МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Модель аналізу рейтингу Сумського державного університету з використанням алгоритмів машинного навчання»

за напрямом підготовки 6.050101 «Комп'ютерні науки»

Виконавець роботи: студент групи ІТ-52 Макаренко Дмитро Валерійович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2019 р.

Науковий керівник _____ к.т.н., доц., Гайдабрус Б.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____В. В. Шендрик
«____» _____ 2019 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Макаренко Дмитро Валерійович

1 Тема роботи *Модель аналізу рейтингу Сумського державного університету з використанням алгоритмів машинного навчання.*

керівник роботи *Гайдабрус Б.В., к.т.н., доцент*,

затверджені наказом по університету від « 17 » травня 2019 р. №_____

2 Строк подання студентом роботи «3» червня 2019 р.

3 Вхідні дані до роботи Методика визначення рейтингу інститутів, факультетів та кафедр Сумського державного університету (версія 012). Контрольні показники(узагальнена форма) факультетів.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *Аналіз предметної області, Постановка задачі та дослідження алгоритмів, Моделювання, Проектування моделі аналізу, Висновок.*

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *презентація 21 слайд.*

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	**Завдання прийняв**
Аналіз предметної області	*Гайдабрус Б.В.*		
Постановка задачі та дослідження алгоритмів	*Гайдабрус Б.В*		
Моделювання	*Гайдабрус Б.В.*		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної галузі	03.09.2018	14.09.2018
3	Обробка даних	15.09.2018	19.10.2018
4	Нормалізація даних	20.10.2018	04.11.2018
5	Дослідження моделей	05.11.2018	26.12.2018
6	Аналіз моделей	27.12.2018	30.01.2019
7	Побудова моделі ДР	01.02.2019	07.03.2019
8	Побудова моделі ВЛ	08.03.2019	03.04.2019
9	Побудова моделі ГБ	04.04.2019	30.04.2019
10	Тестування моделей	01.05.2019	15.05.2019
11	Визначення важливості показників	16.05.2019	29.05.2019

Студент _____ Макаренко Д.В.
(підпис)

Керівник роботи _____ к.т.н., доц. Гайдабрус Б.В.
(підпис)

РЕФЕРАТ

Тема бакалаврської роботи «Модель аналізу рейтингу Сумського державного університету з використанням алгоритмів машинного навчання».

Пояснювальна записка містить вступ, 4 розділи, висновок та список використаних джерел, включає 63 сторінок, 3 таблиці, 48 ілюстрацій, 21 джерело.

У першому розділі наведений огляд методики визначення рейтингу СумДУ, машинного навчання, популярні моделі машинного навчання.

Другий розділ включає в себе формування мети, задач, вибір засобів реалізації та вибір інструментів реалізації. Досліджуються моделі для даної роботи.

У третьому розділі побудовані структурно-функціональні моделі методики рейтингу та модель аналізу рейтингу. Описується, як визначаються показники у методиці та як визначається важливість показників у моделі аналізу рейтингу. Також побудована діаграма класів, де описується зв’язок класів.

Четверний розділ описує розробку моделі аналізу рейтингу СумДУ. Детально наводяться етапи створення моделей, тестування та визначення важливості показників.

Результатом проведеної роботи є розроблена модель аналізу рейтингу СумДУ з використанням алгоритмів машинного навчання та визначена важливість показників найточнішої моделі.

Ключові слова: Сумський Державний Університет СумДУ, машинне навчання, моделі навчання, модель, дерево рішень, випадковий ліс, градієнтний бустингу, прогнозування.

Зміст

ВСТУП

На базі Сумського державного університету (СумДУ) створена методика для визначення рейтингу інститутів, факультетів та кафедр СумДУ. Методика передбачає визначення за підсумками календарного року рейтингу структурних підрозділів і реалізується шляхом комп'ютерної обробки статистичної інформації, яка складена відповідними підрозділами – надавачами інформації, у тому числі із врахуванням даних річних звітів інститутів, факультетів, кафедр, викладачів, наявності підтверджуючих документів.

У загальний рейтинг входить більше 270 показників, кожен із яких вираховується власною формулою. Розрахунки, за якими вираховуються показники можуть змінюватися протягом років, тому складно сказати, які саме показники більше впливають на фінальний рейтинг інститутів, факультетів та кафедр. Розрахунок кожного показнику та визначення важливості займає багато часу, тому запропонована модель на основі машинного навчання.

Актуальність використання даного підходу, обумовлено тим, що показники рейтингу та статистична інформація оброблюється на основі аналізу даних, встановлюються закономірності та приймаються рішення з мінімальним втручанням людського фактору.

У результаті дослідження було проаналізовано більшість моделей машинного навчання та створено моделі на алгоритмах машинного навчання, а саме такі:

- дерево рішень;
- випадковий ліс;
- градієнтний бустингу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методика визначення рейтингу

Методика розроблена з метою оцінки потенціалу, рівня роботи структурних підрозділів (інститутів, факультетів, кафедр) та їх стимулювання на підвищення якості роботи за показниками, які безпосередньо визначають ефективність діяльності, а також за тими, які є суттєво впливовими і які застосовуються у глобальних міжнародних та вітчизняних рейтингах та базах даних, при визначенні статусів дослідницького, національного університету, при проведенні акредитації, ліцензування напрямів підготовки, спеціальностей, у річних звітах університету (узагальненому та за напрямами діяльності), що надаються до МОН України.

Відповідно до методики показники потенціалу діяльності ***i*** **– структурного підрозділу** ($П_{ji}$) оцінюються кількістю рейтингових балів (P_{ji}).

Визначення рейтингу проводиться за групами показників – нижченаведених індикаторів ефективності діяльності ($I_j=\sum P_{ji}$):

- I_1 - Науково-педагогічний потенціал;
- I_2 - Диверсифікація рівнів та форм навчання, контингент осіб, що навчається; **показники випуску[2)]**;
- I_3 - Якість навчально-наукової роботи зі студентами та додаткові освітні послуги;
- I_4 - Якість міжнародної діяльності;
- I_5 - Рівень оприлюднення результатів наукової та науково-методичної діяльності;
- **I_6 - Якість підготовки науково-педагогічних кадрів;**
- I_7 - Якість представлення результатів діяльності в Інтернет та медіа - просторі;

- I_8 - Фінансова оцінка результатів інноваційної діяльності;
- **I_9 - Якість позанавчальної діяльності інституту (факультету).**

Методика передбачає можливість входження показника до різних індикаторів з однаковим або різним рейтинговим оцінюванням. Рейтингові визначення (P_{ji}) показників ($П_{ji}$) індикатора (I_j) об'єктивно співвідносяться при їх порівнянні за рахунок введення відповідних коефіцієнтів. Ректоратом можуть призначатися преміальні рейтингові бали за особливо вагомі досягнення, що зазначається у відповідному поданні, у тому числі і за ті, які не враховані показниками методики.

За значенням розрахункових індикаторів I_{ji}, для кожного структурного підрозділу розраховуються індикатори $I_{ji}^{пр}$, як приведені до 100-бальної шкали (за $I_{ji\ max}^{пр}$=100 балів приймається найвище значення I_{ji}; приведені індикатори інших структурних підрозділів вираховуються як відсоток від $I_{ji\ max}$). За кожним індикатором визначається розрахункове рейтингове місце (N_{ji}) **та рейтинговий рівень ($РР_{ji}$)** структурного підрозділу (форма 3). При цьому, на всіх етапах розрахункового ранжування при однакових значеннях $I_{ji}^{пр}$ для декількох структурних підрозділів (група), кожному з них визначається однакове місце, яке умовно відповідає останньому місцю у цій групі структурних підрозділів. Контрольні показники (пji) та методика розрахунку рейтингових балів (рji) представлені у таблиці 1.1.

Таблиця 1.1 – Показники та методика розрахунку рейтингових балів

Показ-ники	**Найменування показника**	**Вимір**	**Надавачі інформ**	**Розрахунок $Р_{ji}$, $П_{ji}$ розрахункового**
$П_{1.1}$	Чисельність штатних науково-педагогічних працівників, всього, у тому числі:	осіб	ВК	$Р_{1.1} = \frac{2{,}5 \cdot 10^2 \cdot П_{1.1}}{П_{1.1}^{у} \cdot П_{1.11}} \le 0{,}5$

Продовження таблиці 1.1.

$П_{1.1.1}$	- з вченими ступенями та науковими званнями			$Р_{1.1.1} = \frac{5 \cdot 10^2 \cdot П_{1.1.1}}{П^{У}_{1.1.1} \cdot П_{1.1}}$
$П_{1.1.1.1}$	**у.т.ч. у віці до 35 років**			$P_{1.1.1.1} = \frac{10^2 \cdot П_{1.1.1.1}}{П^{У}_{1.1.1} \cdot П_{1.1}}$

1.2 Введення в машинне навчання

Машинне навчання – це метод аналізу даних, який автоматизує побудову аналітичної моделі [1]. Це галузь штучного інтелекту, заснована на ідеї, що системи можуть вчитися на основі даних, виявляти закономірності і приймати рішення з мінімальним втручанням людини. Процес навчання починається з спостережень або даних, таких як приклади, безпосередній досвід або інструкції, для того, щоб шукати закономірності в даних і приймати кращі рішення в майбутньому на основі прикладів, які ми надаємо.

Методи машинного навчання поділяються на певні категорії [2], це:

Контрольоване навчання - в цьому методі введення і виведення даних надається комп'ютера разом зі зворотним зв'язком під час навчання. Точність прогнозів за допомогою комп'ютера під час навчання також аналізується. Основна мета цього тренінгу - навчити комп'ютери зіставляти введення і виведення даних.

Нездібне до навчання - в цьому випадку таке навчання не надається, якщо комп'ютери не можуть самостійно знаходити результати. Некероване навчання в основному застосовується до трансакційних даних. Використовується в більш складних завданнях. Він використовує інший підхід ітерації, відомий як глибоке навчання, щоб прийти до деяких висновків.

Зміцнення навчання - цей тип навчання використовує три компонента, а саме - агент, середа, дія. Агент - це той, хто сприймає своє оточення, а середовище - це середовище, з якої агент взаємодіє і діє в цьому середовищі. Основна мета навчання з підкріпленням - знайти найкращу можливу політику.

Переваги машинного навчання.

1. *Прийняття рішень відбувається швидше* - машинне навчання забезпечує найкращі можливі результати, розставляючи пріоритети в рутинних процесах прийняття рішень.
2. *Здатність до адаптації* - машинне навчання дає можливість швидко адаптуватися до нового змінного середовища. Середовище швидко змінюється через те, що дані постійно оновлюються.
3. *Інновація* - машинне навчання використовує передові алгоритми, які покращують загальну здатність приймати рішення. Це допомагає в розробці інноваційних бізнес-послуг та моделей.
4. *Розуміння* - машинне навчання допомагає зрозуміти унікальні шаблони даних і визначити, які конкретні дії можуть бути зроблені.
5. *Зростання бізнесу* - завдяки машинному навчанню загальний бізнес-процес і робочий процес будуть швидше, і, отже, це буде сприяти загальному зростанню і прискоренню бізнесу.

З машинним навчанням якість результату буде покращено з меншими шансами помилки.

1.3 Дослідження моделей машинного навчання

1.3.1 Регресійний аналіз.

Регресійний аналіз – це надійний метод визначення того, які змінні впливають на цікаву для вас тему [3]. Процес виконання регресії дозволяє вам впевнено визначити, які чинники мають найбільше значення, які чинники можна ігнорувати і як ці фактори впливають один на одного.

1.3.2 Метод опорних векторів.

Машина опорних векторів (SVM) - це дискримінаційний класифікатор, формально визначається розділяє гіперплощиною. Іншими словами, з огляду на помічені дані навчання (контрольоване навчання), алгоритм видає оптимальну гіперплощина, яка класифікує нові приклади. У двовимірному просторі ця гіперплощина являє собою лінію, що розділяє площину на дві частини, де в кожному класі лежать по обидва боки.

План прийняття рішень - це той, який розділяє набір об'єктів, що належать до різних класів [8]. Схематичний приклад показаний на рисунку 1.1. У цьому прикладі об'єкти належать або класу ЗЕЛЕНІ, або ЧЕРВОНІ. Розділова лінія визначає межу, на правій стороні якої всі об'єкти ЗЕЛЕНІ, а зліва від якої всі об'єкти ЧЕРВОНІ. Будь-який новий об'єкт (білий круг), що падає вправо, позначається, тобто класифікується як ЗЕЛЕНИЙ (або класифікується як ЧЕРВОНИЙ, якщо він падає зліва від розділової лінії).

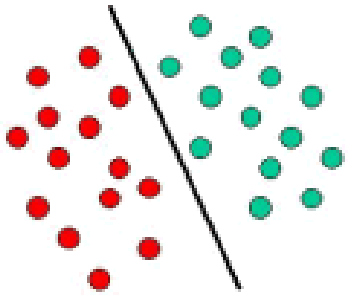

Рисунок 1.1 – Розділення об’єктів на два класи

1.3.3 Дерева рішень.

Одними з найпопулярніших класифікаторів є дерева рішень [4]. У листі такого дерева записані значення цільової функції (або мітки одного з класів), в інших вершинах - умови, що визначають за яким ребром йти. Щоб

класифікувати новий об'єкт, треба спуститися по дереву до листа і видати відповідне значення.

Даний метод має цілий ряд переваг: швидке навчання, здатність працювати з категоріальним атрибутами, простота розуміння і інтерпретації, відсутність необхідності масштабування даних (нормалізація або стандартизація). Завдяки цим перевагам метод добре підходить для складних даних, подібних наявними у нас (критерії оцінювання з різними показниками). З недоліків методу можна виділити схильність методу до перенавчання і відносно низьку точність передбачення. Боротися з цими проблемами можна за допомогою побудови композиції дерев рішень (так званий бустинг).

1.3.4 Бустингу.

Ідея бустинг-підходу полягає в комбінації слабких (з невисокою узагальнюючої здатністю) функцій, які будуються в ході ітеративного процесу, де на кожному кроці нова модель навчається з використанням даних про помилки попередніх. Результуюча функція являє собою лінійну комбінацію базових, слабких моделей.

Бустинг - це процедура послідовного побудови композиції алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів [7]. Існують кілька таких алгоритмів, найпопулярніші: Випадковий ліс, Ada boost, Gradient boosting.

1.3.5 Випадковий ліс.

Випадкові лісу або лісу випадкових рішень - це метод навчання ансамблю для класифікації, регресії і інших задач, який працює шляхом побудови безлічі дерев рішень під час навчання і виведення класу, який є режимом класів (класифікація) або середнім прогнозом (регресія) окремих

дерев [5]. Випадкові лісу рішень коригують звичку дерев рішень відповідати їх тренувального набору.

1.3.6 Метод найменших квадратів.

Метод найменших квадратів є стандартним підходом в регресійному аналізі для апроксимації рішення перевизначених систем, тобто наборів рівнянь, в яких більше рівнянь, ніж невідомих [6]. «Найменші квадрати» означають, що загальне рішення мінімізує суму квадратів залишків, зроблених в результатах кожного рівняння.

Найважливішим додатком є збір даних. Найкраще відповідність в сенсі найменших квадратів зводить до мінімуму суму квадратів залишків (залишкове значення: різниця між спостерігається величиною і підібраною величиною, наданої моделлю).

Завдання найменших квадратів діляться на дві категорії: лінійні або звичайні найменші квадрати і нелінійні найменші квадрати, в залежності від того, чи є залишками лінійними у всіх невідомих. Лінійна задача найменших квадратів виникає в статистичному регресійному аналізі; у нього є рішення в закритій формі. Нелінійна задача зазвичай вирішується шляхом ітеративного уточнення; на кожній ітерації система апроксимується лінійною, і, таким чином, розрахунок ядра в обох випадках однаковий.

1.3.7 Метод найближчих сусідів.

K найближчих сусідів - це простий алгоритм, який зберігає всі доступні випадки і прогнозує числову мета на основі заходи подібності (наприклад, функції відстані) [9].

Проста реалізація регресії KNN полягає в тому, щоб обчислити середню числовий мети K найближчих сусідів. Інший підхід використовує

середньозважене значення по зворотному віддалі K найближчих сусідів. Регресія KNN використовує ті ж функції відстані, що і класифікація KNN.

Вибір числа найближчих сусідів.

При малому числі найближчих сусідів відновлена крива сильно схильна до випадковим коливань. У той же час при занадто великій кількості k буде виявлятися ефект «пере згладжування». При вирішенні конкретних завдань результати визначаються варіабельністю істинної функції регресії і рівнем шуму. Тому значення k повинне підбиратися в залежності від результатів оцінювання.

1.3.8 Лінійна регресія.

Регресія - це метод, який використовується для моделювання і аналізу відносин між змінними, а також для того, щоб побачити, як ці змінні разом впливають на отримання певного результату. Лінійна регресія відноситься до такого виду регресійній моделі, який складається з взаємозв'язаних змінних. Лінійна регресія універсальна в тому сенсі, що вона має можливість запускатися з однієї вхідної змінної (проста лінійна регресія) або з залежністю від кількох змінних (множинна регресія). Суть цього алгоритму полягає в призначенні оптимальних ваг для змінних, щоб створити лінію (ax + b), яка буде використовуватися для прогнозування виведення. Подивіться відео з більш наочним поясненням.

2 ПОСТАНОВКА ЗАДАЧІ ТА ДОСЛІДЖЕННЯ АЛГОРИТМІВ

2.1 Мета та задачі

Мета проекту: створити модель з використанням алгоритмів машинного навчання на основі бізнес процесів прогнозування та визначення показників рейтингів, що дозволить визначити важливість, цінність кожного показника для рейтингу інститутів, факультетів та кафедр Сумського державного університету. За допомогою даної моделі можна передбачити, який показник більше впливає на рейтинг.

2.2 Вибір засобів реалізації

Далі будуть розглянуті алгоритми машинного навчання для завдання класифікації, застосування яких найбільш виправдано в досліджуваній задачі. А саме, ми повинні враховувати різнорідну природу ознак, одержуваних з файлів. Частина ознак являє собою прапори присутності, частина - статистики. Також, ми беремо до уваги швидкість роботи алгоритму. В таких умовах варто перш за все розглянути алгоритми, засновані на деревах рішень(випадковий ліс, градієнтний бустинг). Наведемо розглядаються алгоритми.

2.2.1 Дерево рішень.

Алгоритм дерево рішень (decision tree) тут згадаємо тому, що на ньому гуртуються більш складні, представлені тут алгоритми, і, слід розуміти принцип побудови базових алгоритмів для використання їх в системі.

Дерево рішень, в загальному випадку, є способом уявлення правил в ієрархічній послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення.

Як правило розглядають бінарне вирішальне дерево.

Процес пошуку вузла представлений на рис. 2.1 з відповідним лістингом - рис 2.2.

Введемо позначення:

V - множена вузлів в дереві. β_υ - предикат, функція в вершині υ, яка повертає $\{0, 1\}$. Y - множена класів, в нашому випадку - 2 (positive – позитивно впливає , negative – негативно пливає).

Вибір класу виконується наступним чином:

$$\forall \upsilon \in V_{\text{внутр}} \rightarrow \text{предикат } \beta_\upsilon : X \rightarrow \{0, 1\}, \beta \in B \quad (2.1)$$

$$\forall \upsilon \in V_{\text{лист}} \rightarrow \text{ім'я класу } c_\upsilon \in Y \quad (2.2)$$

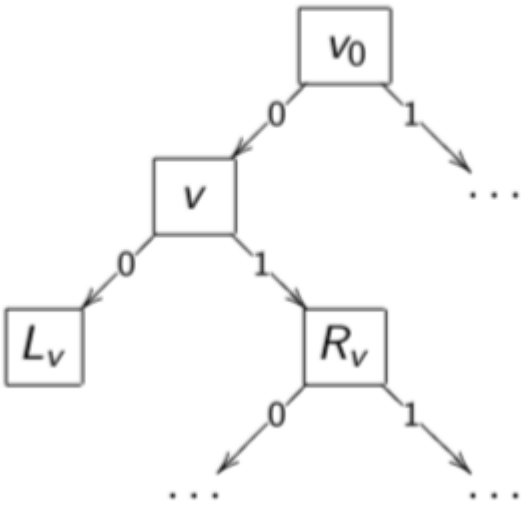

Рисунок 2.1 – Дерево рішень

```
while v ∈ V_внутр do
    if β_v(x) = 1 then
        go to right;
        v := R_v;
    else
        go to left;
        v := L_v;
    end
    return c_v
end
```

Рисунок 2.2 – Програмна реалізація алгоритму(пошуку вузла)

На практиці використовуються одномірні предикати, які порівнюють значення одного із ознак з порогом.

Для побудови дерева існує безліч алгоритмів. Конкретний метод побудови вирішального дерева визначається:

1. видом предикатів в вершинах;
2. функціоналом якості (для прийняття рішення про влаштування навчальної вибірки в розглянутій вершині. Як правило використовується критерій Джині або ентропійний критерій);
3. критерієм зупинки;
4. методом обробки пропущених значень;
5. методом стрижки (видалення деяких вершин з метою зниження складності та підвищення узагальнюючої здатності).

Здатність обробляти пропущені значення корисна і в нашій задачі. Вибрані імена критеріїв можуть бути не знайдені. В такому випадку, для такого файлу частина значень, ознак буде пропущена. Вирішальне дерево може бути навчено і на таких даних. А саме (тут, *U* - безліч об'єктів навчання, *Gain* - деякий вибраний критерій для максимізації в кожній вершині, S_v -функція, при 0 – L_v, 1 – R_v).

На стадії навчання:

$b_v(x_i)$ не визначено => x_i виключається із U для *Gain(*b_v,U*)* (2.3)

$q_{vk} = \frac{|U_k|}{|U|}$ - оцінка ймовірності k-й вітки, $v \in V_{внут}$ (2.4)

$P(y|x,v) = \frac{1}{|U|}\sum_{xi \in U}[y_i = y]$ – для всіх $v \in V_{лист}$ (2.5)

На стадії класифікації:

$b_v(x)$ *визначено => із дочірньої* $s = S_v(b_v(x))$ *взяти* $P(y|x,v) = P(y|x,s)$ (2.6)

$b_v(x)$ *не визначено => пропорційний розподіл:* (2.7)

$P(y|x,v) = \sum_{k \in D} q_{vk} P(y|x, S_v(k))$ (2.8)

Остаточне рішення – найбільш ймовірний клас

$$a(x) = \arg\max_{y \in Y} P(y|x, v_0 \quad (2.9)$$

Для побудови дерева, як правило, використовується жадібний алгоритм з оптимізацією в кожному з вузлів заданого функціонала якості (зазвичай розглядають критерій Джині, ентропійний критерій). Вибір функціоналу якості також сильно впливає на кінцевий результат.

Як готовий алгоритм, вирішальний дерево сьогодні не використовується. Алгоритм схильний до перенавчання, сильний вплив на шуми. Також, не завжди дерево стоїть найкраще поділ (знаменитий приклад із завданням побудови XOR функції). На рис. 2.3 відповідний приклад.

Велика увага приділяється ансамблям дерев рішень. Кожне з дерев здатне до перенавчання, під сильний вплив на шумові об'єкти, але використання їх ансамблів (бустінг, беггінг) перетворює в один з найбільш успішних і застосовних на практиці інструментів машинного навчання.

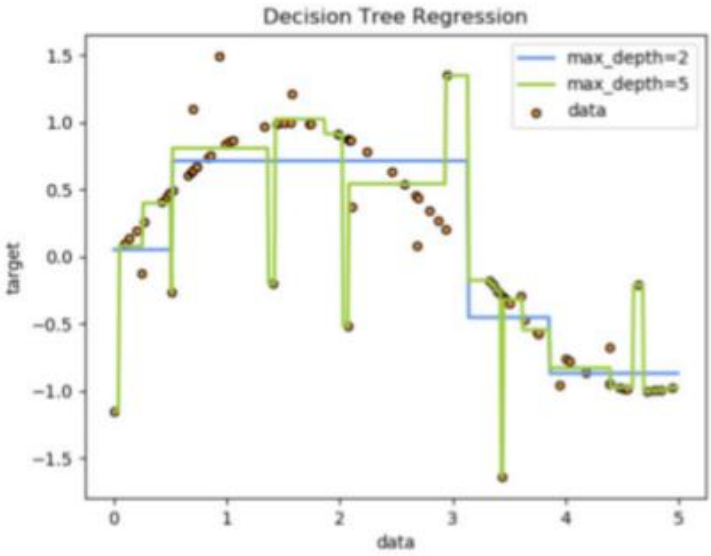

Рисунок 2.3 - Схильність до перенавчання дерева рішень

Побічний ефект від побудови дерева рішень полягає в можливості оцінки важливості ознак. Важливість ознаки розраховується на підставі того, наскільки поліпшується обрана метрика при поділі вибірки за ознакою в черговому вузлі. Це може нам дати уявлення які ознаки для побудови дерева виявилися найбільш вирішальними, а які не були використані. Потрібно розуміти, що важливість ознак – оціночні.

Далі будуть наведені алгоритми, засновані на деревах рішень. Багато властивостей, зазначені тут, вірні і для ансамблів. Наприклад, ми як і раніше можемо отримати важливість ознак через усереднення по кожному дереву.

2.2.2 Випадковий ліс.

Випадковий ліс (Random forest) являє собою ансамбль дерев рішень, які навчаються незалежно. Кожен з вирішальних дерев, як правило, роблять простим. Для прийняття остаточного рішення вибирається значення, за яке проголосувала більшість дерев. Кожне дерево окремо дає низьку якість, але за рахунок їх великої кількості результат виходить хорошим. На рис.2.4 можна бачити приклад випадкового лісу з двох дерев.

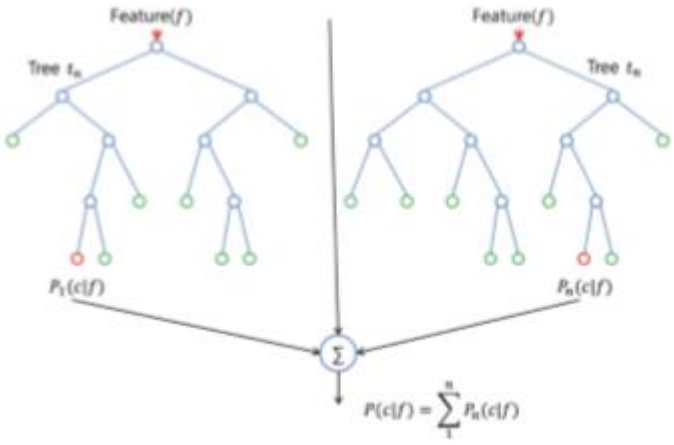

Рисунок 2.4 – Випадковий ліс із двох дерев

У побудові дерев на даних є свої особливості. Зазначимо традиційний підхід в побудові дерева. Нехай N - кількість прикладів на навчанні. тоді:

1. вибирається підвибірка навчальної вибірки розміру N (можливо, з поверненням);
2. будується дерево. При пошуку розбиття при побудові кожного вузла розглядаються в повному обсязі ознаки, а тільки частина з них (sklearn реалізація - корінь з кількості ознак);
3. дерево будується до повного вичерпання підвибірки, або на основі яких-небудь інших критеріїв.

