікальні факультети у вікні.

Метод popularity_of_specialties() візуалізує топ-10 найпопулярніших спеціальностей України за статичними даними, будуючи стовпчиковий графік Most Popular Specialties in 2023.

Метод number_of_sprecialties_per_university() показує розподіл кількості спеціальностей по університетах, отримуючи дані через zakladi_osviti() та university_info() у потоках, і будує діаграму specialty_distribution.png.

Методи модуля GetUniversityData.py надають інтерфейс для отримання даних із відкритого реєстру закладів освіти України (ЄДЕБО) через API. Вони дозволяють завантажувати інформацію про спеціальності, заклади освіти, студентів та випускників, а також зберігати отримані дані локально для офлайн-доступу. У табл.3.5 зведено опис методі модуля GetUniversityData.py.

Таблиця 3.5 - Методи модуля GetUniversityData.py

Метод	Призначення	Основні параметри	Результат
specialnosti	Отримує інформацію про спеціальності закладу	qf - освітній ступінь, sp - код спеціальності, rg - код регіону, id - код закладу, exp - формат	Дані про спеціальності у JSON та локальний файл
zakladi_osviti	Повертає список закладів освіти	ut - категорія закладу, lc - код регіону, exp - формат	Дані про заклади освіти у JSON та файл
university_info	Детальна інформація про ВНЗ	id - код закладу, exp - формат	Дані про заклад у JSON та файл
Метод	Призначення	Основні параметри	Результат
university_educators	Інформація про студентів університетів	dt - дата, qf - ступінь, eb - основа вступу, sp - код спеціальності, rg - код регіону, id - код закладу, exp - формат	Дані про студентів у JSON та файл

university_prof_educators	Дані про студентів ПТНЗ	dt - дата, eb - основа вступу, pr - ідентифікатор професії, rg - код регіону, id - код закладу, exp - формат	Дані про студентів ПТНЗ у JSON та файл

Продовження таблиці 3.5

university_entrant	Інформація про осіб, зарахованих на навчання	y - рік вступу, qf - ступінь, eb - основа вступу, sp - код спеціальності, rg - код регіону, id - код закладу, exp - формат	Дані про вступників у JSON та файл
university_graduate	Інформація про випускників	y - рік випуску, qf - ступінь, eb - основа вступу, sp - код спеціальності, rg - код регіону, id - код закладу, exp - формат	Дані про випускників у JSON та файл

3.3.2 Модуль прогнозування PROPheT

Модуль прогнозування в додатку використовує бібліотеку PROPheT для передбачення кількості студентів за різними категоріями. PROPheT спеціально створений для часових рядів і дозволяє моделювати тренди та сезонні коливання без складного налаштування. Основні завдання модуля:

1. Об’єднати та підготувати дані з різних категорій студентів (full_time_budget, part_time_contract і т.д.) в формат, який сприймає PROPheT.
2. Виконати агрегування даних по датах (date) для формування часових рядів.
3. Навчити модель PROPheT на історичних даних.
4. Зробити прогноз на майбутні періоди (роки).
5. Візуалізувати результат: історичні дані та прогноз на графіку.

Функція forecast_PROPheT_category(df, category='total') прогнозує кількість студентів по обраній категорії (total, full_time, part_time, evening) на наступні 5 років.

Параметри:

- df - DataFrame з даними про студентів.

Має колонки: date, full_time_budget, full_time_contract, part_time_budget, part_time_contract, evening_budget, evening_contract, region

- category - обрана категорія для прогнозу:
  - 'total' - всі студенти разом
  - 'full_time' - студенти на повній формі навчання
  - 'part_time' - студенти на заочній формі
  - 'evening' - вечірня форма навчання

Для уникнення змін оригінального DataFrame застосовується метод copy(), створюючи його повну копію. Далі здійснюється додавання сум за різними категоріями. Зокрема, створюються нові колонки total, full_time, part_time та evening, які представляють суму відповідних підкатегорій:

```
df['total'] = df[['full_time_budget','full_time_contract',
                  'part_time_budget','part_time_contract',
                  'evening_budget','evening_contract']].sum(axis=1)
df['full_time'] = df['full_time_budget'] + df['full_time_contract']
df['part_time'] = df['part_time_budget'] + df['part_time_contract']
df['evening'] = df['evening_budget'] + df['evening_contract']
```

Даний підхід дозволяє легко агрегувати дані за датами та проводити подальший аналіз. Перед підготовкою даних для прогнозування перевіряється коректність вибраної категорії, що забезпечує відсутність помилок при обчисленнях:

```
if category not in ['total','full_time','part_time','evening']:
    print(f"Невідома категорія: {category}")
    return
```

Для моделі PROPheT дані необхідно підготувати у форматі з двома колонками: ds - дати, та y - значення, що прогнозуються. Дані групуються за датами та сумуються для кожного дня:

```
df_cat = df[['date', category]].copy()
df_cat.rename(columns={'date':'ds', category:'y'}, inplace=True)
df_cat = df_cat.groupby('ds')['y'].sum().reset_index()
```

Створення та навчання моделі PROPheT проводиться з урахуванням річної сезонності (yearly_seasonality=True), без врахування щоденної та

щотижневої сезонності (daily_seasonality=False, weekly_seasonality=False), що відповідає специфіці освітніх даних:

```
model = Prophet(yearly_seasonality=True, daily_seasonality=False, weekly_seasonality=False)
model.fit(df_prophet)
```

Прогноз на майбутні періоди здійснюється за допомогою створення датафрейму майбутніх дат з кроком у рік та обчислення прогнозних значень:

```
future = model.make_future_dataframe(periods=5, freq='Y')
forecast = model.predict(future)
```

Візуалізація даних здійснюється побудовою графіку, на якому відображаються історичні дані та прогноз PROPheT, з відповідними підписами та оформленням, що запобігає обрізанню елементів графіку:

```
plt.figure(figsize=(6,4))
plt.plot(df_cat['ds'], df_cat['y'], label='Історичні дані')
plt.plot(forecast['ds'], forecast['yhat'], label=f'Прогноз Prophet ({category})', color='red')
plt.title(f'Прогноз категорії: {category}')
plt.legend()
plt.tight_layout()
plt.show()
```

Таким чином, запропонований підхід забезпечує коректну підготовку даних, навчання моделі та наочну візуалізацію прогнозів за вибраною категорією.

Функція forecast_PROPheT(df) прогнозує загальну кількість студентів (сума всіх категорій) на наступні 5 років. Підготовка даних виглядає так:

```
df_prophet = df.groupby('date')['total'].sum().reset_index()
df_prophet.rename(columns={'date':'ds', 'total':'y'}, inplace=True)
```

Для прогнозування використовуються дані, що зберігаються у форматі JSON у папці saved_data. Кожен JSON-файл містить інформацію про студентів за різними категоріями навчання та фінансування. JSON-файли можуть мати різне кодування, включаючи utf-8-sig, cp1251 та latin1, що обумовлено різними джерелами даних або налаштуваннями систем, де вони були згенеровані. Тому при обробці необхідно коректно визначати кодування або надійно його обробляти, щоб уникнути помилок при завантаженні та конкатенації даних.

Збір даних виконується за допомогою функції load_student_jsons_saved(), яка проходить через всі JSON-файли у вказаній папці та об’єднує записи в один загальний DataFrame. Цей DataFrame стає базовою структурою для подальшого аналізу та прогнозування. Особливу увагу приділено колонкам, що є ключовими для моделювання та агрегування даних. До них належать:

- date - дата запису, яка використовується як часовий індекс для прогнозної моделі;
- full_time_budget та full_time_contract - кількість студентів денного відділення, які навчаються за бюджетною або контрактною формою;
- part_time_budget та part_time_contract - кількість студентів заочної форми навчання з бюджетом або контрактом;
- evening_budget та evening_contract - кількість студентів вечірньої форми навчання за бюджетом та контрактом.

Дані колонки забезпечують можливість подальшого агрегування та створення нових категорій (total, full_time, part_time, evening), необхідних для побудови прогнозу. Таким чином, модуль отримує структуровані дані, готові для обробки та аналізу, гарантуючи цілісність інформації незалежно від формату або кодування окремих файлів.

Механізм роботи простий: спочатку дані з історії студентів збираються та агрегуються за датами для обраної категорії (загальна кількість, повний день, заочники тощо). Потім PROPheT аналізує ці дані, визначаючи довгостроковий тренд та повторювані сезонні коливання, наприклад, щорічні зміни у кількості вступників. На основі цього він будує модель, здатну прогнозувати майбутні значення.

Прогноз можна робити на кілька років вперед, при цьому PROPheT не “вгадує” конкретні події, а лише продовжує виявлені закономірності з минулого. Результат представлений як графік, на якому можна порівняти історичні дані з прогнозованими, а також оцінити можливі коливання в майбутньому. Для зручності користувач може обирати конкретну категорію

студентів і отримувати окремий прогноз для неї. Таким чином, модуль забезпечує швидкий і зрозумілий спосіб передбачення розвитку освітньої діяльності університетів на основі наявних даних, допомагаючи планувати ресурси та аналізувати тенденції.

3.3.3 Модуль прогнозування LSTM

Модуль прогнозування на основі архітектури LSTM реалізовано як окремий програмний компонент аналітичного ядра системи, інтегрований у вкладку Analytical Forecast. Його мета - побудова точного прогнозу кількості випускників українських університетів за обраною спеціальністю та рівнем освіти (бакалавр або магістр) на основі часових рядів, отриманих із накопичених даних платформи.

Робота модуля розпочинається з виклику функції open_graduates_forecast_window(), яка відкриває модальне вікно для вибору параметрів прогнозу. Користувач обирає зі списку код спеціальності (наприклад, 122 - Комп’ютерні науки, 051 - Економіка тощо) та рівень освіти (Bachelor або Master). Після підтвердження вибору натисканням кнопки Run Forecast запускається функція forecast_graduates_lstm(specialty_code, level), яка виконує повний цикл прогнозування.

Перший етап - завантаження історичних даних із внутрішнього сховища (файлів JSON/CSV, створених під час збору даних університетів). З них формується таблиця DataFrame, у якій фіксується кількість випускників за кожен рік для вибраної спеціальності та рівня підготовки. Дані впорядковуються за часовою послідовністю, очищуються від пропусків, а відсутні значення заповнюються лінійною інтерполяцією, щоб зберегти безперервність ряду. Після очищення числові дані нормалізуються за допомогою масштабування Min-Max у діапазон [0, 1]. Що важливо, оскільки мережі LSTM чутливі до масштабу ознак - нормалізація прискорює збіжність і стабілізує градієнти під час навчання.

Для створення навчальних вибірок часовий ряд перетворюється на набір вхідних і цільових послідовностей за принципом ковзного вікна:

кожні n послідовних значень (дані за 5 років) утворюють вхідний вектор, а наступне значення - прогнозовану ціль. Це дозволяє моделі враховувати часовий контекст і тренди.

Архітектура нейронної мережі складається з кількох шарів:

1. Перший LSTM-шар з 64 нейронами, який зчитує вхідні послідовності та формує приховані представлення часових залежностей. Dropout-шар (0.2 - 0.3), який випадково «вимикає» частину нейронів під час навчання для запобігання перенавчанню.
2. Другий LSTM-шар з меншою кількістю нейронів (32 нейрони), який уточнює динамічні закономірності.
3. Вихідний Dense-шар з одним нейроном і лінійною активацією, що видає передбачене числове значення випускників.

Модель компілюється з використанням оптимізатора Adam, який динамічно регулює швидкість навчання, і функції втрат Mean Squared Error (MSE), оскільки вона найкраще підходить для безперервних числових прогнозів. Під час навчання модель проходить кілька епох (зазвичай 100-150) із використанням механізму early stopping - процес зупиняється, коли валідаційна помилка перестає зменшуватися. Для підвищення точності модель автоматично зберігає найкращу конфігурацію ваг (model_checkpoint) на основі мінімального значення помилки на валідаційному наборі.

Після навчання модель переходить у режим передбачення. Для побудови прогнозу використовується рекурсивний підхід: прогнозоване значення за один рік додається до кінця вхідної послідовності й використовується як частина даних для передбачення наступного року. Таким чином, модель може створювати багаторічний прогноз (наприклад, на 5 років уперед). Після завершення прогнозування дані інвертуються з нормалізованого масштабу до початкового, щоб отримати реальні показники кількості випускників. Результат об’єднується з історичними даними, що дозволяє побудувати суцільну динаміку «минуле - майбутнє».

Отримані дані відображаються у вигляді графіка, який будується безпосередньо в інтерфейсі платформи.

Модальне вікно, створене функцією open_graduates_forecast_window(), забезпечує зручність використання для аналітика. Після вибору параметрів користувач натискає Run Forecast, що викликає основну функцію прогнозування, закриває вікно і передає обрані аргументи (specialty_code, level) у модель LSTM. Весь процес відбувається інтерактивно, без необхідності повторного запуску програми.

Модуль LSTM дозволяє враховувати довготривалі часові залежності у кількості випускників, що особливо важливо для системи вищої освіти, де зміни мають інерційний характер. На відміну від класичних статистичних моделей, LSTM адаптивно підлаштовується під нелінійні закономірності, що можуть бути зумовлені реформами освіти, демографічними змінами чи попитом на ринку праці.

3.3.4 Модуль прогнозування XGBoost

Модуль прогнозування на основі алгоритму XGBoost (Extreme Gradient Boosting) реалізований для моделювання динаміки кількості поданих заявок на навчання у вищих навчальних закладах України з розподілом за спеціальностями та типом фінансування (бюджет або контракт). Основна мета модуля - надати університетам і аналітичним органам можливість оцінювати тенденції попиту на освітні напрями, прогнозувати навантаження на бюджетні місця та визначати ефективність приймальних кампаній.

Запуск прогнозування здійснюється через виклик функції open_applications_forecast_window(), що відкриває модальне вікно у вкладці *Analytical Forecast*. Інтерфейс вікна побудований у вигляді двох випадаючих списків (ttk.Combobox):

- спеціальність - користувач обирає один із наявних напрямів підготовки (наприклад, *Computer Science*, *Law*, *Psychology* тощо);

- тип фінансування - *budget* або *contract*.

Після вибору параметрів натискання кнопки Run Forecast викликає функцію forecast_applications_selected(specialty_choice, type_choice), яка виконує основні етапи прогнозування.

На першому етапі функція звертається до внутрішнього сховища системи, зчитуючи історичні дані про кількість заявок за обраною спеціальністю та типом фінансування за останні десять років. Дані формуються у вигляді часової таблиці DataFrame, де:

- стовпець ds містить часову позначку (рік),
- стовпець y - кількість заявок.

Після завантаження дані проходять базову обробку: усунення пропусків, перевірка коректності типів змінних, нормалізація діапазонів значень, за необхідності - усереднення по регіонах чи університетах. Після підготовки даних формується модель градієнтного бустингу XGBoost, яка поєднує у собі велику кількість слабких моделей - дерев рішень, що поступово навчаються на помилках попередніх і таким чином покращують підсумковий прогноз. Архітектура побудована за принципом послідовного навчання ансамблю дерев:

1. Кожне дерево приймає як вхід попередні залишкові похибки та формує корекційні предиктори.
2. Алгоритм мінімізує функцію втрат (у цьому випадку - *Mean Squared Error*) із використанням градієнтного спуску.
3. Отримана модель здатна враховувати нелінійні закономірності між роками, типом фінансування та рівнем попиту на спеціальність.

Особливістю XGBoost є використання регуляризації (λ, α), що зменшує ризик перенавчання, а також оптимізована обробка даних із пропусками, що підвищує ефективність навіть при неповних наборах. Під час навчання модель налаштовується за такими параметрами:

- n_estimators - кількість дерев (приблизно 300-500);
- learning_rate - швидкість навчання (0.05-0.1);

- max_depth - глибина дерев (4-6);
- subsample - частка даних для кожного дерева (0.8 для зменшення варіацій).

Модель поступово наближає прогнозовані значення до фактичних, зменшуючи середньоквадратичну помилку на кожній ітерації. Після навчання XGBoost використовується для передбачення кількості заявок на наступні роки. Для цього створюється розширена часова шкала (future), що охоплює 5 прогнозованих періодів (років). Модель обчислює очікувані значення заявок, враховуючи тенденції попередніх років, нелінійні залежності між попитом на спеціальність, типом фінансування та динамікою освітнього ринку. Отримані результати об’єднуються з історичними даними, формуючи єдиний часовий ряд, що містить як минулі спостереження, так і прогнозовані значення.

3.3.5 Модуль прогнозування Random Forest

Модуль прогнозування на основі алгоритму Random Forest реалізований для оцінки та передбачення популярності спеціальностей у вищих навчальних закладах України. Його головним завданням є визначення тенденцій змін кількості студентів за напрямами підготовки, що дозволяє університетам планувати прийом, розподіл ресурсів і прогнозувати попит на освітні програми.

Для запуску прогнозу використовується модальне вікно, створене функцією open_specialty_forecast_window(). Користувач має можливість обрати конкретну спеціальність із доступного списку через випадаючий комбобокс (ttk.Combobox). Після вибору спеціальності натискання кнопки Run Forecast активує функцію forecast_specialty_popularity_selected(specialty_choice), яка відповідає за формування та візуалізацію прогнозу.

На першому етапі дані завантажуються із збережених аналітичних баз університетів, де зберігається історична інформація про кількість студентів за спеціальностями. Структура даних представлена у вигляді часових рядів:

- колонка ds - рік спостереження,
- колонка y - фактична кількість студентів за спеціальністю.

Дані проходять попередню обробку, яка включає очищення від пропусків, перевірку коректності типів та масштабування при необхідності. Це забезпечує стабільність роботи моделі Random Forest і дозволяє уникати впливу викидів на прогноз.

Для побудови прогнозу застосовується готова модель Random Forest Regressor, яка була навчена на історичних даних. Модель складається з ансамблю дерев рішень, де кожне дерево прогнозує кількість студентів на основі випадкових підмножин даних та ознак. Результат прогнозу формується як середнє значення усіх дерев, що забезпечує стійкість до шуму та підвищує точність передбачень. Кожне дерево враховує часові залежності та закономірності росту чи спадання попиту на спеціальність. Алгоритм автоматично враховує нелінійні взаємозв’язки між роками, популярністю спеціальностей та іншими навчальними характеристиками. Використання готової моделі дозволяє уникнути тривалого процесу навчання і гарантує стабільні прогнози на основі перевірених параметрів. Модель налаштована таким чином, щоб оптимально враховувати коливання історичних даних і передбачати тенденції на кілька наступних років.

Функція forecast_specialty_popularity_selected формує прогнозовані значення на наступні п’ять років. Процес включає:

1. Генерацію часової шкали для майбутніх періодів.
2. Використання Random Forest для обчислення очікуваних значень кількості студентів у кожному році.
3. Поєднання історичних даних та прогнозованих значень для створення єдиного часового ряду.

Таким чином, користувач отримує повну картину динаміки популярності спеціальності від минулих спостережень до очікуваних тенденцій.

3.4 Функціональне тестування

Функціональне тестування є ключовим етапом забезпечення якості програмного забезпечення та цифрових платформ, оскільки дозволяє перевірити відповідність системи вимогам користувача та технічним специфікаціям. Використання функціонального тестування дозволяє виявити помилки на ранніх етапах, забезпечити надійність та стабільність платформи, а також підтвердити її придатність для стратегічного управління розвитком університетів на основі доказових даних. На рис.3.4 відображено головну сторінку додатку.

