Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДУ КЛАСИФІКАЦІЇ З МЕТОЮ ПРОГНОЗУВАННЯ ЦІН НЕРУХОМОСТІ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-21-1
Іващенко О.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Іващенку Олександру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та реалізація методу класифікації з метою прогнозування цін нерухомості

затверджена наказом по університету від 9 листопада 2022 року № 1469Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 листопада 2022 р.

3. Вихідні дані до роботи математичні моделі методів класифікації даних, загальні теоретичні відомості про методи класифікації даних з метою аналізу даних, використання сучасних методів мови програмування Python для вирішення завдань машинного навчання.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів класифікації даних для аналізу даних.
2. Математична модель методу дерева рішень.
3. Математична модель методу випадкового лісу.
4. Комп'ютерна модель методу класифікації даних.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) діаграма роботи методу випадкового лісу, структура дерева рішень, графік взаємодії найбільш передбачуваних та цільової змінної, графік кінцевого прогнозування, структура випадкового лісу.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	09.11.2022	
2	Аналіз завдання, підбір літератури	09.11.22-10.11.22	
3	Аналіз літератури з досліджуваної проблеми	11.11.22-12.11.22	
4	Аналіз технічних засобів реалізації	13.11.22-14.11.22	
5	Розробка методу класифікації даних	15.11.22-18.11.22	
6	Програмна реалізація	19.11.22-23.11.22	
7	Оформлення пояснювальної записки	24.11.22-25.11.22	
8	Перевірка на плагіат	26.11.2022	
9	Рецензування	27.11.2022	
10	Підготовка презентації та доповіді	28.11.2022	
11	Занесення роботи в електронний архів	29.11.2022	
12	Попередній захист кваліфікаційної роботи	30.11.2022	

Дата видачі завдання 9 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Кобилін О.А
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 66 с., 3 табл., 50 рис., 1 дод., 40 джерел.

КЛАСИФІКАЦІЯ ДАНИХ, ВИПАДКОВИЙ ЛІС, ДЕРЕВО РІШЕНЬ, ДИСПЕРСІЯ, СЕРЕДНЬОКВАДРАТИЧНА ПОХИБКА, ДИСПЕРСІЯ, ПРОГНОЗУВАННЯ, PYTHON.

Об'єктом дослідження є прогнозування ціни нерухомості за допомогою методів класифікації даних.

Метою дослідження є розробка методу, що базуються на використанні методів класифікації та регресійного аналізу, які допоможуть у виконанні завдання прогнозування цін нерухомості.

Використано методи класифікації, а саме дерево рішень та випадковий ліс. Проведено дослідження та аналіз даних методів класифікації, визначено слабкі та сильні сторони при виконанні задачі прогнозування. Також аналізовано результати двох обраних алгоритмів та тонкощі розробки застосунку з використанням мови програмування Python.

У результаті проведеного дослідження здійснена розробка застосунку для прогнозування цін нерухомості.

DATA CLASSIFICATION, RANDOM FOREST, DECISION TREE, VARIANCE, MEAN SQUARE ERROR, VARIANCE, PREDICTION, PYTHON.

The object of the research is real estate price forecasting using data classification methods.

The purpose of the research is to develop a method based on the use of classification and regression analysis methods that will help in the task of forecasting real estate prices.

Classification methods were used, namely decision tree and random forest. Research and data analysis of classification methods was carried out, weaknesses and strengths in performing the forecasting task were determined. The results of the two selected algorithms and the intricacies of application development using the Python programming language are also analyzed.

As a result of the conducted research, the development of an application for forecasting real estate prices was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RF – Random Forest. Метод випадковий ліс, який використовується в задачах класифікації та регресійного аналізу

DT – Decision Tree. Метод дерево рішень, який використовується в задачах класифікації та регресійного аналізу

SOP – Sum of Product, диз'юнктивна нормальна форма

ID3 – Iterative Dichotomiser 3. Алгоритм методу дерева рішень

C4.5 – розширена версія алгоритму Iterative Dichotomiser 3. Алгоритм методу дерева рішень

CART – Classification And Regression Trees. Алгоритм методу дерева рішень

CHAID – Chi-square Automatic Interaction Detector. Алгоритм методу дерева рішень

MARS – Multivariate Adaptive Regression Splines. Алгоритм методу дерева рішень

IG – Information Gain. Статистична властивість, яка використовується при побудові дерева рішень

CRM – Customer Relationship Management. Система управління взаємодії з клієнтами

ВСТУП

Сучасні методи класифікації дають змогу обробити набір даних використовуючи просту модель для побудови, забезпечують більшу точність – це дуже важливо при вирішенні завдань прогнозування.

Насамперед, класифікація даних – це техніка, яка класифікує дані за окремою кількістю класів, мітка класифікатора присвоюється кожному класу по черзі. Основна мета класифікації – визначити клас для запуску нових даних шляхом аналізу навчального набору. У даному дослідженні будуть розглянуті, проаналізовані та реалізовані два методи, які більш спеціалізовані саме на прогнозуванні даних, а саме методи дерево рішень та випадковий ліс.

Актуальність предметної області зумовлена потребою швидкого продажу нерухомості. Використовуючи алгоритми класифікації, які розраховані на прогнозування, можна розробити застосунок який допоможе людині в швидкому прийнятті рішення. Наприклад, людина хоче виставити на продаж свою власну нерухомість, для цього вона повинна задати вірну ціну, щоб її дім чи квартира мали успіх на ринку. За допомогою аналізу вхідних даних, таких як кількість квартир, район, доступність до центру міста та різних інших параметрів можна класифікувати ціну нерухомості, яка буде користуватися попитом на ринку. За допомогою сучасних методів машинного навчання таких як випадковий ліс та дерево рішень. Вони дають змогу обробити набір даних використовуючи просту модель для побудови, забезпечують більшу точність – це дуже важливо при вирішенні завдань прогнозування. Застосунок може допомогти не тільки звичайним людям, а і кваліфікаційним агентським компаніям, які займаються продажом нерухомості. На мою думку, обрана предметна область з кожним днем тільки буде набирати свою актуальність.

У результаті проведеного дослідження буде реалізовано програмний застосунок для прогнозування цін нерухомості.

1 ОГЛЯД МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ ДЛЯ АНАЛІЗУ ДАНИХ

1.1 Огляд методу класифікації Random Forest

Random Forest – це контрольований алгоритм машинного навчання, який складається з алгоритмів дерева рішень. Цей алгоритм має широке застосування в класифікації та регресійному аналізі даних. RF застосовується в різних галузях, таких як банківська галузь, маркетингова галузь чи галузі продажу, для прогнозування ціни чи інших показників.

RF використовує ансамблеве навчання, що є технікою, яка поєднує багато класифікаторів для надання рішень різного роду проблем.

Основою побудови випадкового ліс є набір з дерев рішень. Далі згенерований з набору ліс навчається за допомогою пакетування або завантажувального агрегування. Також для покращення точності даного алгоритму використовується ансамблевий метаалгоритм, який називається Bagging. Алгоритм RF встановлює результат на основі передбачень, які були отримані в деревах рішень. Для прогнозування результату алгоритм бере середнє значення отриманих результатів з різних дерев. Чим вища кількість дерев, тим більша точність отриманого результату [1 – 7].

Якщо порівнювати його з відомим алгоритмом DT, то можна сказати, що алгоритм випадкового лісу усуває обмеження використання одного дерева, це зменшує переобладнання наборів даних і також підвищує точність отриманих результатів.

Особливості алгоритму Random Forest:

– він є більш точнішим ніж алгоритм DT;

– він забезпечує ефективний спосіб обробки відсутніх даних;

– він може створити обґрунтований прогноз без налаштування гіперпараметрів;

– він вирішує проблему переобладнання дерев рішень;

– у кожному випадковому дереві кінцевого лісу підмножина функцій вибирається випадковим чином у точці розділення вузла.

Дерева рішень є основною складовою (блоками) алгоритму випадкового лісу. Дерево рішень – це техніка прийняття рішень, яка формує деревоподібну структуру. Дерево рішень складається з трьох компонентів: вузлів рішень, листових вузлів і кореневого вузла. Алгоритм дерева рішень ділить навчальний набір даних на гілки, які далі поділяються на інші гілки. Ця послідовність триває до тих пір, поки не буде досягнуто листовий вузол. Листковий вузол не може бути відокремлений далі [8 – 10].

Вузли в дереві рішень представляють атрибути, які використовуються для прогнозування результату. Вузли рішень забезпечують посилання на листи. На рисунку 1.1 показана структура дерева рішень.

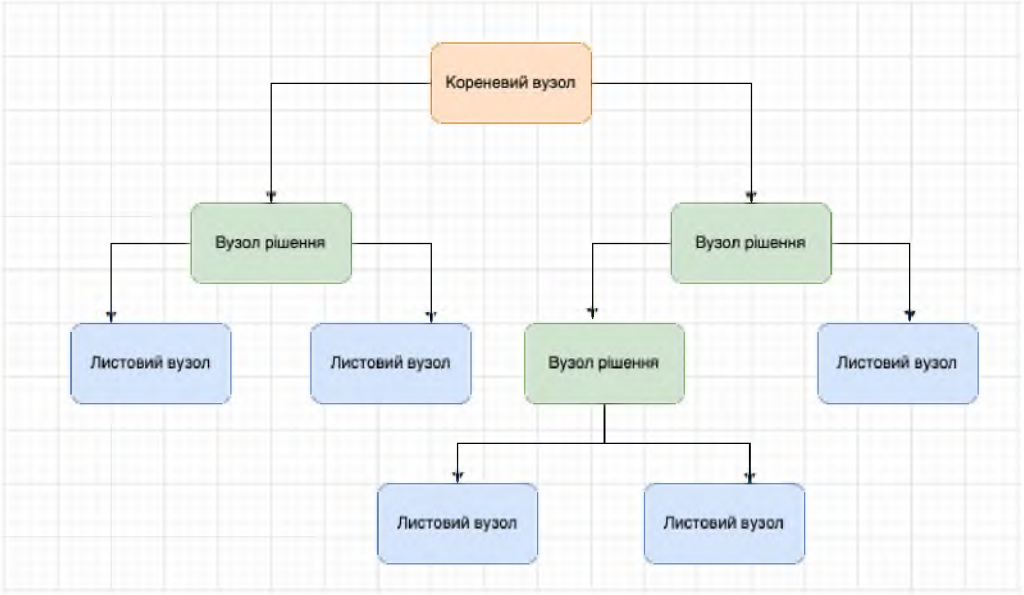

Рисунок 1.1 – Структура Дерева рішень

Результат в будівельних блоках отримується за рахунок ентропії. Ентропія – це метрика для обчислення невизначеності [11]. Приріст інформації Є мірою того, як зменшується невизначеність цільової змінної, враховуючи набір незалежних змінних. Концепція отримання інформації передбачає

використання незалежних змінних для отримання інформації про цільову змінну. Ентропія цільової змінної та умовна ентропія використовуються для оцінки інформаційного простору. У цьому випадку умовна ентропія віднімається від ентропії. Збільшення інформації використовується при навчанні дерев рішень. Це допомагає зменшити невизначеність у цих деревах. Високий інформаційний приріст означає, що високий ступінь невизначеності (інформаційної ентропії) було усунена. Ентропія та приріст інформації важливі для розщеплення гілок, що є важливою діяльністю при побудові дерев рішень [12 – 15]. На рисунку 1.2 зображено приклад використання DT на прикладі прогнозування чи купить клієнт нерухомість.