Кожне з дерев намагаються зробити якомога простіше. При прийнятті рішення вибирається клас, за який проголосувало більшість дерев рішень.

Випадковий ліс дає порівняно погані результати при поділі вибірки в метричному просторі. У тому випадку, коли очікується деякий «правильний» вигляд розділяє гіперплощині. Проте, у випадкового лісу є велике число переваг, які можуть використовуватися в розглянутій задачі визначення важливості показника. Алгоритм не оперує метриками, що дозволяє вільно працювати з ознаками різної природи. Можливо обробляти дані з пропущеними значеннями ознак. Висока швидкість роботи алгоритму, незалежне побудова вирішальних дерев.

2.2.3 Градієнтний бустингу.

При побудові бустингу, базові алгоритми будуються послідовно, а не паралельно, компенсуючи помилку на попередній ітерації. У загальному вигляді алгоритм навчання градієнтного бустингу представлений нижче, рис.2.5. Тут *L(a, y)* - довільна функція втрат, b_i - *i*-ий базовий алгоритм. f_i являє собою *i*-е наближення.

Маємо:

Лінійну комбінацію базових алгоритмів: $a(x) = \sum_{t=1}^{T} a_t b_t(x)$ (2.10). Це і є алгоритм градієнтного бустингу. Являє лінійну комбінацію базових алгоритмів. Весь фокус полягає в тому, як побудувати такий набір алгоритмів і підібрати ваги.

Функціональна якість з довільною функцією втрат *L(a,y)*.

$$Q(a, b; X^l = \sum_{i=1}^{l} L(\underbrace{\sum_{t=1}^{T-1} a_t b_t}(x_i) + ab(x_i), y_i) \rightarrow \min_{a,b} \quad (2.11)$$

$f_{T-1} = (f_{T-1,i})^l_{i=1}$ – поточне приближення

$f_T = (f_{T,i})^l_{i=1}$ – наступне приближення

На кожній ітерації, при пошуку чергового базового алгоритму, мінімізується даний функціонал Q (α, b; Xl).

Data: навчальна вибірка X^l, параметр T;

Result: базові алгоритми і їх ваги $a_t b_t$, *t=1,…,T;*

Ініціалізація: f_i:= 0; *u=1,…,l;*

for t $\leftarrow$ **1** ***to T do***

базовий алгоритм, наближающий алгоритм:

$$b_t := arg \min_b \sum_{i=1}^{l} (b(x_i) + L'(f_i, y_i))^2;$$

$$a_t := arg \min_{a>0} \sum_{i=0}^{l} L(f_i + ab_t(x_i), y_i);$$

онолення вектора значення на об′єктах вибірки:

$$f_i := f_i + a_t b_t(x_i);$$

$$i = 1, \dots, l;$$

end

Рисунок 2.5 – Програма реалізація алгоритму(Градієнтний бустингу)

У разі градієнтного бустингу на вирішальних деревах, очевидно, базові алгоритми являють собою дерева рішень. Параметр T - кількість вибудовується дерев.

Найбільш популярні реалізації градієнтного бустингу на вирішальних деревах: XGBoost[10], LightGBM[11], CatBoost[12].

Xgboost представляє лише найбільш вдалу реалізацію ідей, запропонованих раніше.

LightGBM алгоритм реалізовувався з метою прискорити існуючі реалізації бустингу, а саме, XGBoost.

Основний час на побудову алгоритму йде на пошук кращої точки розбиття тренувальних даних у вузлі за певною ознакою. У XGBoost реалізовані 2 механізму:

1. для знаходження кращого розбиття ознаки перемішуються і знаходиться найкраще розбиття;
2. алгоритм на основі гістограми. ознаки об'єктів розбиваються на групи.

LightGBM пропонує 2 техніки для багаторазового прискорення швидкості роботи алгоритму:

1. Gradient-based One-Side Sampling (GOSS). Для побудови чергового дерева використовуються не всі об'єкти. Видаляються з розгляду об'єкти з малими градієнтами

2. Exclusive Feature Bundling (EFB). Упаковка ознак. При великій кількості ознак дані з високою ймовірністю розріджені і є можливість зменшити кількість ефективних ознак

У статті про LightGBM було проведено порівняння алгоритму з XGBoost як за швидкістю роботи, так і за якістю на декількох наборах даних. Розбиралися завдання класифікації і ранжування. Було продемонстровано від 2

- 20 кратне збільшення швидкості роботи алгоритму бустінга при тій же точності на відкладеної вибірці. CatBoost ставить своїм завданням ефективну роботу з категоріальним ознаками (catboost == categorial boosting), а також пропонує нову схему для підрахунку значень у вузлах, що дозволяє зменшити перенавчання. Ще одне нововведення в тому, що при побудові чергового дерева поділ в вузол може відбуватися за сукупністю двох категоріальних ознак. Підтримка GPU. У статті проводиться порівняння з XGBoost, LightGBM, H2O реалізаціями. Показано, що CatBoosst навіть з параметрами за замовчуванням дозволяє домогтися кращої якості (Logloss) на 8 розглянутих наборах даних. Показані результати порівняння часів передбачення алгоритмів CatBoost, XGBoost, LightGBM. CatBoost виявився на порядок швидше як в одно потоковому режимі, так і в багато потоковому. CatBoost знаходиться у вільному доступі і активно розвивається.

2.3 Вибір інструментів реалізації

Завдяки широкому розвитку концепції «штучний інтелект», поширення одержав такий розділ інформатики, як наука про дані (machine learning). Клас методів штучного інтелекту, які працюють не прямим вирішенням задачі, а навчання в процесі застосування рішень множини вихідних задач. Найбільш поширеними інструментами для роботи з аналізу даних на сьогоднішній день є R і Python.

2.3.1 Огляд мови програмування R.

R - мова програмування, призначений для статистичної обробки даних і роботи з графікою, але в той же час це вільна програмне середовище з відкритим вихідним кодом, що розвивається в рамках проекту GNU. R отримав

широке поширення в сферах, де проводиться робота з даними. Основна обчислювальна потужність R полягає в статистичному аналізі, проте він також володіє широким функціоналом для первинного аналізу даних (побудова графіків і таблиць спряженості) і математичного моделювання.

2.3.2 Огляд онлайн сервісу Amazon Machine Learning.

Ще один інструмент роботи з великими даними - сервіс Amazon Machine Learning, в основі якого лежить застосування методів машинного навчання. Даний сервіс спрощує використання технологій машинного навчання для розробників будь-якого рівня кваліфікації.

Amazon надає підручники, керівництва, корисні пояснення основних методів машинного навчання і досить рад, для здійснення перших самостійних кроків у сфері застосування цих методів.

Даний сервіс надає можливість складання прогнозу цільової змінної за допомогою заданих початкових умов: набір значень спостережень і моделі.

При ініціалізації моделі в налаштуваннях можна задати кількість проходів по набору даних і використовуваний спосіб регуляризації. Як параметр оцінки точності передбачення моделі використовується середньоквадратичне відхилення. Даний сервіс також надає можливість відсортувати дані за ступенем їх впливу на цільову змінну. Крім того, Amazon Machine Learning включає інструменти візуалізації для спрощення процесу створення моделей машинного навчання.

2.3.3 Огляд мови програмування Python.

Python - високорівнева універсальна інтерпретована мова сценаріїв. При розробці мовою Python велика увага приділяється простоті і зрозумілості синтаксису, що не тільки скорочує час вивчення його основ, але і підвищує

швидкість розробки в цілому. Це далеко не всі переваги даного мови, основні з них:

- об'єктно-орієнтованість;
- вільне поширення і широка підтримка;
- кросплатформеність;
- розвинені функціональні можливості.

Мова Python об'єднує дві парадигми програмування - об'єктно, яка є потужним засобом структурованого програмного коду для багаторазового використання, і процедурну, що розширює коло вирішення завдань, дозволяючи використовувати засоби python при вирішенні тактичних завдань з відсутністю фази проектування. Об'єктна модель Python підтримує поняття поліморфізм, перевантаження операторів і множинне спадкування.

Кросплатформеність мови досягається завдяки його реалізації на переносимій ANSI C, що дозволяє програмам, написаним на мові Python, однаково добре компілюватиметься і виконуватися на будь-яких платформах, де встановлена сумісна версія Python.

Гібридна природа Python об'єднує в собі простоту і зручність мов сценаріїв і потужності компілює мов, що робить Python зручним засобом розробки додатків різного типу. Однак найбільша ефективність мови досягається при вирішенні задач аналізу даних і автоматизації процесів. Python широко використовується в дослідницьких роботах. Мова програмування Python має потужний вбудованим інструментарієм (вбудовані типи об'єктів і динамічна типізація, автоматичне керування пам'яттю) і можливістю використання зовнішніх бібліотек і утиліт сторонніх розробників для вирішення більш вузькоспеціалізованих завдань.

Python використовується не тільки окремими користувачами, але і компаніями, включаючи комерційне використання. Наприклад:

- Компанія Google широко використовує Python в своїй пошуковій системі і для створення фреймворка App Engine.

- Служба колективного використання відеоматеріалів YouTube в значній мірі реалізована на мові Python.
- Популярна програма BitTorrent написана на мові Python.
- Такі компанії, як EVE Online і Massively Multiplayer Online Game, широко використовують Python в своїх розробках.
- Потужна система тривимірного моделювання і створення мультиплікації Maya підтримує інтерфейс для управління з сценаріїв на мові Python.
- Виробники електронних пристроїв і комп'ютерних компонентів (Такі, як Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm і IBM) використовують Python для тестування апаратного забезпечення.
- Кіностудії, що займаються створенням візуальних ефектів до кінофільмів (Industrial Light & Magic, Pixar і інші) використовують Python у виробництві анімаційних фільмів.
- Найбільші фінансові конгломерати (JPMorgan Chase, Union Bank of Switzerland, Federal Credit Union) застосовують Python для прогнозування фінансового ринку.
- Науково-дослідні центри (NASA, Los Alamos, Fermilab, Jet Propulsion Laboratory і інші) використовують Python для наукових обчислень.
- Компанія з розробки та продажу робото-техніки iRobot використовує Python в розробці комерційних роботизованих пристроїв.
- Компанія з виробництва геоінформаційних систем (Environmental Systems Research Institute) використовує Python в якості інструменту налаштування своїх програмних продуктів під потреби кінцевого користувача.

- Агентство національної безпеки Сполучених штатів (National Security Agency) використовує Python для шифрування і аналізу розвідданих.

2.3.4 Бібліотека scikit-learn.

Завдяки широкому поширенню Python зібрав навколо себе активну спільноту розробників, які в рамках різних проектів розробляють модулі для вузькоспеціалізованих завдань.

Одним з таких проектів став Google Summer of Code 2007, в рамках якого David Cournapeau розробив бібліотеку scikit-learn. Розробка даної бібліотека є однією з причин популяризації застосування мови Python в області аналізу даних за допомогою методів машинного навчання.

Бібліотека scikit-learn надає реалізацію ряду алгоритмів як для навчання з учителем (Supervised learning), так і для навчання без (Unsuoervised learning).

Scikit-learn побудована на основі стека SciPy (Scientific Python), який включає в себе:

- NumPy додає підтримку великих багатовимірних масивів і матриць, а також бібліотеку високорівневих математичних функцій для операцій з ними.
- SciPy - відкрита бібліотека високоякісних наукових інструментів для мови програмування Python.
- Matplotlib - бібліотека для візуалізації двовимірної і тривимірної графіки.
- IPython - інтерактивна оболонка для мови програмування Python, яка надає розширену інтроспекцію, додатковий командний синтаксис, підсвічування коду і автоматичне доповнення.
- Sympy - бібліотека для роботи з символьними обчисленнями.
- Pandas реалізує різні структури даних і аналіз.

Бібліотека scikit-learn складається з 35 модулів, які можна поділити на модулі кластеризації, модулі оцінки моделі і кількісного визначення якості прогнозів, модулі роботи з наборами даних (попередня обробка, нормалізація), модулі роботи з ознаками (витяг і виявлення найбільш значущих), модулі, реалізують різні алгоритми рішення задач класифікації і регресії. Кожен модуль складається з класів і функцій і вирішує такі завдання, як:

- Кластеризація (Clustering) - угруповання нерозмічену даних.
- Перехресне перевірка (Cross Validation) - оцінка ефективності роботи моделі на незалежних даних.
- Набори даних (Datasets) - для зберігання тестових наборів даних і для генерації наборів даних з певними властивостями для дослідження поведінкових властивостей моделі.
- Скорочення розмірності (Dimensionality Reduction) - набір алгоритмів для зменшення кількості атрибутів для візуалізації та відбору ознак (Feature Selection), наприклад, метод головних компонент (Principal Component Analysis).
- Алгоритмічні композиції (Ensemble Methods) - набір методів для комбінування прогнозів декількох моделей.
- Витяг ознак (Feature Extraction) - процес визначення атрибутів в даних.
- Відбір ознак (Feature Selection) - набір алгоритмів для виявлення значущих атрибутів на основі яких буде побудована модель.
- Оптимізація параметрів алгоритму (Parameter Tuning) - методи для отримання максимально ефективної віддачі від моделі.
- Множина навчання (Manifold Learning) - підхід нелінійного скорочення розмірності даних.

Окремо слід виділити методи реалізують навчання з учителем (Supervised Models). Даний набір методів включає в себе:

- узагальнені лінійні моделі (Generalized Linear Models);

- методи дискримінантного аналізу (Discriminate Analysis);
- наївний баєсовський класифікатор (Naive Bayes);
- нейронні мережі (Neural Networks);
- метод опорних векторів (Support Vector Machines);
- дерева прийняття рішень (Decision Trees).

3 МОДЕЛЮВАННЯ

3.1 Структурно-функціональне моделювання «методики визначення рейтингу»

IDEF0 реалізує методику функціонального моделювання складних систем. Найбільш відомою реалізацією IDEF0 є методологія SADT (Structured Analysis and Design Technique). Функціональна модель IDEF0 являє собою набір блоків, кожен з яких представляє собою «чорний ящик» з входами і виходами, управлінням та механізмами, які деталізуються (декомпозуються) до необхідного рівня. Найбільш важлива функція розташована у верхньому лівому кутку. А з'єднуються функції між собою за допомогою стрілок і описів функціональних блоків. При цьому кожен вид стрілки або активності має власне значення. Дана модель дозволяє описати всі основні види процесів, як адміністративні, так і організаційні.

Стрілки можуть бути:

- Вхідні - вступні, які ставлять певне завдання.
- Вихідні - виводять результат діяльності.
- Керуючі (зверху вниз) - механізми управління (положення, інструкції тощо).
- Механізми (від низу до верху) - що використовується для того, щоб зробити необхідну роботу.

Діаграми декомпозиції призначені для деталізації функцій і виходять при розбиванні контекстної діаграми на великі підсистеми (функціональна декомпозиція) і описують кожну підсистему та їх взаємодію. Єдина функція, представлена на діаграмі контекстного верхнього рівня, може бути розкладена на основні підфункції за допомогою створення дочірньої діаграми. Структурно-функціональна модель першого рівня має наступний вигляд, рис.3.1. Цілю даної

структурно-функціональної моделі є показати, як на даний момент визначається рейтингові рівні кафедр.

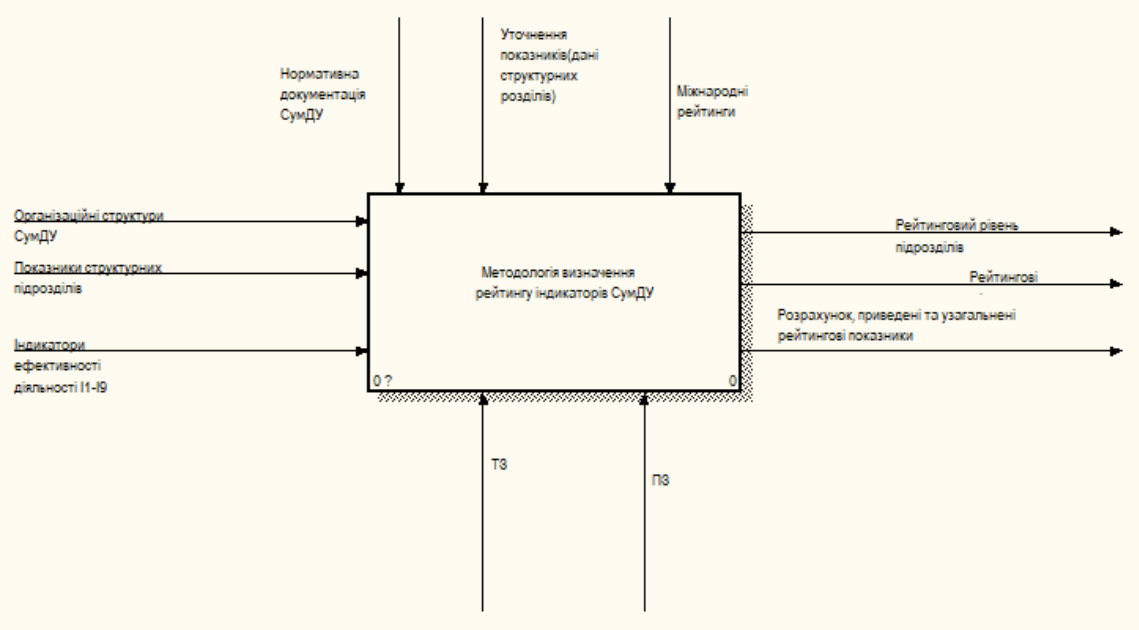

Рисунок 3.1 – Структурно-функціональна модель

Основний блок – Методологія визначення рейтингу індикаторів СумДУ. Вхідні стрілки – «Організаційні структури СумДУ», «Показники структурних підрозділів», «Індикатори ефективності діяльності І1-І9». Це ті вхідні, які потрібні для роботи. Управляючі для даної моделі є – «Нормативна документація СумДУ», «Уточнення показників(дані структурних розділів)», «Міжнародні рейтинги». У ролі «механізмів» – «ПЗ (програмне забезпечення)», «ТЗ (технічне завдання)». На виході із цієї моделі маємо: «Рейтинговий рівень підрозділів», «Рейтингові місця» та «Розрахунок, приведені та узагальнені рейтингові показники». Бізнес-процес «визначення рейтингу індикаторів СумДУ» можна деталізувати, для цього декомпозуємо на зв’язані між собою елементи. Декомпозиція першого рівня моделі зображена на рисунку 3.2.

Для цієї методики робота ділиться на 9 основних етапів:

1. *Визначення показників І1* – Науково-педагогічний потенціал.

2. *Визначення показників І2* – Диверсифікація рівнів та форм навчання, контингент осіб.
3. *Визначення показників І3* – Якість навчально-наукової роботи зі студентами.
4. *Визначення показників І4* – Якість міжнародної діяльності.
5. *Визначення показників І5* – Рівень оприлюднення результатів наукової та науково-методичної діяльності.
6. *Визначення показників І6* – Якість підготовки науково-педагогічних кадрів.
7. *Визначення показників І7* – Якість представлення результатів діяльності в Інтернеті.
8. *Визначення показників І8* – Фінансова оцінка результатів інноваційної діяльності.
9. *Визначення показників І9* – Якість позанавчальної діяльності інституту.

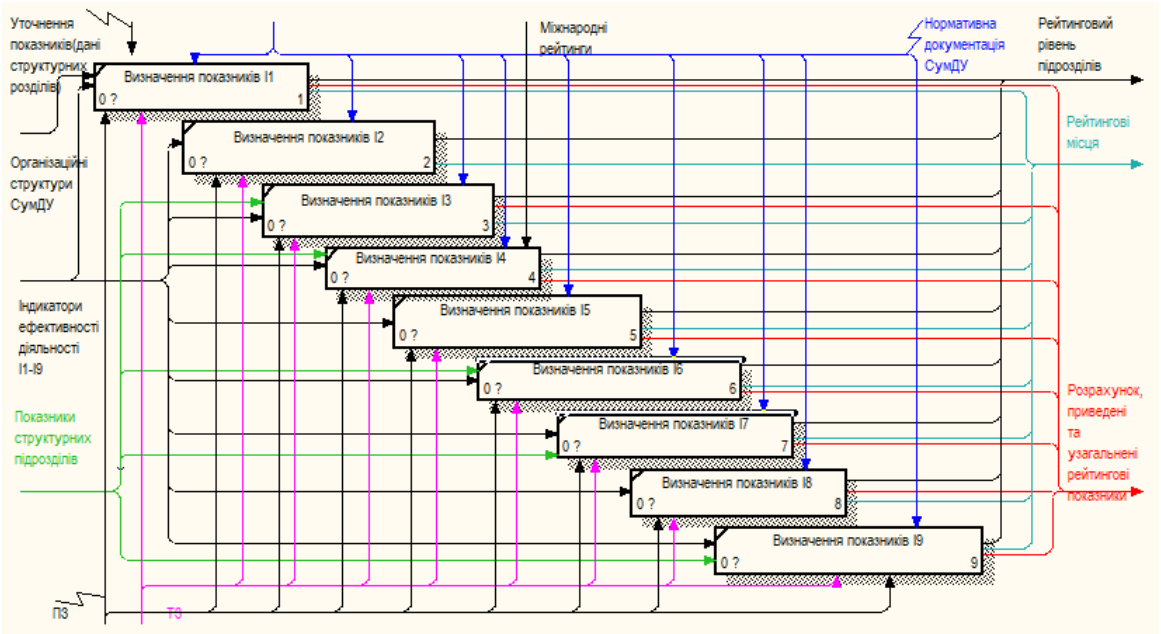

Рисунок 3.2 – 1-ий рівень декомпозиції основного блоку

Стрілка «Індикатори ефективності діяльності І1-І9» є вхідної для всіх блоків. Стрілка «Організаційні структури СумДУ» є вхідною тільки для

першого блоку, а «Показники структурних підрозділів» – вхідною для усіх, крім першого блоку. Стрілка управління «Нормативна документація СумДУ» має вплив на всі блоки декомпозиції першого рівня, коли стрілки «Міжнародні рейтинги» відносяться до блоку «*Визначення показників I4*», а стрілка «Уточнення показників(дані структурних розділів)» до блоку «*Визначення показників I1*». Усі стрілки виходу і «механізмів» відносяться до кожного блоку декомпозиції. У свою чергу кожен індикатор ділиться на показники, які вираховуються за власними формулами. Як видно на рисунку, кожен блок є незалежним друг від друга.

Тому ми бачимо, що для визначення рейтингів потрібно порахувати кожен показники у індикаторі.

3.2 Структурно-функціональне моделювання «модель аналізу рейтингу СумДУ»

У даній структурно-функціональній моделі продемонстровано модель аналізу рейтингу СумДУ, рис.3.3. Цілю даної моделі є показати, як визначається рейтинг показників у моделі аналізу. Основним блоком являться – Модель аналізу рейтингу СумДУ. Вхідними даними для нього є: «Рейтинг факультетів» та «Рейтинг СумДУ». Управляючі це – «Методика СумДУ»(яка була представлена у розділі 3.1) та «Специфіка алгоритмів». Стрілки, які виступають у ролі «механізмів» це – «ПЗ – програмне забезпечення». На виході маємо: «Рейтинг показників» у якому представлені важливості показників, які отримали за допомогою моделей машинного навчання.

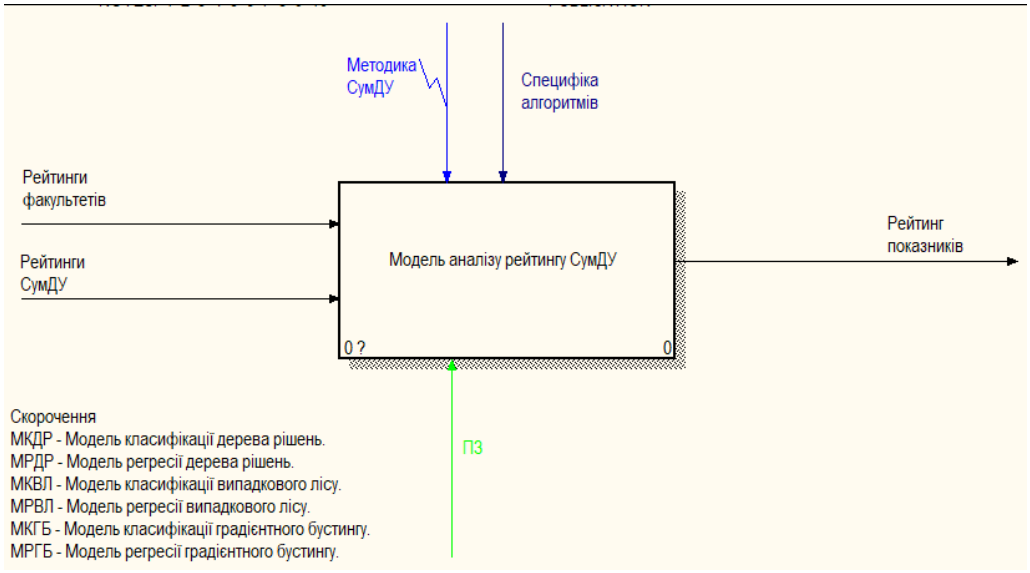

Рисунок 3.3 – IDEF0 – моделі визначення рейтингу СумДУ

Для цієї методики робота ділить на 4 етапи:

1. Робота з даними.
2. Визначення впливу показників моделі Дерева рішень.
3. Визначення впливу показників моделі Випадкового лісу.
4. Визначення впливу показників моделі Градієнтного бустингу.