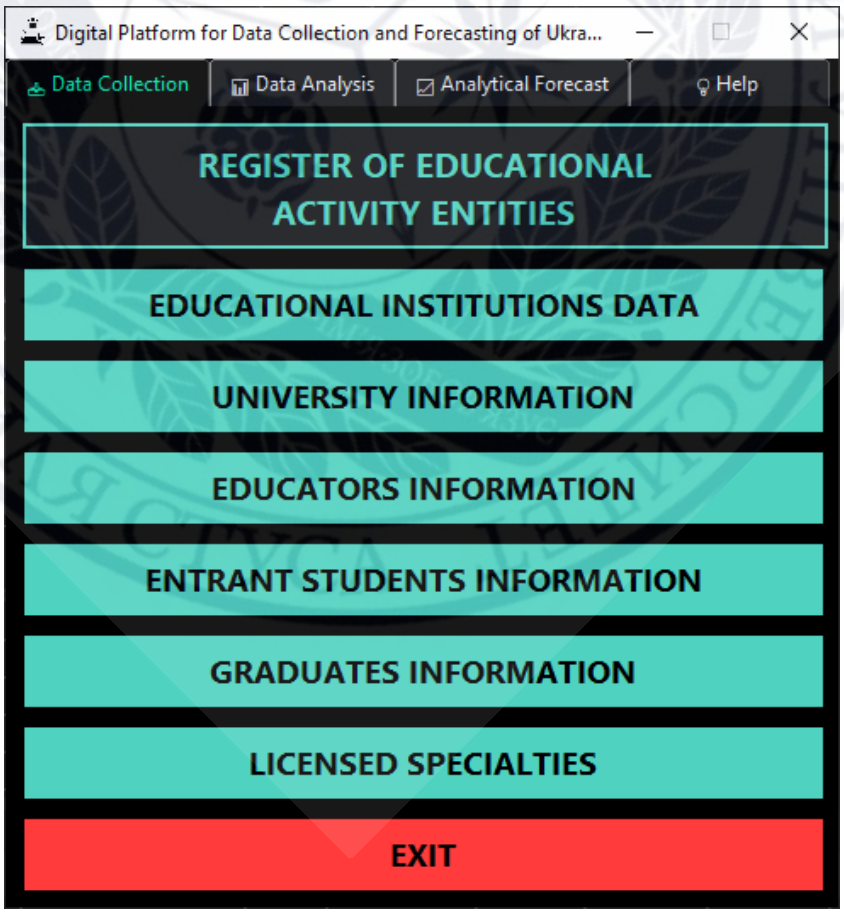

Рисунок 3.4 - Головна сторінка

Головна сторінка має заголовок "REGISTER OF EDUCATIONAL ACTIVITY ENTITIES" (Реєстр суб'єктів освітньої діяльності) і містить меню з кількома великими кнопками для доступу до різних розділів збору даних:

1. EDUCATIONAL INSTITUTIONS DATA (Дані про освітні установи)
2. UNIVERSITY INFORMATION (Інформація про університети)
3. EDUCATORS INFORMATION (Інформація про викладачів)
4. ENTRANT STUDENTS INFORMATION (Інформація про студентів-вступників)
5. GRADUATES INFORMATION (Інформація про випускників)
6. LICENSED SPECIALTIES (Ліцензовані спеціальності)
7. EXIT (Вихід)

У верхній частині екрана також розташоване навігаційне меню з вкладками: Data Collection (Збір даних, яка зараз активна), Data Analysis (Аналіз даних), Analytical Forecast (Аналітичний прогноз) та Help (Допомога). На рис.3.5 відображено діалогове вікно "Institution Information", яке призначене для пошуку освітніх установ.

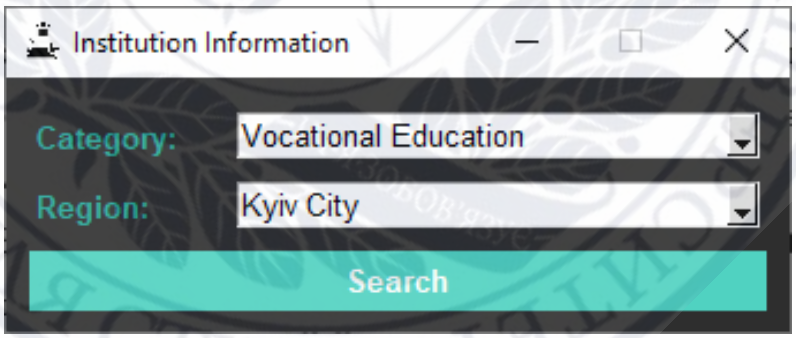

Рисунок 3.5 - Вікно Institution Information

У вікні представлені два поля для фільтрації даних:

1. Category: (Категорія) - поле для вибору типу освітньої установи. Наразі обрано "Vocational Education" (Професійно-технічна освіта).
2. Region: (Регіон) - поле для вибору географічного розташування. Наразі обрано "Kyiv City" (Місто Київ).

Під полями для вибору розташована кнопка Search (Пошук) для ініціації пошуку установ, які відповідають обраним критеріям.

На рис.3.6 відображено вікно "Institution Data", що відкривається після натискання кнопки "Search" на попередньому екрані фільтрації (де було обрано "Vocational Education" та "Kyiv City").

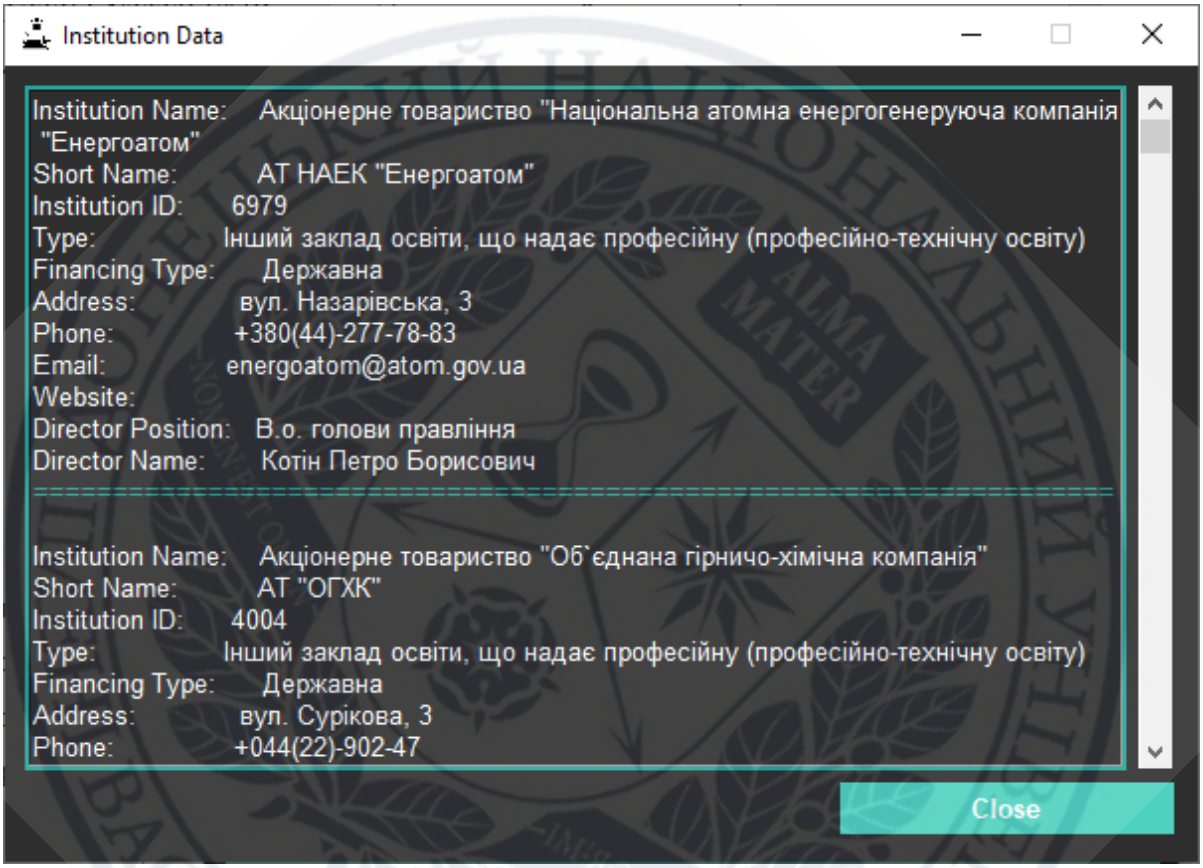

Рисунок 3.6 - Результати Institution Data

Дане вікно відображає результати пошуку, які відповідають заданим критеріям. Наступною кнопкою є Institution Information. При натисканні якого відкривається форма вводу Institution ID, що відображено на рис.3.7.

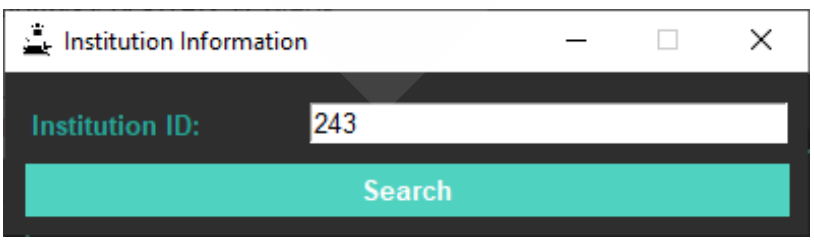

Рисунок 3.7 - Вікно вводу Institution ID

Після вводу Institution ID відкривається вікно University Data.

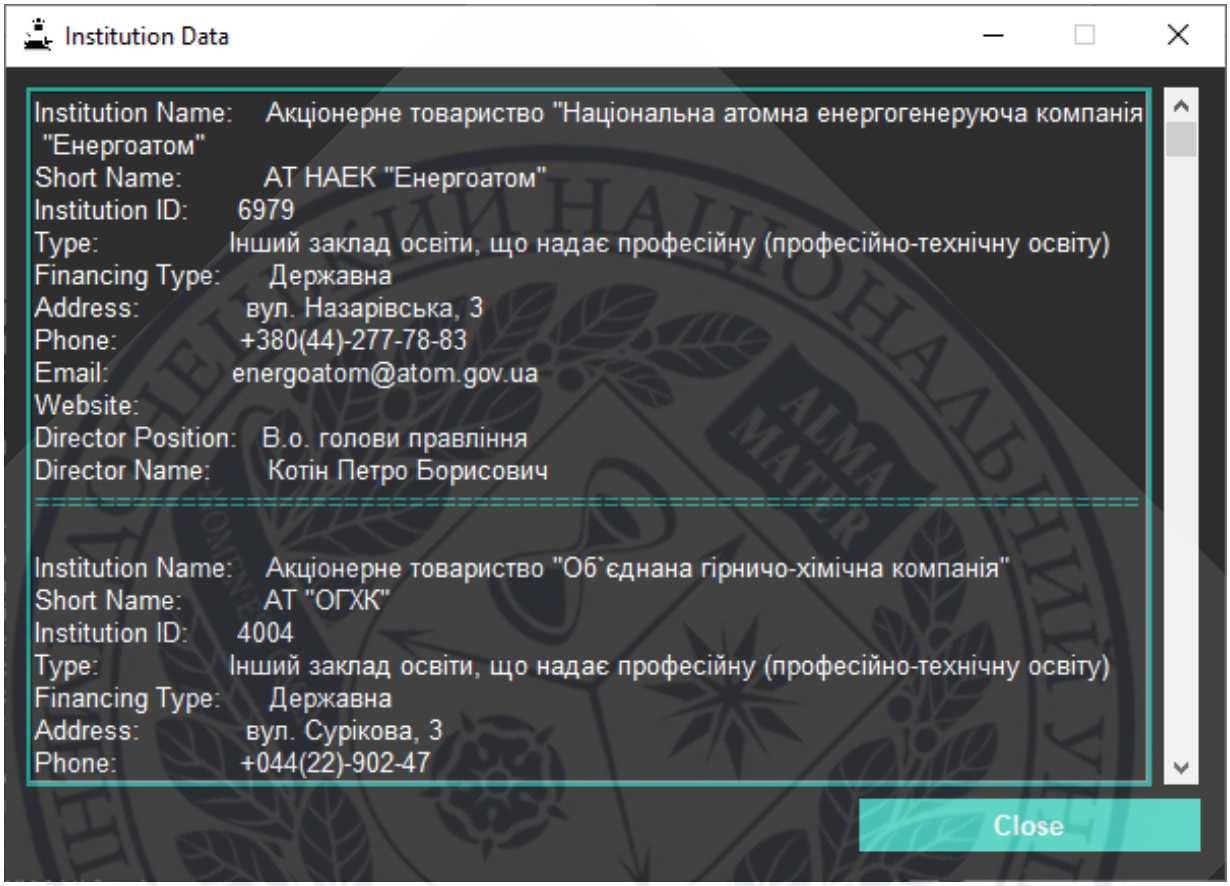

Рисунок 3.8 - Вікно University Data

На рис.3.8 представлено вікно "University Data" (Дані про університет), що відображає детальну інформацію про конкретний заклад вищої освіти. Вікно містить повні реєстраційні та контактні дані (назва, ID, тип, форма фінансування - Приватна, адреса, контакти, ПІБ директора), а також перелік його структурних підрозділів (Faculties). Фактично, це розгорнута картка установи, яка відкривається для перегляду після її вибору у реєстрі.

На рис.3.9 зображено форму введення даних для пошуку у розділі "Educators Information". Це вікно відкривається після натискання

відповідної кнопки на головній сторінці додатку. Форма дозволяє користувачу фільтрувати дані про викладачів за кількома критеріями:

1. Date - поле, де вручну введено дату 01.10.2018.
2. Education Level - випадаючий список, де обрано "Bachelor" (Бакалавр).
3. Entry Basis - випадаючий список, де обрано "Complete General Secondary Education" (Повна загальна середня освіта).
4. Specialty Code - поле для введення коду. Введено 122.
5. Region - випадаючий список, де обрано "Kyiv City" (Місто Київ).
6. Inst. ID (opt.) - поле для необов'язкового введення ідентифікатора конкретної освітньої установи. Поле порожнє.

Внизу форми розташована кнопка "Search" (Пошук) для виконання запиту з введеними параметрами.

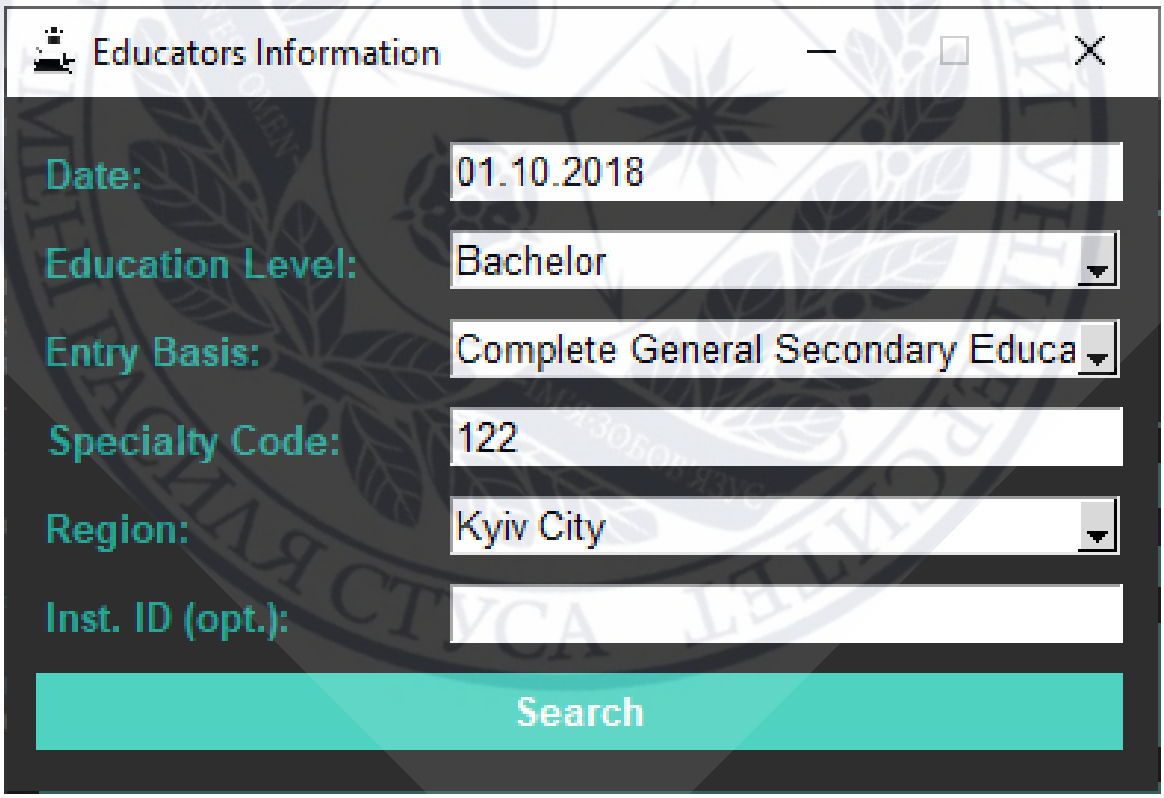

Рисунок 3.9 - Форма вводу даних для пошуку Educators Information

На рис.3.10 зображено вікно "Educators Data" - результат пошуку за фільтрами. Воно містить зведену статистику станом на 01.10.2018 у м. Києві для спеціальності "Комп'ютерні науки" (код 122).

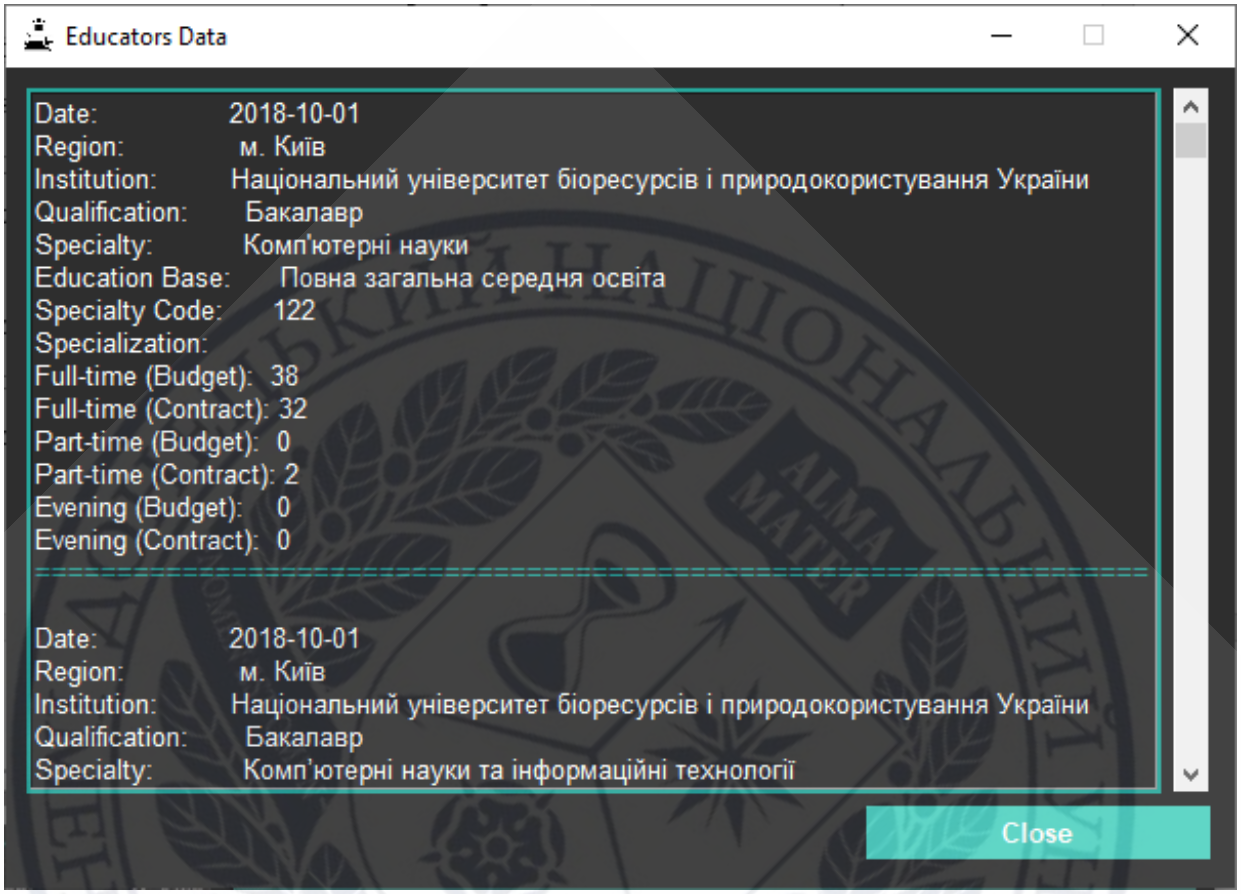

Рисунок 3.10 - Educators Data

Ключові дані - це розподіл студентів, яких навчають ці викладачі, за формами навчання (денна/заочна/вечірня) та джерелами фінансування (бюджет/контракт). На рис.3.11 зображено форму введення даних для пошуку у розділі "Enrolled Students Information" (Інформація про зарахованих студентів). Ця форма призначена для фільтрації та пошуку даних про студентів-вступників або зарахованих студентів за кількома параметрами, аналогічними до попередніх форм пошуку:

1. Year - введено 2018.
2. Education Level - обрано "Bachelor" (Бакалавр).
3. Entry Basis - обрано "Complete General Secondary Education" (Повна загальна середня освіта).

4. Specialty Code - введено 122.
5. Region - обрано "Kyiv City"
6. Inst. ID (opt.) - поле для необов'язкового введення ідентифікатора конкретної освітньої установи. Поле порожнє.

Внизу форми розташована кнопка "Search" (Пошук) для ініціації запиту.