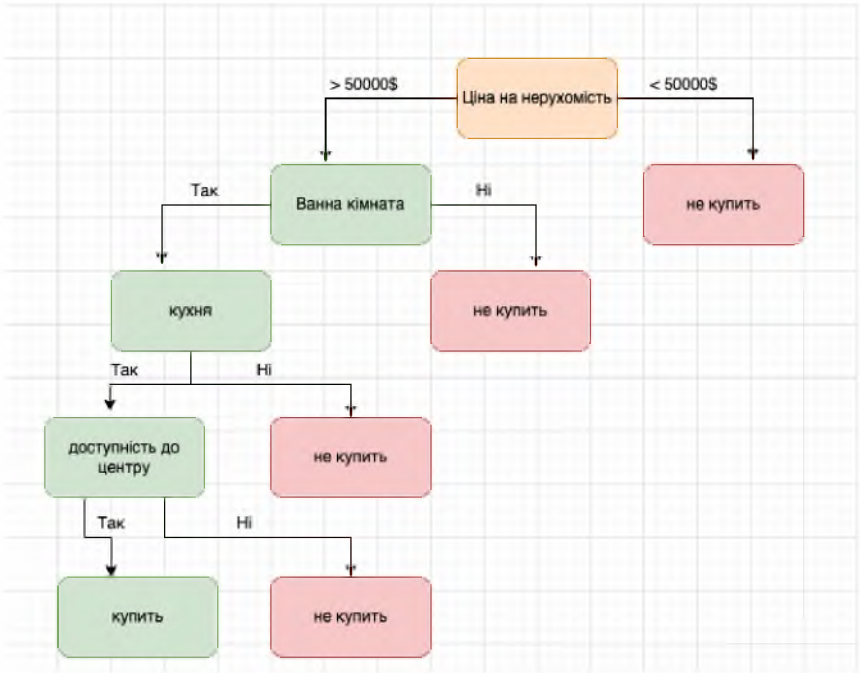

Рисунок 1.2 – Використання RF на прикладі прогнозування чи купить клієнт нерухомість

Основна відмінність між алгоритмами DT та RF полягає в тому, що встановлення кореневих вузлів і саме розділення вузлів виконується випадковим чином. Випадковий ліс використовує метод пакетування для створення необхідного прогнозу [16].

Пакетування передбачає використання різних зразків навчальних даних, а не лише одного зразка. Дерева рішень дають різні результати, залежно від навчальних даних, що передаються алгоритму випадкового лісу. Ці результати будуть класифіковані за рангом та найвищий буде обрано як кінцевий результат.

Розглядаючи приклад, який зображений на рисунку 1.2 можна виділити чотири кореневі вузли, які впливають на вибір клієнта. Випадковий ліс розподілить вузли шляхом випадковості. Кінцевий прогноз буде обрано на основі результату чотирьох дерев.

Результат, який буде обрано більшістю дерев рішень, буде являтися остаточним вибором. Якщо три дерева передбачають покупку, а одне дерево не покупку, то остаточним прогнозом буде покупка. Тобто результатом даного прогнозування буде те, що клієнт купить нерухомість.

Класифікація у RF використовує методологію ансамблю для досягнення результату. Кількість дерев рішень напряму залежить від початкових навчальних даних. Цей набір даних складається зі спостережень та функцій, які будуть вибрані випадковим чином під час розподілу вузлів [17].

Система випадкового лісу спирається на різні дерева рішень. Кожне дерево рішень складається з вузлів рішень, листових вузлів та кореневого вузла. Листовий вузол кожного дерева є кінцевим результатом, отриманим окремим деревом рішення. Вибір остаточного результату відбувається за мажоритарною системною голосування [18]. Таким чином результат, обраний більшістю дерев рішень, стає кінцевим результатом системи випадкового лісу.

Регресія – це інше завдання, яке виконує алгоритм випадкового лісу. Регресія RF дотримується концепції простої регресії. Значення ознаки та незалежних змінних передаються в моделі випадкового лісу.

На рисунку 1.3 зображена діаграма виконання методу випадкового лісу.

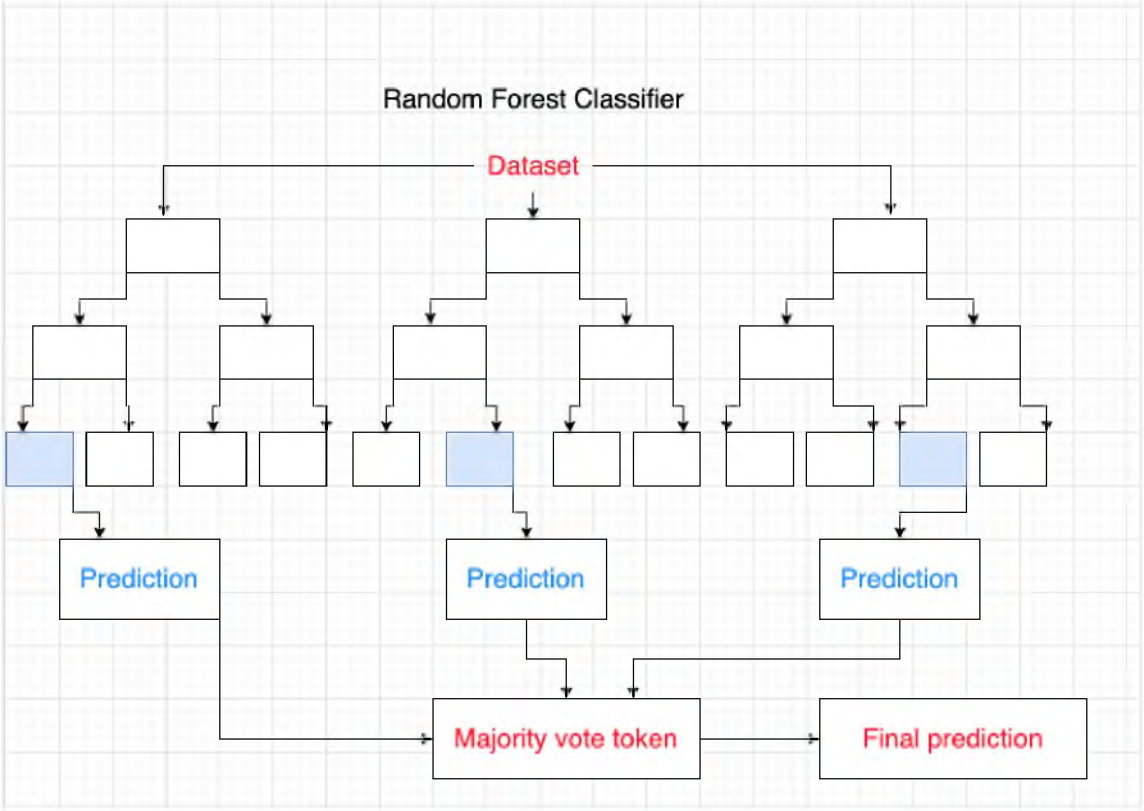

Рисунок 1.3 – Діаграма роботи RF класифікація

У регресії випадкового лісу кожне дерево створює певний прогноз. Результатом регресії є середнє передбачення окремих дерев [19]. Це суперечить випадковій класифікації лісу, результатом якої класифікатор дерев рішень. Також регресія випадкового лісу та лінійна регресія дотримуються однієї концепції, тобто схожі за побудовою, але відрізняються за функціями.

Переваги використання алгоритму випадкового лісу:

– алгоритм може виконувати завдання як класифікації так і регресії;

– в результаті отримуються прогнозу легкі для сприйняття;

– метод ефективно оброблю великі об’єми даних;

– алгоритм RF має більш високу точність ніж алгоритм DT.

Недоліки використання алгоритму випадкового лісу:

– оскільки алгоритм RF можу обробляти великі набори даних, вони надають точніші прогнози, але ці дані обробляються досить повільно, оскільки вони оброблюють дані для кожного дерева рішень окремо;

– так як метод випадкового лісу обробляє значний об’єм даних, то йому для цього знадобиться великий об’єм ресурсів для зберігання цих даних;

– передбачення з одного дерева рішень легше інтерпретувати ніж з лісу дерев.

За результатами дослідження даного методу можна сказати, що алгоритм випадкового лісу – це гнучкий алгоритм машинного навчання, який використовує ансамблеве навчання, що дозволяє легко вирішувати завдання класифікації та регресійного аналізу [20 – 24]. Метод пошуку гарно вирішує завдання прогнозування, яке обширно застосовується в різних галузях.

1.2 Огляд методу класифікації Decision Tree

1.2.1 Загальна характеристика методу Decision Tree

DT – це найпопулярніший алгоритм класифікації, який являється легким для розуміння та інтерпретації. Алгоритм дерева рішень належить до класу алгоритмів керованого навчання. Він гарно підходить для вирішення завдань класифікації та регресійного аналізу. Метою використання цього алгоритму є створення навчальної моделі, яка використовується для прогнозування класу або значення цільової змінної шляхом вивчення простих правил прийняття рішень, отриманих з навчальних даних. У деревах рішень прогнозування мітки класу для запису починається з кореня дерева. В результаті порівнюється значення кореневого атрибуту з атрибутом запису. На основі порівняння відбувається пересування по гілці, яка відповідає обраному значенню і переходить до наступного вузла [25].

Існує два типи дерев рішень, які базуються на типі цільової змінної, яка використовується:

– дерево рішень категоріальної змінної: дерево рішень, яке має категоріальну цільову змінну, називається деревом рішень категоріальної змінної;

– дерево рішень безперервної змінної: дерево рішень, яке має безперервну цільову змінну, називається деревом рішень безперервної змінної.

Існує декілька важливих частин при створенні дерева рішень:

– кореневий вузол, який представляє усю загальну вибірку, яка далі поділяється на два чи більше однорідних наборів;

– вузол прийняття рішення, коли вузол розбивається на додаткові вузли, він називається вузлом прийняття рішення;

– розщеплення, процес поділу вузла на два або більше вузлів;

– листовий або кінцевий вузол, вузол, який не розбивається, називають листовим або кінцевим вузлом;

– скорочення, процес, видалення підвузлів вузла рішення, цей процес називається скороченням. Можна сказати, що процес скорочення протилежний процесу розщеплення;

– гілкою називається підрозділ цілого дерева;

– батьківським вузлом називається вузол, який розділений на підвузли.

На рисунку 1.4 ілюстровано роботу алгоритму дерева випадкового лісу.

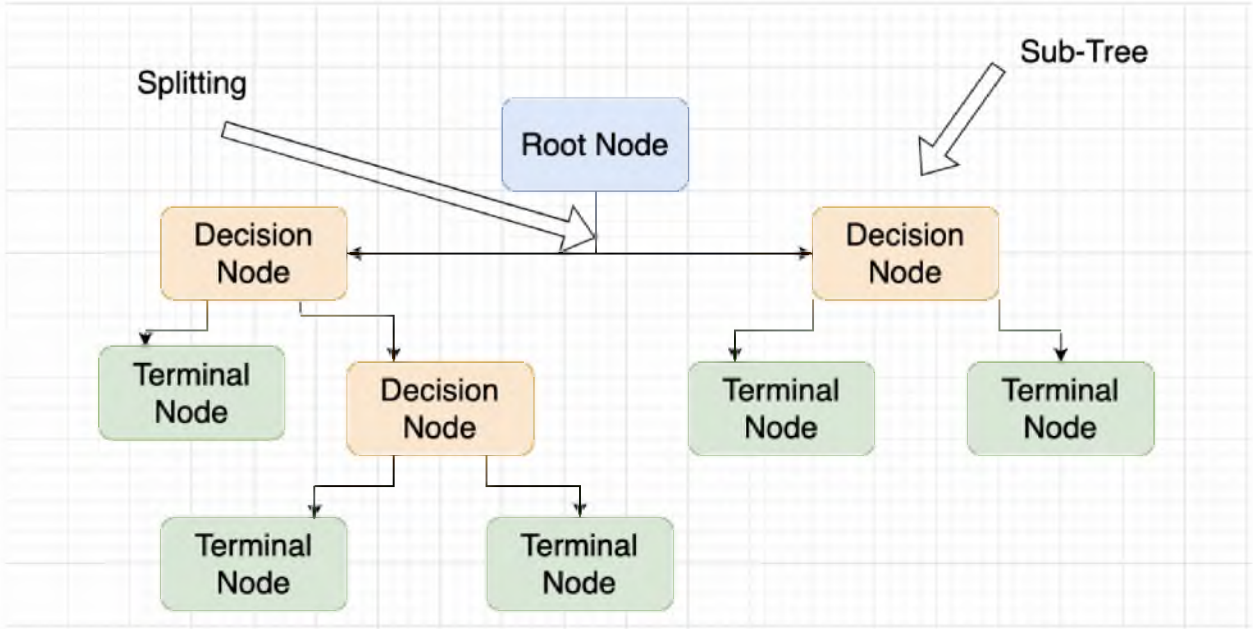

Рисунок 1.4 – Діаграма роботи алгоритму дерева випадкового лісу

Дерева рішень класифікують приклади, сортуючи їх по дереву від кореня до деякого листового вузла, причому листовий вузол забезпечує класифікацію прикладу. Кожен вузол у дереві діє як тестовий приклад для певного атрибута, і кожне ребро, що йде від вузла, відповідає можливим відповідям на тестовий приклад.