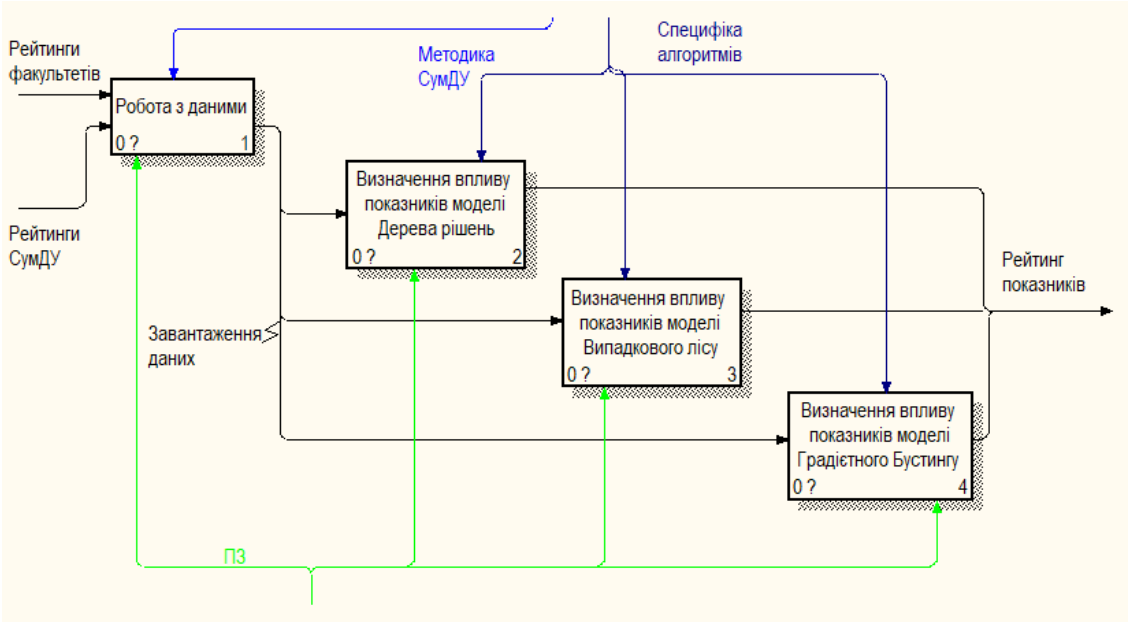

Рисунок 3.4 – 1-ий рівень декомпозиції моделі рейтингу

Вхідні дані подаються до блоку «Робота з даними» де завантажуються з pdf файлів, обробляються та зберігаються у csv файлах. Декомпозиція блоку «Робота з даними» представлена на рисунку В.1. Стрілка «Методика СумДУ» управління відносить до блоку «Робота з даними», а стрілка «Специфіка алгоритмів» до усіх інших блоків. Після визначення впливу показників моделі маємо вихідні дані. Кожний блок визначення впливу показників ділиться на моделі класифікації та регресії. У свою чергу ці блоки також діляться на побудову моделей, навчання, визначення точності та визначення показників. Моделі випадково лісу та градієнтного бустингу мають блок «Визначення кількості дерев» у якому визначається відношення кількості дерев до точності моделі. Рисунки всіх рівнів декомпозиції моделей зображені на рис В.2 – В.13.

3.3 Модель структури ієрархії класів моделі

Діаграма класів визначає типи класів системи і різного роду статичні зв'язки, які існують між ними. На діаграмах класів зображуються також атрибути класів, операції класів та обмеження, які накладаються на зв'язку між класами. Вид і інтерпретація діаграми класів істотно залежить від точки зору (рівня абстракції): класи можуть представляти сутності предметної області (в процесі аналізу) або елементи програмної системи (в процесах проектування і реалізації[21].

Основними елементами є класи і зв'язку між ними. Класи характеризуються за допомогою атрибутів і операцій.

Атрибути описують властивості об'єктів класу. Більшість об'єктів в класі отримують свою індивідуальність через відмінності в їх атрибутах і взаємозв'язку з іншими об'єктами. Однак, можливі об'єкти з ідентичними значеннями атрибутів і взаємозв'язків. Тобто індивідуальність об'єктів визначається самим фактом їх існування, а не відмінностями в їх властивості. Ім'я атрибута повинно бути унікально в межах класу. За ім'ям атрибута може слідувати його тип і значення за замовчуванням.

Операція є функція або перетворення. Операція може мати параметри і повертати значення.

Види зв’язків:

- залежність;
- асоціація;
- агрегація;
- успадкування.

Створена діаграма класів була представлена на рисунку 3.5.

Були розроблені класи діаграми:

- Data – дані показників.
- DTC – Модель класифікації дерево рішень.
- DTR – Модель регресії дерева рішень.
- RFC – Модель класифікації випадкового лісу.
- RFR – Модель регресії випадково лісу.
- XGBC – Модель класифікації градієнтного бустингу.
- XGBR – Модель регресії градієнтного бустингу.
- Feature Analysis – Визначення та аналіз.

Клас «Data» приймає на вхід дані з csv файлі, які попередньо були взяті з pdf файлів «Контрольних показників факультетів». Класу має операції для видалення пропусків, заміни тексту та приведення даних до цифр для арифметичних дій. Клас «Data» з’єднаний залежністю з класами: DTC, DTR, RFC, RFR, XGBC, XGBR. Залежність - семантично представляє собою зв'язок між двома елементами моделі, в якій зміна одного елемента (незалежного) може привести до зміни семантики іншого елемента (залежного). Це означає, що зміна даних вплине на побудову моделі. Визначення усіх моделей представлено у таблиці В.2.

Дані приводяться до структури DataFrame наступної роботи з ними. Вони діляться на тестові та навчальні. На навчальних даних модель навчається, а на тестових проходить перевірку на точність.

Атрибути класів моделей:

- X_test: DataFrame;
- Y_test: DataFrame;
- X_train: DataFrame;
- Y_train: DataFrame;

Моделі RFC, RFR, XGBC, XGBR мають додатковий атрибут *n_estimators* який визначає кількість дерев моделі. Атрибут має тип *int,* тому що кількість дерев повинна бути цілим числом. Також моделі мають операції(методи) навчання моделі (*fit*), визначення точності та визначення важливості показників(*feature_importances_*). Класи моделей з’єднані з класом Feature Analysis зв’язком асоціації. Асоціація – це структурна зв'язок між елементами моделі, яка описує набір зв'язків, що існують між об'єктами. Асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності таким чином, що можна переміщатися від об'єктів одного класу до іншого. До класу Feature Analysis передаються дані у вигляді масиву. За допомогою методів будується графік важливості показників та зображується на екрані. Тому ми бачимо, що об’єкт «importances» залежить від даних, які передаються від класів моделей.

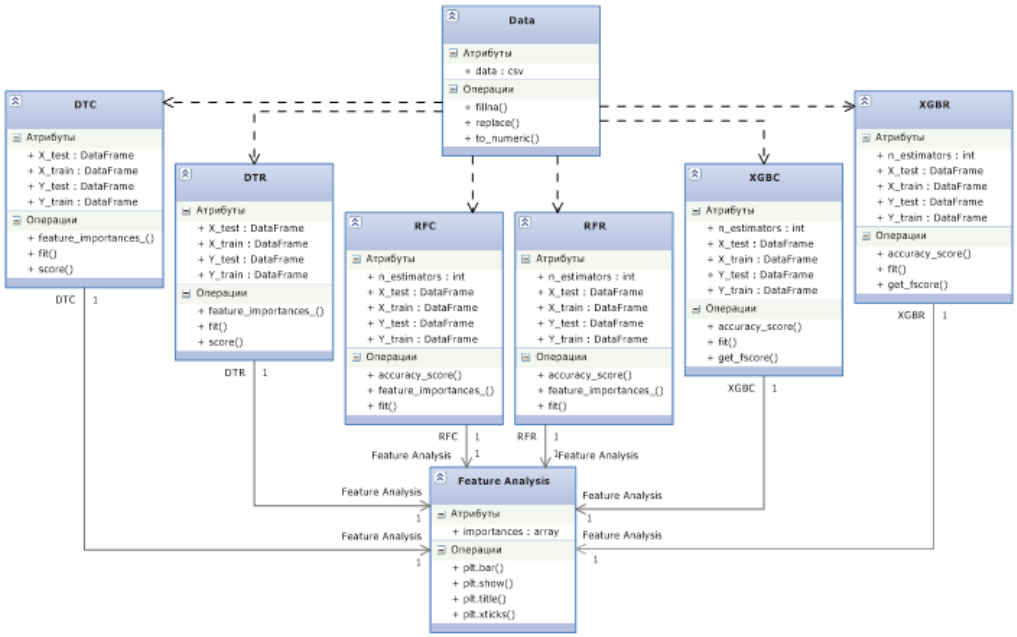

Рисунок 3.5 – Діаграма класів

3.4 Діаграма послідовностей моделі

Діаграма послідовності є однією з різновиду діаграм взаємодії та призначена для моделювання взаємодії об'єктів системи в часі, а також обміну повідомленнями між ними. Діаграма послідовності використовується головним чином для відображення взаємодій між об'єктами в послідовному порядку, в якому ці взаємодії відбуваються. Вони(діаграми) зазвичай містять об'єкти, які взаємодіють в рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, пов'язані з повідомленнями. Втім, часто повертаються результати позначають лише в тому випадку, якщо це не очевидно з контексту[22].

Об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їх від класів).

Повідомлення (виклики методів) - лініями зі стрілками.

Повертаються результати - пунктирними лініями зі стрілками.

Лінія життя об'єкту (object lifeline) зображується пунктирною вертикальною лінією, асоційованою з єдиним об'єктом на діаграмі послідовності. Лінія життя служить для позначення періоду часу, протягом якого об'єкт існує в системі і, отже, може потенційно брати участь у всіх її взаємодіях. Якщо об'єкт існує в системі постійно, то і його лінія життя повинна тривати по всій площині діаграми послідовності від самої верхньої її частини до найнижчої.

На діаграмі послідовності ми можемо побачити такі аспекти:

- повідомлення, які спонукають об'єкт до дії;
- дії, які викликаються повідомленнями (методи) - найчастіше це передача повідомлення наступного об'єкти або повернення певних даних об'єкта;
- послідовність обміну повідомленнями між об'єктами.

Ця діаграма послідовності відображає потік подій в рамках «Побудова моделі аналізу даних». Дії починаються, коли менеджер проекту завантажує дані показників – цей об’єкт показаний у прямокутники у верхній частині діаграми, рис.3.6. Після завантаження даних іде обробка даних(видалення непотрібних даних, нормалізація, приведення до DataFrame структури). Коли дані оброблені починається побудова першої моделі – Модель дерева рішень. Будується модель, навчається на даних та тестується на точність. Результати, які повертаються до менеджера – важливість показників. Будується модель випадкового лісу, визначається кількість дерев для моделі, навчається та тестується. Також результатом є – важливість показників моделі. Для моделі градієнтного бустингу всі дії схожі, як і для моделі випадкового лісу.

Таким чином, діаграма послідовності ілюструє послідовність дій визначення важливості показників у моделі аналізу рейтингу СумДУ.

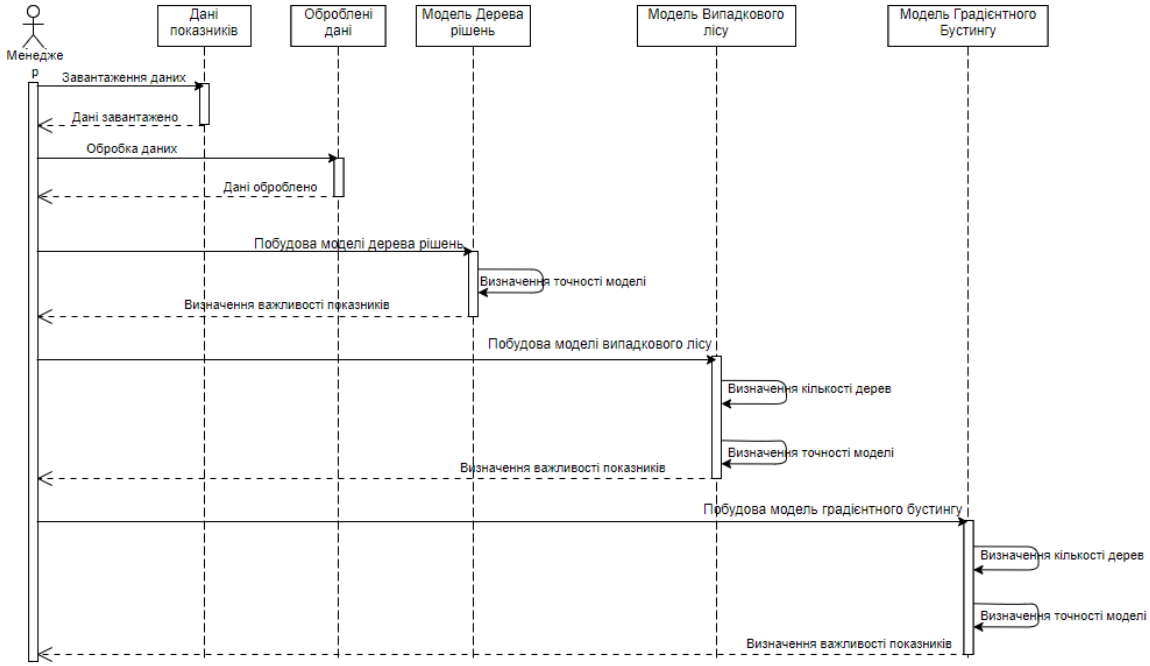

Рисунок 3.6 – Діаграма послідовностей «Моделі аналізу даних»

Тези даної роботи були опубліковані на Науково-технічні конференції “Інформатика, математика, автоматика :: 2019” (IMA-2019), Додаток З.

4 ПРОЕКТУВАННЯ МОДЕЛІ АНАЛІЗУ

4.1 Попередня обробка і аналіз вихідних даних

Контрольні показники по кожному факультету знаходяться у PDF-файлах:

- forma1fk_elit.pdf – файл з показниками факультету ЕлІТ;
- forma1fk_fem.pdf – файл з показниками факультету ФЕМ;
- forma1fk_ifsk.pdf – файл з показниками факультету ІФСК;
- forma1fk_ip.pdf – файл з показниками Інституту Права;
- forma1fk_ki.pdf – файл з показниками Конотопського Інституту;
- forma1fk_mi.pdf – файл з показниками Медичного Інституту;
- forma1fk_shi.pdf – файл з показниками Шосткинський Інститут;
- forma1fk_teset.pdf – файл з показниками факультету ТеСЕТ;
- forma1fk_uabs.pdf – файл з показниками факультету УАБС.

На рисунку В.1 зображена перша сторінка з файлу forma1fk_elit.pdf . Перед початком роботи данні потрібно завантажити. Код завантаження даних з файлу forma1fk_elit.pdf зображено на рис.4.1.

```
#шлях до файлу
way = 'pdf/forma1fk_elit.pdf'
PDF_ELIT = pd.DataFrame()
for i in range(1,40):
    #зчитування інформації з кожної сторінки файлу
    pdf = read_pdf(way,java_options="-Dfile.encoding=UTF8", pages=i,lattice=True)
    #об'єднання кожної сторінки у файл
    PDF_ELIT = pd.concat([PDF_ELIT, pdf])
```

Рисунок 4.1 – Код завантаження даних з файлу forma1fk_elit.pdf

Після завантаження і об’єднання даних в DataFrame структуру вони мають вигляд, рис. 4.2. DataFrame – це проіндексований багатовимірний масив значень, відповідно кожен стовпець DataFrame, є структурою Series.

```
1 pdf
```

НайменуванняЗначення показника ПоказникВимір показника університетфакультеткафедри факультету НЕМАiМОЕЗПФел.енергетЕКТПМтаМССКН	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unna
0 NaN	NaN	NaN	університет	факультет	кафедри факультету	NaN	NaN	NaN	NaN	
1 NaN	NaN	NaN	NaN	NaN	НЕ	МАiМО	ЕЗПФ	ел.енергет	ЕКТ	ПМта
2 П1.1	Чисельність штатних науково-\гпедагогічних пра...	ос.	933	136	11	11	24	8	13	
3 П1.1.1	Чисельність штатних науково-\гпедагогічних пра...	ос.	796	123	11	8	24	8	11	
4 П1.1.1.1	Чисельність штатних науково-\гпедагогічних пра...	ос.	193	26	3	1	5	2	3	

Рисунок 4.2 – Дані завантажені з файлу forma1fk_elit.pdf

Дані у такому вигляді не зручні для роботи та аналізу, вони мають не коректні та не зрозумілі надписи, у таблиці є пропуски. Тому дані були оброблені для зручності (див. рис. 4.3). Код моделі зображений у лістингу Ж.3. У представленій таблиці використовуються наступні позначення колонок:

- показник – номер показнику, за яким було оцінювання;
- університет – значення показників по університету;
- факультет - значення показників по факультету(у даному випадку ЕлІТ);
- НЕ - значення показників кафедри наноелектроніки;
- МАiМО – значення показників кафедри атематичного аналізу і методів оптимізації;
- ЕЗПФ – значення показників кафедри електроніки, загальної та прикладної фізики;
- Ел.енергет – значення показників кафедри електроенергетики;
- ЕКТ – значення показників кафедри електроніки та комп’ютерної техніки;
- ПМтаМСС – значення показників кафедри прикладної математики та моделювання складних систем;
- КН – значення показників кафедри комп’ютерні науки.

```
1 PDF_ELIT[:10]
```

	Показник	університет	факультет	НЕ	МАіМО	ЕЗПФ	ел.енергет	ЕКТ	ПМтаМСС	КН
2	П1.1	933	136	11	11	24	8	13	21	48
3	П1.1.1	796	123	11	8	24	8	11	18	43
4	П1.1.1.1	193	26	3	1	5	2	3	6	6
5	П1.1.2	126	21	2	2	8	0	2	4	3
6	П1.1.3	500	81	8	7	16	4	8	9	29
7	П1.1.3.1	50	8	1	1	2	0	1	0	3
3	П1.1.4	1	0	0	0	0	0	0	0	0
4	П1.1.5.1	5	3	0	0	0	0	1	0	2
5	П1.1.5.2	8	0	0	0	0	0	0	0	0
6	П1.1.6	262	101	23	3	26	3	10	17	22

Рисунок 4.3 – Оброблені дані за файлу forma1fk_elit.pdf

Для більш зручної роботи таблиця з даними була збережена у .csv файл, рисунок 4.4.

```
1 PDF_ELIT.to_csv('csv/Elit.csv', index=False)
```

Рисунок 4.4 – Збереження даних в у .csv файл

Дані для кожного файлу були опрацьовані на збережені у відповідних .csv файлах. Для аналізу рейтингу всі дані з .csv файлів були об’єднані в DataFrame таблицю, рис. 4.5. Таблиця має наступний вигляд, рис. 4.6.

```
1 PDF_SSU = pd.DataFrame()
2 PDF_SSU = pd.concat([PDF_SSU, PDF_ELIT])
3 PDF_SSU = pd.concat([PDF_SSU, PDF_FEM[3:]])
4 PDF_SSU = pd.concat([PDF_SSU, PDF_IFSK[3:]])
5 PDF_SSU = pd.concat([PDF_SSU, PDF_IP[3:]])
6 PDF_SSU = pd.concat([PDF_SSU, PDF_KI[3:]])
7 PDF_SSU = pd.concat([PDF_SSU, PDF_MI[3:]])
8 PDF_SSU = pd.concat([PDF_SSU, PDF_SHI[3:]])
9 PDF_SSU = pd.concat([PDF_SSU, PDF_UABS[3:]])
```

Рисунок 4.5 – Об’єднання даних в таблицю

```
1 PDF_SSU
```

	0	1	2	3	4	5	6	7	8	9	...	268	269	270	271	272	273	274	275	276
показник	П1.1	П1.1.1	П1.1.1.1	П1.1.2	П1.1.3	П1.1.3.1	П1.1.4	П1.1.5.1	П1.1.5.2	П1.1.6	...	П9.5	П9.6	П9.7	П9.8	П9.9	П9.10	П9.11.1	П9.11.2	0.0
університет	933	796	193	126	500	50	1	5	8	262	...	114	20	9	13	21	21	0	0	0.0
факультет	136	123	26	21	81	8	0	3	0	101	...	14	0	2	0	1	3	0	0	0.0
НЕ	11	11	3	2	8	1	0	0	0	23	...	0	0	0	0	0	0	0	0	1.0
МАіМО	11	8	1	2	7	1	0	0	0	3	...	0	0	0	0	0	0	0	0	7.0
ЕЗПФ	24	24	5	8	16	2	0	0	0	26	...	0	0	0	0	0	0	0	0	2.0
ел.енергет	8	8	2	0	4	0	0	0	0	3	...	0	0	0	0	0	0	0	0	6.0
ЕКТ	13	11	3	2	8	1	0	1	0	10	...	0	0	0	0	0	0	0	0	5.0
ПМтаМСС	21	18	6	4	9	0	0	0	0	17	...	0	0	0	0	0	0	0	0	5.0
КН	48	43	6	3	29	3	0	2	0	22	...	0	0	0	0	0	0	0	0	5.0
ФіП	30	26	4	6	20	4	0	0	0	5	...	0	0	0	0	0	0	0	0	4.0
ЕПБА	32	30	9	7	23	1	0	0	1	11	...	0	0	0	0	0	0	0	0	1.0

Рисунок 4.6 – Таблиця, з об’єднаними даними

4.2 Попередній огляд даних

Рейтинг інститутів та факультетів Сумського державного університету за результатами діяльності у 2018 році за узагальненим рейтинговим рівнем показний на рисунку 4.7. Ці дані будуть використовуватися для всіх моделей алгоритмів машинного навчання. Діаграма значення кафедр по показнику П.1.1, рисунок 4.8. Діаграма значення факультетів/інститутів по показнику П.1.1, рисунок 4.9.

Назва структурного підрозділу	**Узагальнений рейтинговий рівень**
Факультет електроніки та інформаційних технологій	1
Навчально-науковий інститут фінансів, економіки та менеджменту імені Олега Балацького	1
Факультет технічних систем та енергоефективних технологій	2
Медичний інститут	3
Навчально-науковий інститут бізнес-технологій "УАБС"	3
Навчально-науковий інститут права	B
Факультет іноземної філології та соціальних комунікацій	BC
Факультет денної форми навчання ШІ	HC
Факультет денної форми навчання КІ	HC

Рисунок 4.7 – Рейтинг інститутів та факультетів

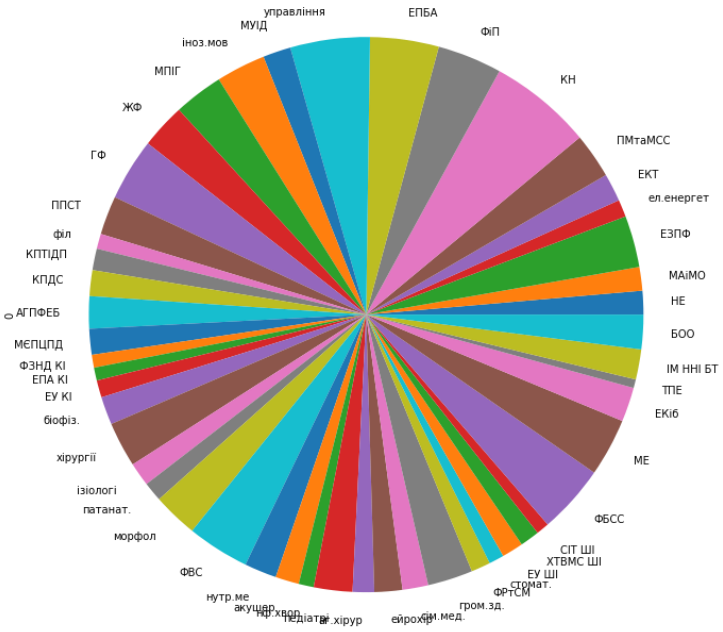

Рисунок 4.8 – Діаграма значення кафедр, інститутів

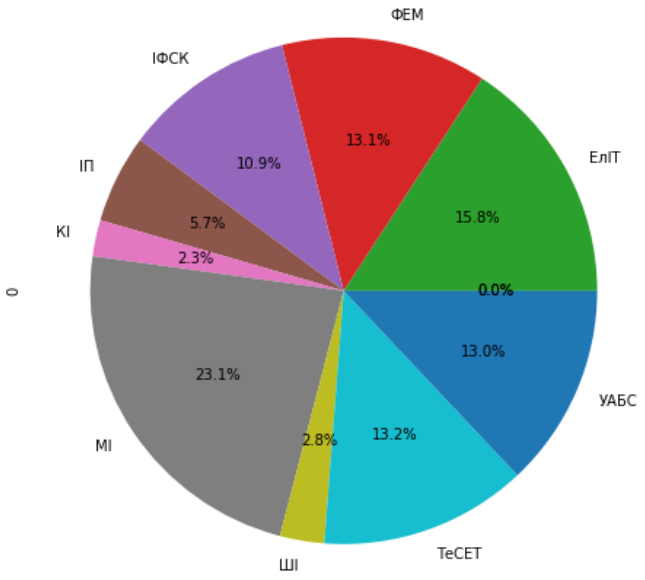

Рисунок 4.9 – Значення факультетів/інститутів

Була визначена сума всіх показників інститутів та факультетів, рис. 4.10. Як видно на рисунку 4.10, найбільша сума показників у Медичного інституту, на другому місці факультет ЕлІТ, на третьому факультет ІФСК. Але у представленому рейтингу, рис. 4.7, ми бачимо, що на першому місці факультет ЕлІТ, на другому факультет ФЕМ, на третьому ТеСЕТ. Це означає, що

показники мають не однаковий вплив на фінальний рейтинг. І задача стоїть в тому, щоб дізнатися вплив кожного показника на рейтинг.

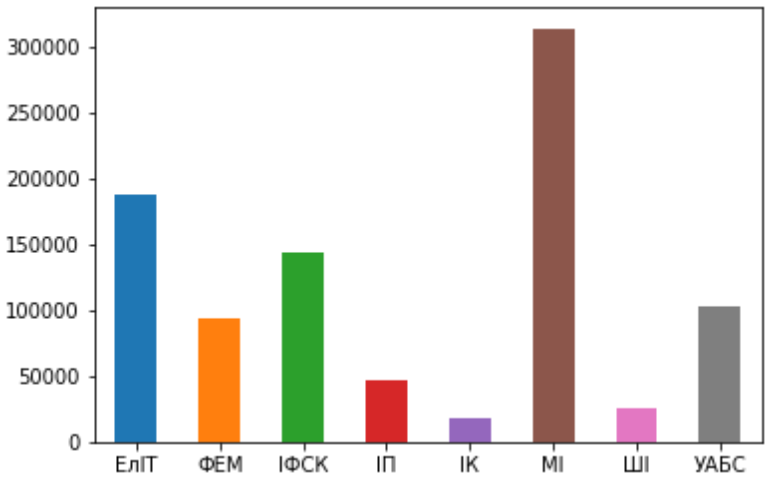

Рисунок 4.10 – Сума показників інститутів та факультетів

4.3 Дерево рішень

Для визначення важливості показників було побудовані дві моделі дерев рішень:

- модель класифікації – кожний показник відноситься до певного класу;
- модель регресії – кожному показнику представлене числове значення.