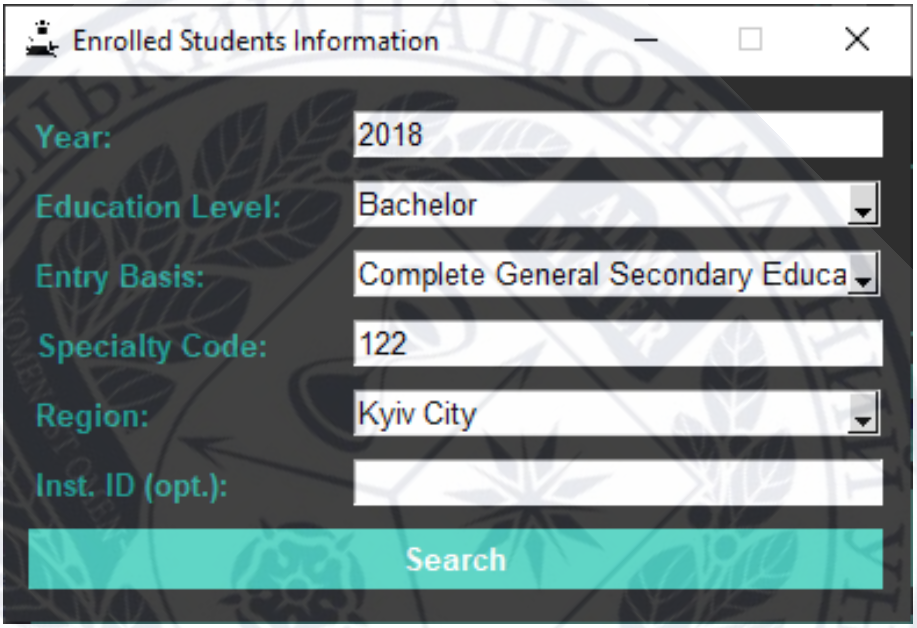

Рисунок 3.11 - Educators Data

На рис.3.12 зображено вікно "Enrolled Students Data" - це результат пошуку за критеріями (2018 рік, м. Київ, Бакалавр, код 122). Вікно містить зведену статистику зарахування студентів, де для кожної установи відображаються:

1. Загальні параметри: рік (2018), регіон (м. Київ), установа, кваліфікація (Бакалавр), спеціальність ("Комп'ютерні науки").
2. Кількісні показники: розподіл зарахованих студентів за формами навчання (денна/заочна/вечірня) та джерелами фінансування (Бюджет/Контракт).

Наприклад, для Національного університету біоресурсів і природокористування України відображено: 21 студент на денній формі за бюджетом і 18 за контрактом.

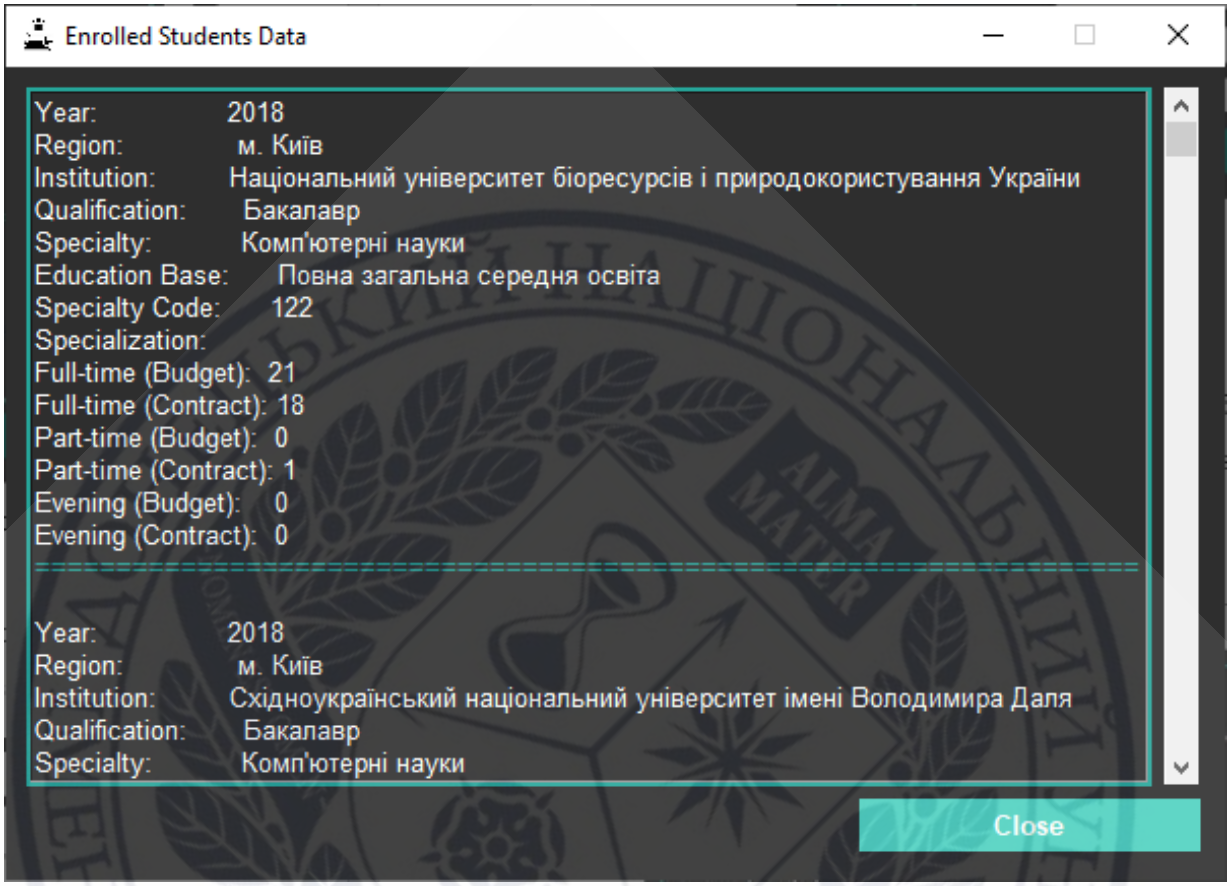

Рисунок 3.12 - Знайдена інформація про зарахованих студентів

У вікні представлені дані щонайменше для двох університетів, а смуга прокрутки дає змогу для перегляду всіх знайдених результатів за запитом. На рис.3.13 зображено форму пошуку "Graduated Students Information" (Інформація про випускників). Це вікно дозволяє фільтрувати дані про студентів, які завершили навчання, за такими основними критеріями:

- Рік випуску
- Освітній рівень
- Спеціальність
- Регіон

Форма використовується для збору статистики про випуск фахівців за конкретний рік та спеціалізацію.

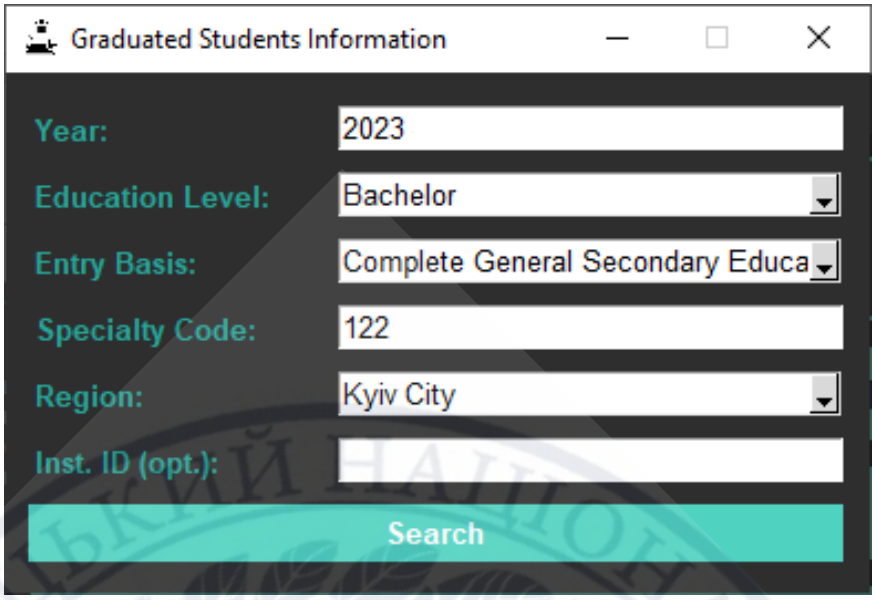

Рисунок 3.13 - Форма вводу для пошуку інформації про випускників

На рис. 3.14 зображено вікно "Graduated Students Data" - результат пошуку випускників 2023 року за спеціальністю "Комп'ютерні науки" (код 122) у м. Києві.

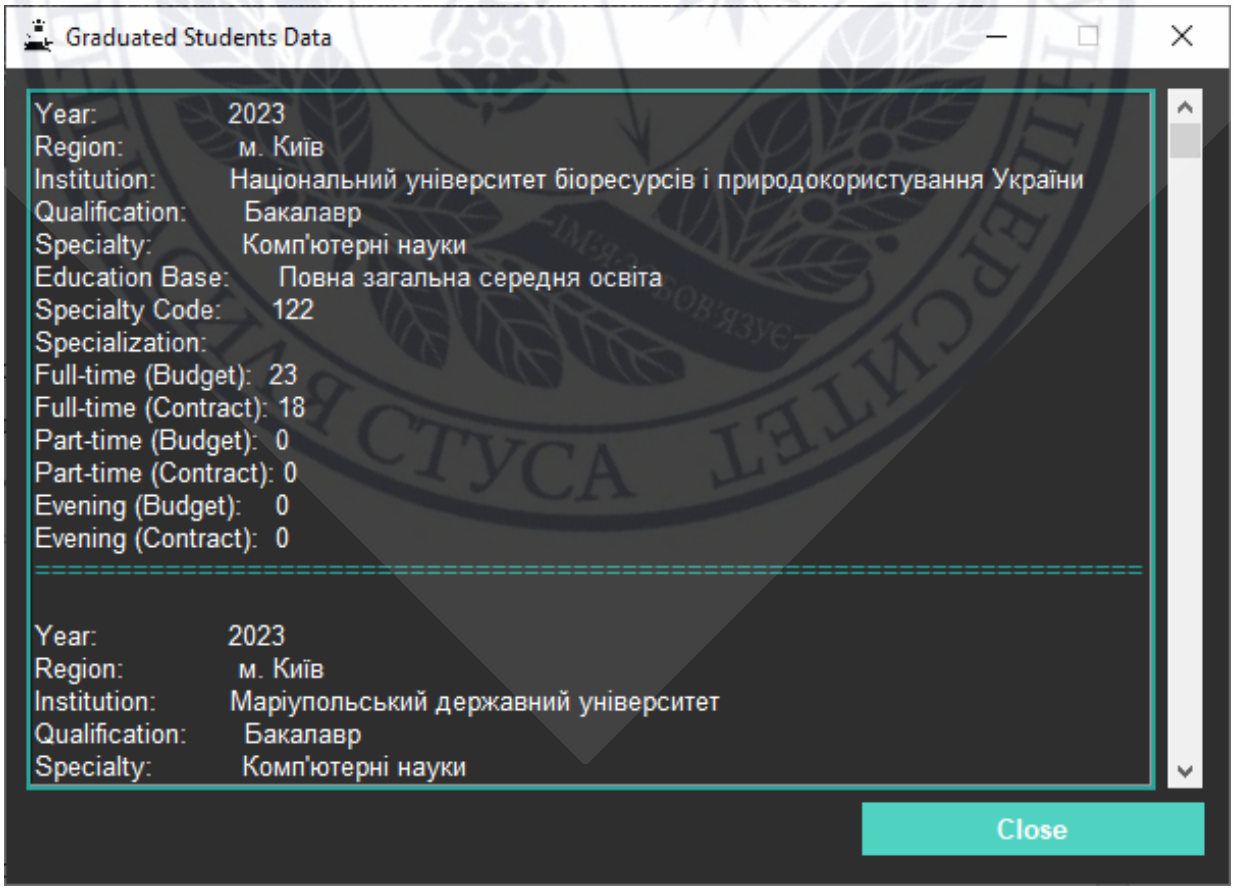

Рисунок 3.14 - Результат пошуку інформації про випускників

Вікно містить зведену статистику випуску студентів-бакалаврів. Для кожної установи (наприклад, Національний університет біоресурсів і природокористування України) відображається розподіл випускників за формами навчання та джерелами фінансування:

- Денна форма (Бюджет): 23 випускники.
- Денна форма (Контракт): 18 випускників.
- Інші форми (заочна, вечірня) не мають випускників у цьому блоці.

На рис 3.15 зображено форму пошуку "Licensed Specialties Information" (Інформація про ліцензовані спеціальності). Це вікно дозволяє користувачу знайти, які освітні установи мають ліцензію на викладання конкретної спеціальності за певними параметрами.

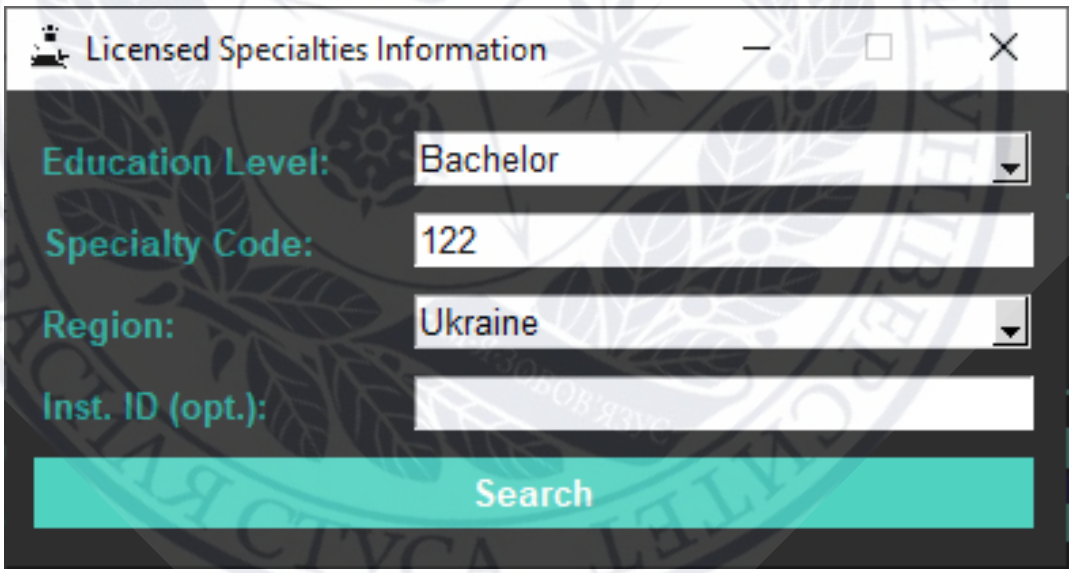

Рисунок 3.15 - Форма вводу пошуку ліцензій по конкретному напрямку

На рис. 3.16 зображено вікно "Licensed Specialties Data" (Дані про ліцензовані спеціальності) - результат пошуку за ліцензією (Бакалавр, код 122, Регіон: Україна).

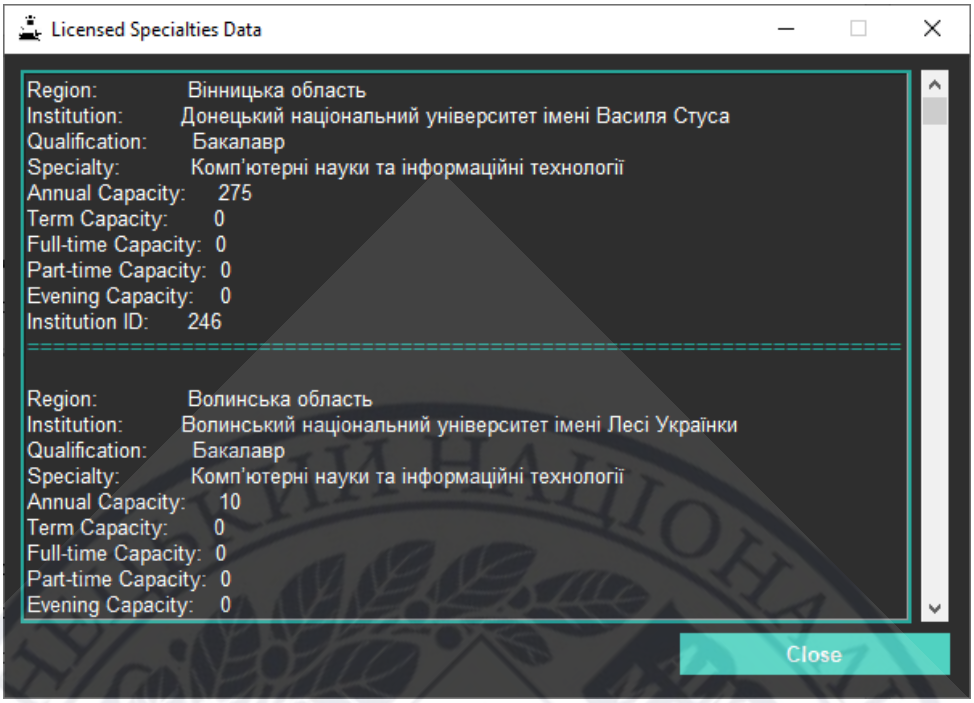

Рисунок 3.16 - Результат пошуку ліцензій по конкретному напрямку

Вікно містить перелік установ, які мають ліцензію на спеціальність "Комп'ютерні науки та Інформаційні технології" з деталізацією по регіонах. Перейдемо до вкладки "Data Analisys". На рис. 3.17 відображено розподіл студентів по регіоні за спеціальністю 122.

Рисунок 3.17 - Розподіл студентів по регіоні за спеціальністю 122

Діаграма відображає загальну кількість студентів (за бюджетом та контрактом) за спеціальністю 122 в усіх регіонах. На рис. 3.18 зображено аналітичне вікно "Comparison of University Specialties" .

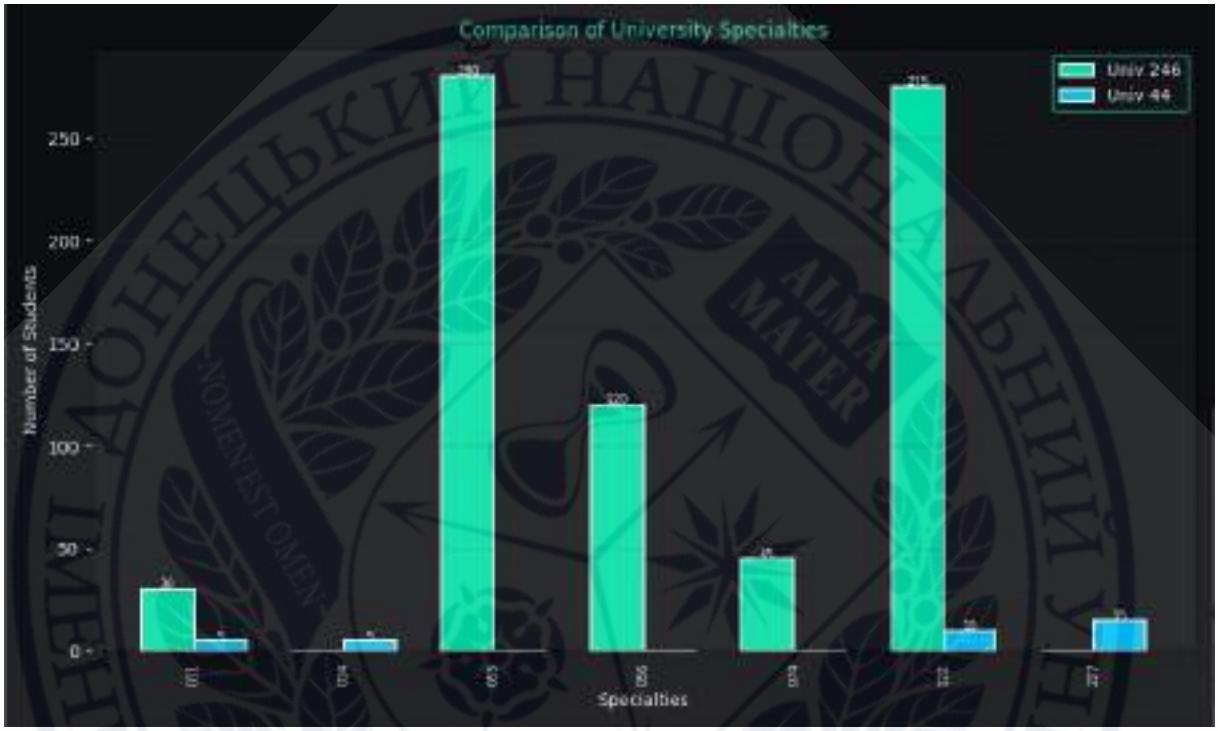

Рисунок 3.18 - Розподіл студентів по регіоні за спеціальністю 122

Діаграма демонструє концентрацію студентів за освітніми програмами в кожному з двох університетів: Університет 246 має значно більшу кількість студентів і концентрується на спеціальностях 055 (280 студентів) та 122 (275 студентів). Він також має студентів на 011, 056 та 074. Університет 44 має загалом набагато менший контингент і концентрується на спеціальностях 056 (120 студентів). Він має невелику кількість студентів на 011, 014, 122 та 227. Ця діаграма дозволяє візуально оцінити профільну спрямованість та масштаб двох освітніх закладів. На рис. 3.19 зображено аналітичну вертикальну стовпчасту діаграму "Most Popular Specialties in 2023".

Рисунок 3.19 - Найпопулярніші спеціальності у 2023 році

Діаграма відображає рейтинг 11 найбільш популярних спеціальностей за загальною кількістю студентів у 2023 році. Даний графік слугує для визначення загальнонаціонального попиту на освітні програми.