Цей процес є рекурсивним за своєю природою і повторюється для кожного піддерева з коренем у новому вузлі [26].

На початку використання алгоритму весь навчальний набір вважається коренем. Бажано, щоб значення ознак були категоричними. Якщо значення неперервні, то вони оброблюються дискретним методом перед побудовою моделі. Записи розподіляються рекурсивно на основі значень атрибутів. Порядок розміщення атрибутів як кореня або внутрішнього вузла дерева виконується за допомогою деякого статистичного підходу. SOP також відома як диз'юнктивна нормальна форма. Для класу кожна гілка від кореня дерева до листового вузла, що має той самий клас є кон'юнкцією (добутком) значень, різні гілки, що закінчуються в цьому класі, утворюють диз'юнкцію (суму) [27]. Основною проблемою реалізації дерева рішень є визначення, які атрибути ми повинні розглядати як кореневий вузол так і кожен рівень. Ця обробка називається вибором атрибутів. Існують різні способи вибору атрибуту, щоб визначити головне значення на кожному рівні.

Рішення про стратегічні поділи сильно впливають на точність дерева рішень. Так як критерії прийняття рішень різні для дерев класифікації та регресії. Дерева рішень використовують кілька алгоритмів, щоб вирішити чи розділяти вузол на два чи більше підвузлів. Створення нових підвузлів збільшує однорідність результуючих підвузлів. Тобто можна сказати, що чистота вузла збільшується в залежності від цільової змінної. Дерево рішень розбиває вузли на всі доступні змінні, а потім обирає розподіл, який призводить до найбільш однорідних підвузлів. Вибір алгоритму також базується на типі цільових змінних.

Першою перевагою методу DT є те, що його легко читати та інтерпретувати. Одна з найважливіших переваг методу дерева рішень полягає в тому що результати легко читати та інтерпретувати. Наприклад, використовуючи дерева рішень для представлення демографічної інформації про клієнтів, співробітники відділу маркетингу можуть читати та інтерпретувати графічне представлення даних, не використовуючи статичні знання. Дані також можуть генерувати важливу інформацію про ймовірності, витрати та альтернативи різним стратегіям, які були сформульовані відділом маркетингу.

Наступною перевагою методу дерева рішень є легкість у приготуванні початкових даних. У порівнянні з іншими методами прийняття рішень, дерева рішень потребують менше зусиль для підготовки даних. Однак користувачі повинні мати готову інформацію для створення нових змінних із можливістю прогнозування цільової змінної. Вони також можуть створювати класифікації даних без виконання складних обчислень. Для складних ситуацій користувачі можуть комбінувати дерева рішень з іншими методами [28 – 30].

Останньою розглянутою перевагою є потреба меншого очищення даних. Ця перевага полягає в тому, що після створення змінних потрібно чистити дані в малому обсязі. Випадки відсутніх значень і викидів мають малі значення для даних дерева рішень.

Одним з недоліків методу дерева рішень є нестійка поведінка методу. Метод є значною мірою нестабільним порівняно з іншими методами прийняття рішень. Невелика зміна в даних може призвести до серйозних змін у структурі дерева рішень, що може змінити результат та він буде мати велику відмінність порівняно з тим, який користувач отримує у звичайному випадку використання методу. Отриманим результатом можна керувати за допомогою алгоритмів машинного навчання, таких як прискорення та пакетування.

Іншим недоліком DT є його ефективність у прогнозуванні результату безперервної змінної. Дерева рішень менш ефективні для прогнозування, коли основною метою є прогнозування результату безперервної змінної, це

пояснюється тим, що дерева рішень мають тенденцію втрачати інформацію під час категоризації змінних на кілька категорій.

У висновку можна сказати, що методу дерев рішень гарно підходить для виконання задач прогнозування, він є легкий для розуміння, також він є універсальний і використовується у багатьох алгоритмах машинного навчання.

1.2.2 Алгоритми Decision Tree

Дерева рішень можуть запускати різноманітні алгоритми для поділу вузла на додаткові підвузли. Технічно дерево рішень використовує всі доступні змінні для поділу вузлів, але все таки обирає поділ, який дає найбільш однорідні підвузли. Тип цільової змінної відіграє найважливішу роль при виборі алгоритму дерева рішень.

Алгоритм ID3 генерує дерева рішень з усім набором даних «*X*» як кореневим вузлом. Потім він повторює інструкції для кожного атрибуту та використовуючи показники, такі як ентропія або приріст інформації, розділює інформацію на підмножини. Після розбиття алгоритм отримує рекурсію на кожній підмножині, враховуючи атрибути, які були раніше не розглянуті в ітераціях. Алгоритм ID3, як правило переповнює дані, також розділення даних може зайняти багато часу, коли розглядаються безперервні змінні. Областями використання алгоритму є обробка природної мови та машинне навчання.

C4.5 є вдосконаленою версією алгоритму ID3. Він розглядає класифіковані зразки як дані. Алгоритм використовує нормалізований приріст інформації для виконання розбиття вузлів. Функція, яка має найбільший приріст інформації, приймає остаточне рішення щодо розподілу даних. На відмінну від алгоритму ID3, C4.5 ефективно керує як дискретними, так і безперервними атрибутами. Також після побудови остаточного дерева рішень

алгоритм проходить процес скорочення, під час якого видаляються всі гілки, що мають низьку важливість або релевантність [31].

Алгоритм CART вирішує проблеми регресії та класифікації. Також він створює точки прийняття рішень за допомогою метрики індексу Gini, на відміну від алгоритмів ID3 та C4.5, які використовують приріст інформації або ентропію та коефіцієнт приросту для розділення наборів даних. Процес розподілу в CART дотримується жадібного підходу, де метою є зменшення функцію витрат. Для задач класифікації індекс Gini використовується як функція вартості для визначення чистоти листових вузлів. Алгоритм вибирає помилку суми квадратів як функцію вартості для регресії, щоб визначити найкращий прогноз.

Алгоритм CHAID розкриває зв'язок між змінними всіх типів, включаючи номінальні, порядкові або безперервні. Підхід CHAID створює дерево, яке визначає як найкраще об'єднати змінні, щоб розкрити результат для даної залежної змінної. Під час створення дерева рішень алгоритм CHAID розглядає всі можливі комбінації для кожного категоріального предиктора та продовжує процес до моменту, коли подальше розбиття стає неможливим. Тобто це є ознакою того, що нарешті досягнуто найкращого результату. Процес розробки дерева рішень починається з визначення кореневого вузла дерева, який представляє цільову або залежну змінну. Також цільова змінна ділиться на кілька батьківських вузлів. В результаті ці вузли потім поділяються на дочірні вузли за допомогою статичних алгоритмів. В алгоритмі CHAID об'єднання змінних здійснюється на основі тестів.

MARS алгоритм зазвичай використовується в задачах регресії, в яких дані є нелінійними. Це адаптивний алгоритм, який розділяє дані та запускає модель лінійної регресії для кожного окремого розділу. MARS закладає основу для нелінійного моделювання та тісно пов'язана з моделями множинної регресії. Алгоритм є адаптацією CART, що дозволяє додавати нові терміни в існуючу модель [32].

1.3 Постановка задачі дослідження

Таким чином, задача прогнозування є актуальним завданням для класифікації та регресії даних. Ставиться завдання створення універсального методу класифікації даних для прогнозування цін нерухомості.

Об'єктом дослідження є прогнозування ціни нерухомості за допомогою методів класифікації даних.

Метою дослідження є розробка методу, що базуються на використанні методів класифікації та регресійного аналізу, які допоможуть у виконанні завдання прогнозування цін нерухомості.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів класифікації даних;

– розробити універсальний алгоритм, який буде виконувати завдання прогнозування ціни;

– реалізувати алгоритм класифікації даних з метою прогнозування ціни нерухомості;

– реалізувати комп'ютерну модель алгоритму класифікації на базі методів випадкового лісу та дерева рішень;

– протестувати створений застосунок використовуючи обрану предметну область.

2 МАТЕМАТИЧНІ МОДЕЛІ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

2.1 Математична модель методу Decision Tree

Дерева рішень використовують кілька алгоритмів, щоб вирішити чи розділяти вузол на два або більше підвузлів. Створення підвуздів збільшує однорідність результуючих підвузлів. Тобто з цього випливає, що чистота вузла збільшується в залежності від цільової змінної. Дерево рішень розбиває вузли на всі доступні змінні, а потім обирає поділ, який призводить до найбільш однорідних підвузлів.

Вибір алгоритму базується на типі цільової змінної. Нижче наведено приклади алгоритмів, які використовуються в розробці дерев рішень:

– ID3 – алгоритм розширення D3;

– C4.5 – алгоритм, який є певним розширенням алгоритму ID3;

– CART – алгоритм, який використовується для побудови дерева класифікації або регресійного аналізу;

– CHAID – алгоритм, який використовує автоматичне виявлення взаємодії *xi*-квадрат. Також виконує багаторівневе розбиття під час обчислення дерев класифікації;

– MARS – алгоритм, який використовує сплайни багатовимірної адаптивної регресії.

Алгоритм ID3 будує дерева рішень, використовуючи підхід жадібного пошуку зверху вниз у просторі можливих гілок без повернення назад. Жадібний алгоритм завжди робить вибір, який здається найкращим на даний момент. Алгоритм ID3 використовує такі кроки:

– починається з початкового набору *S*, який водночас являється кореневим вузлом;

– кожна ітерація алгоритму полягає в тому, що він перебирає невикористаний атрибут набору S та обчислює для нього ентропію та інформаційний приріст;

– далі алгоритм обирає атрибут, який має найменшу ентропію або найбільший приріст інформації;

– набір S розбивається обраним атрибутом для отримання підмножини даних;

– алгоритм продовжує повторюватися для кожної підмножини, враховуючи лише атрибути, які раніше не вибиралися [33 – 35].

Якщо набір даних складається з N атрибутів, то рішення про те, який атрибут розмістити в корені або на різних рівнях дерева як внутрішні вузли являється складним кроком. Простий випадковий вибір будь-якого вузла як кореневого не вирішує проблему. Також якщо дотримуватися підходу випадкового вибору, то можна отримати поганий результат з низькою точністю.

Для вирішення проблеми вибору атрибутів, використовуються певна низка критеріїв, таких як: ентропія, приріст інформації, індекс Джині, коефіцієнт посилення, зменшення дисперсії, *xi*-квадрат. Дані критерії обчислюють значення кожного атрибуту. Значення сортуються, а атрибути розміщуються в дереві відповідно до порядку, тобто атрибут з найбільшим значенням розміщується в корені. Наприклад використовуючи приріст інформації як критерій, припускається, що атрибути є категоричними, а для індексу Джині атрибути вважаються безперервними.

Ентропія є мірою випадковості інформації, що обробляється. Чим вища ентропія, тим важче зробити певні висновки з даної інформації. Прикладом дії, що надає випадкову інформацію може бути підкидання монети.

На рисунку 2.1 зображено графік залежності ентропії від ймовірності.

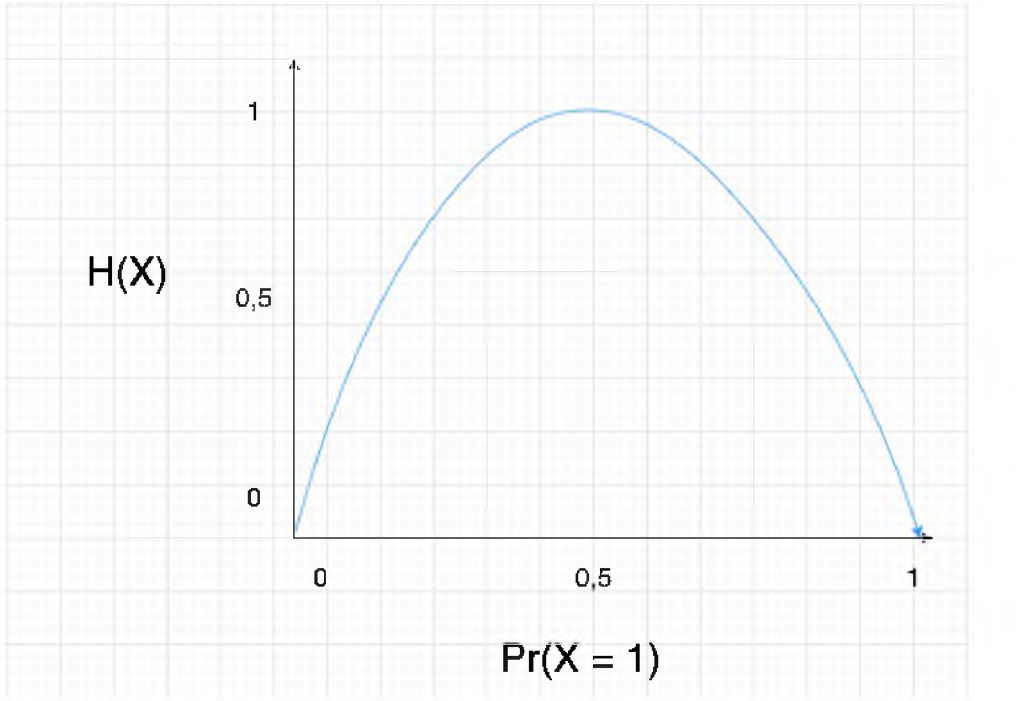

Рисунок 2.1 – Графік залежності ентропії від ймовірності

На рисунку 2.1 показано, що ентропія $H(X)$ дорівнює нулю, коли ймовірність дорівнює 0 або 1. Ентропія є максимальною, коли ймовірність дорівнює 0,5, оскільки вона проектує ідеальну випадковість даних та немає жодного шансу, якщо ідеально визначити результат.