Рейтингові рівні:

- 1, 2, 3 - призові;
- В- високий;
- ВС - вище середнього;
- С - середній;
- НС - нижче середнього;
- Н - низький;
- Кр – критичний.

Відповідність рейтингових рівнів до класів наведено у таблиці 4.1.

Таблиця 4.1 – Відповідність рівнів до класів.

№	Рейтинговий рівень	Клас
1.	1	1
2.	2	2
3.	3	3
4.	B	4
5.	BC	5
6.	C	6
7.	HC	7
8.	H	8
9.	KP	9

Дані для моделей класифікації представлені у файлі dodatok3.pdf і мають вигляд наведені у таблиці Д.1, стовпець «Узагальнений рейтинговий рівень». Для моделей регресії узагальнені рейтингові показники представлені у таблиці Д.1, стовпець «Узагальнений рейтинговий показник».

Для визначення важливості показників використовується метод feature_importances_. feature_importances_ – повертає вектор "Важливості" ознак. Індекс елемента в цьому векторі відповідає індексу ознаки в даних. Значення елемента відображає "важливість" ознаки щодо інших: чим більше значення, тим більше важливість.

4.3.1 Модель класифікації дерева рішень.

Для побудови моделі класифікації дерева рішень буде використовуватися клас DecisionTreeClassifier із бібліотеки sklearn.tree і метод fit() – для навчання моделі, рис.4.11.

```
# створення моделі дерева класифікації
clf1 = DecisionTreeClassifier()
# навчання моделі на даних
clf1 = clf1.fit(X_train, Y_train)
```

Рисунок 4.11 – Побудова моделі класифікації дерева рішень

За допомогою методу *score*, який повертає середню точність даних і даних тесту, було визначено точність моделі. Score() – У класифікації з кількома мітками це точність підмножини, яка є жорсткою метрикою, оскільки для кожної вибірки потрібно, щоб кожен набір міток був правильно зпрогнозований. У даній моделі точність на тестовій вибірці склала: 66.66%, рис. 4.12.

```
1  # Визначення точності моделі
2  clf1.score(X_test,Y_test)
```

```
0.6666666666666663
```

Рисунок 4.12 – Код визначення точності моделі класифікації дерева рішень

Модель класифікації дерева рішень представлена на рис.Е.1.

Після побудови моделі за допомогою метода *feature_importances_* визначаємо «важливість» показників, зображені на рис.4.13. Повний список показників із впливом на модель представлено у Додатку Е таблиця 1. Код моделі зображений у лістингу Ж.1.

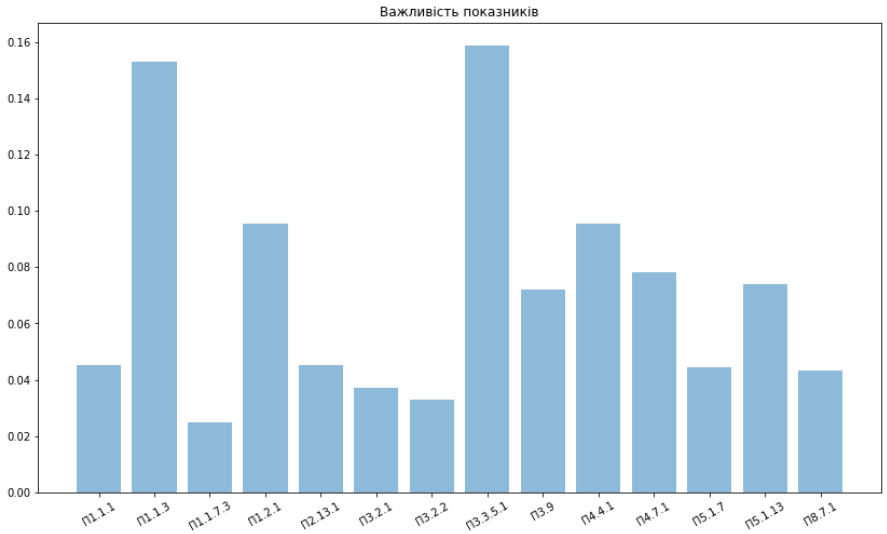

Рисунок 4.13 – Графік важливості показників моделі класифікації дерева рішень

4.3.2 Модель регресії дерева рішень.

Для побудови моделі класифікації дерева рішень буде використовуватися клас DecisionTreeRegressor із бібліотеки sklearn.tree і метод fit() – для навчання моделі, рис.4.14.

```
clf2 = DecisionTreeRegressor(random_state=0)
clf2 = clf2.fit(train_data, predict2_data)
clf2
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
```

Рисунок 4.14 – Побудова моделі регресії

Методом *score* визначена точність моделі, яка скала: 78,64%, рис 4.15.

```
# визначення точності моделі
clf2.score(X_test,Y_test)
```

```
0.78644098305290278
```

Рисунок 4.15 – Код визначення точності моделі регресії дерева рішень

Дерево моделі регресії зображено на рисунку Е.2.

Після побудови моделі регресії за допомогою метода *feature_importances_* визначаємо «важливість» показників та зображуємо на рисунку.4.16 Повний список показників із впливом на модель представлено у Додатку Е таблиця 1. Код моделі зображений у лістингу Ж.1.

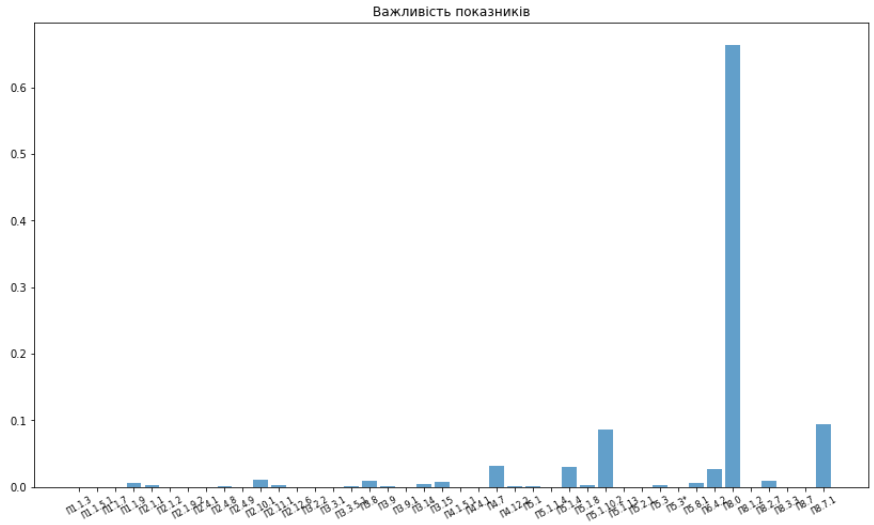

Рисунок 4.16 – Графік важливості показників
моделі регресії дерева рішень

4.4 Випадковий ліс

Для моделей класифікації випадкового лісу будуть використовуватися класи із таблиці Д.1. Для моделей регресії випадкового лісу будуть використовуватися дані з таблиці Д.2.

4.4.1 Модель класифікації випадкового лісу.

Для побудови моделі випадкового лісу використаємо клас *RandomForestClassifier*. Одним із параметрів класу *n_estimators* – є кількість дерев. Тому перед побудовою моделі потрібно визначити кількість дерев, при якій модель буде точніша. Для цього в циклі від 2 до 52 з кроком 2 через метод *accuracy_score* зобразимо на графіку середню точність даних і даних тесту(рис 4.17, рис 4.18). З таблиці були вибрані 6 випадкових записів, які будуть передаватися у метод, як тестові дані для перевірки помилки.

```python
# список чисел кількость дерев
estimators = np.arange(2, 52, 2)
# список для даних точності моделі
predict1 = []
for i in estimators:
    # створення моделі з і-кількість дерев
    clf=RandomForestClassifier(n_estimators=i)
    # навчання моделі на даних
    clf.fit(X_train,Y_train)
    # прогнозування
    y_pred=clf.predict(X_test)
    # запис до масиву та визначення точності моделі
    predict1.append(metrics.accuracy_score(Y_test, y_pred))
# побудова графіка
plt.title("Ефект n-дерев")
plt.grid(True)
plt.xlabel("n-дерев")
plt.ylabel("точність")
plt.plot(estimators, predict1)
```

Рисунок 4.17 – Код пошуку залежності точності від кількості дерев

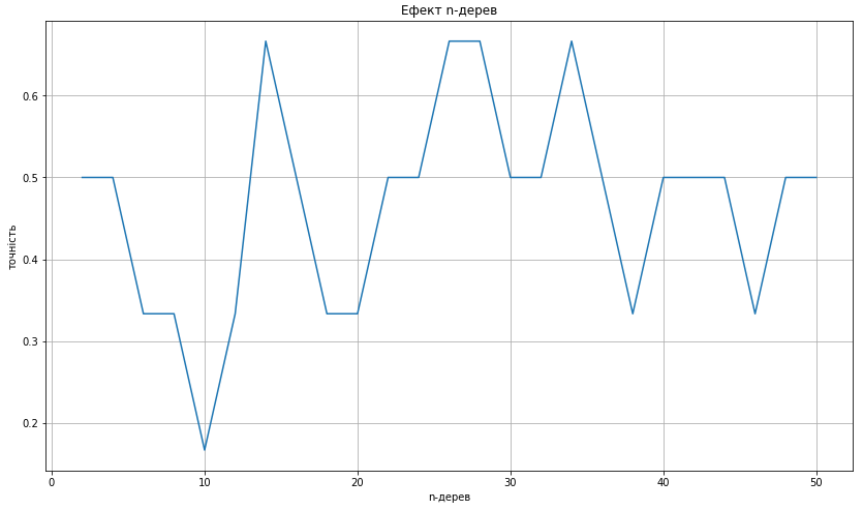

Рисунок 4.18 – Залежність точності від кількості дерев

На графіку видно, що при кількості дерев 26 і 28 найбільша точність, тому для побудування моделі класифікації випадкового лісу було вибрана кількість дерев – 27. Модель класифікації побудована на рис.4.19. Методом *accuracy_score* було визначено точність моделі: 66.66%, рис.4.20.

```
clf2=RandomForestClassifier(n_estimators=27)
clf2.fit(train,test)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=27,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Рисунок 4.19 – побудування моделі класифікації випадкового лісу

```
clf2.score(X_test,Y_test)
y_pred=clf2.predict(X_test)
    # запис до масиву та визначення точності моделі
metrics.accuracy_score(Y_test, y_pred)
```

```
0.6666666666666663
```

Рисунок 4.20 – Визначення точності моделі класифікації випадкового лісу

Після побудови моделі класифікації випадково лісу за допомогою метода *feature_importances_* визначаємо «важливість» показників. Десять показників, які мають найбільший вплив зображені на рис.4.21. Повний список показників із впливом на модель представлено у Додатку Е таблиця 2. Код моделі зображений у лістингу Ж.2.

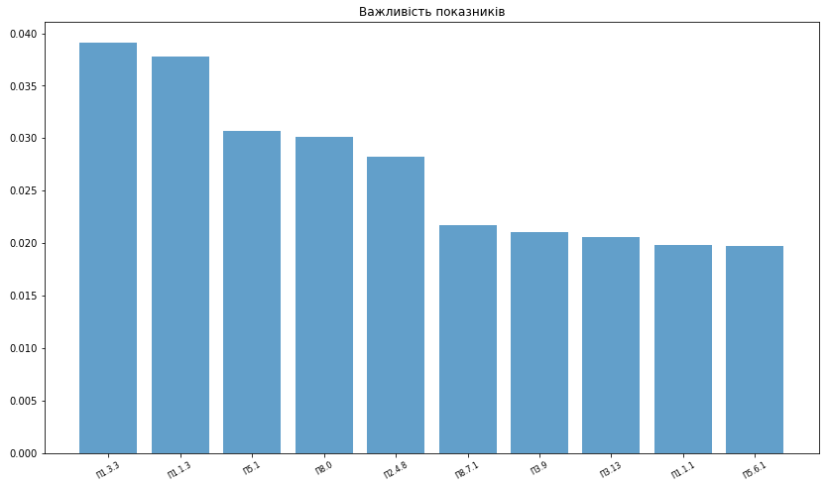

Рисунок 4.21 – Графік важливості показників моделі класифікації випадкового лісу

4.4.2 Модель регресії випадкового лісу.

Для побудови моделі випадкового лісу використаємо клас *RandomForestRegressor*. Як і для моделі класифікації випадкового лісу порівняємо залежність точність моделі від кількості дерев у моделі. Тільки для даної моделі діапазон пошуку збільшився до 102. Код пошуку залежності зображено на рисунку 4.22.

```
# список чисел кількость дерев
estimators = np.arange(2, 102, 2)
# список для даних точності моделі
scores = []
for n in estimators:
    # створення моделі з і-кількість дерев
    regressor = RandomForestRegressor(n_estimators=n, random_state=0)
    # навчання моделі на даних
    regressor.fit(X_train, Y_train)
    # запис до масиву та визначення точності моделі
    scores.append(regressor.score(X_test, Y_test))
 # побудова графіка
plt.title("Ефект n-дерев")
plt.grid(True)
plt.xlabel("n-дерев")
plt.ylabel("точність")
plt.plot(estimators, scores,marker='.')
```

Рисунок 4.22 – Код пошуку залежності точності від кількості дерев

На графіку який зображений в рисунку 4.23 видно, що найкращій показник точності при кількісті дерев 58. Побудова моделі регресії випадкового лісу, рис. 4.24. Як і для моделі регресії дерева рішень, використаємо метод *score* для визначення точності, рис.4.25. Точність побудованої моделі склала – 75.99%.

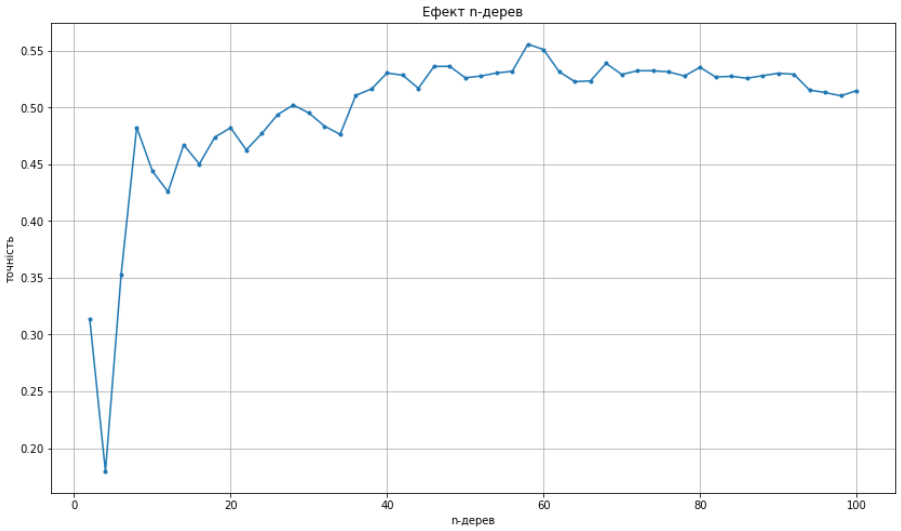

Рисунок 4.23 – Залежність точності від кількості дерев

```
1  regressor = RandomForestRegressor(n_estimators=58, random_state=0)
2  regressor
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=58,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
```

Рисунок 4.24 – Побудова моделі регресії випадкового лісу

```
1  # навчання модели
2  regressor.fit(X_train,Y_train)
3  # прогнозування
4  y_pred=regressor.predict(X_test)
5  # визначення точності
6  regressor.score(X_test, Y_test)
```

```
0.75989789699084942
```

Рисунок 4.25 – Визначення точності моделі регресії випадкового лісу

Після побудови моделі регресії випадково лісу за допомогою метода *feature_importances_* визначаємо «важливість» показників. На рисунку 4.26 наведений графік десяти найважливіших показників моделі регресії

випадкового лісу. Повний список показників із впливом на модель представлено у Додатку Е таблиця 2. Код моделі зображений у лістингу Ж.2.

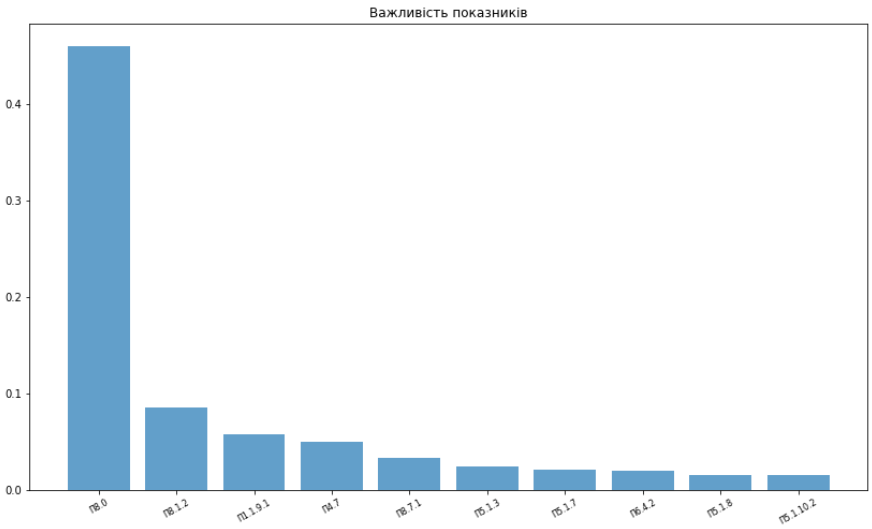

Рисунок 4.26 – Графік важливості показників моделі регресії випадкового лісу

4.5 Градієнтний бустинг

Для моделей класифікації градієнтного бустингу будуть використовуватися класи із таблиці Д.1. Для моделей регресії градієнтного бустингу будуть використовуватися дані з таблиці Д.2.

4.5.1 Модель класифікації градієнтного бустингу.

Для побудови моделі класифікації градієнтного бустингу використаємо клас *XGBClassifier*. Одним із параметрів класу *estimators* – є кількість дерев. Тому перед побудовою моделі потрібно визначити найкращу кількість дерев. Для цього в циклі від 2 до 52 з кроком 2 через метод *accuracy_score* зобразимо на графіку середню точність даних і даних тесту(рис 4.27, рис 4.28). З таблиці були вибрані 6 випадкових записів, які будуть передаватися у метод, як тестові дані для перевірки помилки. На графіку (рис.4.28) видно, що найбільша точність моделі при кількості дерев з 16 до 18. Тому для побудови моделі

бустигу параметр *estimators* буде дорівнювати – 17(рис.4.29). Методом *accuracy_score* визначено точність побудованої моделі: 66.66%, рис.4.30.

```
# список для даних точності моделі
xgb_scoring = []
# список чисел кількость дерев
estimators = np.arange(2, 52, 2)
for i in estimators:
    # створення моделі з i-кількість дерев
    estimator = xgb.XGBClassifier(learting_rate=0.1, n_estimators=i)
    # навчання моделі на даних
    estimator.fit(X_train,Y_train,eval_metric='auc')
    # прогнозування
    y_pred=estimator.predict(X_test)
    # запис до масиву та визначення точності моделі
    xgb_scoring.append(metrics.accuracy_score(Y_test, y_pred))
```

Рисунок 4.27 – Код пошуку залежності точності від кількості дерев

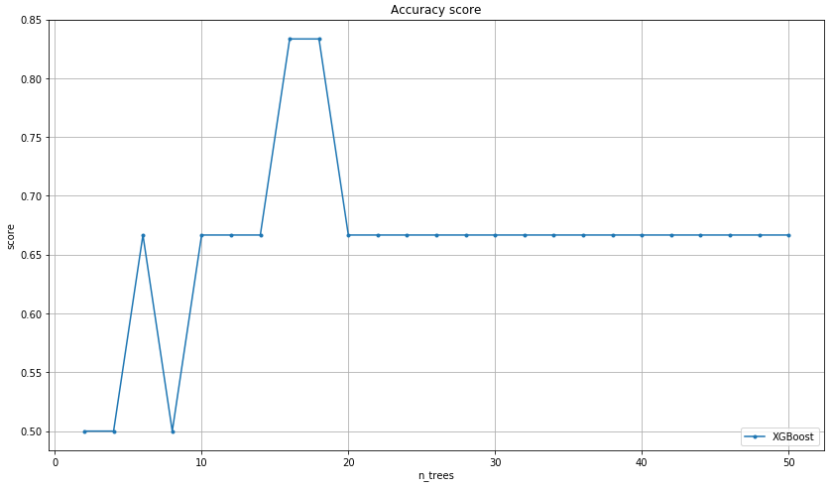

Рисунок 4.28 – Залежність точності від кількості дерев

```
# створення моделі бустингу
estimator = xgb.XGBClassifier(n_estimators=17)
# навчання моделі
estimator.fit(X_train,Y_train)
# бустинг моделі
bst = estimator._Booster
# отримати важливість кожного показника моделі
imps = bst.get_fscore()
```

Рисунок 4.29 – Побудова моделі класифікації градієнтного бустингу

```
y_pred=estimator.predict(X_test)
    # запис до масиву та визначення точності моделі
metrics.accuracy_score(Y_test, y_pred)
```

```
0.6666666666666663
```

Рисунок 4.30 – Визначення точності моделі
класифікації градієнтного бустингу

За допомогою метода get_fscore() можна отримати важливість показника в моделі градієнтного бустингу. Метод _Booster() - це модель xgboost, яка містить процедури низького рівня для навчання, прогнозування та оцінки. На графіку, який зображений на рисунку 4.31 показані десять найважливіших показників. Повний список показників із впливом на модель представлено у Додатку Е таблиця 3. Код моделі зображений у лістингу Ж.2.

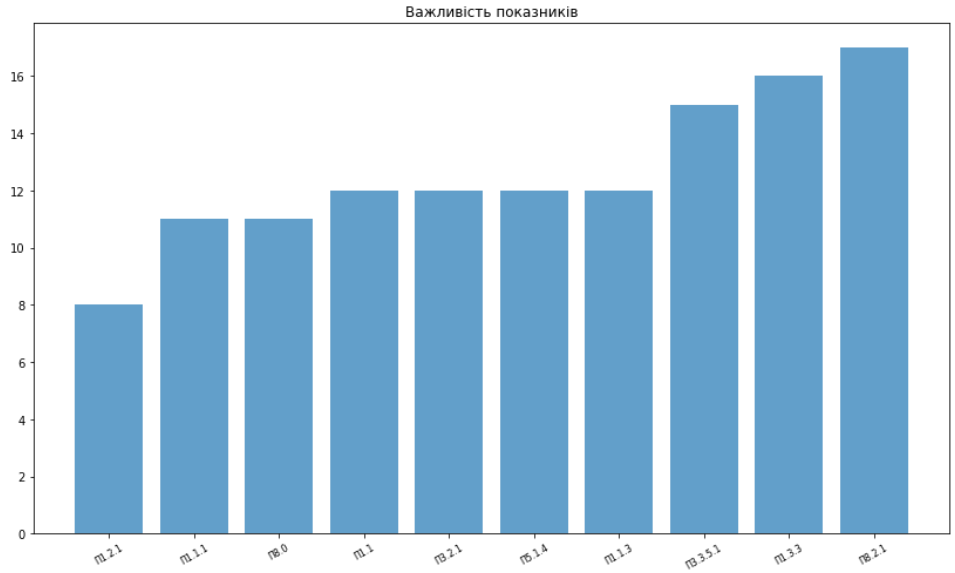

Рисунок 4.31 – Графік важливості показників моделі
класифікації градієнтного бустингу

4.5.2 Модель регресії градієнтного бустингу.

Для побудови моделі регресії градієнтного бустингу використаємо клас *XGBRegressor*. Як і для моделей класифікаці градієнтного бустингу потрібно проаналізувати точність моделі від кількості дерев. Код пошуку залежності представлено на рисунку 4.32, графік залежності(рис. 4.33).

```
# список для даних точності моделі
xgb_scoring = []
# список чисел кількость дерев
estimators = np.arange(2, 52, 2)
for i in estimators:
    # створення моделі з і-кількість дерев
    estimator = xgb.XGBRegressor(n_estimators=i)
    # навчання моделі на даних
    estimator.fit(X_train,Y_train)
    # запис до масиву та визначення точності моделі
    xgb_scoring.append(estimator.score(X_test, Y_test))
```

Рисунок 4.32 – Код пошуку залежності точності від кількості дерев

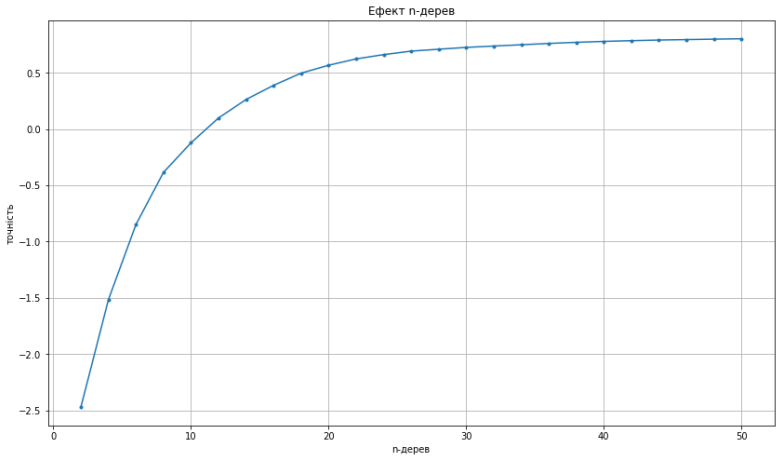

Рисунок 4.33 – Залежність точності від кількості дерев

Як і у розділі 4.5.1 для пошуку важливості показників використаємо метод *get_fscore()*. Створення моделі регресії градієнтного бустингу та визначення важливості показників зображені на рисунку 4.34. На графіку, який зображений на рисунку 4.35 показані десять найважливіших показників моделі. Повний список показників із впливом на модель представлено у Додатку Д таблиця 3. Методом *score()* – було визначено точність моделі: 99.72%, рис.4.36. Код моделі зображений у лістингу Ж.2.