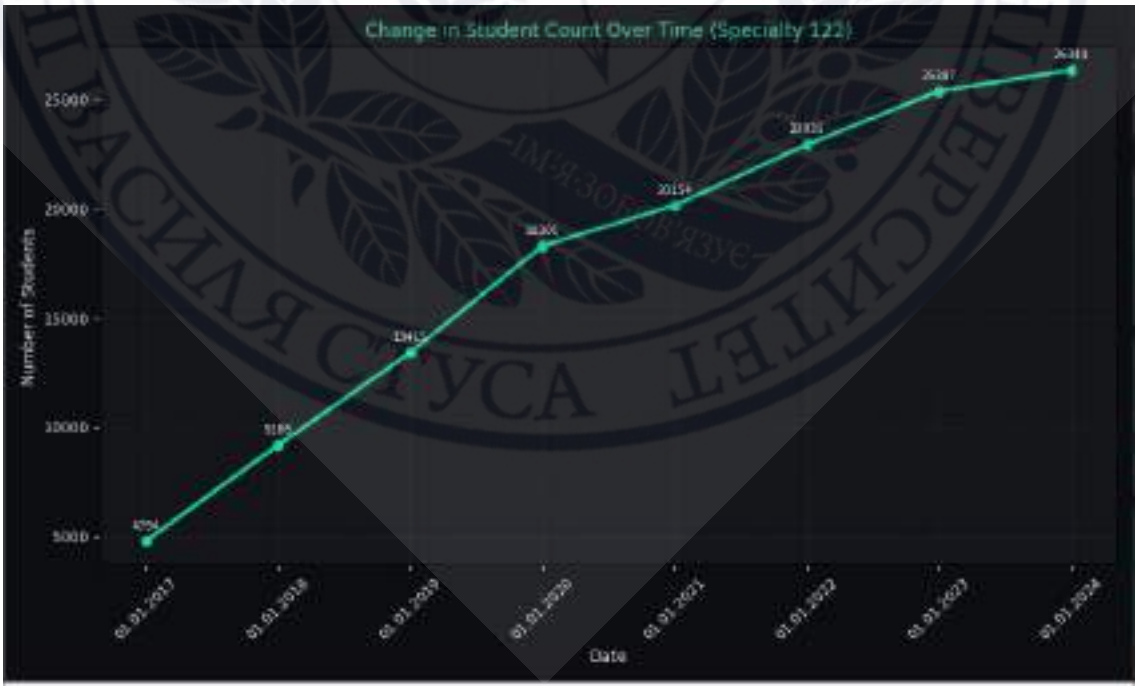

Рисунок 3.20 - Зміна кількості студентів з часом за вибраною спеціальністю

На рис. 3.20 відображено лінійний графік що дозволяє аналізувати зміну кількості студентів за певний період часу та спеціальністю. На рис. 3.21 відображено графік аналізу форм навчання.

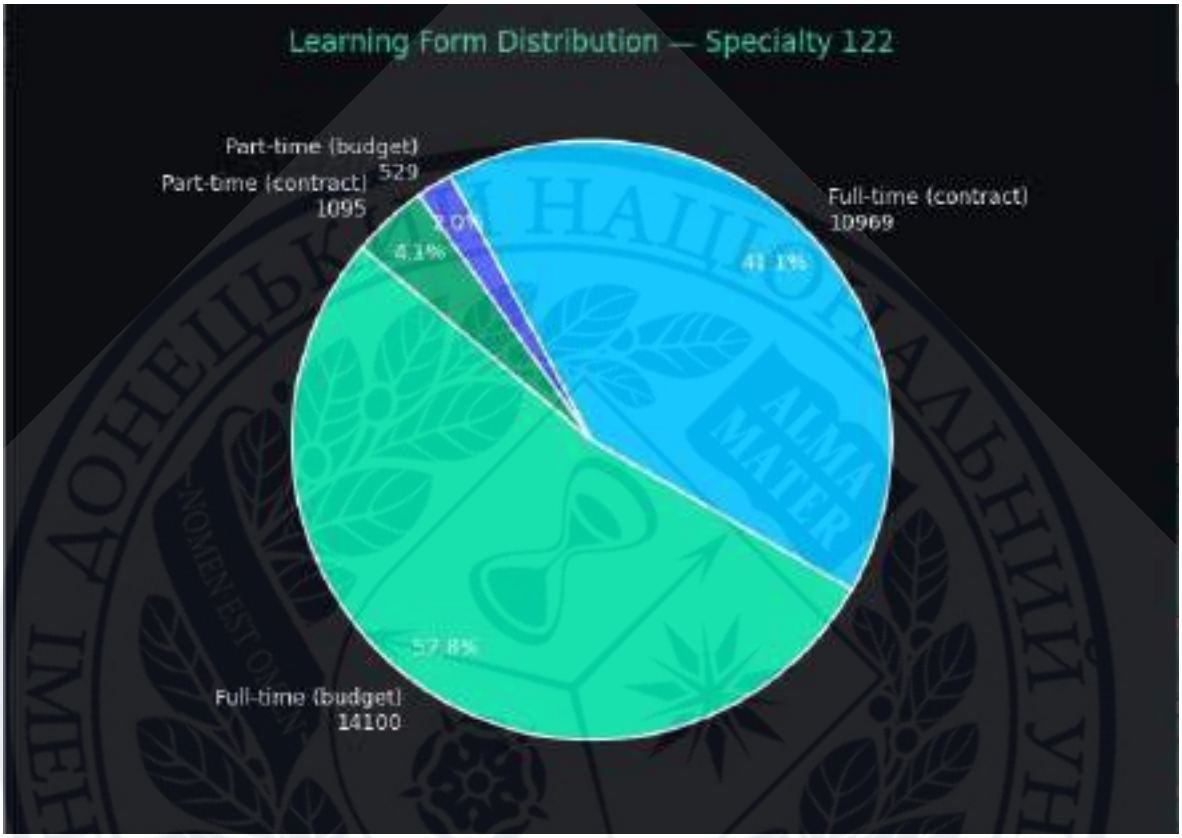

Рисунок 3.21 - Аналіз форм навчання

Додаток надає можливість обирати дату, рівень освіти, основу навчання та спеціальність. Перейдемо до вкладки прогнозування. На рис. 3.22 зображено вміст вкладки "Analytical Forecast" додатку. Дана вкладка є розділом системи, що призначений для прогнозування даних. Вкладка містить два основні розділи/кнопки для ініціювання прогнозування, обидва з використанням моделей прогнозування:

1. PROPHET - TOTAL NUMBER OF STUDENTS
2. PROPHET - CATEGORY FORECAST
3. RANDOM FOREST - POPULARITY FORECAST
4. LSTM - FORECAST OF GRADUATES
5. XGBoost - APPLICATIONS FORECAST

6. MLPNN - STUDENT DEMAND FOR ONLINE LEARNING
7. ARIMA - STUDENT RETENTION FORECAST
8. LSTM - FACULTY DEMAND FORECAST

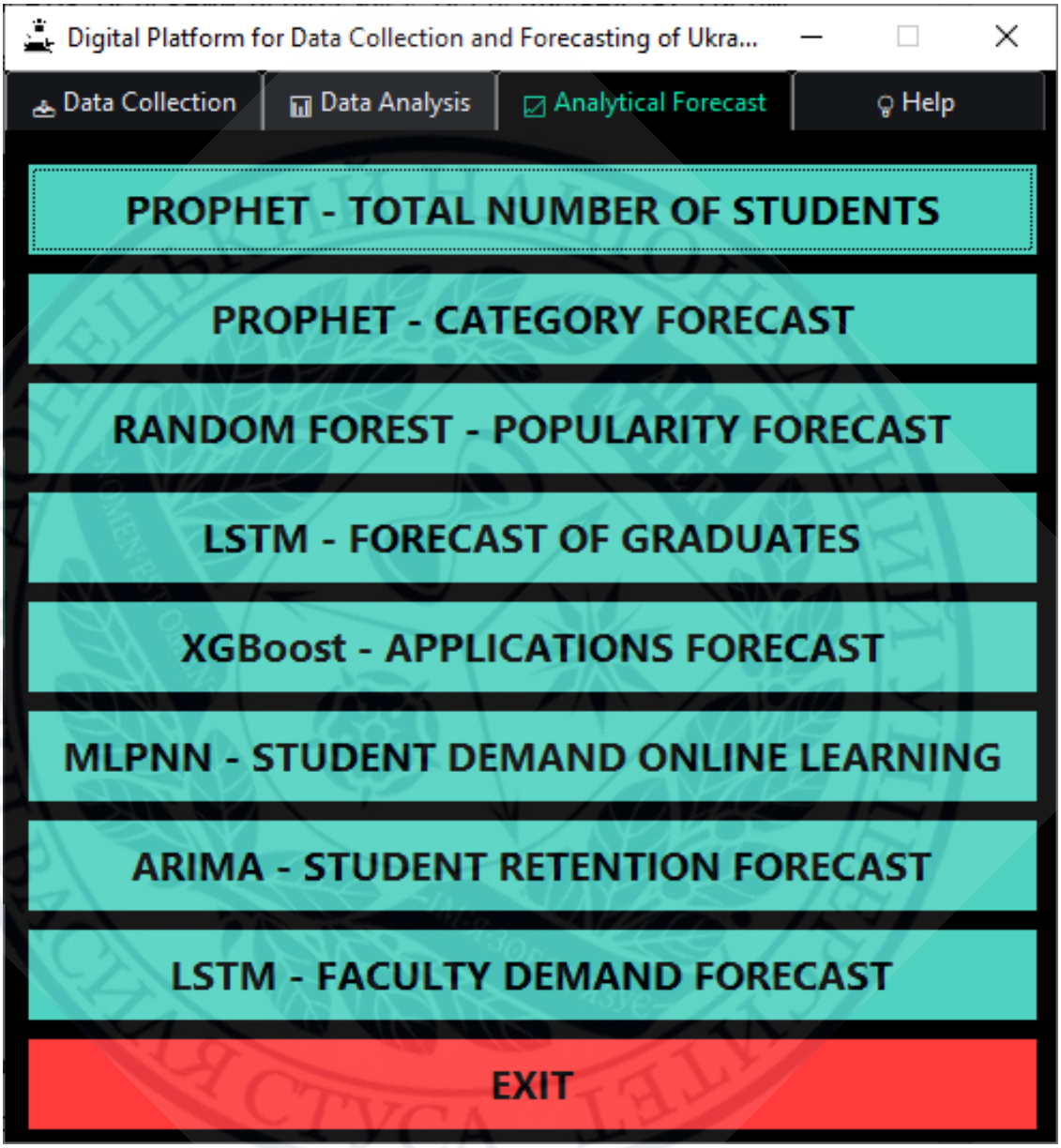

Рисунок 3.22 - Прогнозування для загальної кількості студентів за допомогою PROPheT

Під другою кнопкою розташований елемент керування "Select a category for the forecast:", що є випадаючим списком і дозволяє користувачу уточнити, за якою категорією даних має бути побудований прогноз. На рис. 3.23 зображено лінійний графік "Прогноз загальної кількості студентів",

який є результатом роботи аналітичного інструменту прогнозування. Графік показує зміну загальної кількості студентів з часом і розділений на дві частини:

1. Історичні дані (Синя лінія) - відображає фактичну кількість студентів за минулі роки (з 2017 до 2023 року).
2. Прогноз PROPheT (Червона лінія) - відображає прогнозну кількість студентів на майбутнє.

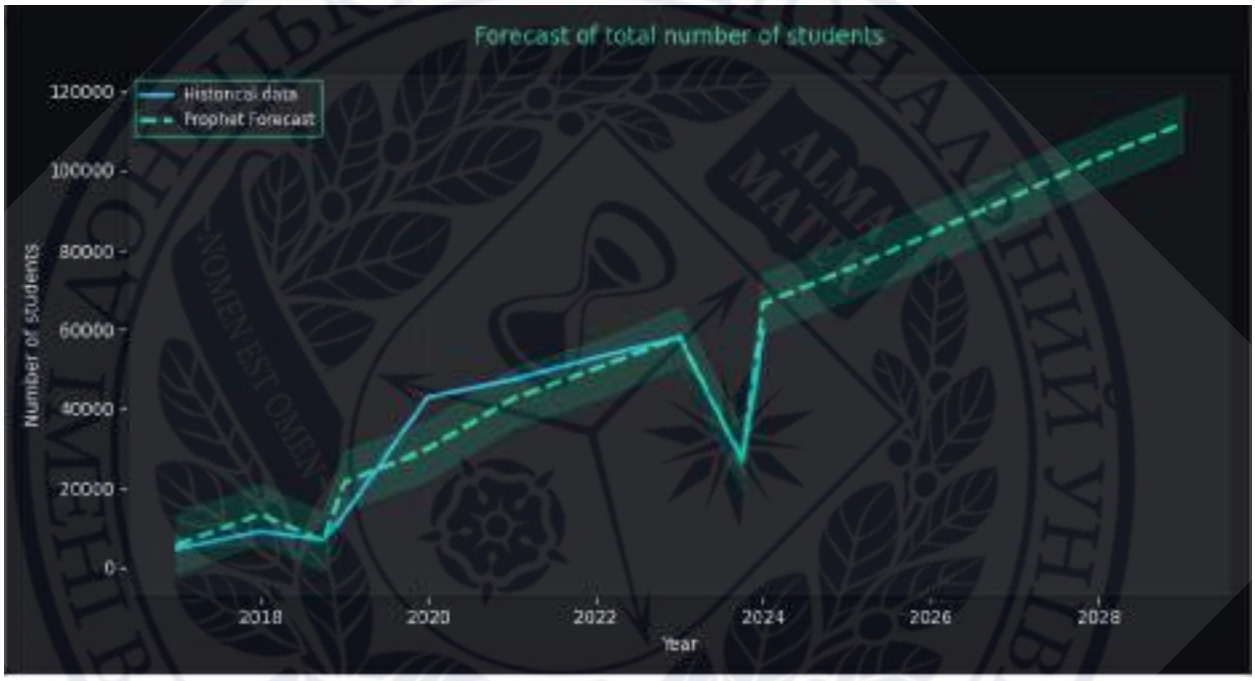

Рисунок 3.23 - Прогнозування для загальної кількості студентів за допомогою PROPheT

Прогноз, побудований на основі історичних даних, показує різке зростання загальної кількості студентів у довгостроковій перспективі - від приблизно 65 000 у 2024 році до понад 110 000 до 2029 року. На рис. 3.24 зображено лінійний графік "Прогноз категорії: part_time".

Таким чином, аналіз прогнозу, побудованого за моделлю PROPHET, підтверджує потенційну тенденцію до збільшення кількості студентів у найближчі роки та демонструє практичну цінність використання автоматизованих інструментів у сфері управління освітою.

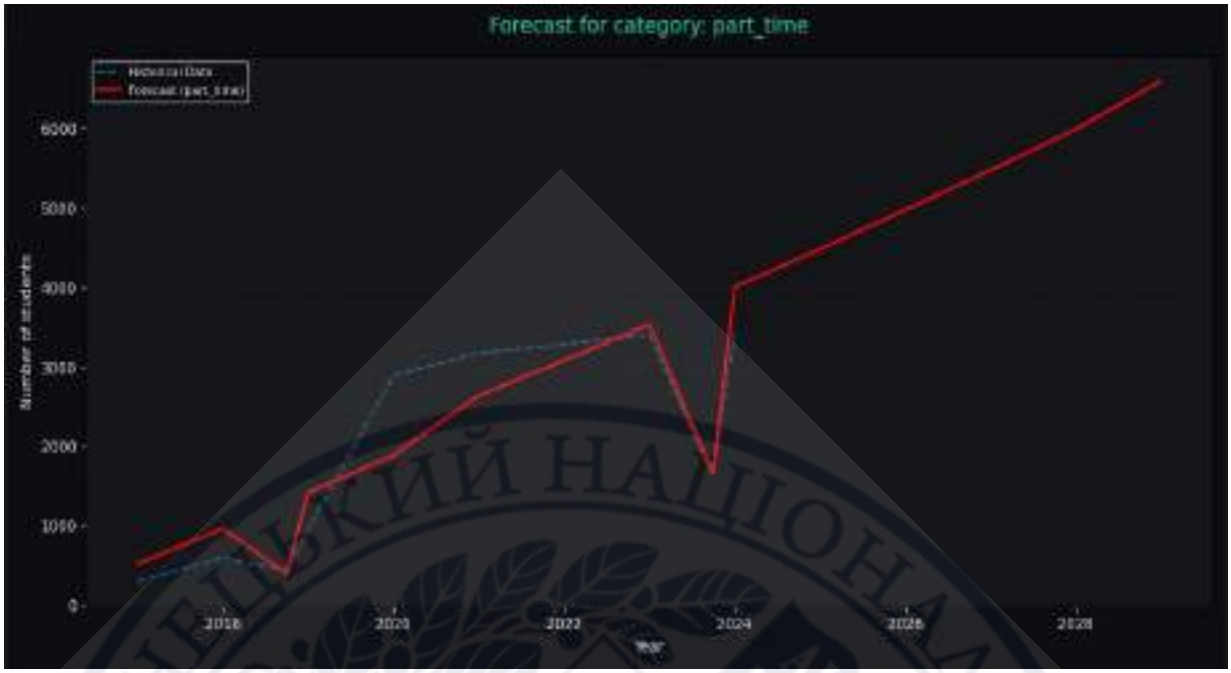

Рисунок 3.24 - Прогнозування для категорії заочної форми навчання за допомогою PROPheT

Прогноз вказує на очікуване постійне та значне зростання кількості студентів на заочній формі навчання: з приблизно 4000 осіб у 2024 році до понад 6500 осіб до 2029 року. На рис. 3.25 зображено лінійний графік "Прогноз категорії: evening".

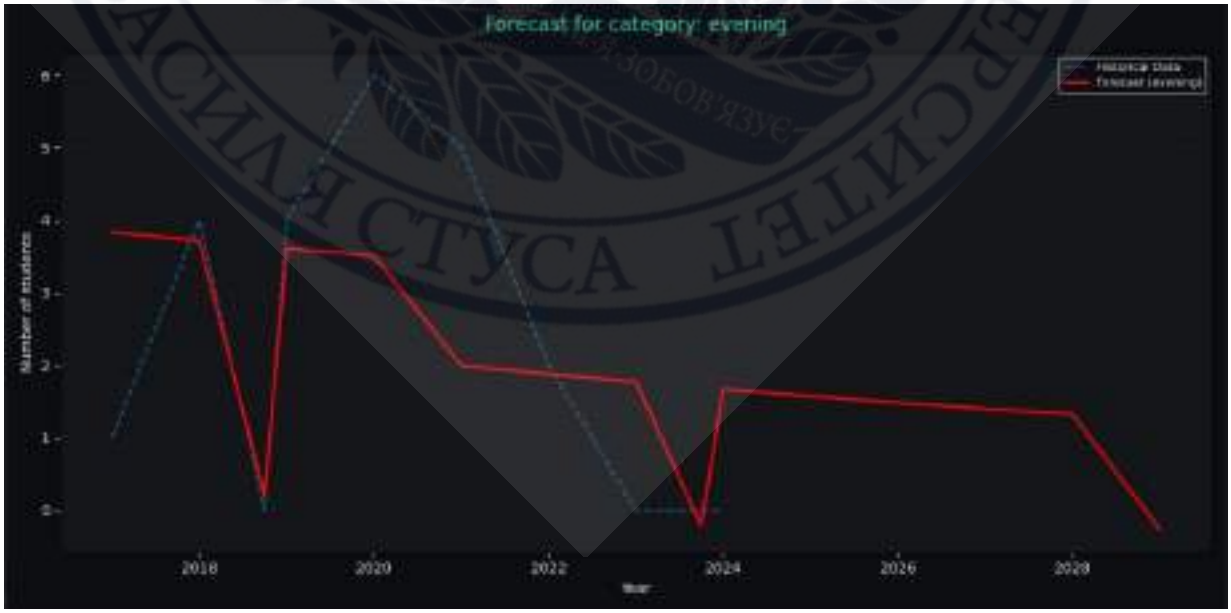

Рисунок 3.25 - Прогнозування для категорії вечірньої форми навчання PROPheT

Прогноз PROPheT показує, що кількість студентів на вечірній формі навчання, ймовірно, залишиться дуже низькою у період з 2024 до 2028 року, і практично впаде до нуля до 2029 року.

На рис. 3.26 відображено прогнозування тренду популярності спеціальностей у вищих навчальних закладах за допомогою алгоритму Random Forest. По горизонтальній осі представлені роки спостережень, що охоплюють історичні дані та прогнозований період, тоді як по вертикальній осі показано кількість студентів за кожною спеціальністю.