Алгоритм ID3 дотримується правила, що гілка з нульовою ентропією є листовим вузлом, а гілка з ентропією більшою за нуль потребує подальшого розбиття. Математично ентропія представляється таким чином:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i, \tag{2.1}$$

де S – поточний стан;

p_i – ймовірність події та стану S або відсоток класу i у вузлі стану S [35].

Математично ентропія для кількох атрибутів представляється таким чином:

$$E(T,X) = \sum_{c \in X} P(c)E(c), \tag{2.2}$$

де T – поточний стан;

X – обраний атрибут [36].

Інформаційний приріст(IG) – це статистична властивість, яка вимірює, наскільки якісно даний атрибут розділяє навчальні дані відповідно до їх цільової класифікації. Основна задача побудови дерева рішень полягає в тому, щоб знайти атрибут, який повертає найбільший приріст інформації та найменшу ентропію. На рисунку 2.2 ілюстровано діаграму інформаційного приросту.

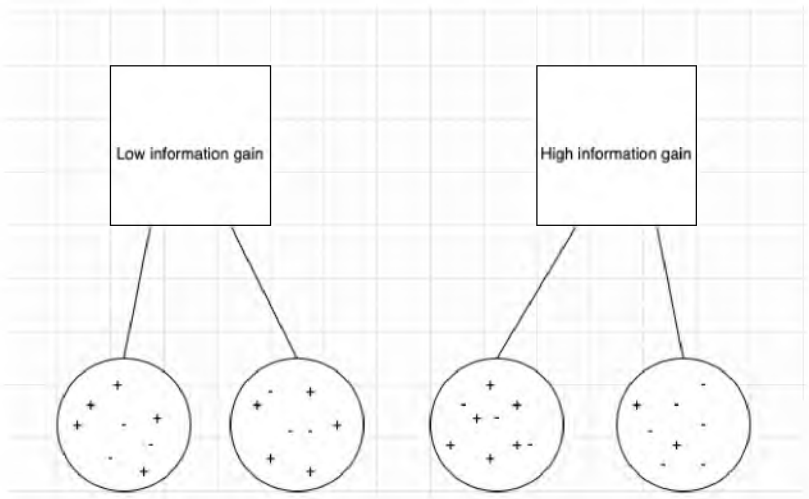

Рисунок 2.2 – Діаграма зображення інформаційного приросту

Приріст інформації – це зменшення ентропії. Він обчислює різницю між ентропією до поділу та середньою ентропією після поділу набору даних на основі заданих значень атрибутів. Алгоритм дерева рішень ID3 використовує дану властивість. Математично IG можна представити у такому вигляді:

$$Information\ Gain(T,X) = Entropy(T) - Entropy(T,X). \tag{2.3}$$

Можна представити дану формулу у більш простому вигляді:

$$Information\ Gain\ =\ Entropy(before)\ -\sum_{j=1}^{K} Entropy(j, after), \qquad (2.4)$$

де *before* – це набір даних до поділу;

K – це кількість підмножин, згенерованих розбиттям;

(j, after) – це підмножина *j* після поділу.

Можна розуміти індекс Джині як функцію вартості, яка використовується для оцінки розбиття набору даних. Індекс Джині обчислюється шляхом віднімання суми квадратів ймовірностей кожного класу з одиниці. IG віддає перевагу більшим розділам, також є простим у застосуванні, але в той же час при отриманні інформації віддає перевагу меншим розділам з різними значеннями. Математично індекс Джині представляється як:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2. \qquad (2.5)$$

Індекс Джині використовує категоріальну цільову змінну, а саме «успіх» або «невдача». Дана властивість виконує лише двійкові розбиття. Чим більше значення індексу Джині, тим більшою є нерівність та однорідність. Для того щоб виконати розрахунок індексу Джині для розбиття, спочатку потрібно обчислити IG для підвузлів, використовуючи формулу наведену нижче:

$$(p^2 + q^2), \qquad (2.6)$$

де p – цільова змінна «успіху»;

q – цільова змінна «невдачі».

Далі потрібно обчислити значення Джині для поділу, використовуючи зважену оцінку Джині кожного вузла цього поділу. Прикладом використання індексу Джині для створення точок розділення є алгоритм дерева рішень CART.

Коефіцієнт підсилення спрямований на вибір атрибутів з великою кількістю значень як кореневих вузлів. Таким чином він надає перевагу атрибуту з великою кількістю різних значень.

C4.5 використовує коефіцієнт підсилення, який є модифікацією підсилення інформації, що зменшує його зміщення та зазвичай виявляється варіантом для вирішення задач алгоритму. Коефіцієнт підсилення долає проблему з отриманням інформації, враховуючи кількість розгалужень, які виникають перед розподілом. Він коригує процес отримання інформації, беручи до уваги внутрішню інформацію розщеплення.

$$Gain\ Ratio = \frac{Entropy(before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j\ log_2\ w_j}, \tag{2.7}$$

де *before* – це набір даних до поділу;

K – це кількість підмножин, які були згенеровані розбиттям;

(j, after) – це підмножина *j* після поділу.

Зменшення дисперсії – це алгоритм, який використовується у роботі з безперервними цільовими змінним, які в свою чергу найчастіше використовуються при вирішенні задач регресії. Цей алгоритм використовує стандартну формулу дисперсії для вибору найкращого розподілу. Поділ з меншою дисперсією обирається як критерій для поділу сукупностей.

$$\frac{\sum(X - \bar{X})^2}{n}, \tag{2.8}$$

де $\bar{X}$ – середнє значення;

X – фактичне значення;

n – кількість значень.

Для обчислення дисперсії потрібно спочатку обчислити дисперсію для кожного вузла, а потім обчислити дисперсію для кожного розбиття як середньоквадратичну дисперсію кожного вузла.

Абревіатура CHAID розшифровується як *xi*-квадрат автоматичного детектора взаємодії – це один з найдавніших методів класифікації дерев. Даний метод визначає статистичну значущість між підвузлами та кореневим вузлом. Вимірюється він сумою квадратів стандартизованих відмінностей між спостережуваною та очікуваною частотами цільової змінної. Працює з категоріальною цільовою змінною «успіх» або «невдача», виконуючи два або більше розбиття. Чим вище значення *xi*-квадрат, тим вище статистична значущість відмінностей між підвузлом та кореневим вузлом. Останнім кроком є генерація дерева CHAID. Математично *xi*-квадрат можна представити як:

$$\sum \frac{(O-E)^2}{E}, \tag{2.9}$$

де O – спостережувана оцінка;

E – очікувана оцінка [37].

Для обчислення *xi*-квадрату для поділу потрібно спочатку обчислити *xi*-квадрат для окремого вузла, при цьому обчисливши відхилення як для «успіху», так і для «невдачі». Розрахувати *xi*-квадрат розділення використовуючи суми усіх *xi*-квадрат кожного вузла розділення.

Переобладнання є поширеною проблемою дерев рішень, особливо коли вхідна таблиця має велику кількість стовбців. Якщо для дерева не встановлено обмеження, це може призвести до погіршення точності використання методу. Є два способи усунення переобладнання:

– обрізка дерев рішень;

– використання методу випадковий ліс.

У висновку можна сказати, що алгоритми методу дерева рішень напряму залежать від налаштування параметрів. Кожен параметр має певну область алгоритмів в яких він є невід'ємною частиною реалізації.

2.2 Математична модель методу Random Forest

Випадковість має продуктивний вимір залежно від використання. Модель випадкового лісу складається з кількох деревних моделей, які створені творчим використанням двох типів випадковості. Кожне дерево будується на випадково обраному наборі зразків шляхом застосування технології Bootstrap до вихідного набору даних. Також при створенні кожного дерева, випадково вибрана підмножина ознак використовується для вибору найкращого розподілу. На рисунку 2.3 представлена схема випадкового лісу з використанням технології Bootstrap.

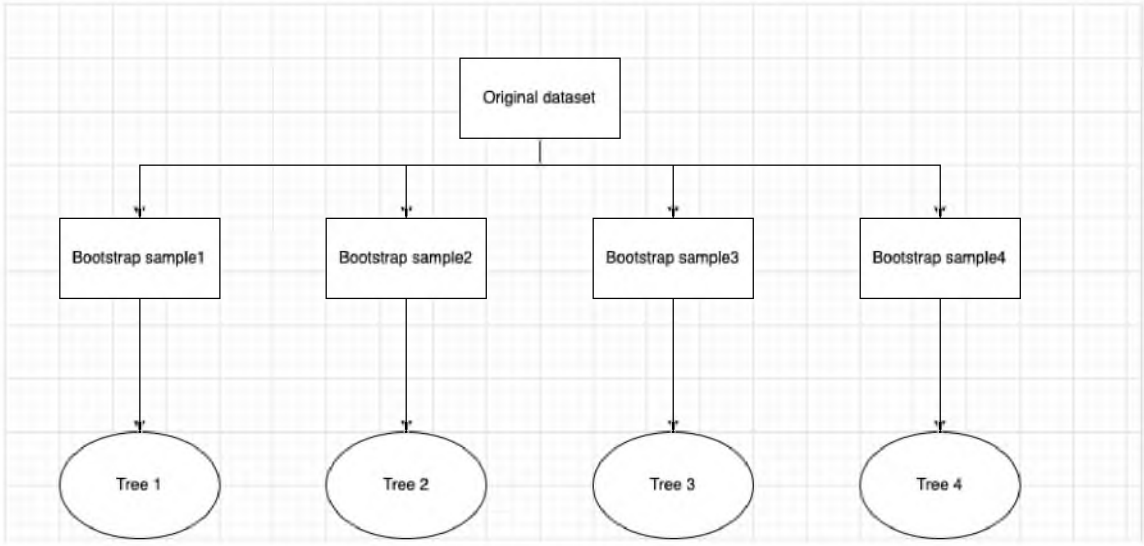

Рисунок 2.3 – Використання Bootstrap для створення дерев рішень

Випадковий ліс має досить чудову продуктивність у багатьох додатках, хоча необов'язково, щоб агрегація багатьох моделей призвела до кращої продуктивності.

Подібно до дерева рішень, процес навчання випадкових лісів слідує структурі алгоритмічного моделювання. Метод використовує організований набір евристики, а не математичну характеристику. Представимо процес побудови моделі випадкового лісу на прикладі з невеликим набором даних, який має два предиктори та змінну результату з двома класами. Отриманий набір даних показано у таблиці 2.1.

Таблиця 2.1 – Приклад набору даних

ID	x_1	x_2	**Class**
1	1	1	C_0
2	1	0	C_1
3	0	1	C_1
4	0	0	C_0

Як показано у таблиці 2.1, кожне дерево побудовано на наборі даних з повторною вибіркою, яка складається з екземплярів даних, випадково вибраних з вихідного набору даних, тобто часто з таким самим розміром вибірки як і вхідний набір даних, називається вибіркою з заміною.

Перший набір даних включає в собі екземпляри даних, які представлені їхніми ідентифікаторами {1,1,3,4} і використовується для побудови першого дерева. Другий набір даних із повторною вибіркою включає екземпляри даних {2,3,4,4} і використовується для побудови другого дерева. Цей процес повторюється, доки не буде побудовано певну кількість дерев. Перше дерево починається з кореневого вузла, який містить екземпляри даних {1,1,3,4}.