```
# створення моделі бустингу
estimator = xgb.XGBRegressor(n_estimators=40)
# навчання моделі
estimator.fit(train,test)
# бустинг моделі
bst = estimator._Booster
# отримати важливість кожного показника моделі
imps = bst.get_fscore()
```

Рисунок 4.34 – Побудова моделі класифікації градієнтного бустингу

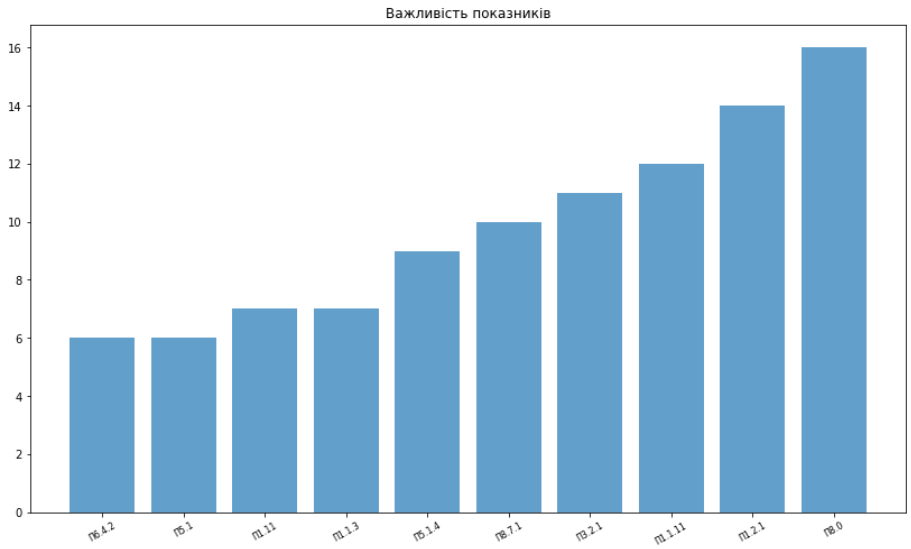

Рисунок 4.35 – Графік важливості показників моделі регресії градієнтного бустингу