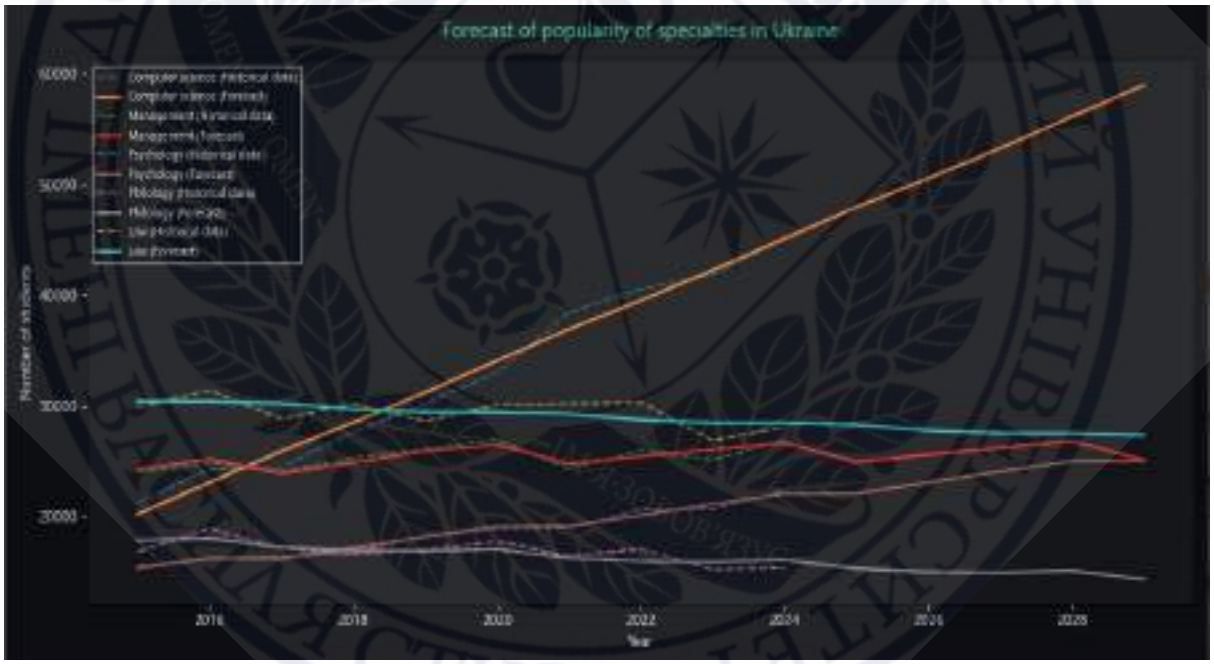

Рисунок 3.26 - Прогнозування тренду метод Random Forest

Візуалізація дозволяє оцінити очікувані зміни тренду на спеціальності. На рис. 3.27 представлено прогнозування кількості випускників за окремими спеціальностями та науковим рівнем із використанням рекурентної нейронної мережі LSTM.

Такий підхід дає змогу виявити довгострокові закономірності та порівняти динаміку розвитку різних освітніх напрямів. Отримані результати можуть бути використані для стратегічного планування та оптимізації освітніх програм у закладах вищої освіти.

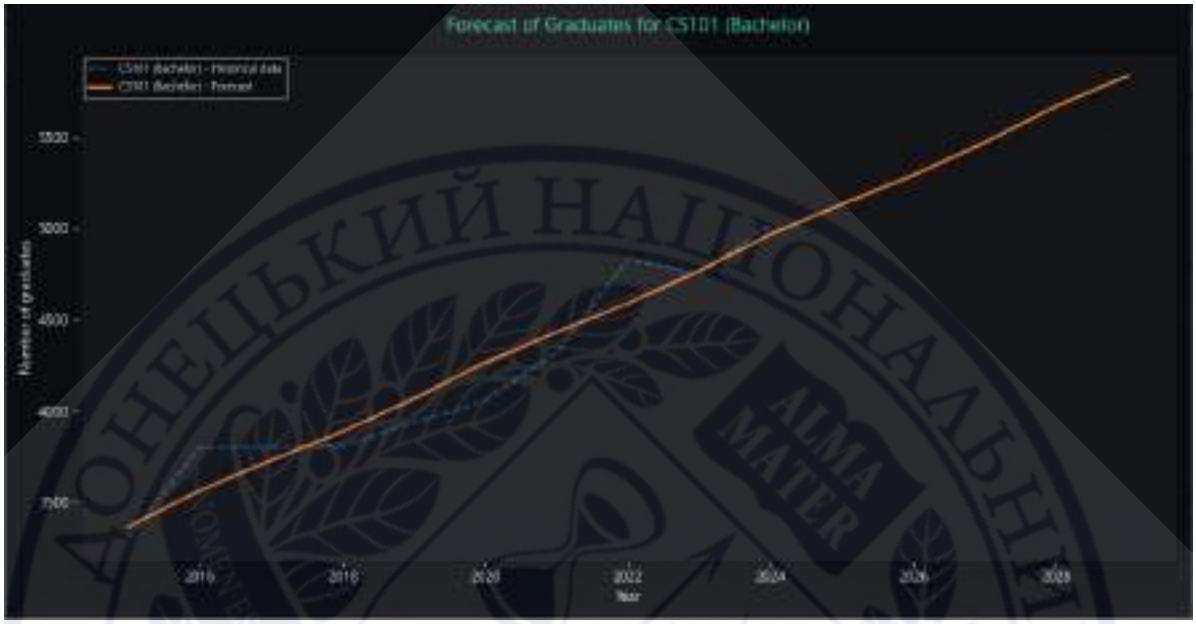

Рисунок 3.27 - Прогнозування кількості випускників метод LSTM

По горизонтальній осі зображено роки спостережень та прогнозу, а по вертикальній осі відображено кількість випускників за відповідною спеціальністю та рівнем освіти. Використання LSTM дозволяє враховувати довготривалі залежності в часових рядах.

На рис. 3.28 представлено прогнозування кількості заявок за окремими спеціальностями та типом фінансування (бюджет чи контракт) із використанням алгоритму XGBoost.

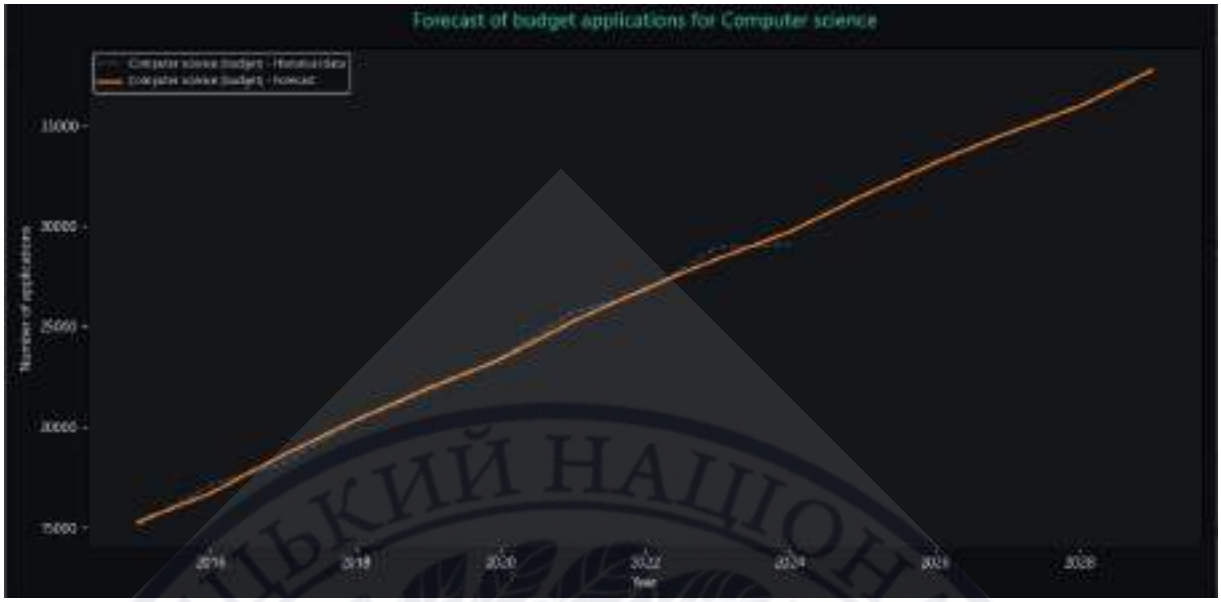

Рисунок 3.28 - Прогнозування кількості заявок XGBoost

По горизонтальній осі показано роки спостережень та прогнозу, а по вертикальній осі – кількість заявок для відповідної спеціальності та типу фінансування. Алгоритм XGBoost, як метод градієнтного бустингу, забезпечує високу точність прогнозу за рахунок послідовного навчання слабких моделей (дерев рішень) на залишках попередніх, що дозволяє ефективно враховувати складні залежності між історичними даними та кількістю заявок.

На рис. 3.29 представлено прогнозування попиту на онлайн-навчання серед студентів за допомогою ARIMA-MLPNN.

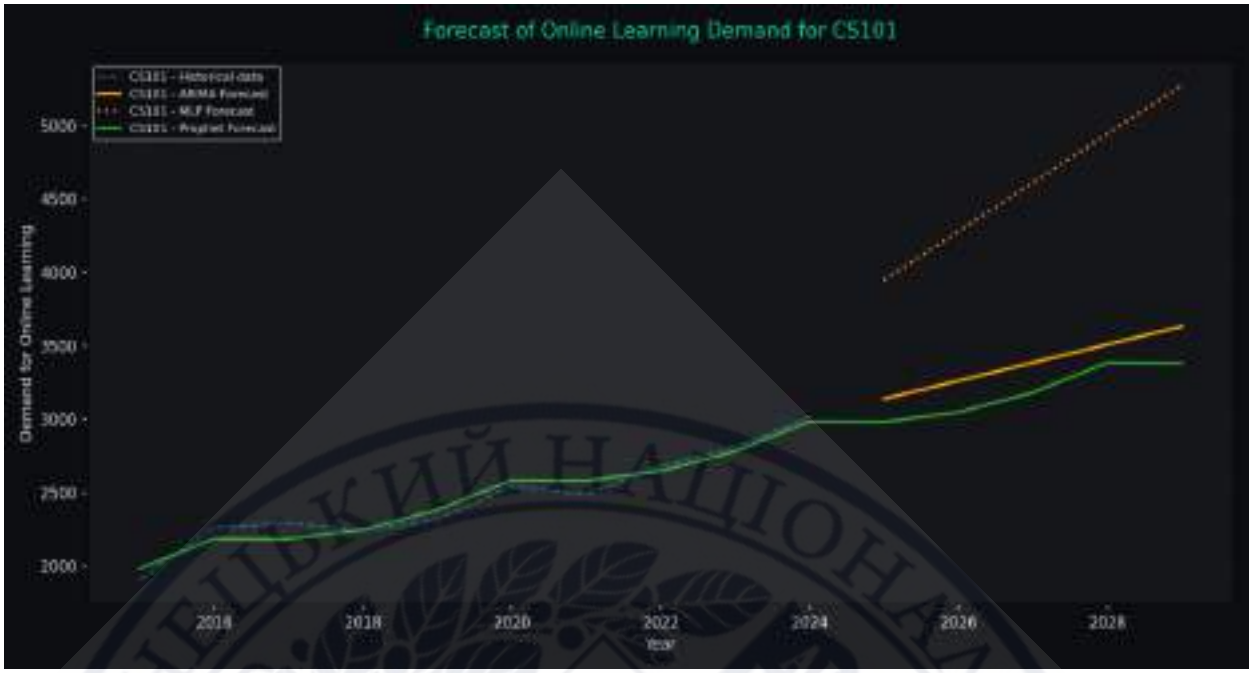

Рисунок 3.29 - Прогнозування попиту на онлайн-навчання серед студентів за допомогою ARIMA-MLPNN

Графік демонструє прогнозування попиту на онлайн-навчання серед студентів, застосовуючи гібридну модель, що поєднує ARIMA (авторегресійна інтегрована модель ковзного середнього) і нейронну мережу багатошарових перцептронів (MLPNN). ARIMA використовується для моделювання часових рядів, а MLPNN допомагає врахувати складні взаємозв'язки та нелінійні тренди. Результати показують точність прогнозування попиту на онлайн-освіту на основі історичних даних, зокрема враховуючи сезонні коливання та інші впливові фактори. На рис. 3.30 представлено прогнозування утримання студентів в університетах на основі різних факторів за допомогою ARIMA.

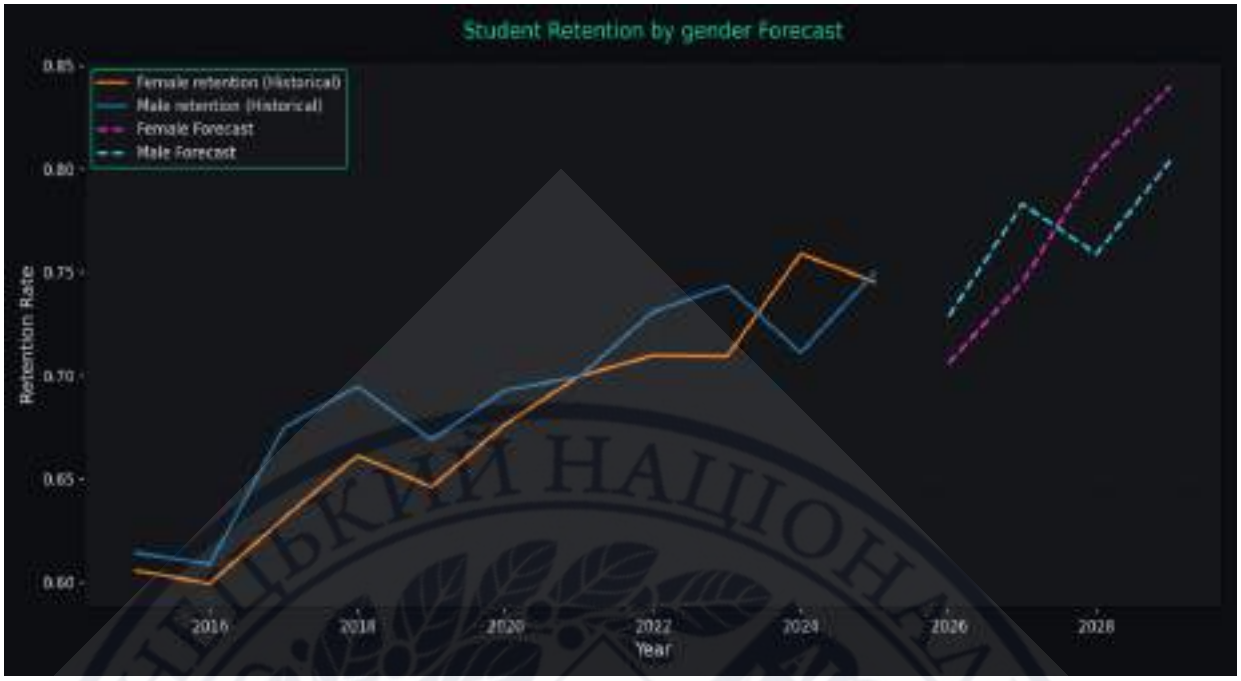

Рисунок 3.30 - Прогнозування утримання студентів з університетів на основі різних факторів за допомогою ARIMA

Графік ілюструє процес прогнозування утримання студентів в університетах із застосуванням моделі ARIMA. Модель використовує історичні дані про утримання студентів для побудови прогнозу на майбутні роки, враховуючи фактори, що можуть впливати на відтік студентів, такі як економічні умови, зміни у політиці університетів або соціокультурні аспекти. Прогнозування утримання допомагає вищим навчальним закладам передбачити тенденції й вжити заходів для покращення утримання студентів.

На рис. 3.31 представлено прогнозування прогнозування потреби в нових викладачах на основі кількості студентів, вибору курсів.

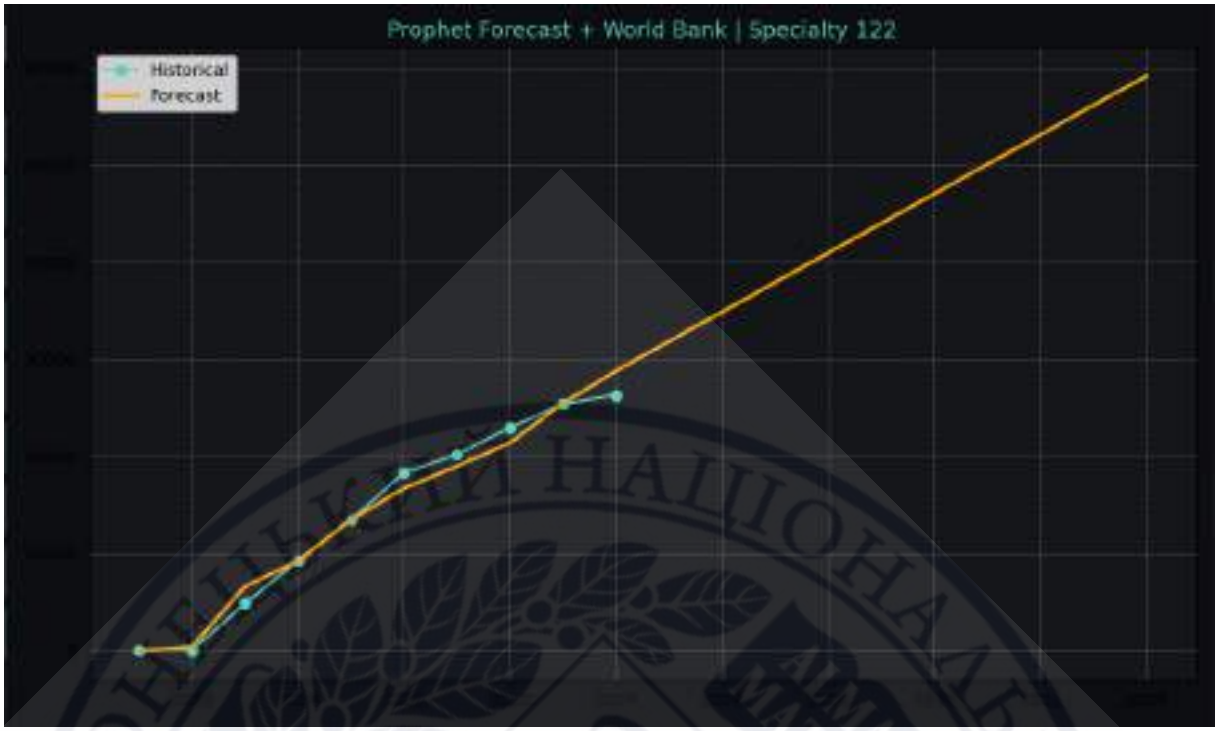

Рисунок 3.31 - Прогнозування потреби в нових викладачах на основі кількості студентів, вибору курсів LSTM

В даному випадку, модель прогнозує, скільки нових викладачів необхідно для забезпечення високої якості навчання в залежності від кількості студентів і вибору навчальних курсів. Цей підхід допомагає вищим навчальним закладам краще планувати кадрові ресурси та адаптувати свою викладацьку команду до змін у попиті на курс.

На рис. 3.32 проілюстровано як задає основні параметри прогнозу: спеціальність, рівень освіти, базу вступу, період історичних даних та горизонт прогнозування. Також він обирає макроекономічні індикатори, що можуть впливати на динаміку попиту на освіту, зокрема показники ВВП на душу населення, чисельності населення, демографічної структури, рівня урбанізації, витрат на освіту та інших соціально-економічних характеристик. Після вибору моделі прогнозування система обчислює майбутні значення, поєднуючи тенденції минулих років із трендами вибраних глобальних показників.

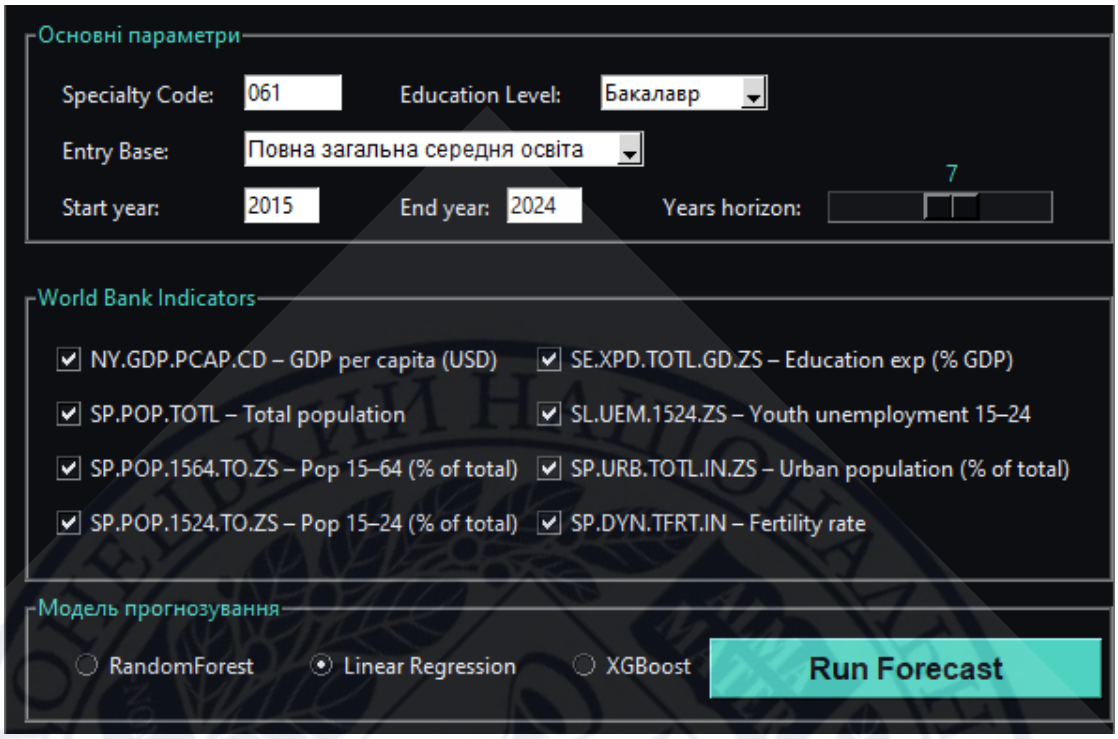

Рисунок 3.32 - Вибір параметрів для прогнозування

На рисунку 3.33 відображено прогнозування кількості студентів спеціальності 061 за період 2015–2024 років. Історична частина демонструє різкі коливання, характерні для сучасної динаміки вступу, тоді як прогноз показує очікуване поступове зростання.