На рисунку 2.4 зображено приклад завантажувальних наборів даних.

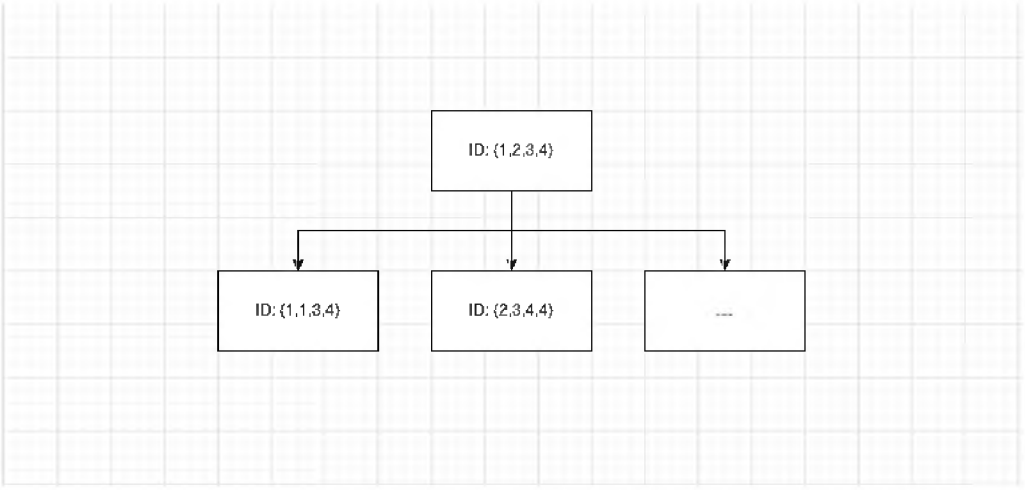

Рисунок 2.4 – Приклад завантажувальних наборів даних

Далі рекурсивно розбивається вузол на два дочірні вузли, щоб зменшити домішки. Цей жадібний рекурсивний процес розбиття також використовується для побудови кожного дерева рішень у моделі випадкового лісу. Також при побудові випадкового лісу використовується індекс Джині для вимірювання домішок замість ентропії. Індекс Джині відіграє ту ж роль, що й ентропія. Подібно до приросту інформації, приріст Джині можна математично визначити як:

$$\nabla Gini = Gini_s - \sum_{i=1,\dots,n} w_i Gini_i, \tag{2.10}$$

де $Gini_s$ – це індекс Джині у вузлі, який потрібно розділити;

w_i та $Gini_i$ – це частка вибірок та індекс Джині у дочірньому вузлі відповідно.

Повертаючись до кореневого вузла, можна його розділити за такими правилами:

– $x_1 = 0$ проти $x_1 \neq 0$;

– $x_2 = 0$ проти $x_2 \neq 0$.

Модель дерева рішень оцінює кожне з можливих правил розбиття та обирає те, що дає максимальний приріст індексу Джині для розбиття вузла.

Однак для випадкових лісів він випадковим чином обирає змінні для розбиття вузла. Так як в даному прикладі присутні дві змінні, можна припустити, що x_1 обрано випадковим чином для розбиття кореневого вузла. Таким чином, $x_1 = 0$ використовується для розбиття кореневого вузла, який генерує модель дерева рішень, схематичне зображення цього процесу показано на рисунку 2.5.

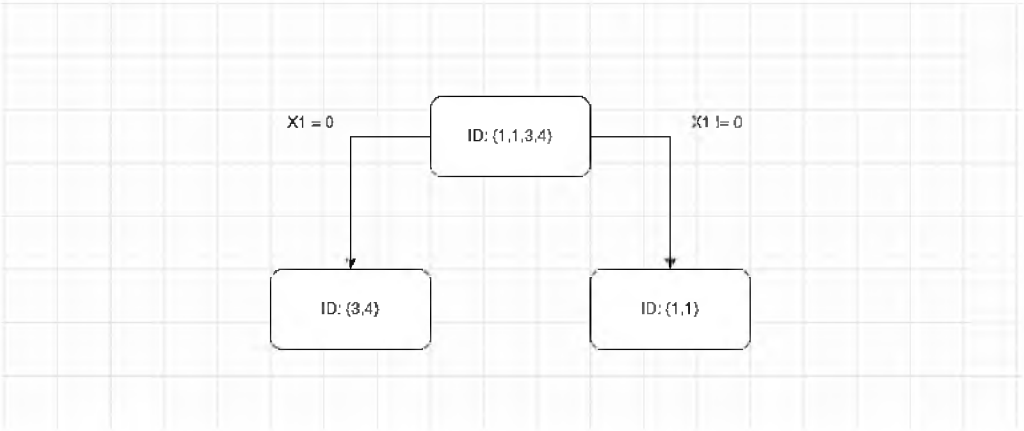

Рисунок 2.5 – Приклад розділення кореневого вузла за допомогою $x_1 = 0$

Розглядаючи правий вузол на рисунку 2.5, можна сказати, що він досяг ідеального стану однорідності. Однак лівий вузол містить два екземпляри {3,4}, які також пов'язані з двома класами. Наступним кроком розкладається лівий вузол, припускаючи, що час обрано випадково. На рисунку 2.6 зображений спосіб розділення лівого вузла.

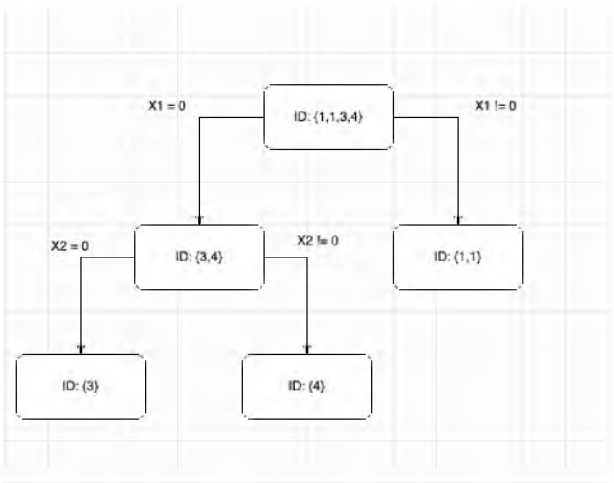

Рисунок 2.6 – Приклад розділення лівого вузла за допомогою $x_2 = 0$

Усі вузли не можна розділити далі. На рисунку 2.7 зображена остаточна модель дерева, де кожен листовий вузол позначений основним класом екземплярів у вузлі, щоб вони перетворилися на вузли прийняття рішень.

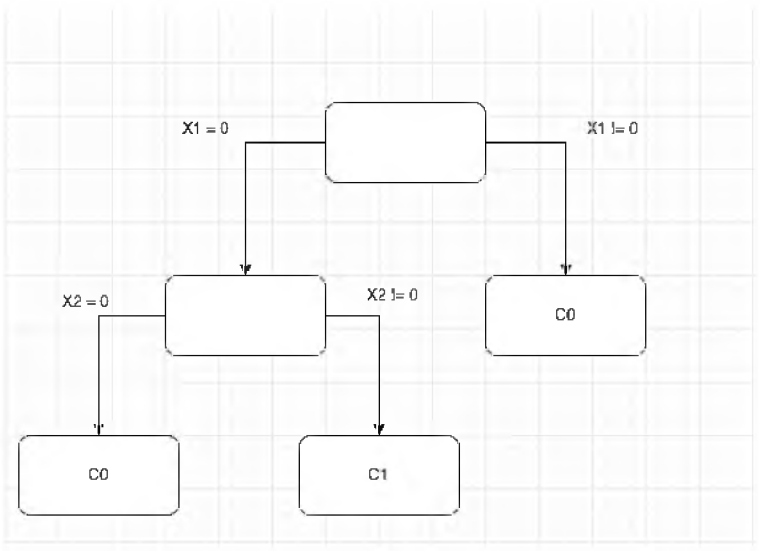

Рисунок 2.7 – Приклад навченої моделі дерева рішень

Використовуючи кінцеву навчену модель дерева рішень, яка зображена на рисунку 2.7 та початкових даних, які описано у таблиці 2.1, можна отримати кінцеве прогнозування для використаного набору даних. Частота помилки становить 25%. Кінцевий результат прогнозування зображений у таблиці 2.2.

Таблиця 2.2 – Кінцевий результат прогнозування

ID	x_1	x_2	**Class**	**Prediction**
1	1	1	C_0	C_0
2	1	0	C_1	C_0
3	0	1	C_1	C_1
4	0	0	C_0	C_1

Таким самим чином можна побудувати всі інші дерева. Зазвичай у випадкових моделях лісу обрізка дерев не потрібна, але в даному прикладі використовувався параметр для керування глибиною моделей дерев, які

потрібно створити. Коли будується модель випадкового лісу, щоб зробити прогноз для точки даних, кожне дерево повинно зробити прогноз, а потім усі прогнози об'єднуються. Наприклад, для безперервної змінної результатом прогнозу є середнє значення усіх прогнозів, в той час для змінної результату класифікації остаточним прогнозом є клас, який виграє більшість серед усіх інших дерев.

Модель випадкового лісу показує, що випадковість, яка зазвичай вважається порушником спокою, має продуктивний вимір. Випадкові ліси разом з іншими моделями, які називають моделями ансамблевого навчання, можуть змусити групу слабких моделей об'єднатися, щоб сформувати сильну модель. Наприклад, розглядаючи випадкову модель лісу, яка містить в собі 100 дерев рішень.

Кожне дерево цього лісу є слабкою моделлю, точність становить 0,6. Дерева є незалежними, тобто передбачення одного дерева не має нічого спільного з передбаченням для іншого дерева. З такою кількістю дерев ймовірність, що модель випадкового лісу правильно виконає процес прогнозування для будь-якої точки даних становить 0,97, так як

$$\sum_{k=51}^{100} C(n,k) \times 0{,}6^{k} \times 0{,}4^{100-k} = 0{,}97. \qquad (2.11)$$

Це припущення не виконується в дійсності в строгому сенсі, тобто в ідеалі, треба мати алгоритм, який знайти багато хороших моделей, але якщо створити ці моделі використовуючи один набір даних, то ці моделі матимуть схожість одна з одною. Якщо зосереджуватись виключно на моделях, які можуть досягти оптимальної продуктивності, то часто виявлені моделі виявляються однаковими. Для уникнення цієї проблеми вводиться певна випадковість, а саме використання Bootstrap технології для рандомізації вибору екземплярів даних та використання випадкового набору ознак для побудови дерев, вводиться в процес побудови моделей, щоб створити

різноманітність моделей. До уваги треба приймати динаміку між ступенем випадковості, продуктивністю кожної окремої моделі та їх продуктивністю.

Чим більше дерев містить в собі випадковий ліс, тим межа рішення стає більш точною та стабільною. На рисунку 2.8 зображено приклад межі рішення випадкового лісів з кількістю дерев 50.

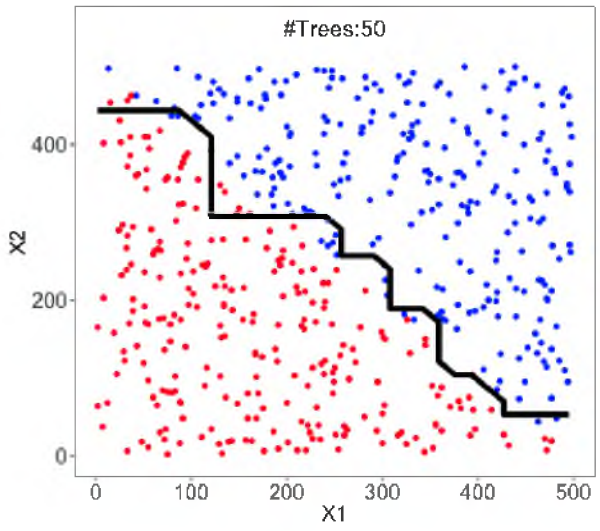

Рисунок 2.8 – Приклад межі рішень з випадкових лісів

Є декілька причин чому випадкові ліси перевершують окремі дерева рішень. Перша з них це вища роздільна здатність у просторі функцій. Дерева рішень являються необрізними. Єдине дерево рішень, наприклад CART, часто обрізається, в той час випадкове дерево лісу повністю вирощується та не обрізається, з цього випливає, що простір функції розбивається на більшу кількість малих регіонів.