```
1  estimator.score(X_test, Y_test)
```

```
0.99722809699056991
```

Рисунок 4.36 – Визначення точності моделі регресії градієнтного бустингу

4.6 Підсумок

У таблиці 4.2 представлені моделі та точність цих моделей. Після аналізу видно, що у моделей регресії більше точність, чим у моделей класифікації. А найбільша точність у моделі регресії градієнтного бустингу склала – 99.72%. Також можна замітити, що точність моделей дерева рішень більше, чим у моделей випадкового лісу, хоча, як було сказано у пункті 2.2.2 випадковий ліс – це ансамбль дерев рішень.

Таблиця 4.2 – Моделі аналізу та точність моделей

№	Модель	Точність,%
1.	Модель класифікації дерева рішень	66.66
2.	Модель регресії дерева рішень	78.64
3.	Модель класифікації випадкового лісу	66.66
4.	Модель регресії випадкового лісу	75.99
5.	Модель класифікації градієнтного бустингу	66.66
6.	Модель регресії градієнтного бустингу	99.72

Список найменування десяти найкращих показників:

1. *П8.0* - Надходження коштів за показниками: $П_{8.1.2} \div П_{8.2.9}; П_{8.3.2} \div П_{8.4.3}$.
2. *П1.2.1* – Чисельність штатних працівників-виконавців НДР, грантів - всього (за виключенням працюючих у поєднанні з навчанням).
3. *П1.1.11* – які є членами редколегій видань, які індексуються у БД Scopus або WoS.
4. *П3.2.1* – Узагальнюючий Показник Узагальнюючий показник.
5. *П8.7.1* – Річний ФОП структ. підрозділу за усіма джерелами фінансування у тому числі за результатами інноваційної діяльності.
6. *П5.1.4* – - які розміщені у звітному році в інституційному репозитарії СумДУ шляхом самоархівування, опубліковані в різні роки не у виданнях СумДУ.
7. *П1.11* – Кількість ставок ПВС, що обліковується у рейтингу.
8. *П1.1.3* – з науковими званнями професор, доцент, старший науковий дослідник, всього.
9. *П1.1* – Чисельність штатних науково-педагогічних працівників, всього.

10.*П1.1.2* – Чисельність штатних науково-педагогічних працівників, всього, у тому числі докторів наук.

ВИСНОВКИ

У результаті виконання бакалаврської дипломної роботи були проаналізовані методи машинного навчання та побудована модель аналізу рейтингу інститутів, факультетів та кафедр СумДУ. Методи машинного навчання дозволяють автоматизувати процес побудови прогнозу. Застосування методів машинного навчання полягає в проведенні серії обчислювальних експериментів, з метою аналізу, інтерпретації та зіставлення результатів моделювання з реальною поведінкою досліджуваного об'єкта і, при необхідності, подальшому уточненні вхідних параметрів.

За допомогою методів машинного навчання на етапі аналізу даних і обробки ознак були виявлені найбільш значимі ознаки (добре корелюють з цільової змінної). Таким чином, вдалося мінімізувати помилку передбачення.

Результатом виконання випускної кваліфікаційної роботи є розроблена модель аналізу рейтингу, яка визначає важність кожного показника для фінального оцінювання інститутів, факультетів та кафедр.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Машинное обучение. [Електронний ресурс]. https://ru.wikipedia.org/wiki/ машинное_обучение.

2. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. – М.: ДМК Пресс, 2015. – 400 с.

3. Регресійний аналіз. [Електронний ресурс]. http://www.machinelearning.ru/wiki/ index.php?title= Регресия.

4. Дерево решений. [Електронний ресурс]. https://ru.wikipedia.org/wiki/Дерево_решений.

5. Wikipedia. Random forest // Википедия, свободная энциклопедия. — https://en.wikipedia.org/wiki/Random_forest (дата обращения: 22.05.2015).

6. Метод наименьших квадратов. [Електронний ресурс]. https://ru.wikipedia.org/wiki/ метод наименьших квадратов.

7. Бустинг (обучение машин) аналіз. [Електронний ресурс]. https://ru.wikipedia.org/wiki/ Бустинг_(обучение машин).

8. Support Vector Machines (SVM) Introductory Overview – http://www.statsoft.com/textbook/support-vector-machines

9. K Nearest Neighbors – Regression. [Електронний ресурс]. . https://www.saedsayad.com/ k_nearest_neighbors_reg.htm

10. Chen T., C. Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedingsofthe22ndACMSIGKDDInternationalConferenceonKnowledge Discovery and Data Mining.

11. LightGBM: A Highly Efficient Gradient Boosting Decision Tree / Ke G. [et al.] // Advances in Neural Information Processing Systems 30.

12. Dorogush A. V., Ershov V., Gulin A. CatBoost: gradient boosting with categorical features support.

13. Воронцов К. Курс лекций. Презентации. [Електронний ресурс]. http://www. machinelearning.ru/wiki/images/0/0d/Voron-ML-Compositions.pdf

14. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Ma- chine.

15. Amazon Machine Learning. [Електронний ресурс] . - https://aws.amazon.com/ru/machine-learning/

16. Documentation Python.org. [Електронний ресурс] . - https://www.python.org/

17. Eli Bressert, – 1005 Gravenstein Highway North, Sebastopol : O'Reilly Media, Inc., 2012 p. 214.

18. Scikit-learn. Machine Learning in Python. [Електронний ресурс] . - http://scikit-learn.org

19. Лутц М., Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: СимволПлюс.

20. Лутц М., Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс.

21. UML-диаграммы классов. [Електронний ресурс]. - https://prog-cpp.ru/uml-classes/

22. Диаграмма последовательности (Sequence diagram). [Електронний ресурс]. - https://flexberry.github.io/ru/fd_sequence-diagram.html

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку «Модель аналізу рейтингу Сумського державного університету з використанням алгоритмів машинного навчання»

Суми 2019

1.1 Загальні вимоги.

Побудувати модель аналізу рейтингу Сумського державного університету з використанням алгоритмів машинного навчання для прогнозування та визначення важливості впливу кожного показника для фінального рейтингу інститутів, факультетів, кафедр університету.

1.2 Вимоги до роботи моделі.

Модель повинна правильно і точно прогнозувати вплив показника на фінальний рейтинг.

Також модель повинна працювати з даними, реагувати на шуми або випадкових вкидань, встановлювати статистичний взаємозв'язок двох або більше випадкових величин при необхідності.

1.3 Вимоги до апаратного забезпечення.

- встановлений Python 3;
- процесор з тактовою частотою 800MHz або більше;
- оперативна пам'ять 256 Мб або більше;
- вільне місце на жорсткому дискі від 50 Мб;
- архітектура з розрядністю 32 біт або 64 біт.

ДОДАТОК Б. ПЛАНУВАННЯ РОБІТ

Планування змісту структури робіт IT-проекту (WBS).

В основі планування проекту лежить складання структурованого переліку робіт, реалізація яких дозволяє досягти цілей проекту.

Суть декомпозиції - в поділі (розбитті) процесу на частини з того чи іншою ознакою: за типом виробленого під час робіт продукту, по функціональності, по етапах життєвого циклу і т.д .. Важливо, щоб такий поділ виділяла достатньо прості складові - більш керовані елементи, що допомагають краще контролювати реалізацію завдань і досягнення мети. Однак ієрархічна структура робіт відрізняється від просто графіка реалізації, від документа, аналогічного плану проекту та від переліку віх.

Структурної декомпозицією робіт (Work Breakdown Structure) називають представлення проекту, виконане у вигляді ієрархічної структури робіт, що досягається за допомогою послідовної декомпозиції. Інструмент спрямований на детальне планування, оцінку вартості, визначення та розподіл персональної відповідальності виконавців і т. Д. - тобто, на основні роботи і результати, що визначають зміст проекту.

Кожен нижчий рівень в ієрархії деталізує який-небудь елемент вищого рівня, але при цьому повинні бути дотримані такі принципи: Принцип «Правильного дерева». Ідея в тому, щоб у кожного залежного елементу (у кожної «гілки» або, при ще більш докладний поділ, - у кожного «листа») був тільки один батьківський елемент. Принцип повноти і логічної стрункості. У СДР повинні враховуватися всі елементи проекту, однак нічого не повинно дублюватися. Принцип єдності критерію. Процес декомпозиції повинен відбуватися за одним критерієм, тому, наприклад, не можна упереміж працювати відразу і з продуктом, і з функціональними завданнями. Принцип глибини СДР. Поділ має проводитися до тих пір, поки вийшла структура не буде легко керованою і контрольованою. Найчастіше цей процес закінчується

при структуруванні до рівня елементарної роботи - такий, яку здатний виконати один співробітник, або яку можна контролювати як окрему одиницю. Втілення цих принципів у проекті призводить до того, що ієрархічна структура робіт набуває певні характеристики, а саме: завжди визначає зміст робіт з необхідною (достатньої) точністю, охоплює весь обсяг - 100% - робіт за проектом (якщо якісь роботи не входять до СДР, значить, вони випадають і зі змісту), структура декомпозіруется на ієрархічно вибудувані елементи (пакети, субпакети і т. д.)., причому верхні рівні відповідають основним етапам проекту, і в міру наближення до нижніх рівнів визначення робіт се більш деталізується, блоки робіт в структурі автономні, роботи мають вимірний (або порівнянний) результат, а кожен блок робіт - вихідний результат (при цьому легко оцінюються час і витрати), стартові і завершальні заходи визначаються чітко і однозначно.

При цьому, нижнім рівнями пакетів робіт відповідають відносно невеликі обсяги робіт, що спрощує оцінювання параметрів виконання, а також допомагає ясно визначити дії, що призводять до досягнення проектних цілей. Такий підхід формує основу для встановлення вартості і трудомісткості як вимірних показників.

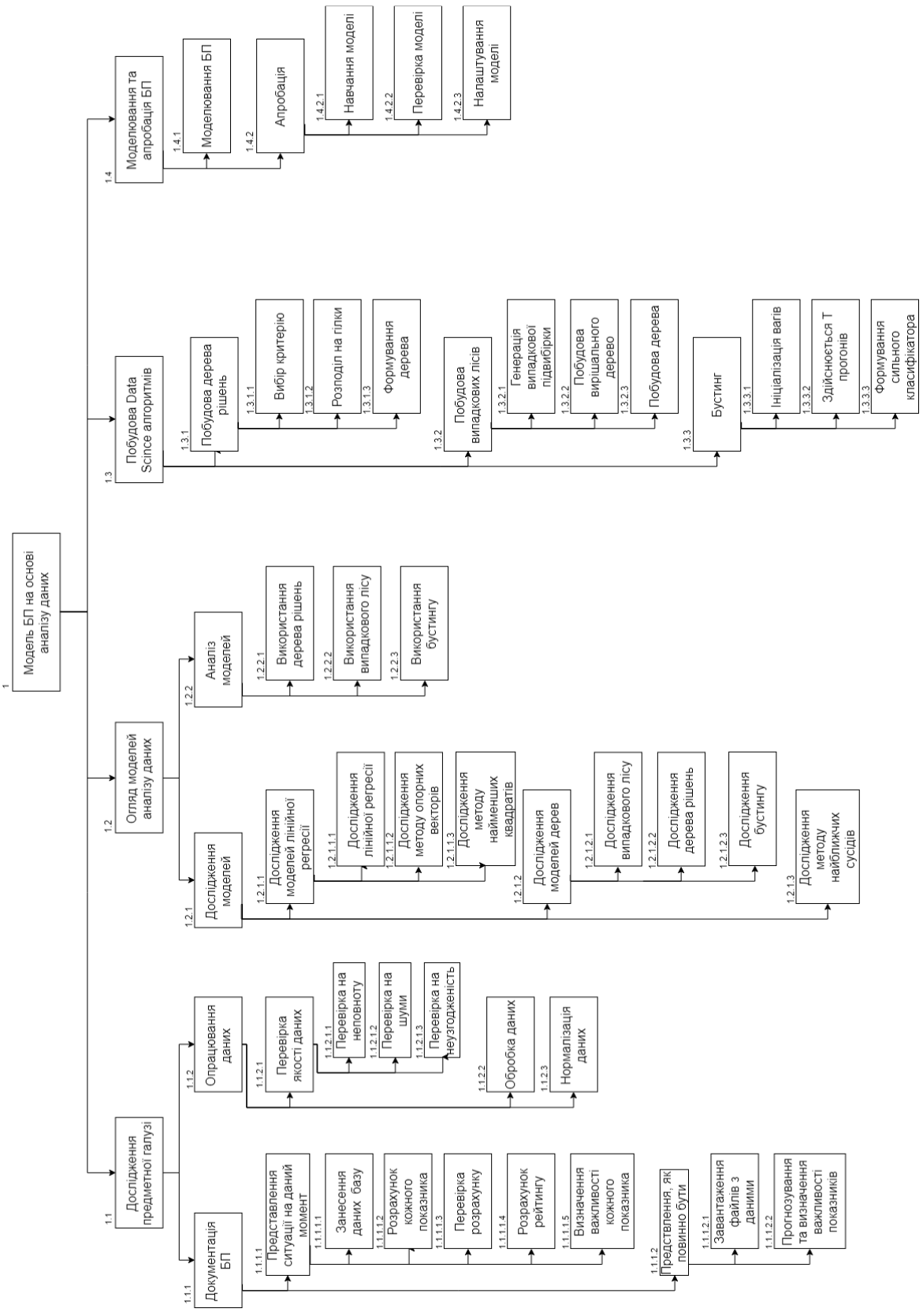

Рисунок Б.1 – таблиця WBS (work breakdown structure)

Організаційна структура проекту (OBS).

Термін організаційна структура організаційної структури терміна управління проектом (також званий трибуквених анаграмою OBS) відноситься конкретно до інструменту, який може використовуватися групою управління проектом і / або керівником групи управління проектом ієрархічним чином для цілей проведення та створення докладний і чітко окреслений опис організації проекту з метою створення угоди з метою встановлення відносин між різними робочими пакетами, пов'язаними з проектом, а також між цими робочими п кетамі і заздалегідь визначеними організаційними одиницями проекту. Важливо пам'ятати, що структура організаційної структури також записується і реєструється як організація, структура розбивки із застосуванням того ж визначення і зазвичай з використанням тієї ж трибуквених анаграми OBS. Організаційна структура повинна бути створена на початку діяльності, щоб допомогти в цілях організації; Проте, це можна проводити на постійній основі.

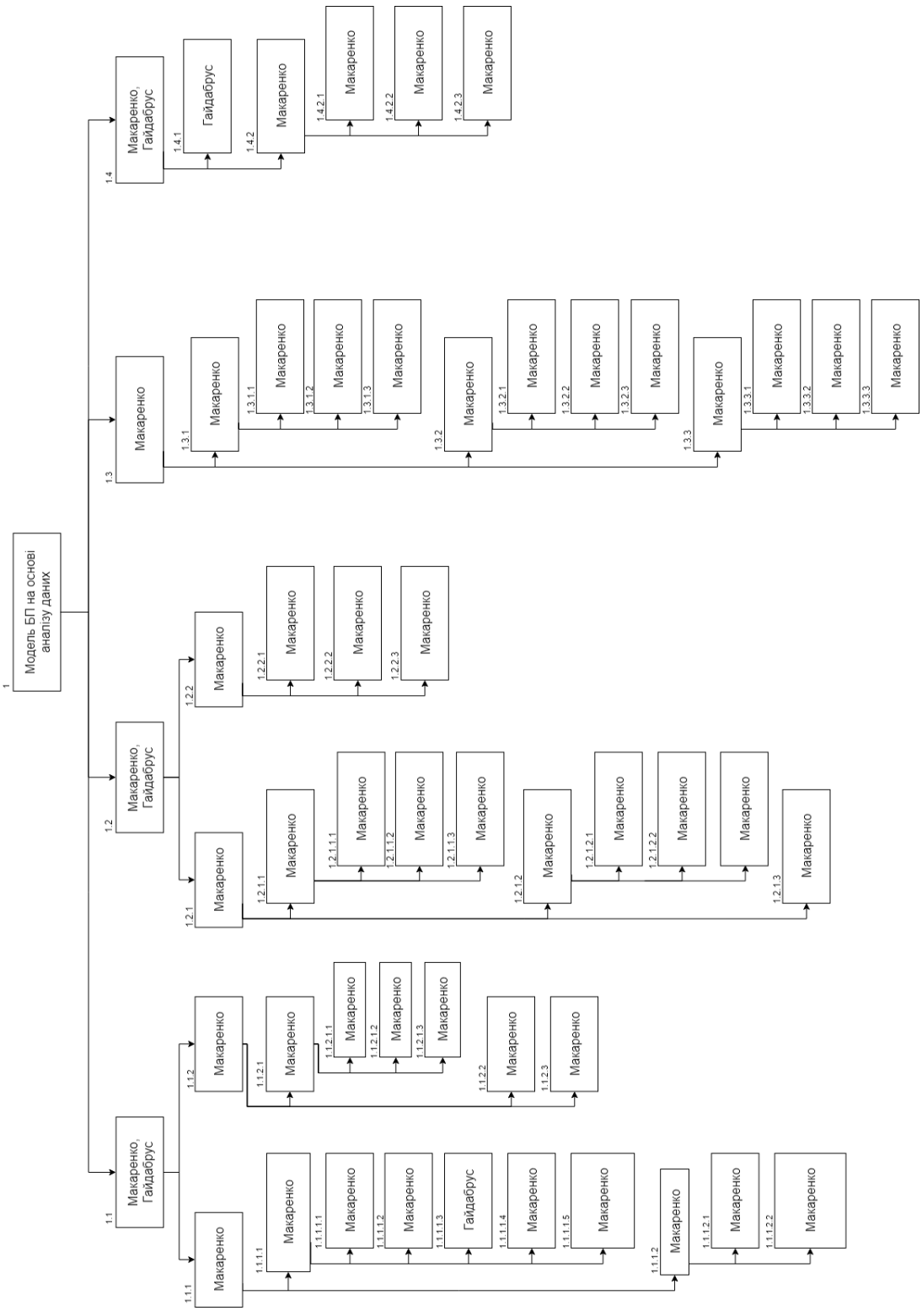

Рисунок Б.2 – таблиця OBS (Organization Breakdown Structure)

Побудова календарного графіка виконання ІТ-проекту.

Для визначення часу здійснення заходів, спрямованих на досягнення цілей проекту, і для встановлення взаємозв'язків між ними по тимчасовому параметру з урахуванням найбільш ризикових подій, складається календарний план проекту. Календарне планування полягає в створенні і подальшому уточненні розкладу, яке враховує склад робіт, ризики, обмеження.

В результаті створення календарного плану виходить повне проектне розклад, що враховує тривалість робіт і ресурсну базу, необхідну для виконання проекту. Календарне планування, в цілому, включає кілька основних стадій, серед яких: планування проектного змісту і побудова структури декомпозиції робіт, вибудовування послідовності робіт і мережевого графіка, складання плану термінів, тривалості, узгодження логічних зв'язків робіт і відображення їх на діаграмах Ганта або в таблицях, визначення ресурсних потреб (в персоналі, механізмах, матеріалах і т. д.) і складання плану використання ресурсів, розрахунок проектних трудовитрат і інших витрат.

Діграма Ганта представляється смугами, зорієнтовані уздовж осі (шкали) часу так, що кінець і початок кожної смуги відповідає часу початку і кінця роботи по виконанню завдання. Відповідно, довжина діаграми Ганта дозволяє визначити тривалість виробленої роботи. На інший (перпендикулярній) шкалою шикуються виконувані завдання. Стовпчики, що представляють ці завдання, взаємопов'язані між собою, а їх зв'язок відбивається фігурними стрілками.

За допомогою програми Microsoft Project Professional побудовано діаграму Ганта (рис. Б.3).

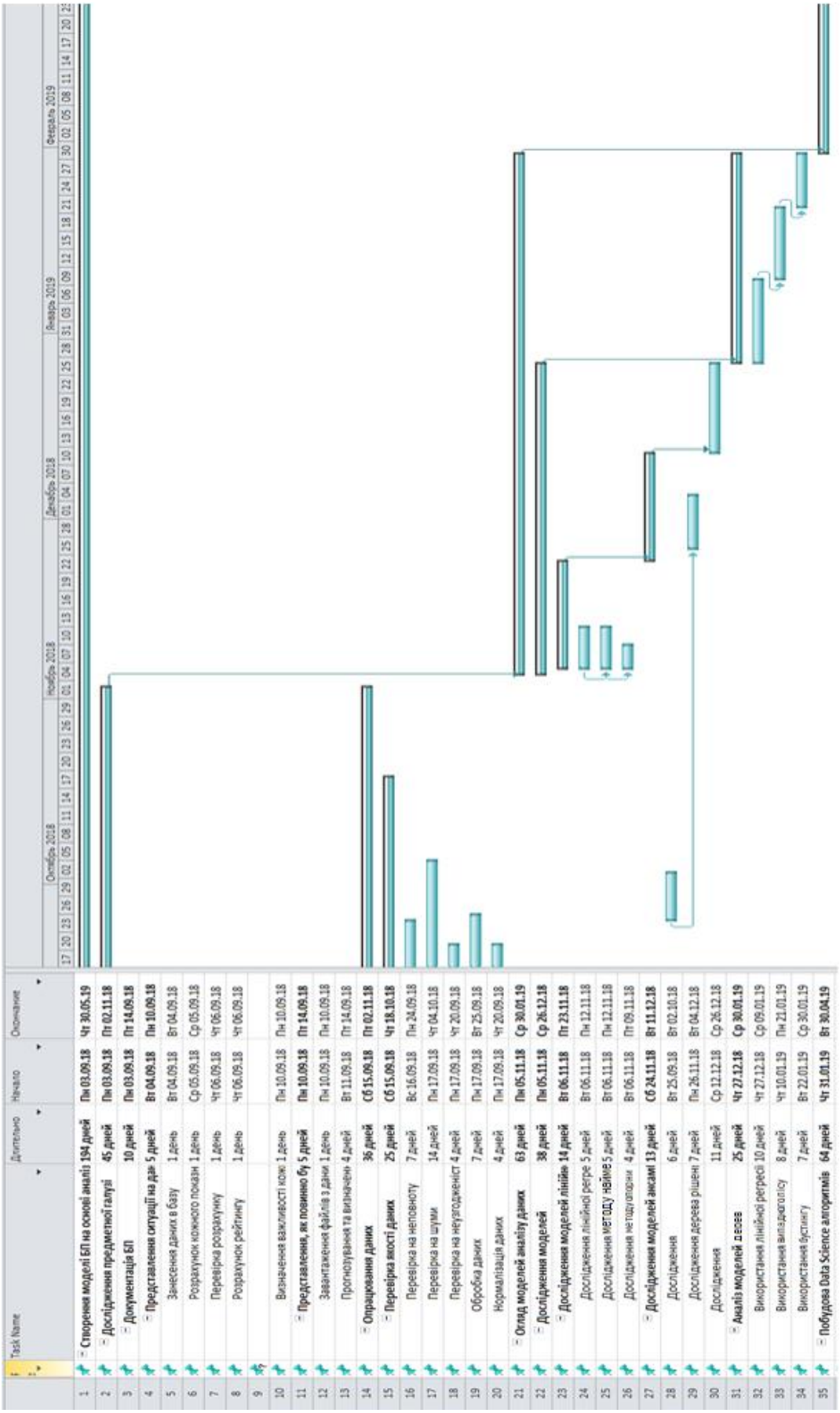

Рисунок Б.3 – Діаграма Ганта

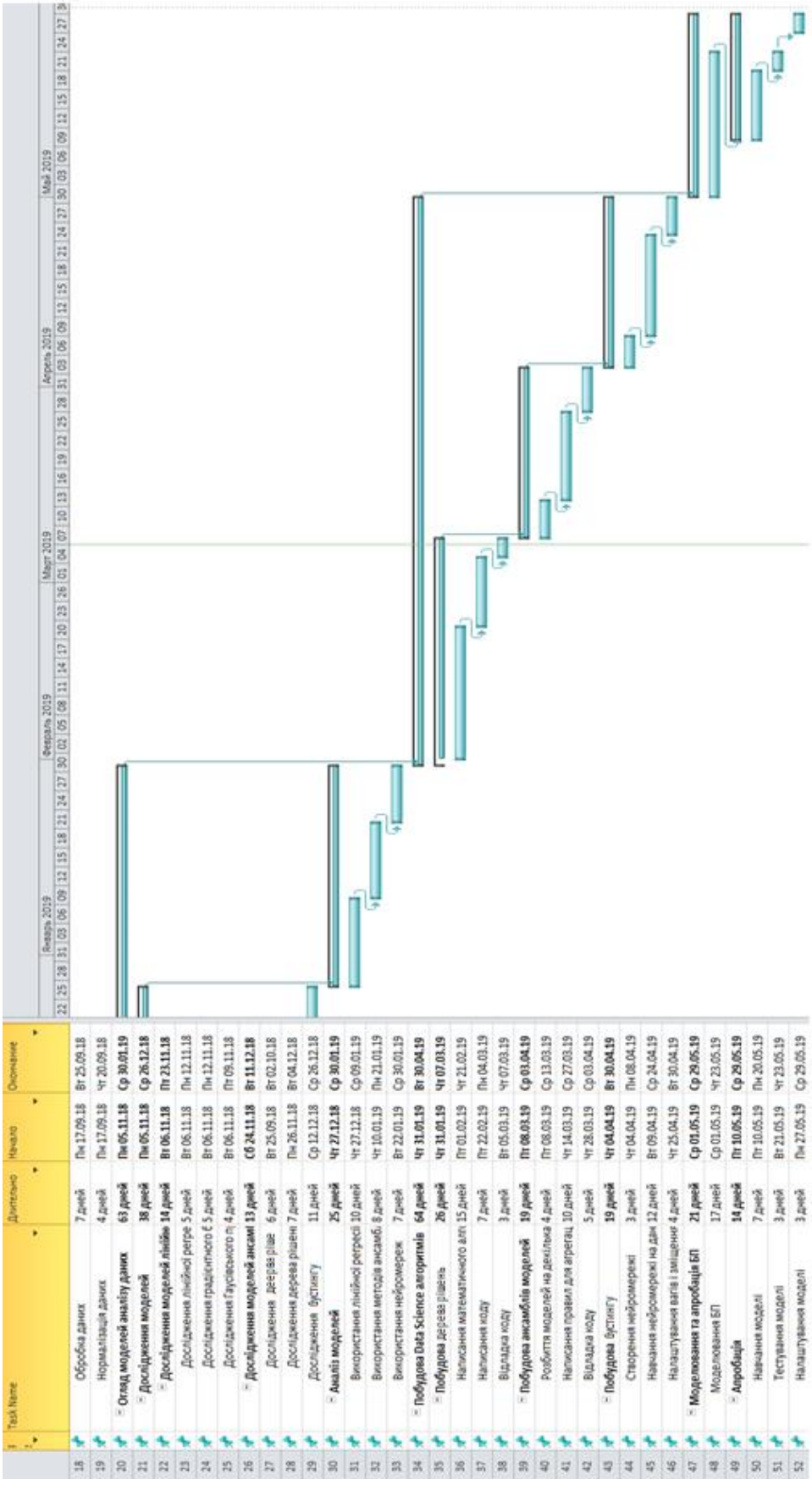

Рисунок Б.4 – Продовження діаграми Ганта

Управління ризиками.

До основних ризиків моделей аналізу рейтингу СумДУ з використанням алгоритмів машинного навчання:

− зміна цілей у ході реалізації проекту;

− зміна строків виконання роботи;

− не правильно оброблені дані;

− людський фактор;

− відсутність кваліфікованого програміста;

− зростання вимог до проекту;

Шкала оцінки ризику може відповідати емпіричній шкалі оцінки ризику(див табл. Б.1):

5 бали - критичний ризик (0,81 - 1);

4 бали - вагомий ризик (0,61 - 0,8);

3 бали – помірний ризик (0,41 - 0,6);

2 бали - незначний ризик (0,31 - 0,4);

1 бал - ігнорується ризик (0 - 0,3).

R = P * L, де R – рівень ризику; L – втрати, P – ймовірність виникнення.

Таблиця Б.1 – Ризики проекту

№	**Об’єкт ризику**	**P**	**L**	**R**
1	Зміна цілей проекту	0.5	0.4	0.2
2	Зміна строків проекту	0.5	0.5	0.25
3	Неправильно оброблені дані	0.7	0.8	0.56
4	Людський фактор	0.3	0.2	0.06
5	Відсутність кваліфікованого програміста	0.3	0.5	0.15
6	Зростання вимог до моделі	0.3	0.3	0.09

ДОДАТОК В. МОДЕЛЮВАННЯ

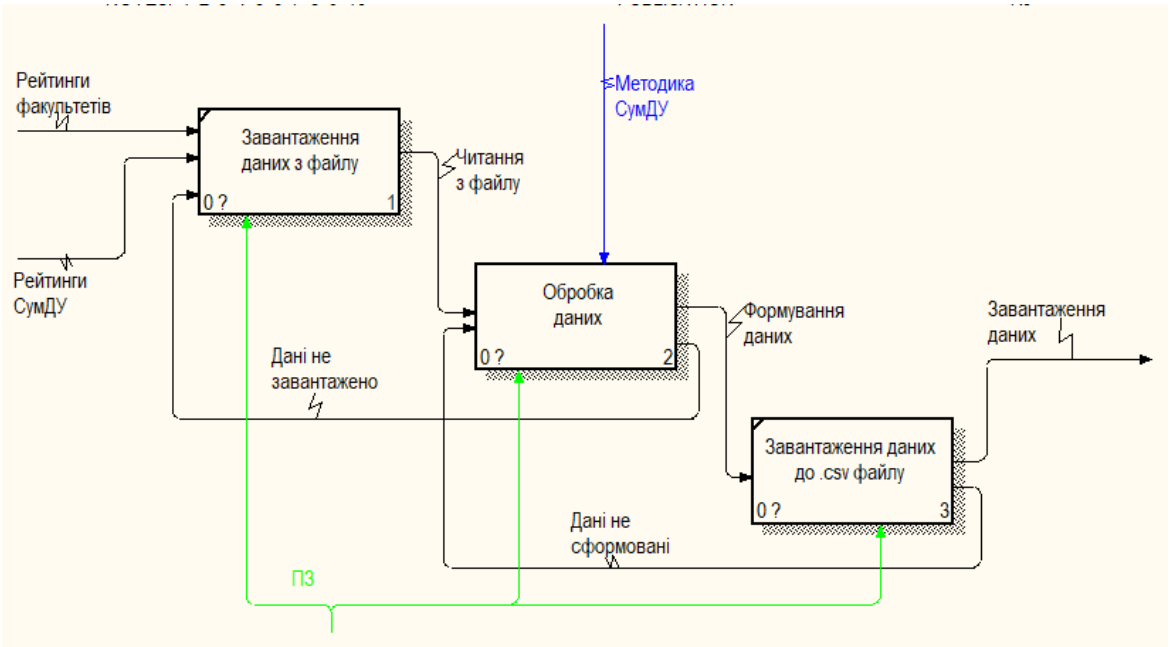

Рисунок В.1 – Декомпозиція блоку «Робота з даними»

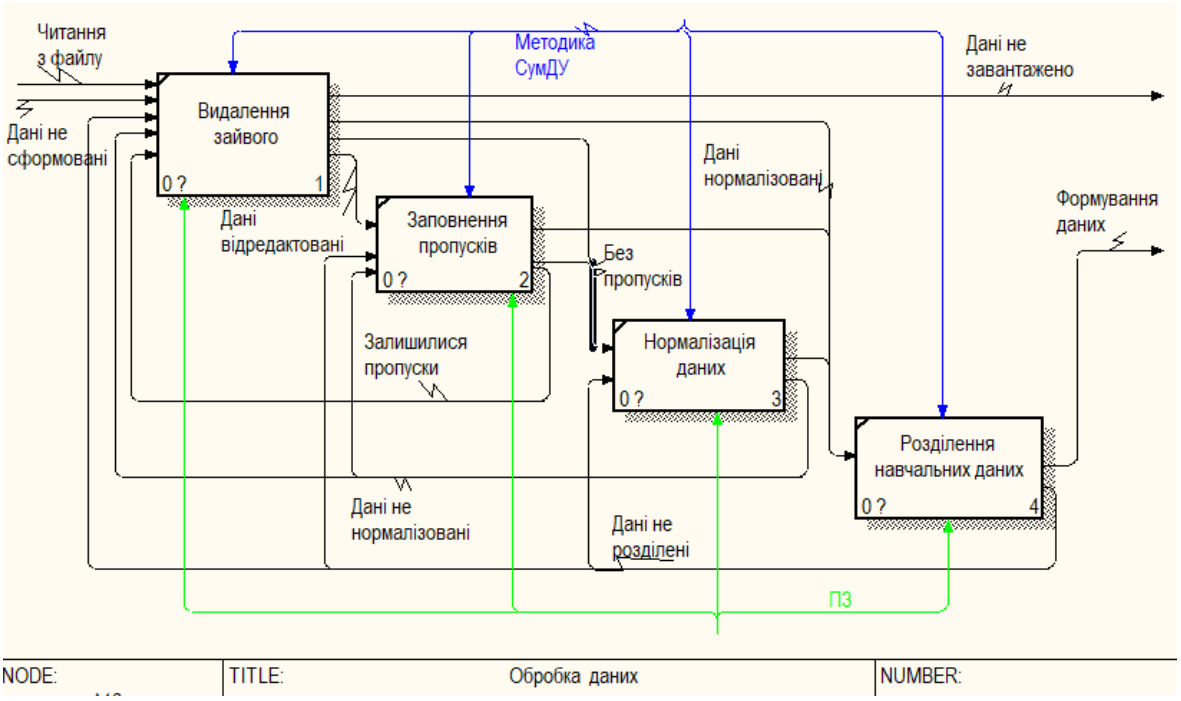

Рисунок В.2 – Декомпозиція блоку «Обробка даних»

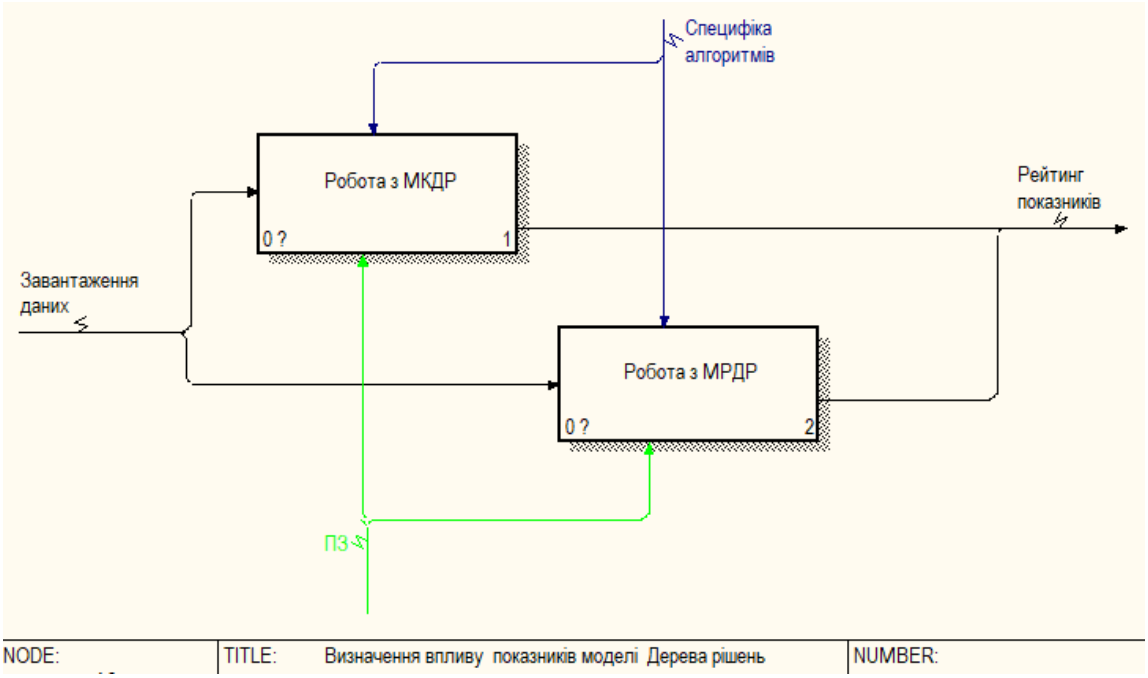

Рисунок В.3 – Декомпозиція блоку «Визначення впливу показників моделі дерева рішень»

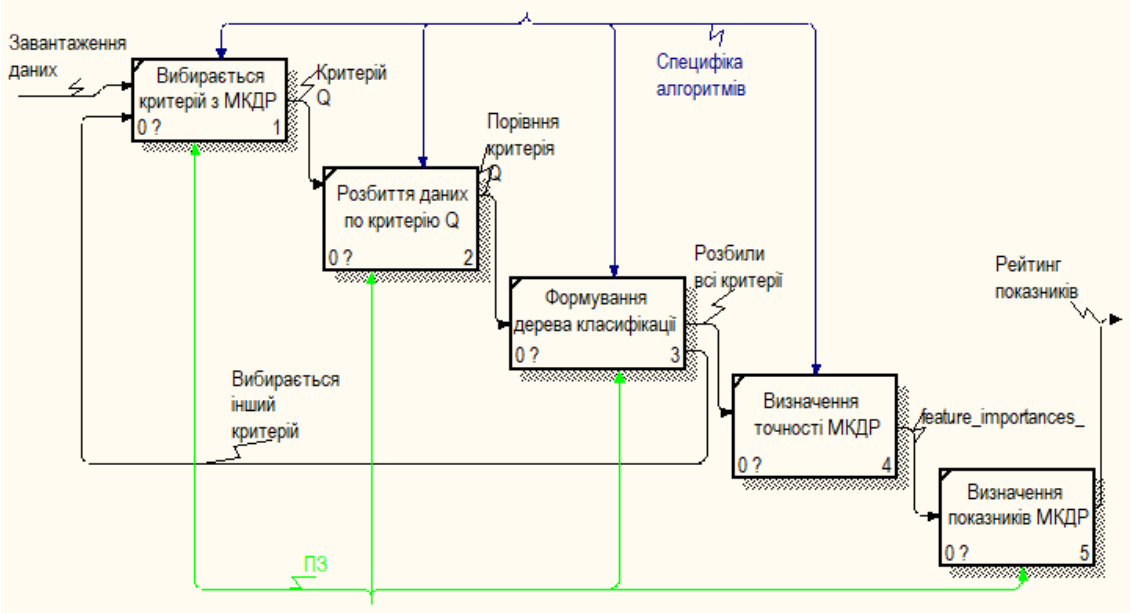

Рисунок В.4 – Декомпозиція блоку «Робота з МКДР»

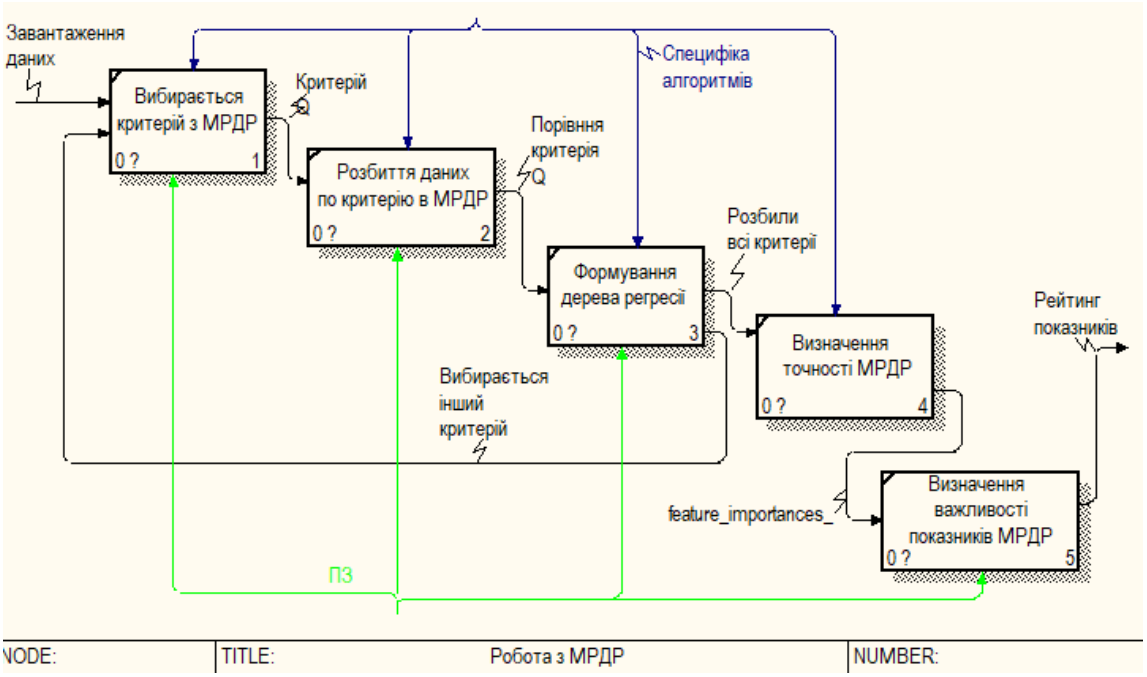

Рисунок В.5 – Декомпозиція блоку «Робота з МРДР»

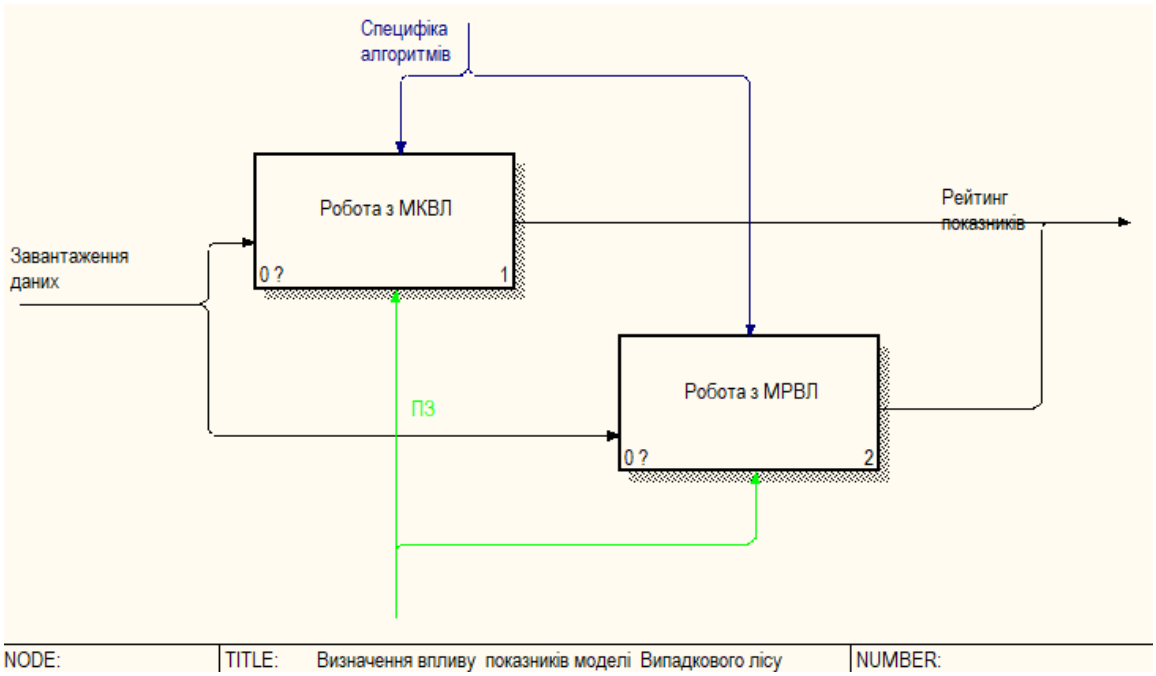

Рисунок В. 6– Декомпозиція блоку «Визначення впливу показників моделі випадкового лісу»

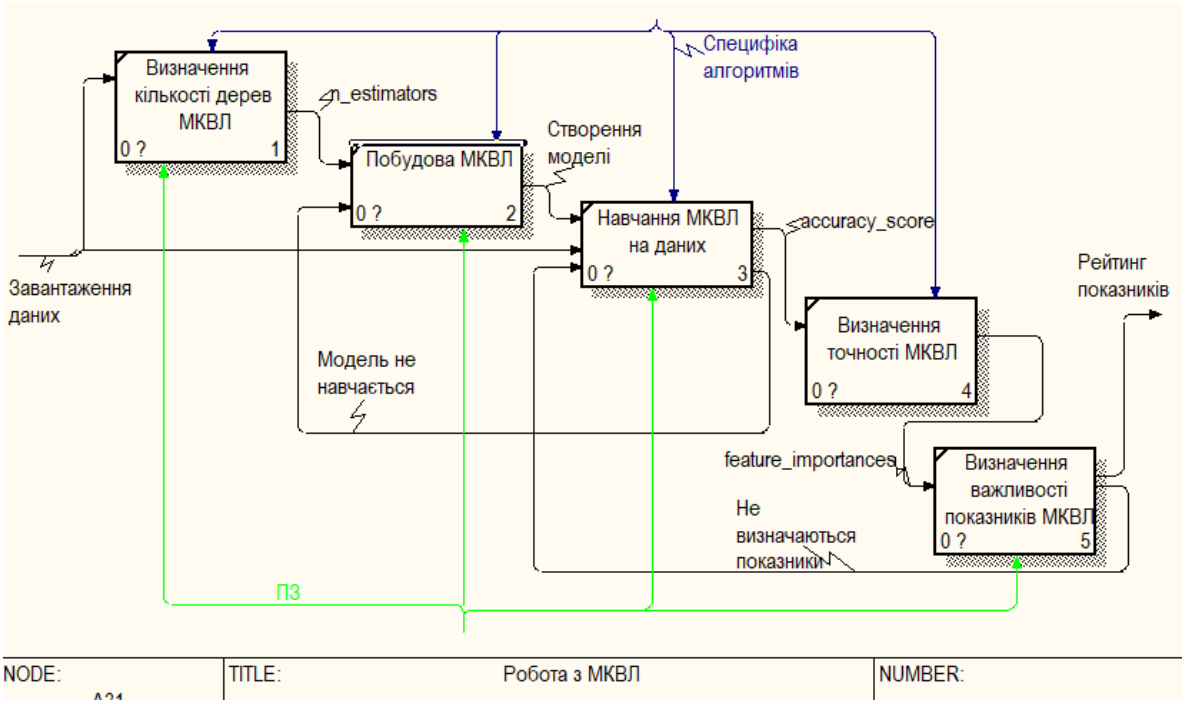

Рисунок В.7 – Декомпозиція блоку «Робота з МКВЛ»

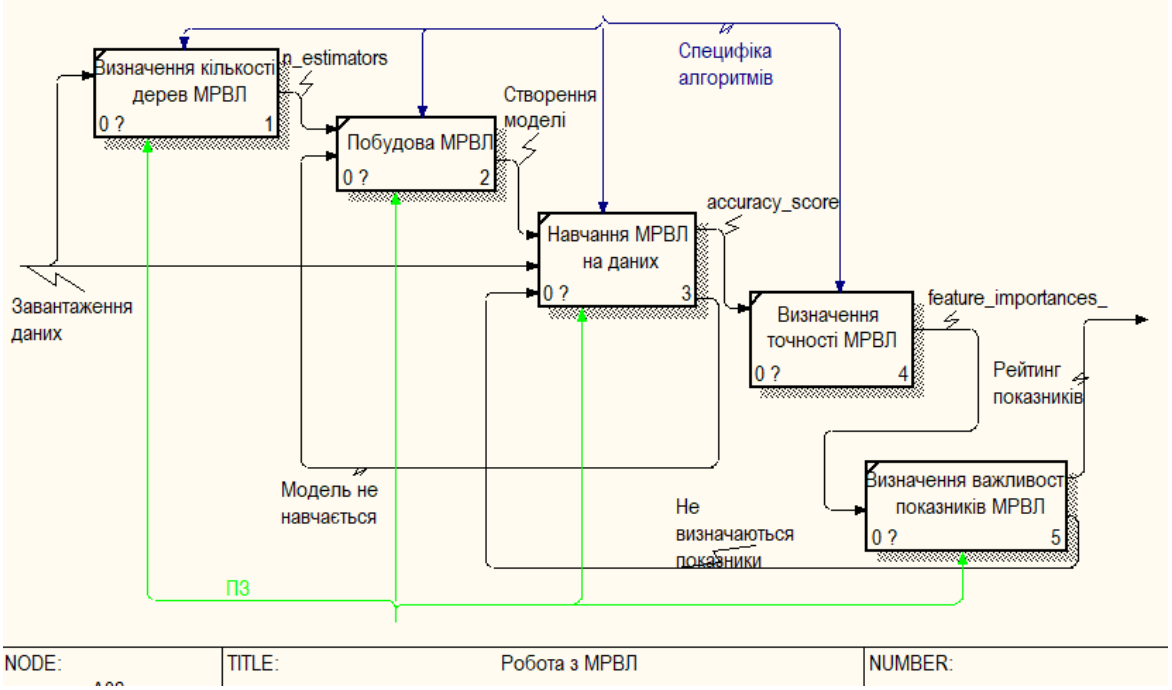

Рисунок В.8 – Декомпозиція блоку «Робота з МРВЛ»

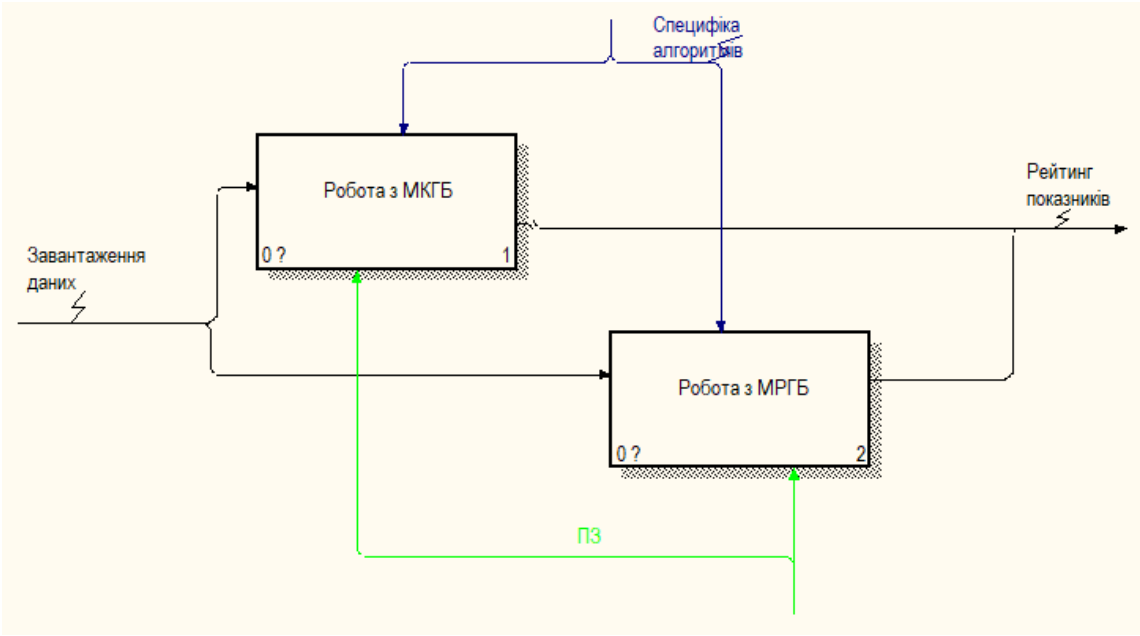

Рисунок В.9 – Декомпозиція блоку «Визначення впливу показників моделі градієнтного бустингу»

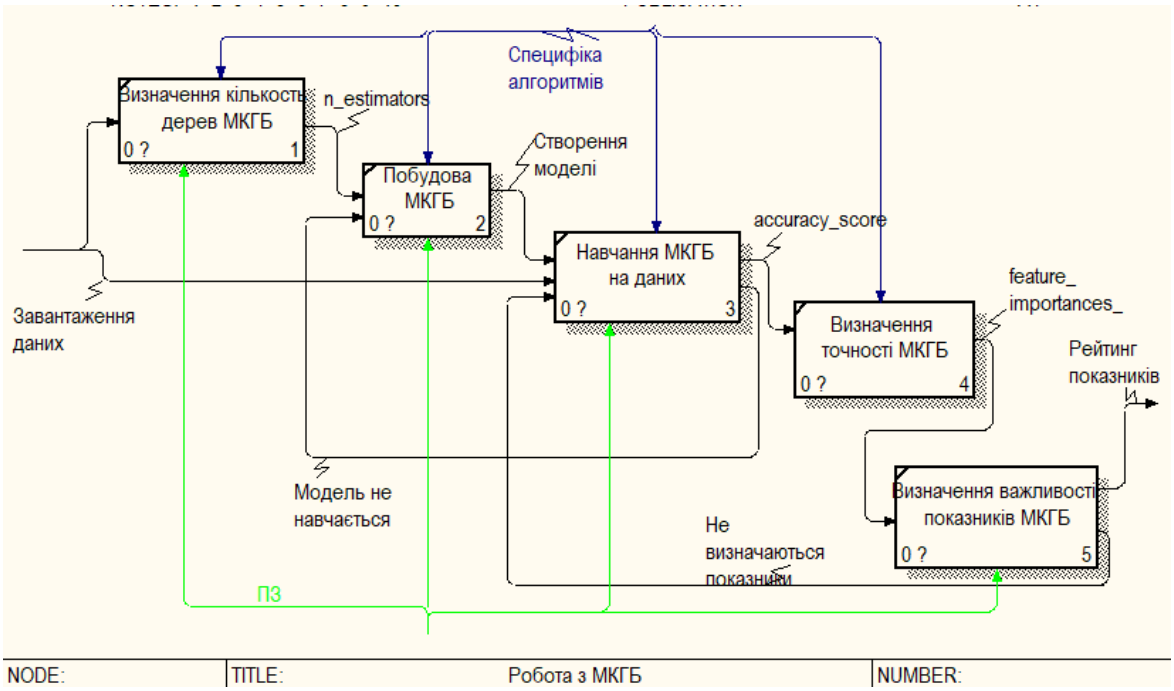

Рисунок В.10 – Декомпозиція блоку «Робота з МКГБ»

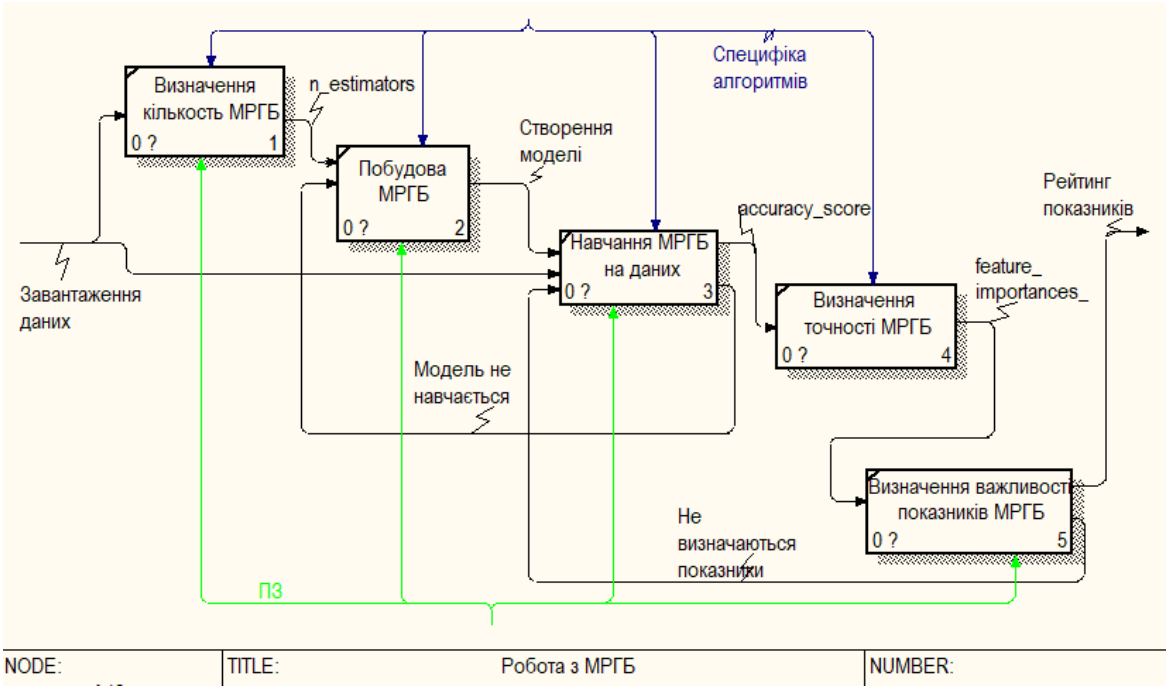

Рисунок В.11 – Декомпозиція блоку «Робота з МРГБ»

ДОДАТОК Д. ДАНІ КЛАСИФІКАЦІЇ ТА РЕГРЕСІЇ

Таблиця Д.1 – Узагальнені рейтингові дані показників.

№	Кафедра	Узагальнений рейтинговий рівень	Клас даних	Узагальнений рейтинговий показник
1.	НЕ	1	1	99.3
2.	МАіМО	НС	7	31.97
3.	ЕЗПФ	2	2	92.96
4.	Ел.Енергет	С	6	37.9
5.	ЕКТ	ВС	5	71.13
6.	ПМтаМСС	ВС	5	59.9
7.	КН	ВС	5	68.87
8.	ФіП	В	4	80.48
9.	ЕПБА	1	1	100
10.	управління	ВС	5	66.67
11.	МУІД	НС	7	32.76
12.	Іноз. мова	НС	7	29.19
13.	МПІГ	Н	8	18.74
14.	ЖФ	ВС	5	71.69
15.	ГФ	С	6	45.96
16.	ППСТ	ВС	5	60.70
17.	Філологія	НС	7	25.66
18.	КПТІДП	С	6	40.77
19.	КПДС	С	6	49.41
20.	АГПФЕБ	3	3	86.4
21.	МЄПЦПД	ВС	5	61.24
22.	ФЗНД КІ	Н	8	11.34
23.	ЕПА КІ	Н	8	20.1
24.	ЕУ КІ	Н	8	11.39
25.	Біофіз	С	6	38.04

Продовження таблиці Д.1.

26.	Хірургії	С	6	51.34
27.	Фізіології	С	6	46.79
28.	Патолог.	З	3	84.63
29.	Морфол	С	6	39.51
30.	ФВС	НС	7	31.2
31.	Внутр. Мед.	С	6	35.33
32.	Акушер	НС	7	31.91
33.	Інф.хвор	С	6	36.10
34.	Педіатрія	С	6	40.73
35.	Заг.хірур	НС	7	30.46
36.	Нейрохір	С	6	41.71
37.	Сім.мед.	С	6	45.76
38.	Гром.зд.	ВС	5	72.66
39.	ФРтаСМ	НС	7	32.23
40.	Стомат.	НС	7	28.07
41.	ЕУ ШІ	Н	8	17.19
42.	ХТВМС ШІ	Н	8	20.74
43.	СІТ ШІ	Н	8	16.18
44.	ФБСС	ВС	5	62.25
45.	МЕ	2	2	90.3
46.	ЕКіб	ВС	5	69.01
47.	ТПЕ	Н	8	20.17
48.	ІМ ННІ БТ	С	6	35.7
49.	БОО	ВС	5	60.01

ДОДАТОК Е. ВАЖЛИВІСТЬ ПОКАЗНИКІВ МОДЕЛЕЙ

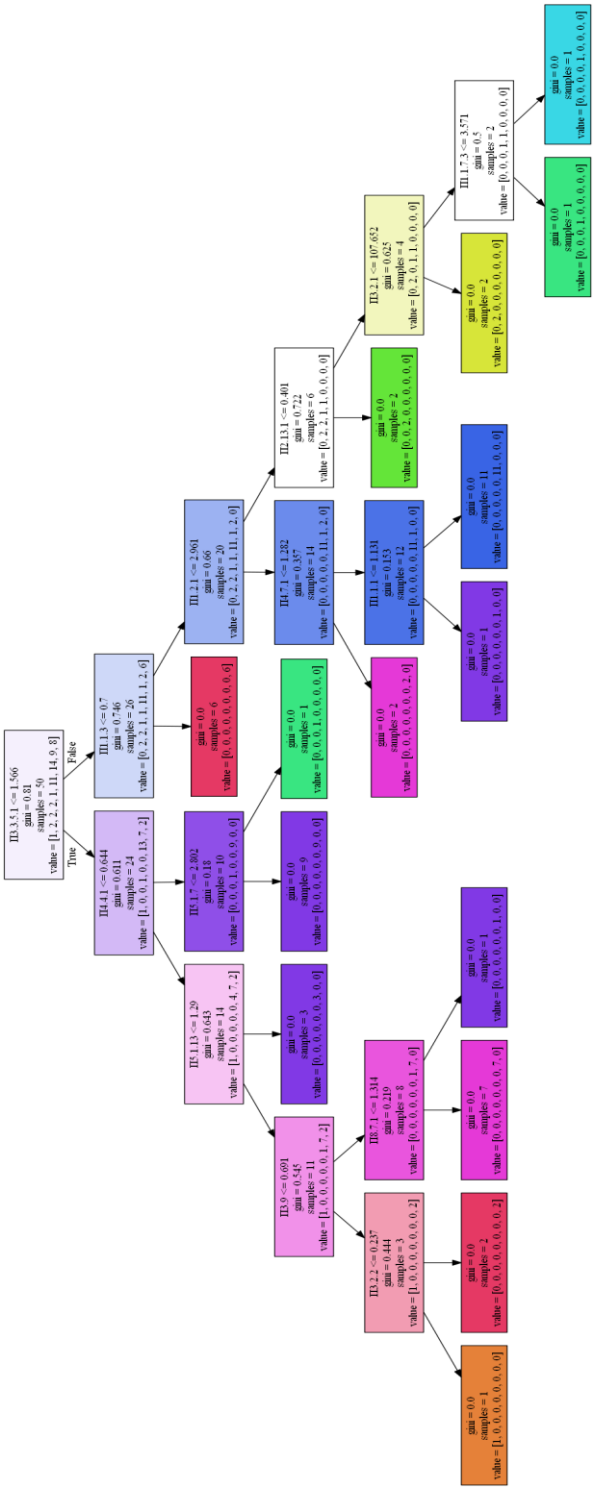

Рисунок Е.1 – Дерево рішень моделі класифікації дерева рішень

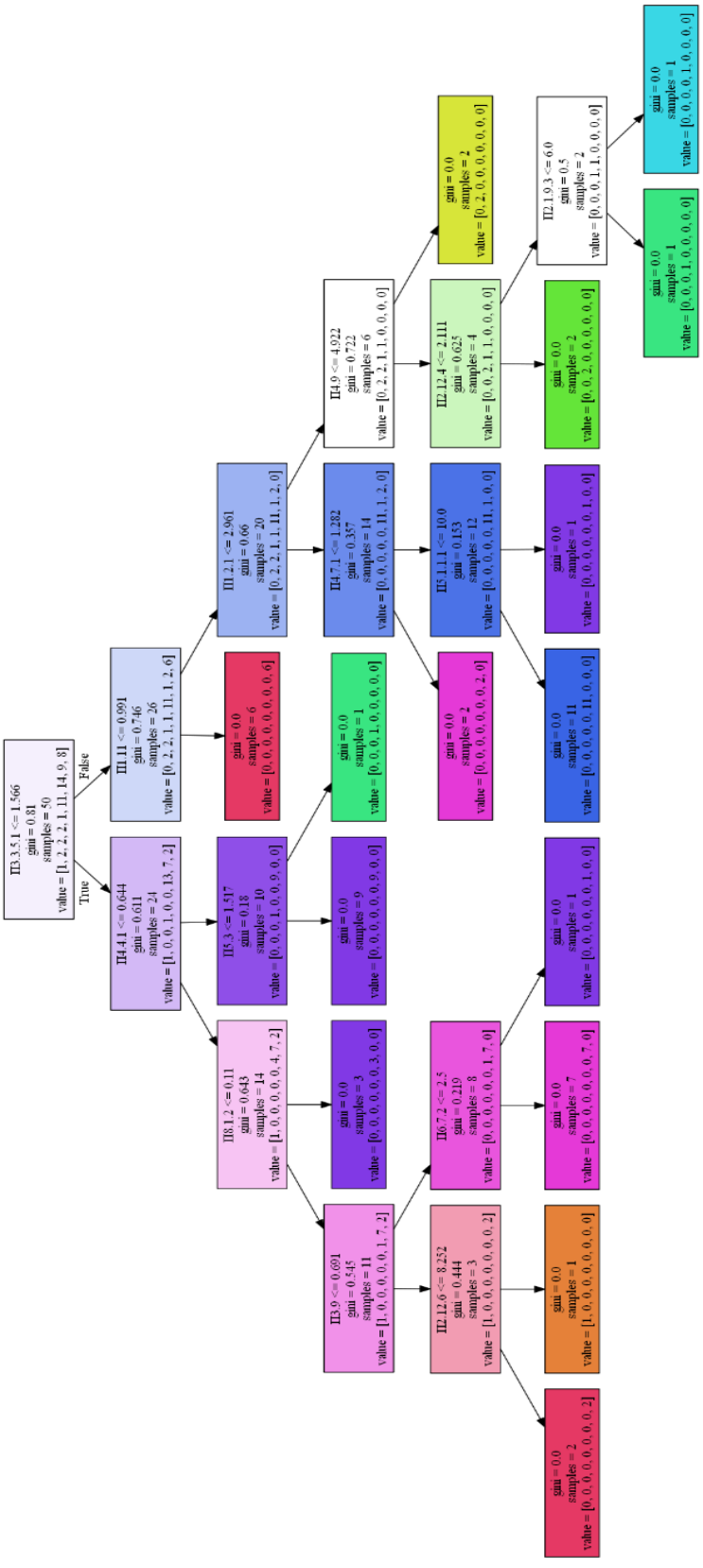

Рисунок Е.2 - Дерево рішень моделі регресії

Таблиця Е.1 – важливість показників рейтингу моделей дерева рішень.

	Модель класифікації дерева рішень		Модель регресії дерева рішень	
№	Показник	Важливість показнику у моделі	Показник	Важливість показнику у моделі
1.	П3.3.5.1	0.15881220229	П8.0	0.6643616
2.	П1.1.3	0.152782000608	П8.7.1	0.0938794
3.	П1.2.1	0.0955204216074	П5.1.10.2	0.0867474
4.	П4.4.1	0.0955204216074	П4.7	0.0308849
5.	П4.7.1	0.0782279314888	П5.1.4	0.0306992
6.	П5.1.13	0.0741106719368	П6.4.2	0.0265883
7.	П3.9	0.0720520421607	П2.10.1	0.0112015
8.	П1.1.1	0.0452898550725	П3.8	0.0095179
9.	П2.13.1	0.0452898550725	П8.2.7	0.0085724
10.	П5.1.7	0.0444664031621	П3.15	0.0068921
11.	П8.7.1	0.0432312252964	П5.8.1	0.0057353
12.	П3.2.1	0.0370553359684	П1.1.9	0.0056478
13.	П3.2.2	0.0329380764163	П3.14	0.0038398
14.	П1.1.7.3	0.0247035573123	П5.3	0.0026777
15.			П5.1.8	0.0022715
16.			П2.11.1	0.0022384
17.			П2.1.1	0.0021224
18.			П3.9	0.0015485
19.			П4.12.2	0.0012235
20.			П3.3.5.1	0.0009283
21.			П5.1	0.0006945
22.			П2.4.8	0.0005021
23.			П3.9.1	0.0002576
24.			П1.1.5.1	0.0001996
25.			П8.1.2	0.0001710
26.			П2.1.9.2	0.0001175
27.			П5.1.1.4	0.0001158
28.			П2.12.6	0.0001028

Продовження таблиці Е.1.

29.			П5.1.13	0.0000638
30.			П1.1.7	0.0000610
31.			П3.3.1	0.0000513
32.			П1.1.3	0.0000189
33.			П2.4.1	0.0000167
34.			П4.1.5.1	0.0000157
35.			П2.4.9	0.0000121
36.			П5.3*	0.0000086
37.			П8.7	0.0000080
38.			П4.4.1	0.0000026
39.			П8.3.3	0.0000018
40.			П2.1.2	0.0000003
41.			П3.2.2	0.0000002
42.			П5.2.1	0.0000001

Таблиця Е.2 – важливість показників рейтингу моделей випадкового лісу.

	Модель класифікації випадкового лісу		Модель регресії випадкового лісу	
№	Показник	Важливість показнику у моделі	Показник	Важливість показнику у моделі
1.	П1.3.3	0.0391224279223	П8.0	0.4603563
2.	П1.1.3	0.0377797883906	П8.1.2	0.0847576
3.	П5.1	0.0306918664007	П1.1.9.1	0.0580098
4.	П8.0	0.0301512709042	П4.7	0.0492963
5.	П2.4.8	0.0282630207314	П8.7.1	0.0327665
6.	П8.7.1	0.0217163988512	П5.1.3	0.0238262
7.	П3.9	0.0210908903104	П5.1.7	0.0211157
8.	П3.13	0.0206202539556	П6.4.2	0.0191970
9.	П1.1.1	0.0198558854335	П5.1.8	0.0149365
10.	П5.6.1	0.0197718856237	П5.1.10.2	0.0147401

Продовження таблиці Е.2.

11.	П3.3	0.0194266694866	П8.2.7	0.0135050
12.	П2.4.9	0.018094967577	П1.2.1	0.0134269
13.	П5.1.4	0.0173842179338	П3.9.1	0.0125503
14.	П5.1.7	0.0173502024102	П5.1.4	0.0105636
15.	П1.1.10	0.016518125788	П5.8.1	0.0099022
16.	П2.10.1	0.0161733226661	П1.1.2	0.0079286
17.	П3.15	0.0160789532693	П4.4.1	0.0078735
18.	П1.1	0.0159853537349	П1.3.3	0.0069574
19.	П3.2.1	0.0154503019013	П1.1.3	0.0064036
20.	П3.3.5.1	0.0141889124183	П5.6	0.0062182
21.	П2.4.2	0.0141592765609	П1.1.1	0.0058443
22.	П5.1.8	0.014079591048	П3.15	0.0054410
23.	П5.1.11	0.0131364998188	П1.11	0.0053603
24.	П3.3.1	0.0130619403819	П3.2.1	0.0048743
25.	П5.1.3	0.0125745822819	П3.14	0.0048319
26.	П5.6	0.0124820487715	П6.4.2.1	0.0043756
27.	П3.3.5	0.0122819102825	П5.1.13	0.0036199
28.	П5.1.13	0.0115928418598	П5.1.1.2	0.0034405
29.	П1.11	0.0109418941374	П4.3.1	0.0034161
30.	П4.6	0.0105137870641	П1.1.6	0.0033505
31.	П2.1.1	0.0103788904736	П5.3*	0.0033094
32.	П4.7	0.0103316489054	П5.5.1	0.0032098
33.	П6.3.1	0.00995710030699	П8.3.2	0.0031842
34.	П6.4.2	0.00971243912191	П5.1	0.0029340
35.	П8.1.2	0.0096061829187	П3.3	0.0028518
36.	П2.1	0.00935902882375	П5.1.11	0.0027271
37.	П3.2.2	0.00935588757003	П6.2	0.0025900
38.	П1.1.3.1	0.0092504694463	П1.1.10	0.0024928
39.	П8.2.1	0.00827339926784	П2.5.2	0.0022874

Продовження таблиці Е.2.

40.	П8.3.2	0.00823583996321	П5.3	0.0022324
41.	П2.4	0.00774920930513	П1.1	0.0021184
42.	П4.3.4	0.00768862470406	П8.2.1	0.0019001
43.	П5.8.1	0.00760571812804	П2.10.2	0.0017985
44.	П3.14	0.00740260779933	П2.12.7	0.0017926
45.	П5.3	0.00673154209744	П6.1.2	0.0017355
46.	П1.1.7	0.00670271087311	П6.5.2	0.0016337
47.	П4.3.4.1	0.00657949591052	П1.1.9	0.0015821
48.	П6.5.2	0.0065518712475	П3.3.1	0.0015674
49.	П4.3.1	0.00624235100818	П4.12.3	0.0015634
50.	П5.1.9	0.00615146597548	П5.2.2	0.0015526
51.	П1.2.1	0.00605540053301	П1.1.1.1	0.0015207
52.	П4.11.3	0.00593071965209	П4.9	0.0013796
53.	П3.3.2	0.00590793792501	П4.12.1	0.0012969
54.	П3.10.3	0.00587183710068	П6.1.1	0.0010987
55.	П2.12.7	0.00571940401035	П3.9	0.0010097
56.	П8.2.7	0.00564299993082	П4.6	0.0009987
57.	П2.4.6	0.00563347016573	П6.3.2	0.0009494
58.	П5.2.1	0.00557230006717	П2.1	0.0009466
59.	П2.11.2	0.00555985646605	П1.1.7	0.0008803
60.	П4.7.1	0.00550648351294	П4.9.1	0.0008118
61.	П2.13.1	0.00549722135989	П2.12.2	0.0007841
62.	П5.6.2	0.00543143410715	П2.12.1	0.0007552
63.	П6.6.1	0.0052420475637	П5.6.1	0.0007180
64.	П4.4.2	0.00521457479816	П8.7	0.0007127
65.	П4.9.1	0.00517835581043	П2.4.9.2	0.0006839
66.	П8.2.8	0.00491471180558	П2.10.1	0.0006567
67.	П2.1.2	0.00469498145718	П5.2.1	0.0006294
68.	П4.4.1	0.00467504053297	П3.6	0.0006049

Продовження таблиці Е.2.

69.	П4.1.5	0.00460268035334	П2.4.1	0.0005759
70.	П1.1.1.1	0.00446947004855	П3.3.5.1	0.0005443
71.	П3.11.2	0.00427604818114	П8.8	0.0005420
72.	П6.4.2.1	0.00418233859154	П6.6.1	0.0005297
73.	П2.4.1	0.00411375031933	П2.4.6	0.0005296
74.	П4.9	0.00408611562164	П2.12.8	0.0005166
75.	П2.4.10	0.00386071433289	П1.1.11	0.0004902
76.	П5.2.2	0.00384969921727	П2.1.9.3	0.0004677
77.	П8.7	0.00378816389151	П5.1.10	0.0004673
78.	П4.11.1	0.00378639711769	П3.13	0.0004497
79.	П8.3.3	0.00370053349712	П4.11.3	0.0004406
80.	П6.2	0.00369942154434	П2.5.4.1	0.0004218
81.	П5.1.10.2	0.00366116528725	П1.1.3.1	0.0004213
82.	П1.1.6	0.00366065248291	П2.4.6.1	0.0004210
83.	П6.1.2	0.00362683610048	П4.1.5	0.0004078
84.	П6.7.2	0.00357672892384	П3.2.2	0.0004062
85.	П6.3.2	0.00347617033718	П2.4.9	0.0003787
86.	П2.5.2	0.00346307566349	П8.3.3	0.0003667
87.	П1.1.2	0.00344826021661	П2.4.5	0.0003583
88.	П2.4.9.2	0.00339780319558	П2.4	0.0003576
89.	П3.9.1	0.00332734585387	П2.1.1	0.0003430
90.	П1.1.7.3	0.0032452698963	П3.3.2	0.0003322
91.	П4.12.2	0.0032235098798	П8.2.2	0.0003168
92.	П7.4.1	0.00309102723585	П4.1.1	0.0003164
93.	П3.6	0.00306629272328	П5.5.3	0.0003075
94.	П2.12.6	0.00306390909209	П2.13.1	0.0003022
95.	П2.4.2.1	0.00305166468308	П2.11.2	0.0002913
96.	П2.12.4	0.002828352957	П2.4.8	0.0002854
97.	П2.5.4	0.00265625212161	П2.12.6	0.0002684

Продовження таблиці Е.2.

98.	П2.10.2	0.00265080425401	П1.3.2	0.0002597
99.	П8.8	0.00260369653378	П3.10.3	0.0002586
100.	П2.5.3	0.00255216627874	П2.4.4.1	0.0002517
101.	П1.2.2	0.00242026104244	П2.12.3	0.0002423
102.	П6.1.1	0.00239672310125	П4.3.4	0.0002360
103.	П2.12.3	0.00237670825906	П2.4.4	0.0002111
104.	П2.4.3	0.00235875235875	П6.4.3	0.0002094
105.	П2.12.8	0.00235577001186	П2.4.3	0.0002030
106.	П2.4.4.1	0.00234936428966	П3.8	0.0001930
107.	П2.5.1	0.00227214497881	П3.3.5	0.0001914
108.	П5.5.1	0.00225722138859	П2.4.2	0.0001874
109.	П5.1.1.4	0.00214114422748	П2.1.9.2	0.0001836
110.	П1.1.9.1	0.00201625411006	П2.1.6	0.0001777
111.	П2.1.9.1	0.00191307009489	П1.1.7.3	0.0001764
112.	П1.1.9	0.00191033428781	П8.6.1	0.0001742
113.	П2.12.5	0.00188879164646	П8.2.5	0.0001720
114.	П5.7.1	0.00184815554077	П5.1.1.3	0.0001709
115.	П3.4	0.00182748538012	П6.3.1	0.0001577
116.	П3.12	0.00177195288538	П8.4.2	0.0001572
117.	П2.4.6.1	0.00172815251401	П1.1.12	0.0001519
118.	П4.11.2	0.00167393633303	П4.16	0.0001472
119.	П5.1.1.2	0.00163579110948	П2.1.3	0.0001462
120.	П2.8	0.00159795461809	П2.8	0.0001380
121.	П1.3.1	0.00157470395566	П4.8	0.0001362
122.	П5.1.1.3	0.00154630248151	П3.10.1	0.0001362
123.	П2.1.7	0.00153680651606	П2.6	0.0001344
124.	П4.1.5.2	0.00152868573921	П7.4.1	0.0001332
125.	П3.8	0.00151083484417	П1.1.7.2	0.0001275
126.	П8.2.2	0.00148892611204	П8.2.6	0.0001255

Продовження таблиці Е.2.

127.	П5.3*	0.00146312313644	П5.1.9	0.0001087
128.	П6.6.2	0.0014558583741	П2.2	0.0001082
129.	П2.6	0.0014497796335	П8.4.3	0.0001078
130.	П1.1.12	0.00139867964641	П4.3.4.1	0.0001052
131.	П5.8.2	0.00136433093211	П2.4.9.1	0.0001035
132.	П4.1.4	0.00136115534866	П2.5.1	0.0000805
133.	П1.1.11	0.00128067209672	П6.4.2.2	0.0000799
134.	П2.1.3	0.000956535047444	П2.1.7	0.0000792
135.	П4.15	0.000944822373394	П7.3.1	0.0000729
136.	П2.11.1	0.000925000925001	П2.7.1	0.0000688
137.	П2.4.5	0.000924077770385	П4.5	0.0000610
138.	П5.6.3	0.00092131932928	П4.1.4	0.0000606
139.	П2.4.9.1	0.000887752565605	П2.4.4.2	0.0000597
140.			П2.12.4	0.0000558
141.			П5.7.2	0.0000540
142.			П8.2.8	0.0000524
143.			П5.7.1	0.0000498
144.			П4.11.1	0.0000496
145.			П1.1.5.2	0.0000257
146.			П2.11.1	0.0000250
147.			П6.7.2	0.0000214
148.			П4.11.2	0.0000202
149.			П4.3.3	0.0000198
150.			П2.12.5	0.0000187
151.			П4.1.5.1	0.0000169
152.			П2.1.2	0.0000166
153.			П4.10	0.0000160
154.			П4.7.1	0.0000152
155.			П4.15	0.0000151

Продовження таблиці Е.2.

156.			П3.11.2	0.0000083
157.			П2.4.2.1	0.0000078
158.			П2.5.3	0.0000074
159.			П2.1.9.1	0.0000039
160.			П6.7.1	0.0000029
161.			П2.1.5	0.0000023
162.			П8.4.4	0.0000020
163.			П6.6.2	0.0000020
164.			П2.5.4	0.0000019
165.			П6.4.1	0.0000019
166.			П1.1.4	0.0000013
167.			П3.11.1	0.0000006
168.			П3.4	0.0000004
169.			П4.4.2	0.0000004

Таблиця Е.3 – важливість показників рейтингу моделей градієнтного бустингу.

	Модель класифікації градієнтного бустингу		Модель регресії градієнтного бустингу	
№	Показник	Важливість показнику у моделі	Показник	Важливість показнику у моделі
1.	П8.2.1	0,060932	П8.0	0,080402
2.	П1.3.3	0,057348	П1.2.1	0,070352
3.	П3.3.5.1	0,053763	П1.1.11	0,060302
4.	П1.1	0,043011	П3.2.1	0,055276
5.	П3.2.1	0,043011	П8.7.1	0,050251
6.	П5.1.4	0,043011	П5.1.4	0,045226
7.	П1.1.3	0,043011	П1.11	0,035176
8.	П1.1.1	0,039427	П1.1.3	0,035176
9.	П8.0	0,039427	П1.1	0,030151
10.	П1.11	0,028674	П1.1.2	0,030151

Продовження таблиці Е.3.

11.	П5.1	0,028674	П6.4.2	0,030151
12.	П1.2.1	0,028674	П5.1	0,030151
13.	П1.1.2	0,021505	П3.9	0,025126
14.	П3.9	0,021505	П6.4.2.1	0,025126
15.	П3.3	0,021505	П3.3	0,025126
16.	П4.12.1	0,021505	П8.2.7	0,025126
17.	П5.1.13	0,021505	П1.1.1.1	0,020101
18.	П1.1.10	0,021505	П2.10.1	0,020101
19.	П4.6	0,017921	П5.1.7	0,020101
20.	П3.9.1	0,017921	П5.8.1	0,020101
21.	П8.4.2	0,014337	П5.1.1.3	0,020101
22.	П4.7.1	0,014337	П4.7.1	0,015075
23.	П1.1.9	0,010753	П6.2	0,015075
24.	П4.4.1	0,010753	П3.3.2	0,015075
25.	П4.1.5	0,010753	П5.1.8	0,015075
26.	П6.4.2.1	0,010753	П4.6	0,015075
27.	П5.1.3	0,010753	П8.1.2	0,015075
28.	П2.10.1	0,010753	П2.1.9.3	0,015075
29.	П5.1.7	0,010753	П5.1.10.2	0,01005
30.	П6.4.2	0,010753	П5.2.1	0,01005
31.	П5.1.10.2	0,010753	П1.1.1	0,01005
32.	П1.1.6	0,007168	П1.1.7.3	0,01005
33.	П6.5.2	0,007168	П3.15	0,01005
34.	П1.1.1.1	0,007168	П1.1.3.1	0,01005
35.	П8.2.2	0,007168	П2.2	0,01005
36.	П4.9.1	0,007168	П8.4.2	0,005025
37.	П4.3.1	0,007168	П5.1.10	0,005025
38.	П2.4.2.1	0,007168	П1.12.2	0,005025
39.	П3.6	0,007168	П3.3.5	0,005025
40.	П2.4.5	0,007168	П4.7	0,005025
41.	П2.1.2	0,007168	П3.2.2	0,005025

Продовження таблиці Е.3.

42.	П3.3.5	0,007168	П1.1.12	0,005025
43.	П6.7.2	0,007168	П4.3.4	0,005025
44.	П5.6	0,007168	П5.1.3	0,005025
45.	П8.8	0,007168	П2.4.1	0,005025
46.	П5.1.10	0,003584	П5.1.11	0,005025
47.	П5.1.11	0,003584	П1.1.6	0,005025
48.	П1.1.7	0,003584	П2.4.6	0,005025
49.	П8.3.2	0,003584	П3.13	0,005025
50.	П5.1.8	0,003584		
51.	П3.2.2	0,003584		
52.	П3.15	0,003584		
53.	П2.12.1	0,003584		
54.	П4.1.5.2	0,003584		
55.	П4.11.3	0,003584		
56.	П2.1.5	0,003584		
57.	П1.1.7.2	0,003584		
58.	П2.4.8	0,003584		
59.	П2.4.4	0,003584		
60.	П1.1.11	0,003584		
61.	П2.4.1	0,003584		
62.	П2.6	0,003584		
63.	П8.1.2	0,003584		
64.	П1.1.9.1	0,003584		
65.	П3.3.2	0,003584		
66.	П2.1.9.3	0,003584		
67.	П5.1.1.3	0,003584		
68.	П5.1.1.1	0,003584		
69.	П2.12.2	0,003584		
70.	П8.7	0,003584		
71.	П5.2.2	0,003584		
72.	П2.11.1	0,003584		

ДОДАТОК Ж. ЛІСТИНГ ПРОГРАМНОГО КОДУ

Лістинг Ж.1 – код файлу tree-decision.ipynb.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor
from sklearn.tree import export_graphviz
from graphviz import Source
from sklearn import tree
from sklearn.model_selection import cross_val_score
from dtreeviz.trees import *
In [3]:
data = pd.read_csv('../csv/s1.csv')
data1 = pd.read_csv('../csv/all.csv')
In [4]:
train_data = data
In [5]:
del train_data['Unnamed: 0']
del train_data['276']
del train_data['277']
In [6]:
indicator = data1.iloc[1]
In [7]:
del indicator['276']
del indicator['277']
del indicator['Unnamed: 0']
In [8]:
predict1_data = data1.iloc[3:]['276']
predict2_data = data1.iloc[3:]['277']
In [14]:
replace_with = np.diag(train_data)
train_data = train_data.replace('ні', 0)
train_data = train_data.replace('так', 1)
train_data = train_data.replace('', 0)
train_data = train_data.replace(' ', 0)
train_data = train_data.fillna(0)

# train_data = pd.to_numeric(train_data)
indicator = pd.to_numeric(indicator)
indicator = indicator.replace('ні', 0)
indicator = indicator.replace('так', 1)
indicator = indicator.fillna(0)
In [10]:
```

```
list_in = train_data.columns.values
for i in range(len(indicator)):
    if indicator[i] == 0:
        train_data[list_in[i]] = train_data[list_in[i]] * 0
    else:
        train_data[list_in[i]] = train_data[list_in[i]] * 100 / indicator[i]
```