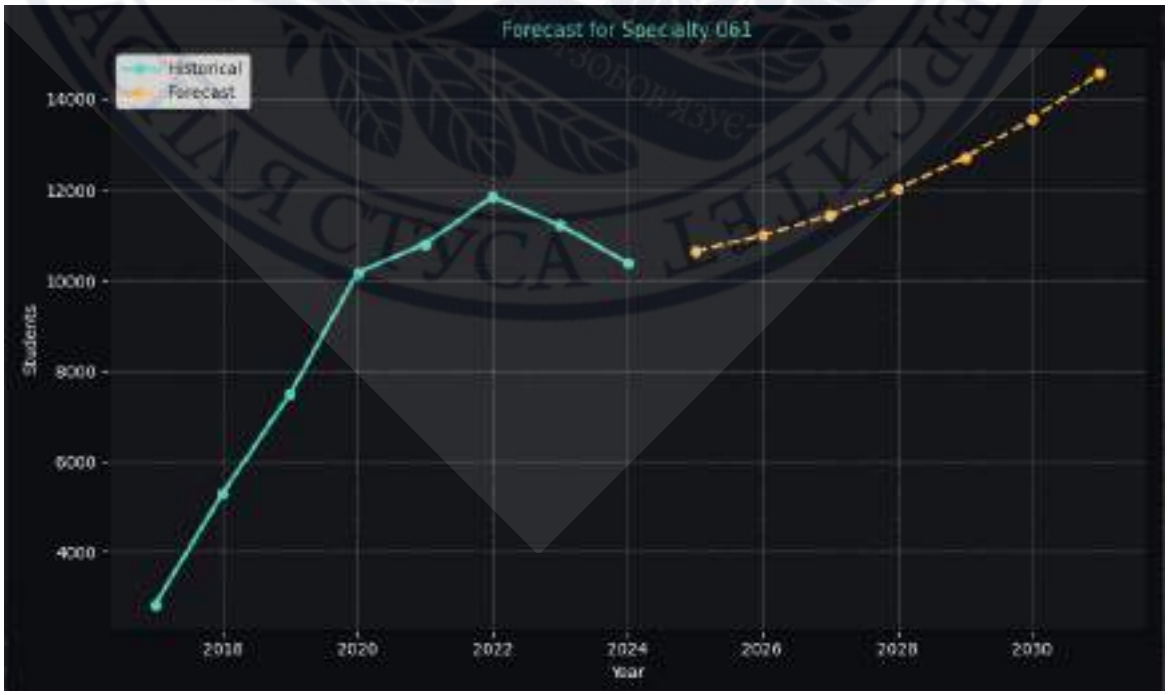

Рисунок 3.33 - Прогнозування студентів сп. 061 2015–2024 р.

На рисунку 3.34 відображено вікно вибору параметрів для прогнозування кількості вступників та випускників по даним з ЄДБО та World Bank Open Data.

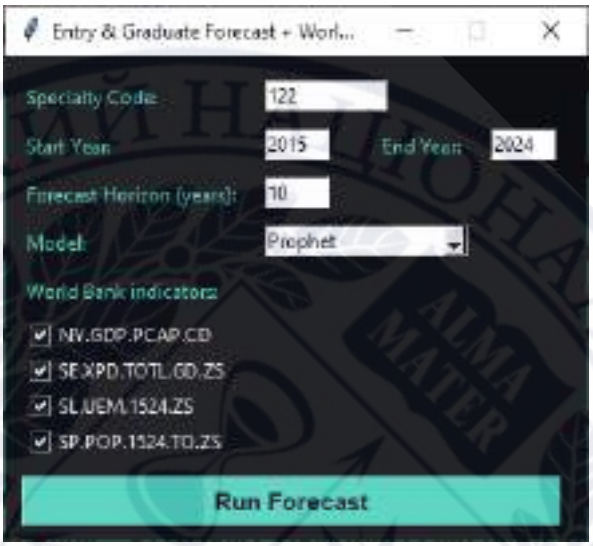

Рисунок 3.34 - Вибір параметрів прогнозування вступників та випускників

На рисунку 3.35 відображено роботу попередньої функції після вибірки параметрів: спеціальність 122, 10 років, модель Prophet. Прогнозування здійснено на основі навчальних даних ЄДЕБО та макроекономічних індикаторів World Bank.

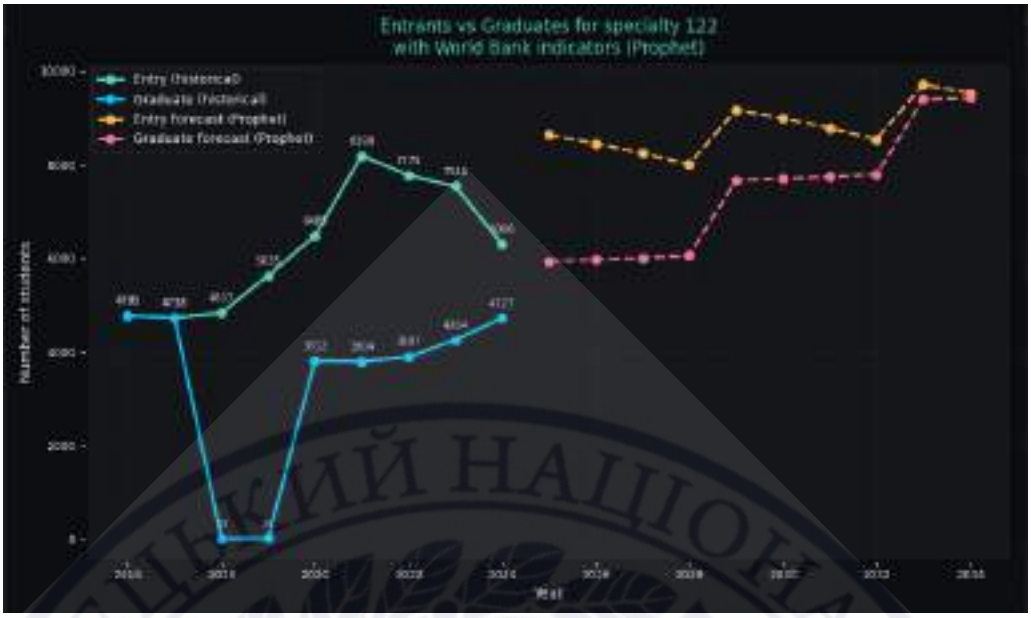

Рисунок 3.35 - Прогнозування динаміки вступників та випускників

На рис. 3.36 зображено вміст останньої вкладки "Help" - довідкового розділу додатку в якому є можливість прочитати про всі можливості платформи, описані у вигляді гортаного тексту.

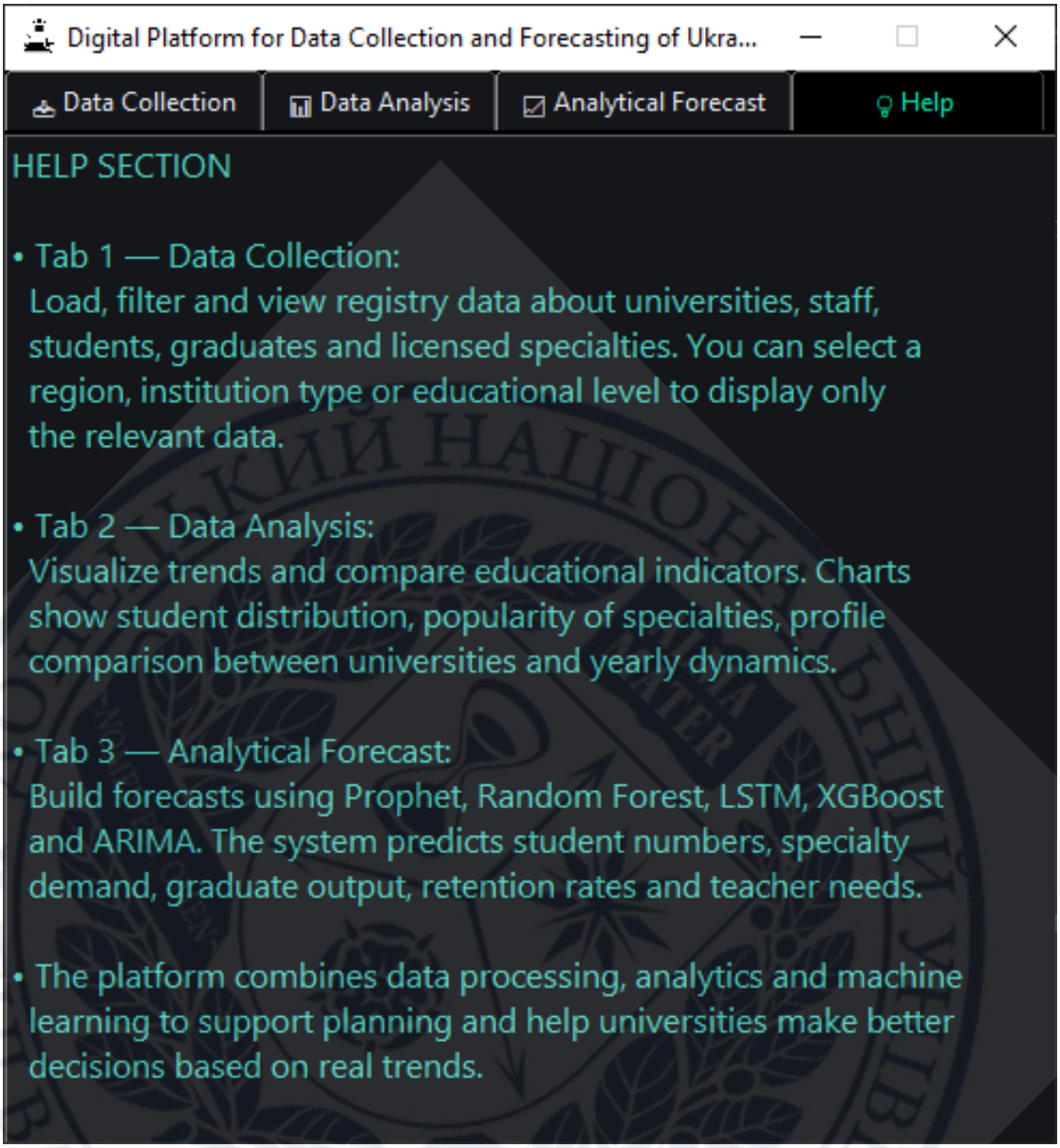

Рисунок 3.36 - Вкладка “Help”

Вкладка пояснює, що система розділена на три основні функціональні блоки:

1. Data Collection для введення та перегляду реєстрових даних про освітні установи та їхніх суб'єктів (викладачі, студенти, ліцензії).
2. Data Analysis для візуалізації та аналізу поточних трендів і статистики.
3. Analytical Forecast для формування прогнозів майбутніх тенденцій на основі зібраної інформації.

На основі проведеного тестування функціональних розділів та візуалізації даних, можна зробити такі загальні висновки щодо

працездатності та функціональності додатку. Система успішно відображає детальні реєстрові дані про освітні установи, викладачів, вступників, випускників та ліцензії після застосування фільтрів (регіон, спеціальність, рік, освітній рівень). Система генерує коректні візуалізації для аналізу трендів: порівняння кількості студентів за регіонами, порівняння профілів університетів та рейтинги найпопулярніших спеціальностей.

Система успішно використовує бібліотеку PROPheT для побудови прогнозів загальної кількості студентів та студентів за окремими категоріями (part-time, evening), відображаючи як історичні дані, так і прогнозну криву. Форми вводу даних для пошуку (за кодом спеціальності, регіоном, рівнем навчання та роком) працюють коректно, видаючи відповідні результати (Institution Data, Educators Data, Enrolled Students Data тощо). Аналітичні графіки є зрозумілими та інформативними, дозволяючи швидко оцінити тенденції (наприклад, постійне зростання популярності спеціальності 122).

Наявність розділу прогнозування (PROPheT) свідчить про використання сучасних інструментів для аналізу освітнього ринку, що є ключовою перевагою платформи. Тестування підтвердило, що додаток працює вірно, а його архітектура дозволяє ефективно збирати, аналізувати та прогнозувати дані, пов'язані з українською освітньою діяльністю.

3.5 Висновки до розділу

Розділ 3 демонструє поетапну розробку цифрової платформи для збору, обробки та прогнозування даних щодо розвитку університетів. Було проведено детальний аналіз вимог до системи, визначено цільову аудиторію та її потреби, а також окреслено функціональні та нефункціональні вимоги, що забезпечують надійну, ефективну та безпечну роботу платформи.

Проєктування архітектури з використанням UML дозволило чітко візуалізувати взаємодію користувача з системою та внутрішню структуру її компонентів, що підвищує зрозумілість і підтримуваність програмного

забезпечення. Діаграми використання, послідовності та компонентів відображають логіку роботи системи — від завантаження та обробки даних до побудови прогнозів і візуалізації результатів.

Реалізація ключових модулів підтвердила ефективність обраного підходу: модуль збору та обробки даних забезпечує інтеграцію з офіційними API, локальними файлами та відкритими джерелами, а модуль прогнозування на основі Prophet та інших алгоритмів машинного навчання таких як LSTM, Random Forest, Xgboost дозволяє створювати точні часові прогнози для загальної кількості студентів, окремих категорій та спеціальностей. Завдяки попередній підготовці, очищенню та агрегуванню даних забезпечується коректність прогнозів і наочне відображення результатів у вигляді графіків.

Функціональне тестування показало стабільну роботу платформи, коректне відображення інформації про освітні заклади, викладачів, студентів та ліцензовані спеціальності, а також зрозумілість і інформативність аналітичних та прогнозних графіків.

Таким чином, розроблена платформа успішно реалізує завдання збору, аналізу та прогнозування освітніх даних, надаючи користувачам можливість приймати обґрунтовані управлінські та стратегічні рішення на основі цифрових даних.

ВИСНОВКИ

У ході виконання магістерської роботи було проведено комплексне дослідження щодо створення цифрової платформи збору, обробки та прогнозування даних про розвиток університетів України. Результати дослідження підтвердили актуальність проблеми централізованого збирання освітніх даних, а також важливість впровадження сучасних аналітичних та прогнозних інструментів для стратегічного управління закладами вищої освіти.

У першому розділі було проаналізовано сучасний стан університетів України, визначено проблеми цифрової зрілості, наявність розрізнених інформаційних систем та відсутність уніфікованих форматів даних. Доведено необхідність створення єдиної цифрової платформи, яка забезпечує інтеграцію даних, автоматизацію процесів збору та верифікації, централізоване зберігання інформації та використання аналітичних інструментів для оцінки ефективності діяльності університетів.

У другому розділі було розглянуто основні методи прогнозування та моделі аналітики, що можуть бути застосовані для оцінки розвитку університетів. Досліджено класичні статистичні методи (регресійний аналіз, аналіз часових рядів, кластеризація), а також сучасні алгоритми машинного навчання. Встановлено, що поєднання класичних та сучасних методів дозволяє отримувати високоточні прогнози та забезпечує можливість сценарного планування розвитку освітньої системи.

У третьому розділі виконано проєктування та реалізацію цифрової платформи. Розроблено архітектуру системи з урахуванням вимог цільових користувачів - Міністерства освіти і науки, університетів та аналітичних центрів. Створено модулі збору даних із відкритих джерел (API ЄДЕБО, локальні файли та відкриті статистичні набори), модуль обробки та підготовки даних, а також модуль прогнозування на основі бібліотеки PROPheT, моделі Xgboost, рекурентної нейронної мережі LSTM та Random

Forest. Реалізовано графічний інтерфейс користувача для збору, аналізу та візуалізації даних, а також побудови прогнозів. Проведене функціональне тестування підтвердило працездатність системи, коректність відображення даних, точність прогнозів та зрозумілість аналітичних графіків.

В результаті дослідження отримано:

1. Реалізовану архітектуру та програмну платформу, яка включає модулі збору даних, їх обробки, прогнозування та візуалізації.
2. Прогнозні моделі розвитку контингенту студентів на основі історичних даних, що дозволяють оцінювати динаміку вступної кампанії, ефективність освітніх програм та потребу у викладацьких кадрах.
3. Функціональні інструменти аналітики, які дають змогу порівнювати університети за спеціальностями, регіонами та формами навчання, а також оцінювати популярність освітніх програм.
4. Практичне підтвердження ефективності платформи через тестування, що показало стабільну роботу системи та зручність користувацького інтерфейсу.

Таким чином, магістерська робота забезпечила створення ефективного інструменту стратегічного управління університетами України, який поєднує збір, обробку, аналітику та прогнозування даних, сприяє оптимізації ресурсів, підвищенню якості освітніх послуг та сталому розвитку вищої освіти на національному рівні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Колпакова О. В. Національна вища школа в Україні (1917-1921 рр.) // Україна XX ст. : Культура, ідеологія, політика : Збірник статей, вип. 2. - К., 1996.
2. Баніт, О., & Гришко, В. В. (2022). Сучасний стан розвитку університетів третього віку в Україні . Імідж сучасного педагога, (2(197), 18-22. https://doi.org/10.33272/2522-9729-2021-2(197)-18-22
3. Арешонков, В. Ю. (2020). Цифровізація вищої освіти: Виклики та відповіді. Вісник Національної академії педагогічних наук України, 2(2). https://doi.org/10.37472/2707-305X-2020-2-2-13-2
4. Офіційний сайт ЄДЕБО. URL: https://info.edbo.gov.ua
5. Вербовський І. А. Ефективність цифровізації в управлінні освітніми ресурсами: аналіз та стратегії оптимізації. Академічні візії. 2024. № 27.
6. Офіційна сторінка платформи Moodle. URL: https://moodle.org
7. Що таке CRM і як отримати максимум від її впровадження. URL: https://ce.smart-it.com/blog-post/what-is-crm/
8. Офіційна сторінка платформи Classroom. URL: https://edu.google.com/workspace-for-education/products/classroom/
9. Zavalii, T. (2025). Від традиційного до цифрового маркетингу: аномалії як індикатори парадигмального зсуву. Economic Scope (Економічний простір). https://doi.org/10.30838/EP.202.107-114
10. Rouaud, Mathieu (2013). Probability, Statistics and Estimation
11. Asteriou, Dimitros; Hall, Stephen G. (2011). "ARIMA Models and the Box-Jenkins Methodology". Applied Econometrics (Second ed.). Palgrave MacMillan. pp. 265-286. ISBN 978-0-230-27182-1.
12. Murphy, K. P. (2021). Probabilistic Machine Learning: An Introduction. MIT Press.

13. Time Series Forecasting With PROPheT in Python. URL: https://machinelearningmastery.com/time-series-forecasting-with-PROPheT-in-python/
14. Mitzias, Panagiotis & Riga, Marina & Kontopoulos, Efstratios & Stavropoulos, Thanos & Andreadis, Stelios & Meditskos, Georgios & Kompatsiaris, Ioannis. (2016). User-Driven Ontology Population from Linked Data Sources. 649. 10.1007/978-3-319-45880-9_3.
15. Random Forest. URL: https://medium.com/@denizgunay/random-forest-af5bde5d7e1e
16. Ding, Hanwei. (2024). Establishing a soil carbon flux monitoring system based on support vector machine and XGBoost. Soft Computing. 28. 1-24. 10.1007/s00500-024-09641-y.
17. Wang, Yuanchao & Pan, Z. & Zheng, J. & Qian, L. & Mingtao, Li. (2019). A hybrid ensemble method for pulsar candidate classification. Astrophysics and Space Science. 364. 10.1007/s10509-019-3602-4.
18. Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-Term Memory. Neural Computation. 9. 1735-1780. 10.1162/neco.1997.9.8.1735.
19. Mohsen, Saeed & Elkaseer, Ahmed & Scholz, Steffen. (2021). Industry 4.0-Oriented Deep Learning Models for Human Activity Recognition. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3125733.
20. Lutz, Mark (2013). Learning Python (5th ed.). O'Reilly Media. ISBN 978-0-596-15806-4.
21. Summerfield, Mark (2009). Programming in Python 3 (2nd ed.). Addison-Wesley Professional. ISBN 978-0-321-68056-3.
22. Офіційний сайт JSON. https://www.json.org/json-en.html
23. Tkinter. URL: https://wiki.python.org/moin/TkInter
24. World Bank Open Data. URL: https://data.worldbank.org/
25. Fowler, Martin (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley. ISBN 0-321-19368-7.
26. Fowler, Martin (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley. ISBN 0-321-19368-7.