Кожне випадкове дерево лісу вивчається на випадковій вибірці та в кожному вузлі розглядається випадковий набір ознак для розділення. Обидва механізми створюють різноманітність серед дерев. На рисунку 2.9 ілюстровано два випадкових дерева з одним розбиттям. Для кожного дерева можна призначити дві області з різними мітками. Об’єднавши два дерева вийде чотири регіони, які позначаються по-різному.

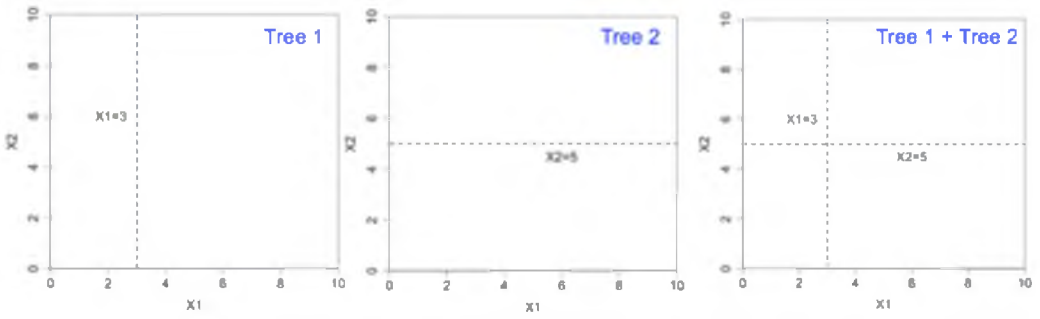

Рисунок 2.9 – Приклад двох випадкових дерев з одним розбиттям

Необразні та різноманітні дерева забезпечують високу роздільну здатність у просторі елементів [38]. Для безперервних функцій це означає більш плавну межу прийняття рішень, як це зображено на рисунку 2.10.

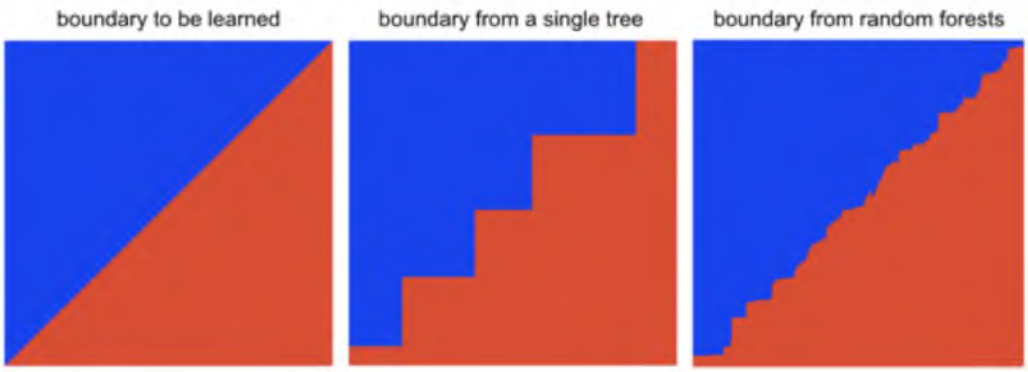

Рисунок 2.10 – Приклад порівняння меж прийняття рішень між одним деревом та випадковим лісом

Одне дерево рішень потребує обрізки, щоб уникнути переобладнання. Варто зазначити, що межа більш гладка, але допускає очевидні помилки, а саме переобладнання.

На рисунку 2.11 зображено межу рішення для необрізаного дерева, для двох класів, а саме синій та червони, обидва розбиття x_1= 3 та x_2=3 можуть повністю розділити два класи, що ілюстровано на рисунку 2.12.

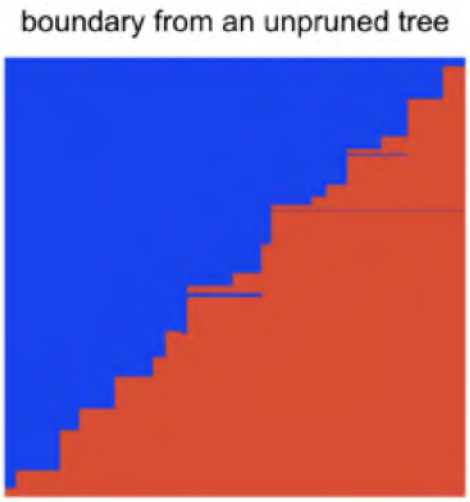

Рисунок 2.11 – Приклад межі рішення для необрізаного дерева

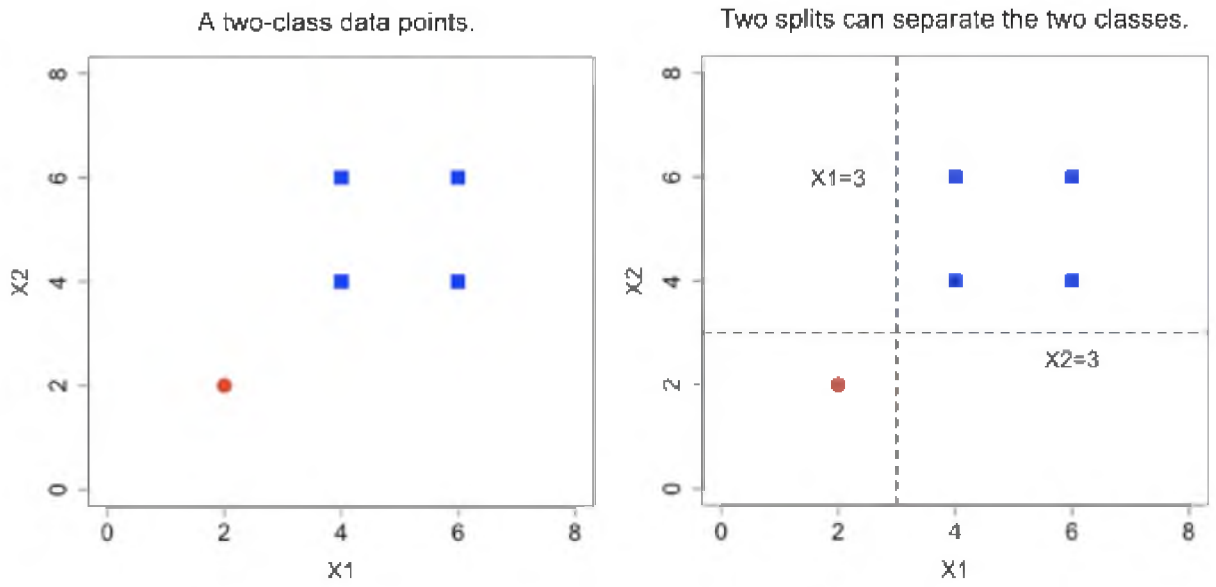

Рисунок 2.12 – Приклад розбиття двох класів

Однак два поділи призводять до дуже різних меж прийняття рішень. Дерева рішень часто використовують першу змінну для поділу, тому порядок змінних у навчальних даних визначає межу рішень.

На рисунку 2.13 зображений приклад межі рішень, яка виникає внаслідок використання першої змінної для поділу.

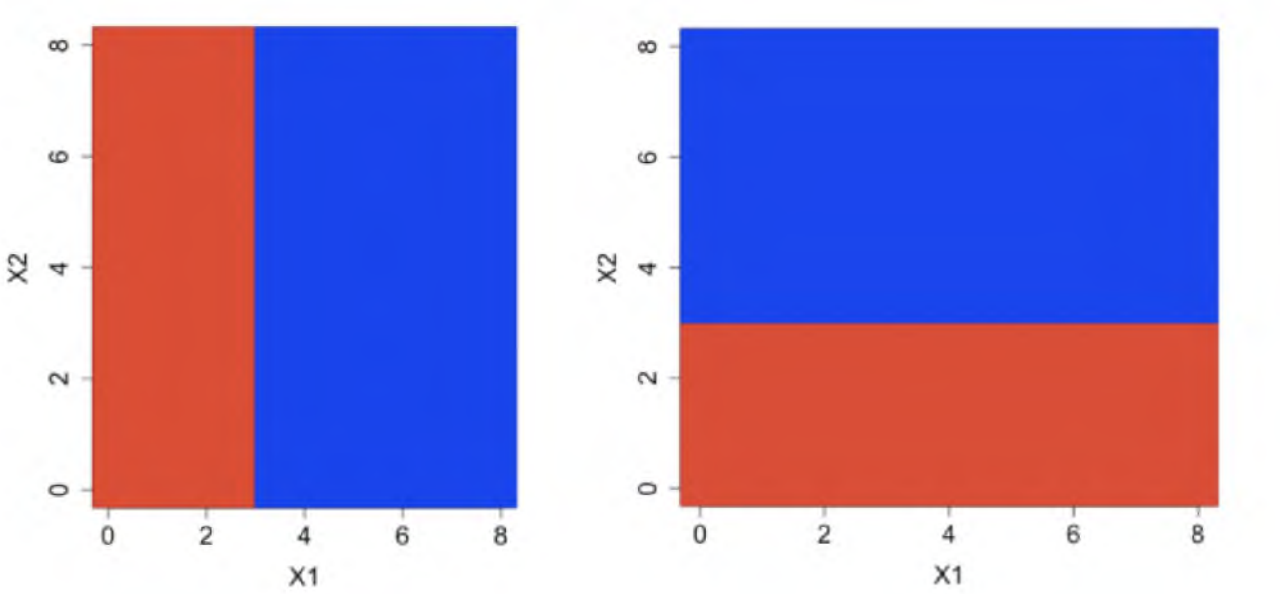

Рисунок 2.13 – Приклад межі рішень, яка виникає внаслідок використання першої змінної для поділу

Розглядаючи випадкові ліси, для кожної випадкової вибірки, яка використовується для навчання дерева, ймовірність того, що червона точка відсутня у вибірці становить тридцять три відсотки. Таким чином, приблизно одне з трьох дерев будується з усіма даними синього кольору і завжди передбачає синій клас. Інші 2/3 дерев матимуть червону точку в навчальних даних. Оскільки в кожному вузлі розглядається випадкова підмножина функцій, то очікується, що приблизно 1/3 дерев буде використовувати x_1, а решта використовуватиме x_2. На рисунку 2.14 показано приклад поділу двох типів дерев.

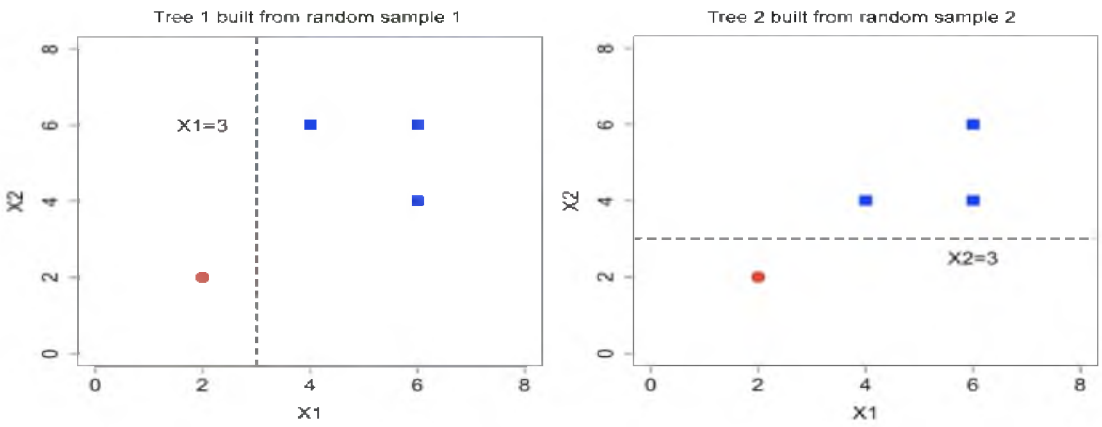

Рисунок 2.14 – Приклад поділу двох типів дерев

Поки буде існувати достатня кількість дерев, межа буде стабільною та незалежною від нерелевантної інформації, такої як порядок змінних. На рисунку 2.15 зображено об’єднання трьох типів дерев.