DecisionTreeClassifier

```
In [15]:
# створення моделі дерева класифікації
clf1 = DecisionTreeClassifier()
# навчання моделі на даних
clf1 = clf1.fit(train_data, predict1_data)
clf1
Out[15]:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
In [16]:
cvs = cross_val_score(clf1, train_data, predict1_data, cv=10)
In [38]:
importances  = clf1.feature_importances_
In [19]:
list_n = []
full_list_name = data1.iloc[0][1:-2]
list_name = []
for i in range(len(importances)):
    if importances[i] != 0:
        list_n.append(importances[i])
        list_name.append(full_list_name[i])
In [20]:
plt.rcParams['figure.figsize'] = (14,8)
In [21]:
plt.bar(range(len(list_n)), list_n,alpha=0.5)
plt.xticks(range(len(list_n)), list_name, fontsize=10, rotation=30)
plt.title('Важливість показників')
plt.show()

In [25]:
for i in range(len(list_n)):
    print(list_name[i], '\t', list_n[i])
In [22]:
export_graphviz(clf1, out_file='dec-tree-class.dot', feature_names=data1.iloc[0
][1:-2], filled=True)
In [23]:
```

```
import os
os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'
# Convert to png using system command (requires Graphviz)
from subprocess import call
call(['dot', '-Tpng', 'dec-tree-class.dot', '-o', 'dec-tree-class.png'])

# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'dec-tree-class.png')
```