ДОДАТКИ

ДОДАТОК А

```
from tkinter import *
import tkinter as tk
import webbrowser, os,time
from tkinter import PhotoImage, Tk, ttk
from InfoTabDefs.info_tabs_defs import educational_institutions_data,
graduates_information,university_information,educators_information,entrant_students_information,
licensed_specialties
from AnalizTabDefs.analiz_tabs_defs import
geographical_distribution_of_students,distribution_of_institutions_by_region,time_based_speciality
_popularity,entrant_graduate_trends,analysis_of_learning_forms,rozpodil_facultetiv,university_profi
le_comparison,popularity_of_specialties,number_of_sprecialties_per_university
from ForecastDefs.forecast_defs import forecast_faculty_demand, forecast_student_retention,
forecast_online_learning,load_student_jsons_saved,forecast_prophet_category,forecast_prophet,for
ecast_specialty_popularity_selected,forecast_prophet_category_selected,forecast_graduates,forecas
t_applications_selected,run_forecast_for_selected_category,open_applications_forecast_window,op
en_graduates_forecast_window,open_specialty_forecast_window,open_specialty_forecast,open_cat
egory_forecast_window,forecast_specialty_with_worldbank,forecast_specialty_with_worldbank_pro
phet,forecast_specialty_entry_graduate_worldbank
from AdditionalDefs.additional_defs import make_buttons

design_folder = os.path.join(os.getcwd(), 'DesignImages')
os.makedirs(design_folder, exist_ok=True)

image_paths = {'icon': os.path.join(design_folder, 'icon.ico')}

##########################################################################################
################################
# Main window

window = tk.Tk()
window.title("Digital Platform for Data Collection and Forecasting of Ukrainian Universities
Development")
window.geometry("490x500+500+300")
window.resizable(False, False)

if os.path.exists(image_paths['icon']):
   window.iconbitmap(image_paths['icon'])

# Colors
COLOR_BG = "#000000"
COLOR_PANEL = "#14161A"
```

```
COLOR_ACCENT = "#50d2c1"   # green
COLOR_ACCENT2 = "#00C2FF"
COLOR_TEXT = "#E8EAF0"
COLOR_DANGER = "#FF3B3B"

window.configure(bg=COLOR_BG)

style = ttk.Style()
style.theme_use("default")
style.configure("TNotebook", background=COLOR_BG, borderwidth=0)

style.configure(
   "TNotebook.Tab",
   background="#14161A",
   foreground="#E8EAF0",
   padding=[10, 5]
)
style.map(
   "TNotebook.Tab",
   background=[("selected", "#000000")],
   foreground=[("selected", "#00E0A6")]
)

tab_control = ttk.Notebook(window)


tab1 = ttk.Frame(tab_control, style="TFrame")
tab2 = ttk.Frame(tab_control, style="TFrame")
tab3 = ttk.Frame(tab_control, style="TFrame")
tab4 = ttk.Frame(tab_control, style="TFrame")

for tab in (tab1, tab2, tab3, tab4):
   tab.configure(style="My.TFrame")

style.configure("My.TFrame", background="#000000")


tab_control.add(tab1, text="📥 Data Collection")
tab_control.add(tab2, text="📊 Data Analysis")
tab_control.add(tab3, text="📈 Analytical Forecast")
tab_control.add(tab4, text="        💡 Help        ")
tab_control.pack(expand=1, fill='both')
################################################################################
################################
```

```
#################################################################################################################################
# Tab1 Data Collection

label_title = Label(tab1, text="REGISTER OF EDUCATIONAL\nACTIVITY ENTITIES",font=("Segoe UI", 15, "bold"), fg=COLOR_ACCENT, bg=COLOR_PANEL,pady=5, padx=5,width=38, highlightthickness=2, highlightbackground=COLOR_ACCENT)
label_title.place(x=10, y=10)

button_font = ("Segoe UI", 14, "bold")

make_buttons("EDUCATIONAL INSTITUTIONS DATA", 95, lambda: educational_institutions_data(image_paths), tk, COLOR_BG, button_font, COLOR_ACCENT, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("UNIVERSITY INFORMATION", 149, lambda: university_information(image_paths), tk, COLOR_BG, button_font, COLOR_ACCENT, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("EDUCATORS INFORMATION", 203, lambda: educators_information(image_paths), tk, COLOR_BG, button_font, COLOR_ACCENT, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("ENTRANT STUDENTS INFORMATION", 257, lambda: entrant_students_information(image_paths), tk, COLOR_BG, button_font, COLOR_ACCENT, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("GRADUATES INFORMATION", 311, lambda: graduates_information(image_paths), tk, COLOR_BG, button_font, COLOR_ACCENT, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("LICENSED SPECIALTIES", 365, lambda: licensed_specialties(image_paths), tk, COLOR_BG, button_font, COLOR_ACCENT, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("EXIT", 419, lambda: window.destroy(), tk, COLOR_BG, button_font, COLOR_DANGER, tab1, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
#################################################################################################################################
#################################


#################################################################################################################################
#################################
# Tab2 Data Analysis

button_font2 = ("Segoe UI", 14, "bold")

make_buttons("GEOGRAPHICAL DISTRIBUTION OF STUDENTS", 20, lambda: geographical_distribution_of_students(), tk, COLOR_BG, button_font2, COLOR_ACCENT, tab2, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
```

```
make_buttons("UNIVERSITY PROFILE COMPARISON", 70, lambda: university_profile_comparison(), tk,
COLOR_BG, button_font2, COLOR_ACCENT, tab2, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
make_buttons("POPULARITY OF SPECIALTIES", 120, lambda: popularity_of_specialties(), tk,
COLOR_BG, button_font2, COLOR_ACCENT, tab2, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
make_buttons("TIME-BASED SPECIALTY POPULARITY", 170, lambda:
time_based_speciality_popularity(), tk, COLOR_BG, button_font2, COLOR_ACCENT, tab2,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("NUMBER OF SPECIALTIES PER UNIVERSITY", 220, lambda:
number_of_sprecialties_per_university(), tk, COLOR_BG, button_font2, COLOR_ACCENT, tab2,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("DISTRIBUTION OF INSTITUTIONS BY REGION", 270, lambda:
distribution_of_institutions_by_region(), tk, COLOR_BG, button_font2, COLOR_ACCENT, tab2,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
make_buttons("ENROLLMENT VS GRADUATION TRENDS", 320, lambda: entrant_graduate_trends(),
tk, COLOR_BG, button_font2, COLOR_ACCENT, tab2, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
make_buttons("ANALYSIS OF LEARNING FORMS", 370, lambda: analysis_of_learning_forms(), tk,
COLOR_BG, button_font2, COLOR_ACCENT, tab2, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
make_buttons("EXIT", 420, lambda: window.destroy(), tk, COLOR_BG, button_font2,
COLOR_DANGER, tab2, COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
################################################################################################
################################


################################################################################################
################################
# Tab3 Analytical Forecast

checkpoint=0
def change_buttons_tab3():
    global checkpoint
    if checkpoint==0:
        make_buttons("FORECAST + WORLD BANK", 15, lambda: forecast_specialty_with_worldbank(),
tk,COLOR_BG, button_font, COLOR_ACCENT, tab3, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
        make_buttons("PROPHET + WORLD BANK", 66, lambda:
forecast_specialty_with_worldbank_prophet(), tk, COLOR_BG, button_font,COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("ENTRANTS & GRADUATES + WORLD BANK", 117, lambda:
forecast_specialty_entry_graduate_worldbank(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
```

```
        make_buttons("LSTM - FORECAST OF GRADUATES", 168, lambda:
open_graduates_forecast_window(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("XGBoost - APPLICATIONS FORECAST", 219, lambda:
open_applications_forecast_window(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("MLPNN - STUDENT DEMAND ONLINE LEARNING", 270, lambda:
forecast_online_learning(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3, COLOR_ACCENT2,
COLOR_DANGER, COLOR_ACCENT)
        make_buttons("ARIMA - STUDENT RETENTION FORECAST", 321, lambda:
forecast_student_retention(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3, COLOR_ACCENT2,
COLOR_DANGER, COLOR_ACCENT)
        make_buttons("LSTM - FACULTY DEMAND FORECAST", 372, lambda: open_specialty_forecast(),
tk, COLOR_BG, button_font,COLOR_ACCENT, tab3, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
        make_buttons("NEXT ANALYTICAL PAGE ↓↑", 423, lambda: change_buttons_tab3(), tk,
"#000000", button_font,"#4CFF71", tab3, "#5EE8D6", "#FF4D4D", "#4CFF71")
        checkpoint = 1

    else:
        make_buttons("PROPHET - TOTAL NUMBER OF STUDENTS", 15, lambda:
forecast_prophet(load_student_jsons_saved()), tk,COLOR_BG, button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("PROPHET - CATEGORY FORECAST", 66, lambda:
open_category_forecast_window(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("RANDOM FOREST - POPULARITY FORECAST", 117, lambda:
open_specialty_forecast_window(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("LSTM - FORECAST OF GRADUATES", 168, lambda:
open_graduates_forecast_window(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("XGBoost - APPLICATIONS FORECAST", 219, lambda:
open_applications_forecast_window(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3,
COLOR_ACCENT2, COLOR_DANGER, COLOR_ACCENT)
        make_buttons("MLPNN - STUDENT DEMAND ONLINE LEARNING", 270, lambda:
forecast_online_learning(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3, COLOR_ACCENT2,
COLOR_DANGER, COLOR_ACCENT)
        make_buttons("ARIMA - STUDENT RETENTION FORECAST", 321, lambda:
forecast_student_retention(), tk, COLOR_BG,button_font, COLOR_ACCENT, tab3, COLOR_ACCENT2,
COLOR_DANGER, COLOR_ACCENT)
        make_buttons("LSTM - FACULTY DEMAND FORECAST", 372, lambda: open_specialty_forecast(),
tk, COLOR_BG,button_font, COLOR_ACCENT, tab3, COLOR_ACCENT2, COLOR_DANGER,
COLOR_ACCENT)
        make_buttons("NEXT ANALYTICAL PAGE ↓↑", 423, lambda: change_buttons_tab3(), tk,
"#000000", button_font,"#4CFF71", tab3, "#5EE8D6", "#FF4D4D", "#4CFF71")
```

```
    checkpoint=0

change_buttons_tab3()
################################################################################################################################

################################################################################################################################
# Tab4 Help
info_textbox = Text(tab4, font=("Segoe UI", 12), height=27, width=106,fg=COLOR_ACCENT,
bg=COLOR_PANEL,highlightthickness=1,
highlightbackground=COLOR_ACCENT,insertbackground=COLOR_ACCENT, relief="flat",
wrap="word")
info_textbox.place(x=0, y=2)

info_textbox.insert(END, "HELP SECTION\n\n")

info_textbox.insert(END, "• Tab 1 — Data Collection:\n")
info_textbox.insert(END, "  Load, filter and view registry data about universities, staff,\n")
info_textbox.insert(END, "  students, graduates and licensed specialties. You can select a\n")
info_textbox.insert(END, "  region, institution type or educational level to display only\n")
info_textbox.insert(END, "  the relevant data.\n\n")

info_textbox.insert(END, "• Tab 2 — Data Analysis:\n")
info_textbox.insert(END, "  Visualize trends and compare educational indicators. Charts\n")
info_textbox.insert(END, "  show student distribution, popularity of specialties, profile\n")
info_textbox.insert(END, "  comparison between universities and yearly dynamics.\n\n")

info_textbox.insert(END, "• Tab 3 — Analytical Forecast:\n")
info_textbox.insert(END, "  Build forecasts using Prophet, Random Forest, LSTM, XGBoost\n")
info_textbox.insert(END, "  and ARIMA. The system predicts student numbers, specialty\n")
info_textbox.insert(END, "  demand, graduate output, retention rates and teacher needs.\n\n")

info_textbox.insert(END, "• The platform combines data processing, analytics and machine\n")
info_textbox.insert(END, "  learning to support planning and help universities make better\n")
info_textbox.insert(END, "  decisions based on real trends.\n")


window.mainloop()
################################################################################################################################
```