DecisionTreeRegressor

```
In [26]:
clf2 = DecisionTreeRegressor(random_state=0)

cvs2 = cross_val_score(clf2, train_data, predict2_data, cv=10)
In [28]:
clf2 = DecisionTreeRegressor(random_state=0)
clf2 = clf2.fit(train_data, predict2_data)
clf2
Out[28]:
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
In [40]:
export_graphviz(clf1, out_file='dec-tree-regr.dot', feature_names=data1.iloc[0]
[1:-2], filled=True)
In [41]:
import os
os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'
# Convert to png using system command (requires Graphviz)
from subprocess import call
call(['dot', '-Tpng', 'dec-tree-regr.dot', '-o', 'dec-tree-regr.png'])

# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'dec-tree-regr.png')

In [39]:
importances2  = clf2.feature_importances_
In [31]:
list_n2 = []
full_list_name2 = data1.iloc[0][1:-2]
list_name2 = []
for i in range(len(importances2)):
    if importances2[i] != 0:
        list_n2.append(importances2[i])
        list_name2.append(full_list_name2[i])
```

```
In [32]:
plt.bar(range(len(list_n2)), list_n2,alpha=0.7)
plt.xticks(range(len(list_n2)), list_name2, fontsize=8, rotation=30)
plt.title('Важливість показників')
plt.show()

In [37]:
for i in range(len(list_n2)):
    print(list_name2[i], '\t', '%.7f' % list_n2[i])
```

Лістинг Ж.2 – Код файла random-forest.ipynb.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier
from sklearn.tree import export_graphviz
from graphviz import Source
from sklearn import tree
from sklearn.model_selection import cross_val_score
from dtreeviz.trees import *
import xgboost as xgb
In [293]:
data = pd.read_csv('../csv/s1.csv')
data1 = pd.read_csv('../csv/all.csv')
In [294]:
train_data = data
In [295]:
a = train_data['276']
In [296]:
del train_data['Unnamed: 0']
del train_data['276']
del train_data['277']
In [297]:
indicator = data1.iloc[1]
In [298]:
del indicator['276']
del indicator['277']
del indicator['Unnamed: 0']
In [299]:
predict1_data = data1.iloc[3:]['276']
predict2_data = data1.iloc[3:]['277']
In [300]:
replace_with = np.diag(train_data)
train_data = train_data.replace('ні', 0)
train_data = train_data.replace('так', 1)
train_data = train_data.replace('', 0)
```

```
train_data = train_data.replace(' ', 0)
train_data = train_data.fillna(0)
indicator = pd.to_numeric(indicator)
indicator = indicator.replace('ні', 0)
indicator = indicator.replace('так', 1)
indicator = indicator.fillna(0)
In [301]:
list_in = train_data.columns.values
for i in range(len(indicator)):
    if indicator[i] == 0:
        train_data[list_in[i]] = train_data[list_in[i]] * 0
    else:
        train_data[list_in[i]] = train_data[list_in[i]] * 100 / indicator[i]
```

Random Forest Classifier

```
In [248]:
train_data['276'] = a
train_data = train_data.sample(frac=1)
train_data = train_data.replace(' ', 0)
train_data = train_data.replace('ні', 0)
train_data = train_data.replace('так', 1)
train_data = train_data.replace('', 0)
train_data = train_data.replace(' ', 0)
train_data = train_data.fillna(0)
In [249]:
train = train_data
train = train.astype(float)
test = train_data['276']
In [250]:
del train['276']
In [251]:
X_train = train[6:]
Y_train = test[6:]
X_test = train[:6]
Y_test = test[:6]
In [146]:
X_test
In [26]:
# список чисел кількость дерев
estimators = np.arange(2, 52, 2)
# список для даних точності моделі
predict1 = []
for i in estimators:
    # створення моделі з і-кількість дерев
    clf=RandomForestClassifier(n_estimators=i)
    # навчання моделі на даних
    clf.fit(X_train,Y_train)
    # прогнозування
    y_pred=clf.predict(X_test)
    # запис до масиву та визначення точності моделі
```

```
    predict1.append(metrics.accuracy_score(Y_test, y_pred))
# побудова графіка
plt.title("Ефект n-дерев")
plt.grid(True)
plt.xlabel("n-дерев")
plt.ylabel("точність")
plt.plot(estimators, predict1)
Out[26]:
[<matplotlib.lines.Line2D at 0x2b0022f52e8>]
In [16]:
from sklearn import metrics
plt.rcParams['figure.figsize'] = (14,8)
In [21]:
estimators = np.arange(2, 100, 1)
predict1 = []
for i in estimators:
    clf=RandomForestClassifier(n_estimators=i)
    clf.fit(train,test)
    predict1.append(cross_val_score(clf,train,test, cv=3))
predict1 = np.asmatrix(predict1)

plt.title("Ефект n-ітерацій")
plt.grid(True)
plt.xlabel("n-ітерацій")
plt.ylabel("score")
plt.plot(estimators, predict1.mean(axis = 1),linewidth=2, markersize=12,marker=
'.', label='RandomForest')

In [252]:
clf2=RandomForestClassifier(n_estimators=27)
clf2.fit(train,test)
Out[252]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None
,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=27,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
In [254]:
importances1  = clf2.feature_importances_
In [19]:
# список для даних точності моделі
xgb_scoring = []
# список чисел кількость дерев
estimators = np.arange(2, 52, 2)
for i in estimators:
    # створення моделі з і-кількість дерев
```

```
    estimator = xgb.XGBClassifier(learting_rate=0.1, n_estimators=i)
    # навчання моделі на даних
    estimator.fit(X_train,Y_train,eval_metric='auc')
    # прогнозування
    y_pred=estimator.predict(X_test)
    # запис до масиву та визначення точності моделі
    xgb_scoring.append(metrics.accuracy_score(Y_test, y_pred))
In [20]:
plt.plot(estimators, xgb_scoring, marker='.', label='XGBoost')
plt.grid(True)
plt.xlabel('n_trees')
plt.ylabel('score')
plt.title('Accuracy score')
plt.legend(loc='lower right')
In [302]:
# створення моделі бустингу
estimator = xgb.XGBClassifier(n_estimators=17)
# навчання моделі
estimator.fit(X_train,Y_train)
# бустинг моделі
bst = estimator._Booster
# отримати важливість кожного показника моделі
imps = bst.get_fscore()
In [303]:
# imps
list_xgb_name = []
list_xgb_n = []
for key in imps:
    list_xgb_name.append(key)
    list_xgb_n.append(imps[key])
In [304]:
plt.bar(range(len(list_xgb_n)), list_xgb_n,alpha=0.7)
plt.xticks(range(len(list_xgb_n)), list_name_feature[list_xgb_name], fontsize=8
, rotation=30)
plt.title('Важливість показників')
plt.show()
In [305]:
for i in range(len(list_xgb_name)):
    print(list_name_feature[list_xgb_name[i]], '\t', list_xgb_n[i])
In [309]:
import operator
imps1 = sorted_x = sorted(imps.items(), key=operator.itemgetter(1))
In [318]:
imps2 = imps1[-10:]
In [316]:
list_xgb_name = []
list_xgb_n = []
for i in range(len(imps2)):
    list_xgb_name.append(imps2[i][0])
```

```
    list_xgb_n.append(imps2[i][1])
In [317]:
plt.bar(range(len(list_xgb_n)), list_xgb_n,alpha=0.7)
plt.xticks(range(len(list_xgb_n)), list_name_feature[list_xgb_name], fontsize=8
, rotation=30)
plt.title('Важливість показників')
plt.show()

In [255]:
importances1  = clf2.feature_importances_
importances1
In [256]:
list_n2 = []
full_list_name2 = data1.iloc[0][1:-2]
list_name2 = []
for i in range(len(importances1)):
    if importances1[i] != 0:
        list_n2.append(importances1[i])
        list_name2.append(full_list_name2[i])
In [257]:
plt.bar(range(len(list_n2)), list_n2,alpha=0.7)
plt.xticks(range(len(list_n2)), list_name2, fontsize=8, rotation=30)
plt.title('Важливість показників')
plt.show()

In [258]:
indices = np.argsort(importances1)[::-1]
In [259]:
indices
In [260]:
list_name_feature = data1.iloc[0][1:-2]
for f in range(X_train.shape[1]):
    if importances1[indices[f]] != 0:
        print(f + 1,'.\t', list_name_feature[indices[f]],'\t', importances1[ind
ices[f]])
In [261]:
plt.bar(range(10), importances1[indices[:10]],alpha=0.7)
plt.xticks(range(10), list_name_feature[indices[:10]], fontsize=8, rotation=30)
plt.title('Важливість показників')
plt.show()
```

RandomForestRegressor

```
In [220]:
train_data = train_data.sample(frac=1)
train_data = train_data.replace(' ', 0)
train_data = train_data.replace('ні', 0)
train_data = train_data.replace('так', 1)
train_data = train_data.replace('', 0)
train_data = train_data.replace(' ', 0)
```

```
train_data = train_data.fillna(0)
In [221]:
train_data['277'] = a
In [222]:
train = train_data
test = train_data['277']
In [223]:
del train['277']
In [224]:
X_train = train[6:]
Y_train = test[6:]
X_test = train[:6]
Y_test = test[:6]
In [146]:
# список чисел кількость дерев
estimators = np.arange(2, 102, 2)
# список для даних точності моделі
scores = []
for n in estimators:
    # створення моделі з і-кількість дерев
    regressor = RandomForestRegressor(n_estimators=n, random_state=0)
    # навчання моделі на даних
    regressor.fit(X_train, Y_train)
    # запис до масиву та визначення точності моделі
    scores.append(regressor.score(X_test, Y_test))
plt.title("Ефект n-дерев")
plt.grid(True)
plt.xlabel("n-дерев")
plt.ylabel("точність")
plt.plot(estimators, scores,marker='.')

In [57]:
regressor = RandomForestRegressor(n_estimators=25, random_state=0)
In [143]:
regressor.fit(X_train,Y_train)

y_pred=regressor.predict(X_test)
In [147]:
estimator
Out[147]:
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, learting_rate=0.1,
             max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
             n_estimators=2, n_jobs=1, nthread=None, objective='reg:linear',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
In [225]:
regressor = RandomForestRegressor(n_estimators=58, random_state=0)
```

```
regressor
Out[225]:
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=58,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
In [226]:
regressor.fit(X_train,Y_train)

y_pred=regressor.predict(X_test)
In [319]:
importances2  = regressor.feature_importances_
In [230]:
list_n2 = []
full_list_name2 = data1.iloc[0][1:-2]
list_name2 = []
for i in range(len(importances2)):
    if importances2[i] != 0:
        list_n2.append(importances2[i])
        list_name2.append(full_list_name2[i])
             In [231]:
plt.bar(range(len(list_n2)), list_n2,alpha=0.7)
plt.xticks(range(len(list_n2)), list_name2, fontsize=8, rotation=30)
plt.title('Важливість показників')
plt.show()

In [232]:
indices = np.argsort(importances2)[::-1]
In [238]:
list_name_feature = data1.iloc[0][1:-2]
for f in range(X_train.shape[1]):
    if importances2[indices[f]] != 0:
        print(list_name_feature[indices[f]],'\t', '%.7f' % importances2[indices
[f]])

In [234]:
plt.bar(range(10), importances2[indices[:10]],alpha=0.7)
plt.xticks(range(10), list_name_feature[indices[:10]], fontsize=8, rotation=30)
plt.title('Важливість показників')
plt.show()
```

XGBoosting

```
In [144]:
# список для даних точності моделі
xgb_scoring = []
```

```
# список чисел кількость дерев
estimators = np.arange(2, 52, 2)
for i in estimators:
    # створення моделі з і-кількість дерев
    estimator = xgb.XGBRegressor(n_estimators=i)
    # навчання моделі на даних
    estimator.fit(X_train,Y_train)
    # запис до масиву та визначення точності моделі
    xgb_scoring.append(estimator.score(X_test, Y_test))
In [145]:
plt.title("Ефект n-дерев")
plt.grid(True)
plt.xlabel("n-дерев")
plt.ylabel("точність")
plt.plot(estimators, xgb_scoring, marker='.')
Out[145]:
In [ ]:
# estimator = xgb.XGBRegressor(n_estimators=40)
In [181]:
# створення моделі бустингу
estimator = xgb.XGBRegressor(n_estimators=40)
# навчання моделі
estimator.fit(train,test)
# бустинг моделі
bst = estimator._Booster
# отримати важливість кожного показника моделі
imps = bst.get_fscore()
[15:18:34] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
In [205]:
imps = bst.get_fscore()
In [207]:
# imps
list_xgb_name = []
list_xgb_n = []
for key in imps:
    list_xgb_name.append(key)
    list_xgb_n.append(imps[key])
In [185]:
plt.bar(range(len(list_xgb_n)), list_xgb_n,alpha=0.7)
plt.xticks(range(len(list_xgb_n)), list_name_feature[list_xgb_name], fontsize=8
, rotation=30)
plt.title('Важливість показників')
plt.show()

In [320]:
imps2 = imps1[-10:]
In [201]:
```

```
import operator
imps1 = sorted_x = sorted(imps.items(), key=operator.itemgetter(1))
list_xgb_name = []
list_xgb_n = []
for i in range(len(imps2)):
    list_xgb_name.append(imps2[i][0])
    list_xgb_n.append(imps2[i][1])
In [202]:

plt.bar(range(len(list_xgb_n)), list_xgb_n,alpha=0.7)
plt.xticks(range(len(list_xgb_n)), list_name_feature[list_xgb_name], fontsize=8
, rotation=30)
plt.title('Важливість показників')
plt.show()

In [188]:
list_name_feature = data1.iloc[0][:-2]
list_name_feature[list_xgb_name]
In [191]:
imps1 = sorted_x = sorted(imps.items(), key=operator.itemgetter(1))
```

ДОДАТОК З. ТЕЗИ РОБОТИ

Модель аналізу рейтингу СумДУ з використанням алгоритмів машинного навчання

Макаренко Д.В., *студент,* Гайдабрус Б.В., *кандидат технічних наук*
Сумський державний університет, м. Суми, Україна

На базі Сумського державного університету (СумДУ) створена методика для визначення рейтингу інститутів, факультетів та кафедр СумДУ. Методика передбачає визначення за підсумками календарного року рейтингу структурних підрозділів і реалізується шляхом комп'ютерної обробки статистичної інформації, яка складена відповідними підрозділами – надавачами інформації, у тому числі із врахуванням даних річних звітів інститутів, факультетів, кафедр, викладачів, наявності підтверджуючих документів.

У загальний рейтинг входить більше 250 показників, кожен із яких вираховується власною формулою. Розрахунки, за якими вираховуються показники можуть змінюватися протягом років, тому неможливо сказати, які саме показники більше впливають на фінальний рейтинг інститутів, факультетів та кафедр. Розрахунок кожного показнику та визначення важливості займає багато часу, тому запропонована модель на основі машинного навчання. Актуальність використання данного підходу, обумовлено тим, що показники рейтингу та статистична інофрмація оброблюється на основі аналізу даних, встановлюються закономірності та приймються рішення з мінімальним втручанням людського фактору.

У дипломній роботі розглядаються та досліджуються популярні алгоритми машинного навчання, а саме такі, як:

- лінійна регресія, що дозволить побудувати залежності однієї оголошеної змінної (наприклад, структурний підрозділ) від інших змінних, факторів або незалежних змінних (таких ключових елементів рейтингу СумДУ, як індикатори, показники, тощо.);

- дерево рішень для побудови классифікацій та передбачення значення цільової змінної (показник рейтингу структурного підрозділу) на освнові декількох змінних на вході (наприклад, індикаторів);

На цих дослідження була створена модель для прогнозування та визначення показників важливості рейтингу структурних підрозділів СумДУ з використанням алгоритмів машинного навчання.