ЄДБО API requests

```
import os,json,requests
#########################################################################################
#####################################

#qf освітній ступінь
#1 – Бакалавр
#2 – Магістр
#3 – Спеціаліст
#4 – Молодший спеціаліст
#6 – Молодший бакалавр
#9 – Фаховий молодший бакалавр
#7 – Доктор філософії
#10 – Доктор мистецтва
#sp код спеціальності
#g код регіону
#id код закладу освіти
#exp тип експорту (xlsx, xml, json)

def specialnosti(qf, sp, rg, id, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{qf}_{sp}_{rg}_{id}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:
        url = f"https://registry.edbo.gov.ua/api/licenses-
specialities/?qf={qf}&sp={sp}&exp={exp}&id={id}&rg={rg}"
        response = requests.get(url)
        data = response.json()

        with open(file_path, 'w') as file:
            json.dump(data, file)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту")
        print("Спроба знайти офлайн файл...")

        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
                print("Офлайн файл знайдено.")
```

```
            return data

        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None
#specialnosti(qf="1",sp="056",rg="05",id=None,exp="json")

########################################################################################
#######################################

########################################################################################
#######################################

#ДАНІ ЗАКЛАДІВ ОСВІТИ
#ut Категорія закладу освіти
#1 – Заклади вищої освіти
#2 – Заклади професійної (професійно-технічної) освіти
#9 – Заклади фахової передвищої освіти
#8 – Наукові інститути (установи)
#10 – Заклади післядипломної освіти
#lc код регіону
#exp тип експорту (xlsx, xml, json)
def zakladi_osviti(ut, lc, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{ut}_{lc}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:

        url = f"https://registry.edbo.gov.ua/api/universities/?ut={ut}&lc={lc}&exp={exp}"
        response = requests.get(url)
        data = response.json()

        with open(file_path, 'w') as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту:", e)

        try:
```

```
            with open(file_path, 'r') as file:
                data = json.load(file)
                print("Офлайн файл знайдено.")

                return data

        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None
#zakladi_osviti(ut="1", lc="05", exp="json")

#########################################################################################
#####################################

#########################################################################################
#####################################
#ІНФОРМАЦІЯ ПРО КОНКРЕТНИЙ ЗАКЛАД ОСВІТИ
#GET /api/university/
#id код закладу освіти в ЄДЕБО (обов'язковий)
#exp тип експорту (xlsx, xml, json)
def university_info(id, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{id}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:
        url = f"https://registry.edbo.gov.ua/api/university/?id={id}&exp={exp}"
        response = requests.get(url)
        data = response.json()


        with open(file_path, 'w') as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту:", e)


        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
```

```
            print("Офлайн файл знайдено.")

            return data

        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None
#university_info(id="246", exp="json")

#########################################################################################
#####################################

#########################################################################################
#####################################

#ІНФОРМАЦІЯ ПРО ЗДОБУВАЧІВ ОСВІТИ УНІВЕРСИТЕТИ
#dt дата (обов'язковий)
#Формат: dd.mm.yyyy
#dd=01
#mm=01 (січень), 04 (квітень), 07 (липень), 10 (жовтень)
#yyyy=2016, 2017, 2018
#qf освітній ступінь (обов'язковий)
#1 – Бакалавр
#2 – Магістр
#3 – Спеціаліст
#4 – Молодший спеціаліст
#6 – Молодший бакалавр
#9 – Фаховий молодший бакалавр
#7 – Доктор філософії
#10 – Доктор мистецтва
#eb основа вступу (обов'язковий)
#25 – Без базової середньої освіти
#30 – Базова середня освіта
#40 – Повна загальна середня освіта
#510 – Кваліфікований робітник
#530 – Фаховий молодший бакалавр
#520 – Молодший спеціаліст
#610 – Молодший бакалавр
#620 – Бакалавр
#630 – Спеціаліст
#640 – Магістр
#650 – Доктор філософії
#sp код спеціальності
#rg код регіону
#id код закладу освіти
```

```
#exp тип експорту (xlsx, xml, json)
def university_educators(dt, qf, eb, sp, rg, id, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{dt}_{qf}_{eb}_{sp}_{rg}_{id}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:
        url = f"https://registry.edbo.gov.ua/api/university-
educators/?dt={dt}&qf={qf}&eb={eb}&sp={sp}&rg={rg}&id={id}&exp={exp}"
        response = requests.get(url)
        data = response.json()


        with open(file_path, 'w') as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту:", e)


        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
                print("Офлайн файл знайдено.")

                return data

        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None
#university_educators(dt="01.10.2018", qf="1", eb="40", sp="121", rg="05", id="", exp="json")

#########################################################################################
#####################################

#########################################################################################
#####################################

#ІНФОРМАЦІЯ ПРО ЗДОБУВАЧІВ ПРОФЕСІЙНО-ТЕХНІЧНОЇ ОСВІТИ
#dt дата (обов'язковий)
```

```
#Формат: dd.mm.yyyy
#dd=01
#mm=01 (січень), 04 (квітень), 07 (липень), 10 (жовтень)
#yyyy=2016, 2017, 2018
#eb база вступу
#25 – Без базової середньої освіти
#30 – Базова середня освіта
#40 – Повна загальна середня освіта
#510 – Кваліфікований робітник
#pr ідентифікатор професії
#rg код регіону
#id код закладу освіти
#exp тип експорту (xlsx, xml, json)
def university_prof_educators(dt, eb, pr, rg, id, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{dt}_{eb}_{pr}_{rg}_{id}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:
        url = f"https://registry.edbo.gov.ua/api/university-prof-
educators/?dt={dt}&eb={eb}&pr[]={pr}&rg={rg}&id={id}&exp={exp}"
        response = requests.get(url)
        data = response.json()

        with open(file_path, 'w') as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту:", e)


        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
                print("Офлайн файл знайдено.")

                return data

        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None
#university_prof_educators(dt="01.01.2023", eb="30", pr="None", rg=None, id=None, exp="json")
```

```
#########################################################################################
#####################################

#########################################################################################
#####################################

#ІНФОРМАЦІЯ ПРО ОСІБ, ЗАРАХОВАНИХ НА НАВЧАННЯ
#y рік вступу (обов'язковий) = 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023
#qf освітній ступінь (обов'язковий)
#1 – Бакалавр
#2 – Магістр
#3 – Спеціаліст
#4 – Молодший спеціаліст
#6 – Молодший бакалавр
#9 – Фаховий молодший бакалавр
#7 – Доктор філософії
#10 – Доктор мистецтва
#eb основа вступу (обов'язковий)
#25 – Без базової середньої освіти
#30 – Базова середня освіта
#40 – Повна загальна середня освіта
#510 – Кваліфікований робітник
#530 – Фаховий молодший бакалавр
#520 – Молодший спеціаліст
#610 – Молодший бакалавр
#620 – Бакалавр
#630 – Спеціаліст
#640 – Магістр
#650 – Доктор філософії
#sp код спеціальності
#rg код регіону
#id код закладу освіти
#exp тип експорту (xlsx, xml, json)
def university_entrant(y, qf, eb, sp, rg, id, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{y}_{qf}_{eb}_{sp}_{rg}_{id}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:
```

```
        url = f"https://registry.edbo.gov.ua/api/university-
entrant/?y={y}&qf={qf}&eb={eb}&sp={sp}&rg={rg}&id={id}&exp={exp}"
        response = requests.get(url)
        data = response.json()


        with open(file_path, 'w') as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту:", e)


        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
                print("Офлайн файл знайдено.")

                return data

        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None

#university_entrant(y="2020", qf="1", eb="40", sp="122", rg='', id='', exp="json")

##########################################################################################
#####################################

##########################################################################################
#####################################

#ІНФОРМАЦІЯ ПРО ОСІБ, ЯКІ ЗАКІНЧИЛИ НАВЧАННЯ
#y рік закінчення навчання (обов`язковий) = 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023
#qf освітній ступінь (обов`язковий)
#1 – Бакалавр
#2 – Магістр
#3 – Спеціаліст
#4 – Молодший спеціаліст
#6 – Молодший бакалавр
#9 – Фаховий молодший бакалавр
#7 – Доктор філософії
```

```
#10 – Доктор мистецтва
#eb основа вступу (обов`язковий)
#25 – Без базової середньої освіти
#30 – Базова середня освіта
#40 – Повна загальна середня освіта
#510 – Кваліфікований робітник
#530 – Фаховий молодший бакалавр
#520 – Молодший спеціаліст
#610 – Молодший бакалавр
#620 – Бакалавр
#630 – Спеціаліст
#640 – Магістр
#650 – Доктор філософії
#sp код спеціальності
#rg код регіону
#id код закладу освіти
#exp тип експорту (xlsx, xml, json)
def university_graduate(y, qf, eb, sp, rg, id, exp):
    folder_path = os.path.join(os.getcwd(), 'saved_data')
    os.makedirs(folder_path, exist_ok=True)

    file_name = f"{y}_{qf}_{eb}_{sp}_{rg}_{id}_{exp}.json"
    file_path = os.path.join(folder_path, file_name)

    try:
        url = f"https://registry.edbo.gov.ua/api/university-
graduate/?y={y}&qf={qf}&eb={eb}&sp={sp}&rg={rg}&id={id}&exp={exp}"
        response = requests.get(url)
        data = response.json()

        with open(file_path, 'w') as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

        return data

    except requests.RequestException as e:
        print("Помилка під час виконання запиту:", e)


        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
                print("Офлайн файл знайдено.")

                return data
```

```
        except FileNotFoundError:
            print("Офлайн файл не знайдено.")
            return None

#university_graduate(y="2021", qf="1", eb="40", sp="122", rg="", id="", exp="json")

#########################################################################################
#####################################

#Some data defs
import os
import tkinter as tk
from tkinter import PhotoImage, Tk, ttk
from GetUniversityData.GetUniversityData import specialnosti, zakladi_osviti, university_info,
university_educators, university_prof_educators, university_entrant, university_graduate

from tkinter import *
import tkinter as tk
import os
from tkinter import ttk


def display_data_window(title, data, image_paths,is_list=False):
    try:
        window3 = tk.Toplevel()
        window3.title(title)
        window3.geometry("600x400+400+200")
        window3.resizable(False, False)
        window3.configure(bg="#2E2E2E")
        if os.path.exists(image_paths['icon']):
            window3.iconbitmap(image_paths['icon'])

        text_area = Text(window3, font=("Arial", 10), height=21, width=78, fg="#FFFFFF", bg="#2E2E2E",
                         highlightthickness=2, highlightbackground="#26A69A")
        text_area.place(x=10, y=10)
        scrollbar = Scrollbar(window3, orient="vertical", command=text_area.yview)
        scrollbar.place(x=570, y=10, height=343)
        text_area.config(yscrollcommand=scrollbar.set)

        if is_list:
            for item in data:
                for label, value in item:
                    if label ==
"=================================================================":
                        text_area.insert(tk.END, f"{label}\n", "separator")
                    else:
```

```
                text_area.insert(tk.END, f"{label:<20} {value}\n" if value else f"{label}\n")
        else:
            info = [
                ("Institution Name:", data['university_name']),
                ("Short Name:", data['university_short_name']),
                ("Institution ID:", data['university_id']),
                ("Type:", data['university_type_name']),
                ("Financing Type:", data['university_financing_type_name']),
                ("Address:", data['university_address_u']),
                ("Phone:", data['university_phone']),
                ("Email:", data['university_email']),
                ("Website:", data['university_site']),
                ("Director Position:", data['university_director_post']),
                ("Director Name:", data['university_director_fio']),
                ("========================================================================", ""),
                ("", "")
            ]
            for label, value in info:
                if label ==
"========================================================================":
                    text_area.insert(tk.END, f"{label}\n", "separator")
                else:
                    text_area.insert(tk.END, f"{label:<20} {value}\n" if value else f"{label}\n")

            if 'branches' in data:
                text_area.insert(tk.END, "Branches:\n")
                for branch in data['branches']:
                    text_area.insert(tk.END, f"  - {branch['university_name']} ({branch['region_name']})\n")

            if 'facultets' in data:
                text_area.insert(tk.END, "Faculties:\n")
                for faculty in data['facultets']:
                    text_area.insert(tk.END, f"  - {faculty}\n")

            if 'speciality_licenses' in data:
                text_area.insert(tk.END, "Specialties and Licenses:\n")
                for license in data['speciality_licenses']:
                    text_area.insert(tk.END, f"  Qualification: {license['qualification_group_name']}\n")
                    text_area.insert(tk.END, f"  Specialty Code: {license['speciality_code']}\n")
                    text_area.insert(tk.END, f"  Specialty Name: {license['speciality_name']}\n")
                    text_area.insert(tk.END, f"  Number of Places: {license['all_count']}\n")
                    text_area.insert(tk.END, f"  License Expiry: {license['certificate_expired']}\n")
                    text_area.insert(tk.END, f"  License: {license['license_description']}\n")
                    text_area.insert(tk.END, "\n")

        text_area.tag_configure("separator", foreground="#26A69A")
```

```
        tk.Button(window3, text="Close", width=20, fg="#FFFFFF", bg="#50d2c1", font=("Arial", 10,
"bold"),
                borderwidth=0, relief="solid", highlightthickness=2, highlightbackground="#000000",
                command=window3.destroy).place(x=420, y=360)

    except Exception as e:
        # Error window
        window3 = tk.Toplevel()
        window3.title("Error")
        window3.geometry("300x100+500+300")
        window3.resizable(False, False)
        window3.configure(bg="#2E2E2E")
        tk.Label(window3, text="Data not found", fg="#26A69A", bg="#2E2E2E", font=("Arial", 10,
"bold")).place(x=10,
                                                                    y=20)
        tk.Button(window3, text="Close", width=20, fg="#FFFFFF", bg="#50d2c1", font=("Arial", 10,
"bold"),
                borderwidth=0, relief="solid", highlightthickness=2, highlightbackground="#000000",
                command=window3.destroy).place(x=190, y=60)


def educational_institutions_data(image_paths):
    window2 = tk.Toplevel()
    window2.title("Institution Information")
    window2.geometry("340x110+500+300")
    window2.resizable(False, False)
    window2.configure(bg="#2E2E2E")
    if os.path.exists(image_paths['icon']):
        window2.iconbitmap(image_paths['icon'])
    else:
        print(f"File {image_paths['icon']} not found.")

    canvas2 = tk.Canvas(window2, width=450, height=110, bg="#2E2E2E", highlightthickness=0)
    canvas2.place(x=0, y=0)

    categories = ["Higher Education", "Vocational Education", "Pre-Higher Vocational Education",
            "Research Institutes", "Postgraduate Education"]
    tk.Label(canvas2, text="Category:", fg="#26A69A", bg="#2E2E2E", font=("Arial", 10,
"bold")).place(x=10, y=15)
    category_combobox = ttk.Combobox(canvas2, values=categories, width=30, font=("Arial", 10))
    category_combobox.set(categories[1])
    category_combobox.place(x=100, y=15)

    regions = ["Vinnytsia Oblast", "Volyn Oblast", "Dnipropetrovsk Oblast", "Donetsk Oblast",
          "Zhytomyr Oblast", "Zakarpattia Oblast", "Zaporizhzhia Oblast", "Ivano-Frankivsk Oblast",
```

```
            "Kyiv Oblast", "Kirovohrad Oblast", "Luhansk Oblast", "Lviv Oblast",
            "Mykolaiv Oblast", "Odesa Oblast", "Poltava Oblast", "Rivne Oblast",
            "Sumy Oblast", "Ternopil Oblast", "Kharkiv Oblast", "Kherson Oblast",
            "Khmelnytskyi Oblast", "Cherkasy Oblast", "Chernivtsi Oblast", "Chernihiv Oblast",
            "Kyiv City", "Sevastopol City"]
    tk.Label(canvas2, text="Region:", fg="#26A69A", bg="#2E2E2E", font=("Arial", 10,
"bold")).place(x=10, y=45)
    regions_combobox = ttk.Combobox(canvas2, values=regions, width=30, font=("Arial", 10))
    regions_combobox.set("Kyiv City")
    regions_combobox.place(x=100, y=45)

    region_codes = {
        "Autonomous Republic of Crimea": "01", "Vinnytsia Oblast": "05", "Volyn Oblast": "07",
        "Dnipropetrovsk Oblast": "12", "Donetsk Oblast": "14", "Zhytomyr Oblast": "18",
        "Zakarpattia Oblast": "21", "Zaporizhzhia Oblast": "23", "Ivano-Frankivsk Oblast": "26",
        "Kyiv Oblast": "32", "Kirovohrad Oblast": "35", "Luhansk Oblast": "44",
        "Lviv Oblast": "46", "Mykolaiv Oblast": "48", "Odesa Oblast": "51",
        "Poltava Oblast": "53", "Rivne Oblast": "56", "Sumy Oblast": "59",
        "Ternopil Oblast": "61", "Kharkiv Oblast": "63", "Kherson Oblast": "65",
        "Khmelnytskyi Oblast": "68", "Cherkasy Oblast": "71", "Chernivtsi Oblast": "73",
        "Chernihiv Oblast": "74", "Kyiv City": "80", "Sevastopol City": "85"
    }

    def zakladi_osviti3():
        try:
            ut_map = {"Higher Education": "1", "Vocational Education": "2", "Pre-Higher Vocational
Education": "9",
                      "Research Institutes": "8", "Postgraduate Education": "10"}
            ut = ut_map.get(category_combobox.get(), "2")
            g = region_codes.get(regions_combobox.get(), "05")
            data = zakladi_osviti(ut=ut, lc=g, exp="json")

            window3 = tk.Toplevel()
            window3.title("Institution Data")
            window3.geometry("600x400+400+200")
            window3.resizable(False, False)
            window3.configure(bg="#2E2E2E")
            if os.path.exists(image_paths['icon']):
                window3.iconbitmap(image_paths['icon'])

            text_area = Text(window3, font=("Arial", 10), height=21, width=78, fg="#FFFFFF",
bg="#2E2E2E",
                             highlightthickness=2, highlightbackground="#26A69A")
            text_area.place(x=10, y=10)
            scrollbar = Scrollbar(window3, orient="vertical", command=text_area.yview)
            scrollbar.place(x=570, y=10, height=343)
```

```
        text_area.config(yscrollcommand=scrollbar.set)

        for item in data:
            info = [
                ("Institution Name:", item['university_name']),
                ("Short Name:", item['university_short_name']),
                ("Institution ID:", item['university_id']),
                ("Type:", item['university_type_name']),
                ("Financing Type:", item['university_financing_type_name']),
                ("Address:", item['university_address_u']),
                ("Phone:", item['university_phone']),
                ("Email:", item['university_email']),
                ("Website:", item['university_site']),
                ("Director Position:", item['university_director_post']),
                ("Director Name:", item['university_director_fio']),
                ("=============================================================================",
""),
                ("", "")
            ]
            for label, value in info:
                if label ==
"=============================================================================":
                    text_area.insert(tk.END, f"{label}\n", "separator")
                else:
                    text_area.insert(tk.END, f"{label:<20} {value}\n" if value else f"{label}\n")

        text_area.tag_configure("separator", foreground="#26A69A")

        tk.Button(window3, text="Close", width=20, fg="#FFFFFF", bg="#50d2c1", font=("Arial", 10,
"bold"),
                  borderwidth=0, relief="solid", highlightthickness=2, highlightbackground="#000000",
                  command=window3.destroy).place(x=420, y=360)

    except Exception as e:
        window3 = tk.Toplevel()
        window3.title("Error")
        window3.geometry("300x100+500+300")
        window3.resizable(False, False)
        window3.configure(bg="#2E2E2E")
        tk.Label(window3, text="Data not found", fg="#26A69A", bg="#2E2E2E", font=("Arial", 10,
"bold")).place(x=10,
                                                                    y=20)
        tk.Button(window3, text="Close", width=20, fg="#FFFFFF", bg="#50d2c1", font=("Arial", 10,
"bold"),
                  borderwidth=0, relief="solid", highlightthickness=2, highlightbackground="#000000",
                  command=window3.destroy).place(x=190, y=60)
```

```
    window2.destroy()

  tk.Button(canvas2, text="Search", width=39, fg="#FFFFFF", bg="#50d2c1", font=("Arial", 10,
"bold"),
        borderwidth=0, relief="solid", highlightthickness=2, highlightbackground="#000000",
        command=zakladi_osviti3).place(x=10, y=75)



#Some Analytic defs

import os, json, threading
from datetime import datetime
from collections import defaultdict
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import ttk, messagebox
from prophet import Prophet
from pmdarima import auto_arima
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from statsmodels.tsa.arima.model import ARIMA
from GetUniversityData.GetUniversityData import specialnosti, zakladi_osviti,
university_info,university_educators, university_prof_educators,university_entrant,
university_graduate

def geographical_distribution_of_students():
  window2 = tk.Toplevel()
  window2.title("Geographical Distribution of Students")
  window2.geometry("268x170+500+300")
  window2.config(bg="#0D0E11")
  window2.resizable(False, False)

  label_color = "#50d2c1"
  tk.Label(window2, text="Date:", fg=label_color, bg="#0D0E11").place(x=10, y=10)
  entry_data = tk.Entry(window2, width=25)
  entry_data.place(x=100, y=10)
  entry_data.insert(0, "01.01.2024")

  tk.Label(window2, text="Education Level:", fg=label_color, bg="#0D0E11").place(x=10, y=40)
  education_levels = ["Бакалавр", "Магістр", "Спеціаліст", "Молодший спеціаліст",
            "Молодший бакалавр", "Фаховий молодший бакалавр",
            "Доктор філософії", "Доктор мистецтва"]
  education_box = ttk.Combobox(window2, values=education_levels, width=22)
```

```
education_box.set("Бакалавр")
education_box.place(x=100, y=40)

tk.Label(window2, text="Entry Base:", fg=label_color, bg="#0D0E11").place(x=10, y=70)
entry_bases = ["Без базової середньої освіти", "Базова середня освіта",
        "Повна загальна середня освіта", "Кваліфікований робітник",
        "Фаховий молодший бакалавр", "Молодший спеціаліст",
        "Молодший бакалавр", "Бакалавр", "Спеціаліст", "Магістр", "Доктор філософії"]
entry_box = ttk.Combobox(window2, values=entry_bases, width=22)
entry_box.set("Повна загальна середня освіта")
entry_box.place(x=100, y=70)

tk.Label(window2, text="Specialty Code:", fg=label_color, bg="#0D0E11").place(x=10, y=100)
entry_specialty = tk.Entry(window2, width=25)
entry_specialty.place(x=100, y=100)
entry_specialty.insert(0, "122")

def university_educators3():
  try:
    dt = entry_data.get()
    sp = entry_specialty.get()
    rg, id = "", ""

    qf_map = {
      "Бакалавр": 1, "Магістр": 2, "Спеціаліст": 3, "Молодший спеціаліст": 4,
      "Молодший бакалавр": 6, "Фаховий молодший бакалавр": 9,
      "Доктор філософії": 7, "Доктор мистецтва": 10
    }
    qf = qf_map.get(education_box.get(), 1)

    eb_map = {
      "Без базової середньої освіти": 25, "Базова середня освіта": 30,
      "Повна загальна середня освіта": 40, "Кваліфікований робітник": 510,
      "Фаховий молодший бакалавр": 530, "Молодший спеціаліст": 520,
      "Молодший бакалавр": 610, "Бакалавр": 620,
      "Спеціаліст": 630, "Магістр": 640, "Доктор філософії": 650
    }
    eb = eb_map.get(entry_box.get(), 40)

    data = university_educators(dt=dt, qf=qf, eb=eb, sp=sp, rg=rg, id=id, exp="json")

    regions = {}
    for item in data:
      region_name = item['region_name']
      budget = (int(item['full_time_budget']) +
            int(item['part_time_budget']) +
```

```
                int(item['evening_budget']))
            contract = (int(item['full_time_contract']) +
                        int(item['part_time_contract']) +
                        int(item['evening_contract']))
            if region_name not in regions:
                regions[region_name] = [0, 0]
            regions[region_name][0] += budget
            regions[region_name][1] += contract

        region_names = list(regions.keys())
        budget_students = [v[0] for v in regions.values()]
        contract_students = [v[1] for v in regions.values()]

        plt.close('all')
        fig, ax = plt.subplots(figsize=(10, 7))
        bar_height = 0.35
        y_pos = range(len(region_names))

        ax.barh(y_pos, budget_students, bar_height, label='Budget',
                color="#50d2c1", edgecolor='white', linewidth=1.3)
        ax.barh([y + bar_height for y in y_pos], contract_students, bar_height,
                label='Contract', color="#00C2FF", edgecolor='white', linewidth=1.3)

        ax.set_facecolor("#14161A")
        fig.patch.set_facecolor("#0D0E11")
        ax.set_xlabel("Number of Students", color="#E8EAF0")
        ax.set_ylabel("Regions", color="#E8EAF0")
        ax.set_title(f"Students by Region — Specialty {sp}", color="#50d2c1")
        ax.set_yticks([y + bar_height / 2 for y in y_pos])
        ax.set_yticklabels(region_names, color="#E8EAF0", fontsize=8)
        ax.tick_params(axis='x', colors="#E8EAF0")
        ax.legend(facecolor="#14161A", edgecolor="#50d2c1", labelcolor="#E8EAF0")

        for y, b, c in zip(y_pos, budget_students, contract_students):
            ax.text(b + 5, y - 0.1, str(b), color="#FFFFFF", fontsize=6)
            ax.text(c + 5, y + bar_height - 0.1, str(c), color="#FFFFFF", fontsize=6)

        plt.tight_layout()

        os.makedirs("saved_diagrams", exist_ok=True)
        path = os.path.join("saved_diagrams", "students_per_region.png")
        plt.savefig(path, facecolor=fig.get_facecolor(), dpi=200)

        plt.show()

    except Exception as e:
```

```
            print("Error:", e)

    tk.Button(window2, text="Show", width=30, bg="#50d2c1", fg="#0D0E11",
              font=("Arial", 10, "bold"), command=university_educators3).place(x=10, y=130)

    window2.mainloop()



def distribution_of_institutions_by_region():
    window2 = tk.Toplevel()
    window2.title("Distribution of Institutions by Region")
    window2.geometry("268x105+500+300")
    window2.config(bg="#0D0E11")
    window2.resizable(False, False)

    label_color = "#50d2c1"

    tk.Label(window2, text="Education Level:", fg=label_color, bg="#0D0E11").place(x=5, y=5)
    education_levels = ["Bachelor", "Master", "Specialist", "Junior Specialist", "Junior Bachelor",
                        "Professional Junior Bachelor", "Doctor of Philosophy", "Doctor of Arts"]
    education_box = ttk.Combobox(window2, values=education_levels, width=20)
    education_box.set("Bachelor")
    education_box.place(x=120, y=5)

    tk.Label(window2, text="Specialty Code:", fg=label_color, bg="#0D0E11").place(x=5, y=35)
    entry_code = tk.Entry(window2, width=22)
    entry_code.place(x=120, y=35)
    entry_code.insert(0, "122")

    qf_map = {
        "Bachelor": "1", "Master": "2", "Specialist": "3", "Junior Specialist": "4",
        "Junior Bachelor": "6", "Professional Junior Bachelor": "9",
        "Doctor of Philosophy": "7", "Doctor of Arts": "10"
    }

    def show_distribution():
        try:
            qf = qf_map.get(education_box.get(), "1")
            sp = entry_code.get()

            data = specialnosti(qf=qf, sp=sp, rg="", id="", exp="json")

            region_counts = {}
            for item in data:
                region = item['region_name']
```

```
            region_counts[region] = region_counts.get(region, 0) + 1

        if not region_counts:
            raise ValueError("No data found")

        regions = list(region_counts.keys())
        counts = list(region_counts.values())

        plt.close('all')
        fig, ax = plt.subplots(figsize=(10, 7))
        ax.barh(regions, counts, color="#50d2c1", edgecolor='white', linewidth=1.3)

        ax.set_facecolor("#14161A")
        fig.patch.set_facecolor("#0D0E11")
        ax.set_xlabel("Number of Institutions", color="#E8EAF0")
        ax.set_ylabel("Regions", color="#E8EAF0")
        ax.set_title(f"Distribution of Institutions with Specialty {sp}", color="#50d2c1")
        ax.tick_params(axis='x', colors="#E8EAF0")
        ax.tick_params(axis='y', colors="#E8EAF0", labelsize=8)
        ax.grid(axis="x", linestyle="--", alpha=0.5, color="#333333")

        for i, count in enumerate(counts):
            ax.text(count + max(counts) * 0.01, i, str(count),
                    va="center", color="#FFFFFF", fontsize=6)

        plt.tight_layout()

        os.makedirs("saved_diagrams", exist_ok=True)
        path = os.path.join("saved_diagrams", "specialty_distribution.png")
        plt.savefig(path, facecolor=fig.get_facecolor(), dpi=200)

        plt.show()

        window2.destroy()
    except Exception as e:
        messagebox.showerror("Error", f"Data not found or invalid.\n{e}")

tk.Button(window2, text="Show", width=30, bg="#50d2c1", fg="#0D0E11",
          font=("Arial", 10, "bold"), command=show_distribution).place(x=10, y=65)

window2.mainloop()
```

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТУСА
ALMA MATER
NOMEN EST OMEN
ІМ'Я ЗОБОВ'ЯЗУЄ