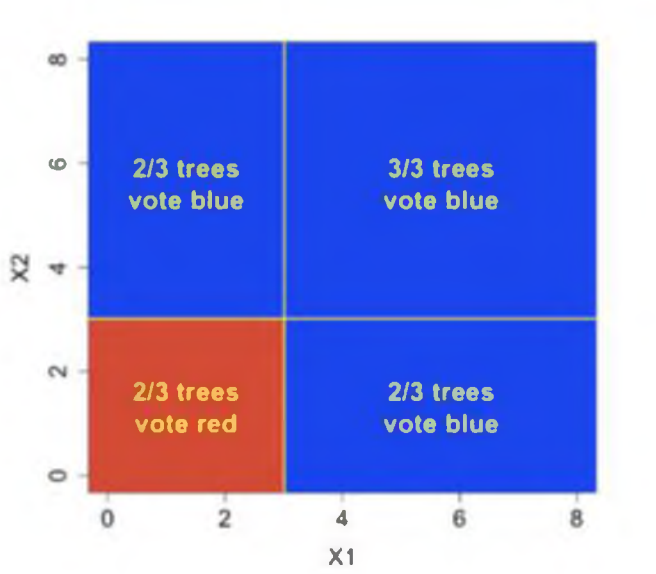

Рисунок 2.15 – Приклад об’єднання трьох типів дерев

Механізм випадковості та голосування у випадкових лісах досить гарно вирішують проблему переобладнання.

3 КОМП'ЮТЕРНА МОДЕЛЬ МЕТОДУ КЛАСИФІКАЦІЇ ДАНИХ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений алгоритм прогнозування цін нерухомості з використанням методів класифікації даних.

Для реалізації даного програмного застосунку використовувалися хмарне середовище Salesforce та програмне середовище, для програмування за допомогою мови програмування Python, яке має назву PyCharm.

Salesforce – це платформа, яка використовується для продажу, обслуговування чи аналізування певної області товарів, в даному випадку нерухомості [39].

Salesforce має все необхідне для ведення бізнесу з будь-якого місцезнаходження. Використовуючи стандартні продукти та функції, можна мати можливість керувати відносинами з потенційними клієнтами, співпрацювати та взаємодіяти зі співробітниками та партнерами, а також, найголовнішою особливістю є безпечне зберігання власних даних у хмарі. Також стандартні продукти та функції – це не всі можливості даної платформи. Її можна налаштовувати та персоналізувати, також її легко розширити різною функціональністю в залежності від потреб користувача.

CRM технологія дозволяє керувати відносинами з клієнтами та потенційними клієнтами та відстежувати дані, які пов'язані з усіма взаємодіями. Технологія також допомагає командам співпрацювати як всередині, так і ззовні, збирати статистику з соціальних мереж, відстежувати важливі показники та спілкуватися електронною поштою, телефоном, соціальними та іншими каналами. У Salesforce вся ця інформація надійно зберігається в хмарі. Саме цим фактором обумовлений вибір цієї платформи як джерела отримання даних для розробки та тестування алгоритму.

На рисунку 3.1 представлено приклад візуалізації однієї одиниці даних для виконання алгоритму з прогнозування ціни.

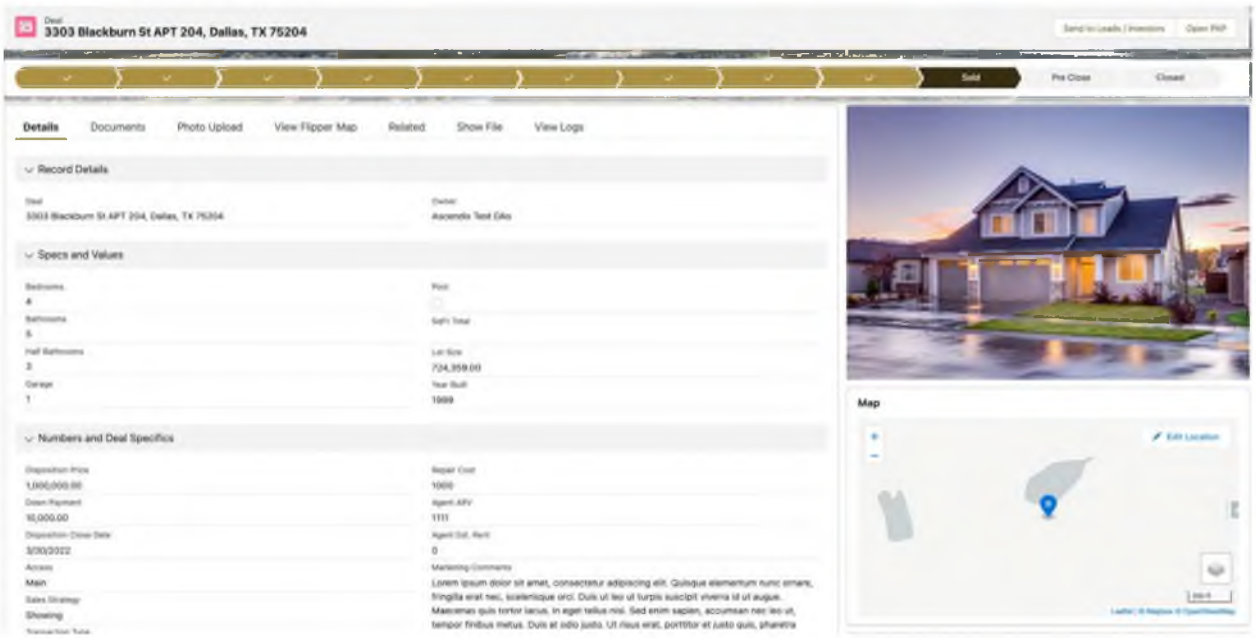

Рисунок 3.1 – Візуальний вигляд однієї одиниці даних

PyCharm – це багатофункціональне та корисне середовище, для програмування мовою Python.

PyCharm має безліч функцій продуктивності, так як підхід програмування з орієнтиром на клавіатуру, зручний командний інтерфейс, які збережуть час та збільшать якість кінцевого програмного застосунку.

За допомогою даного програмного середовища можна забезпечити себе більш якісним кодом, а саме середовище підтримує миттєву перевірку синтаксичних або лексичних помилок та їх виправлення, також надає можливість швидкого переміщення по різним частинам проєкту. Середовище використовує перевірки PEP8, допомагає в тестуванні програмного застосунку також виконує інтелектуальний рефакторинг та безліч інших перевірок.

На рисунку 3.2 зображено середовище розробки програмного застосунку PyCharm за допомогою мови програмування Python.

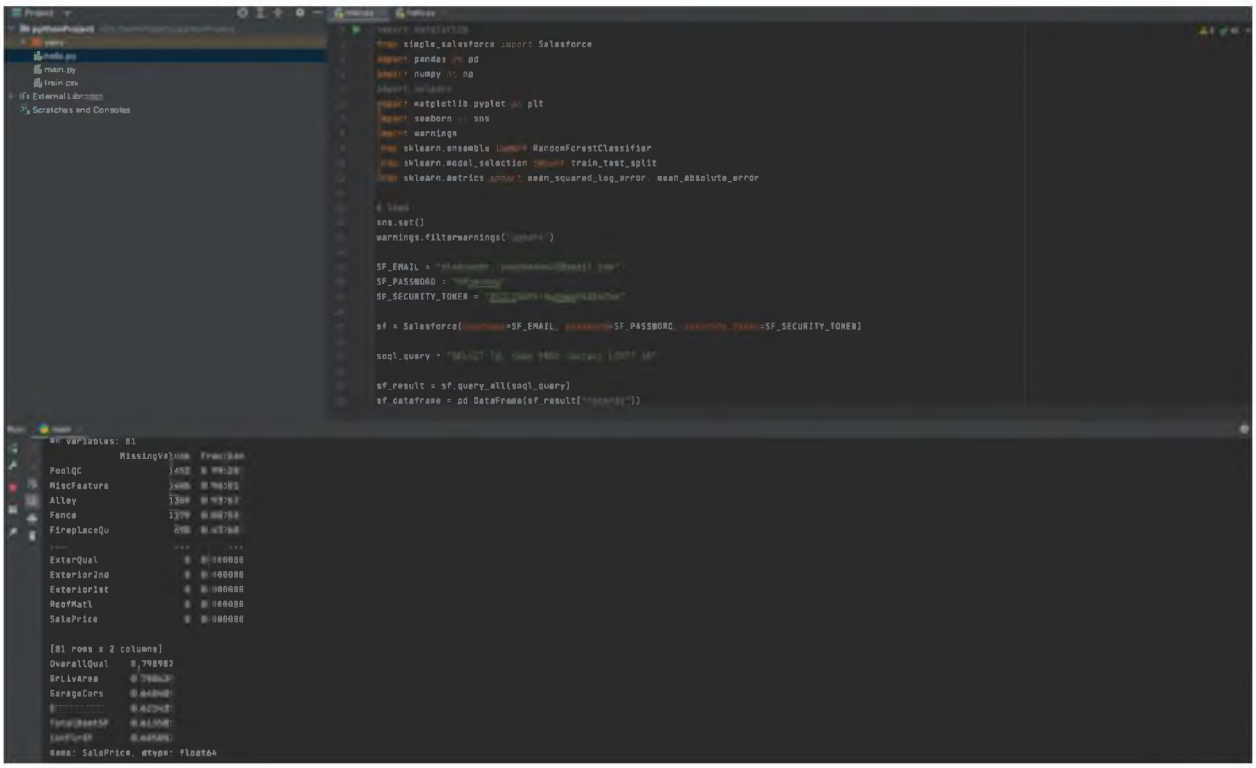

Рисунок 3.2 – Інтерфейс програмного середовища PyCharm

На рисунку 3.3 зображений приклад візуалізації графіка за допомогою середовища PyCharm.

Рисунок 3.3 – Приклад візуалізації за допомогою середовища PyCharm

Для того щоб запустити будь-яку програму або візуалізувати треба лише натиснути кнопку «старт» в середовищі і всі закодовані графіки буде візуалізовано. У висновку можна сказати, що PyCharm – це найзручніший застосунок для розробки будь-яких застосунків мовою програмування Python.

3.2 Програмна реалізація

Першим етапом програмної реалізаціє є налаштування організації Salesforce для підключення до мови програмування Python. Для цього треба перш за все створити Salesforce Connected App. На рисунку 3.4 зображено процес створення Salesforce Connected App.

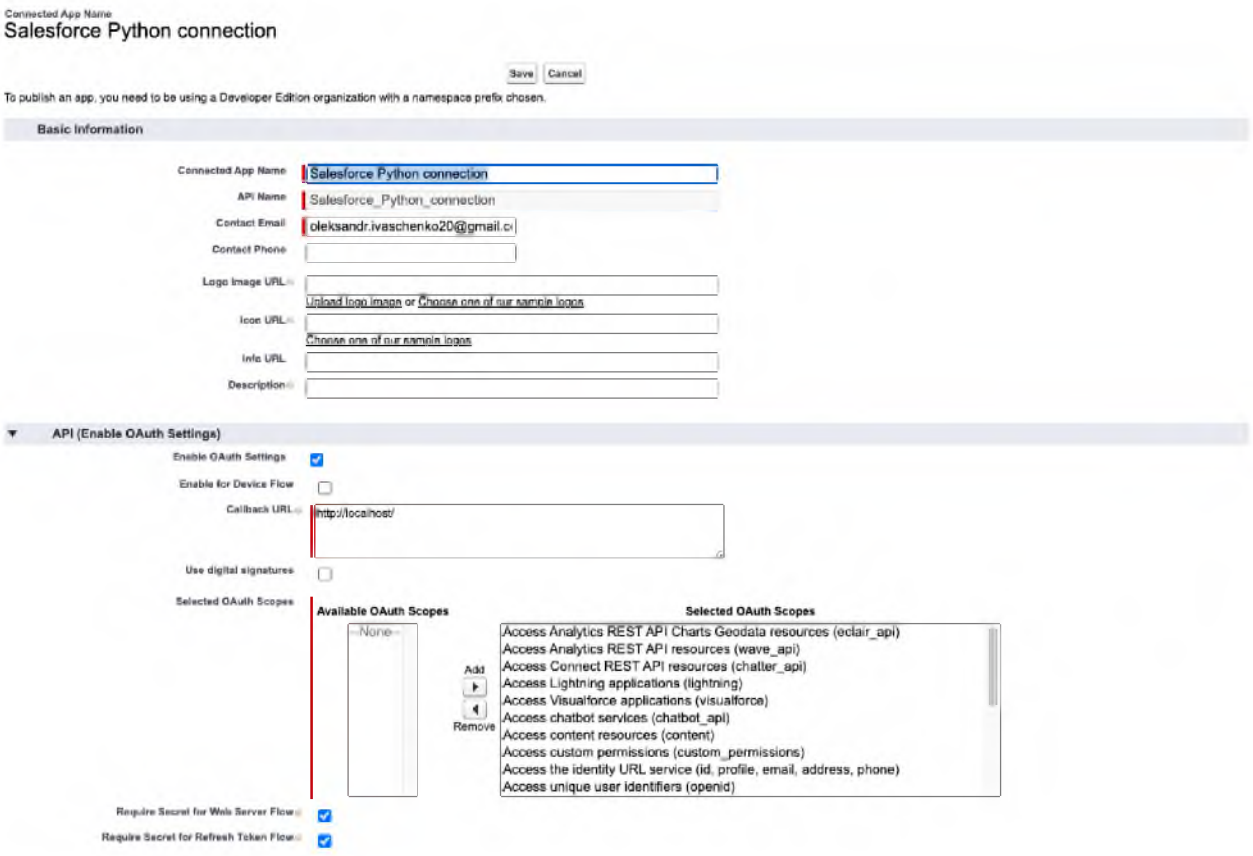

Рисунок 3.4 – Приклад створення Salesforce Connected App

Після цього потрібно створити компонент, який буде зв'язувати Salesforce та Python та посилати запит, який містить в собі набір даних нерухомості для подальшого прогнозування. Перед тим як реалізувати алгоритм потрібно проаналізувати усі Python бібліотеки та обрати ті, які будуть активно використовуватися при розробці алгоритму прогнозування цін нерухомості. Розглянемо обрані бібліотеки, щоб зрозуміти в якій частині нашого застосунку вони будуть використовуватися:

– simple_salesforce – бібліотека підключення платформи Salesforce до мови програмування Python;

– pandas – бібліотека мови програмування Python, за допомогою якої відбуваються процеси маніпулювання та аналізу даних;

– numpy – розширення мови програмування Python, для підтримки багатовимірних масивів та багаторівневих математичних функцій;

– matplotlib – бібліотека мови програмування Python, для візуалізації даних за допомогою графіків;

– scikit-learn – бібліотека машинного навчання для мови програмування Python, яка використовується для створення та тренування різних методів класифікації, кластеризації та регресії;

– seaborn – бібліотека мови програмування Python, яка використовується для побудови статичних графіків.

Після підключення всіх необхідних бібліотек наступає етап створення синхронізації між Salesforce платформою та Python проєктом. Для цього необхідно задати декілька параметрів, а саме пароль, імейл та токен безпеки юзера з'єднання. На рисунку 3.5 зображений процес підключення проєкту Python з платформою Salesforce.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_log_error, mean_absolute_error

# load
sns.set()
warnings.filterwarnings('ignore')

SF_EMAIL = "*******"
SF_PASSWORD = "****"
SF_SECURITY_TOKEN = "*******"

sf = Salesforce(username=SF_EMAIL, password=SF_PASSWORD, security_token=SF_SECURITY_TOKEN)
```

Рисунок 3.5 – Лістинг коду, який ілюструє підключення Python проєкту до Salesforce платформи

Наступним етапом розробки є отримання даних про зареєстровану нерухомість. Даний застосунок використовує дані про нерухомість за останній місяць. Для цього використовується звичайний запит до бази даних платформи Salesforce. Запит будується за допомоги мови програмування, яку використовує платформа, який називається SOQL. Наразі ліміт отримання даних за одну транзакцію дорівнює п’ять тисяч записів. На рисунку 3.6 ілюстровано процес отримання даних з платформи Salesforce.

```
soql_query = "SELECT Id, Name, pba__Address_pb__c, pba__City_pb__c," \
             "pba__PostalCode_pb__c, pba__State_pb__c, pba__Longitude_pb__c" \
            "pba__Latitude_pb__c,Deal_Scalper_ARV__c, pba__Bedrooms_pb__c," \
            "County__c, pba__FullBathrooms_pb__c, pba__HalfBathrooms_pb__c," \
            "Block__c, Legal_Description__c, Lot__c, pba__LotSize_pb__c," \
            "ParkingSpaces__c, Subdivision__c, Year_Built_Text__c, Owner_of_Property__c" \
             "FROM pba__Listing__c WHERE CreatedDate < Last_Month LIMIT 5000"

sf_result = sf.query_all(soql_query)
sf_dataframe = pd.DataFrame(sf_result["records"])
sf_dataframe = sf_dataframe.drop(columns="attributes")
```

Рисунок 3.6 – Лістинг коду, який ілюструю отримання даних про нерухомість за останній місяць

В даному наборі даних приблизно вісімдесят змінних. Для більш кращої оптимізації, потрібно визначити які самі змінні потрібно використовувати, а які ні. Змінна SalePrice є цільовою змінною, вона буде цільовою для передбачення з урахуванням інших доступних змінних. Щоб визначити, які саме змінні можуть бути зайвими, потрібно дізнатися кількість невизначених значень в змінних. Ті змінні, які матимуть багато невизначених значень можна відкинути, так як вони не матимуть значного внеску на передбачуваність цільової змінної. Також ми ліквідуємо будь-яку змінну з відсутнім стовпцем. Цей процес може погіршити кінцевий прогноз, але це гарантує, що не доведеться робити жодних припущень щодо цих змінних, що також може бути небезпечним. На рисунку 3.7 ілюстровано процес чистки невикористовуваних значень.

```
# get all columns
df_train = pd.read_csv('train.csv')
print(df_train.columns.values)
print('No variables.', len(df_train.columns.values))

#clean missing data
num_missing = df_train.isnull().sum()
percent = num_missing / df_train.isnull().count()

df_missing = pd.concat([num_missing, percent], axis=1, keys=['MissingValues', 'Fraction'])
df_missing = df_missing.sort_values('Fraction', ascending=False)
var = df_missing[df_missing['MissingValues'] > 0]
print(df_missing)

variables_to_keep = df_missing[df_missing['MissingValues'] == 0].index
df_train = df_train[variables_to_keep]
```

Рисунок 3.7 – Лістинг коду, який ілюструє процес чистки вхідних даних

Наступним етапом розробки програмного застосунку є аналіз даних. Основною ціллю аналізу даних є аналіз змінних та основних зв'язків між ними. Для цього потрібно спочатку побудувати кореляційну матрицю. На рисунках 3.8 та 3.9 зображено процес побудови кореляційної матриці для всіх змінних та приклад візуалізації кореляційної матриці по всіх змінних відповідно.

```
#Build the correlation matrix
matrix = df_train.corr()
f, ax = plt.subplots(figsize=(16, 12))
sns.heatmap(matrix, vmax=0.7, square=True)
```

Рисунок 3.8 – Лістинг коду, який ілюструє процес побудови кореляційної матриці для всіх змінних

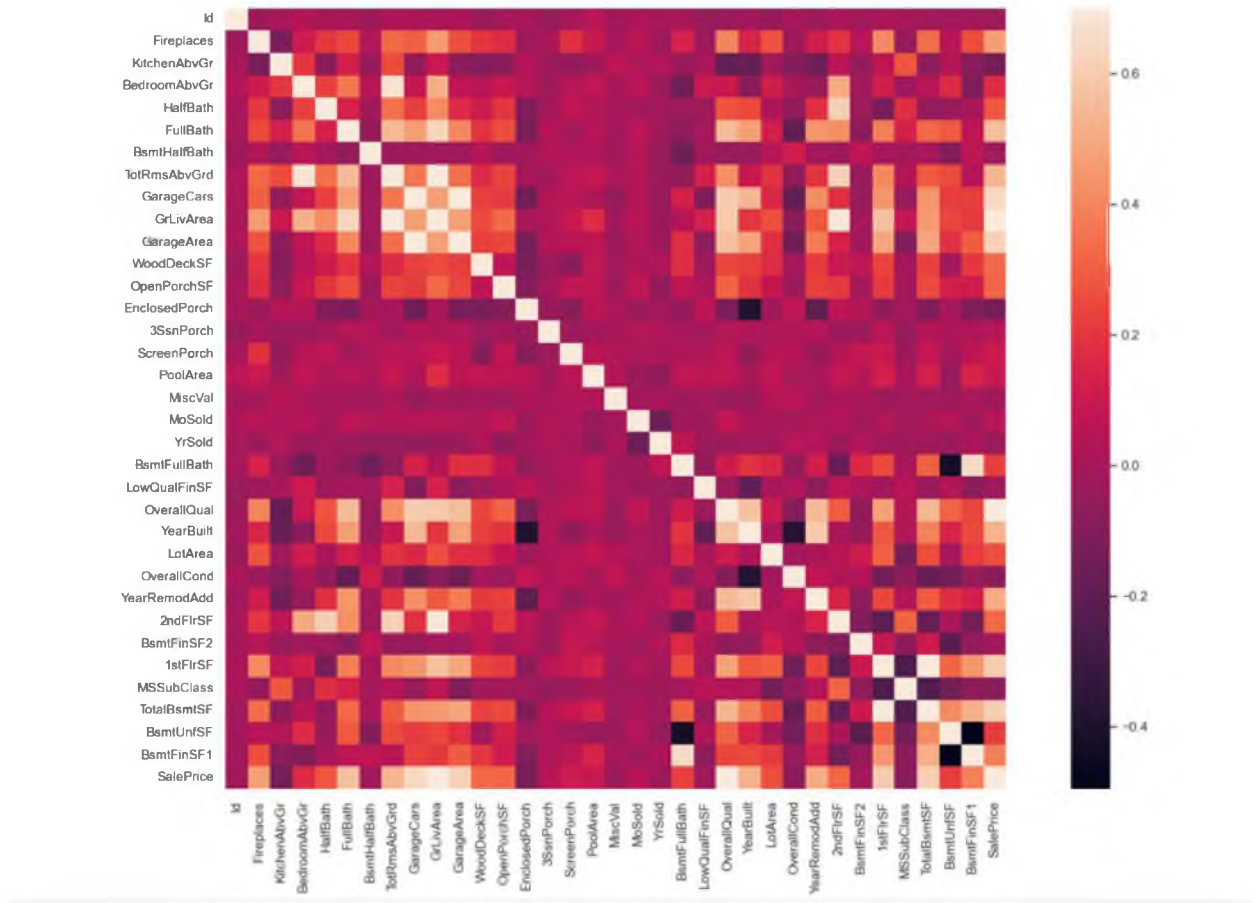

Рисунок 3.9 – Приклад візуалізації кореляційної матриці по всіх змінних

Тепер треба взяти змінну, яка є цільовою, а саме, в даному випадку, це змінна SalePrice та визначити змінні, які є тісно пов'язаними до неї. На рисунках 3.10 та 3.11 зображено визначення змінних, які є тісно пов'язаними з цільовою змінною та приклад отримання тісно пов'язаних змінних відповідно.

```
interesting_variables = matrix['SalePrice'].sort_values(ascending=False)
interesting_variables = interesting_variables[abs(interesting_variables) >= 0.6]
interesting_variables = interesting_variables[interesting_variables.index != 'SalePrice']
print(interesting_variables)
```

Рисунок 3.10 – Лістинг коду, який ілюструє визначення змінних, які є тісно пов’язаними з цільовою змінною

```
OverallQual    0.790982
GrLivArea      0.708624
GarageCars     0.640409
GarageArea     0.623431
TotalBsmtSF    0.613581
1stFlrSF       0.605852
```

Рисунок 3.11 – Приклад отримання тісно пов’язаних змінних

Далі обирається змінна, яка матиме найбільшу якість в взаємодії з цільовою змінною, її можна вважати найбільш передбачуваною змінною. Наступним етапом буде збільшення масштабу найбільш передбачуваної змінної. На рисунку 3.12 ілюстровано аналіз та знаходження всіх унікальних значень найбільш передбачуваної змінної.

```
values = np.sort(df_train['OverallQual'].unique())
print('Unique values of "OverallQual":', values)
```

Рисунок 3.12 – Лістинг коду, який ілюструє пошук унікальних значень найбільш передбачуваної змінної

На рисунку 3.13 можна побачити детальний графік взаємодії найбільш передбачуваної та цільової змінних.

Ця тенденція повинна чітко повторюватися, потрібно також проаналізувати всі інші змінні, які представляють інтерес.

На рисунку 3.14 зображено детальні графіки для усіх змінних, які представляють інтерес.

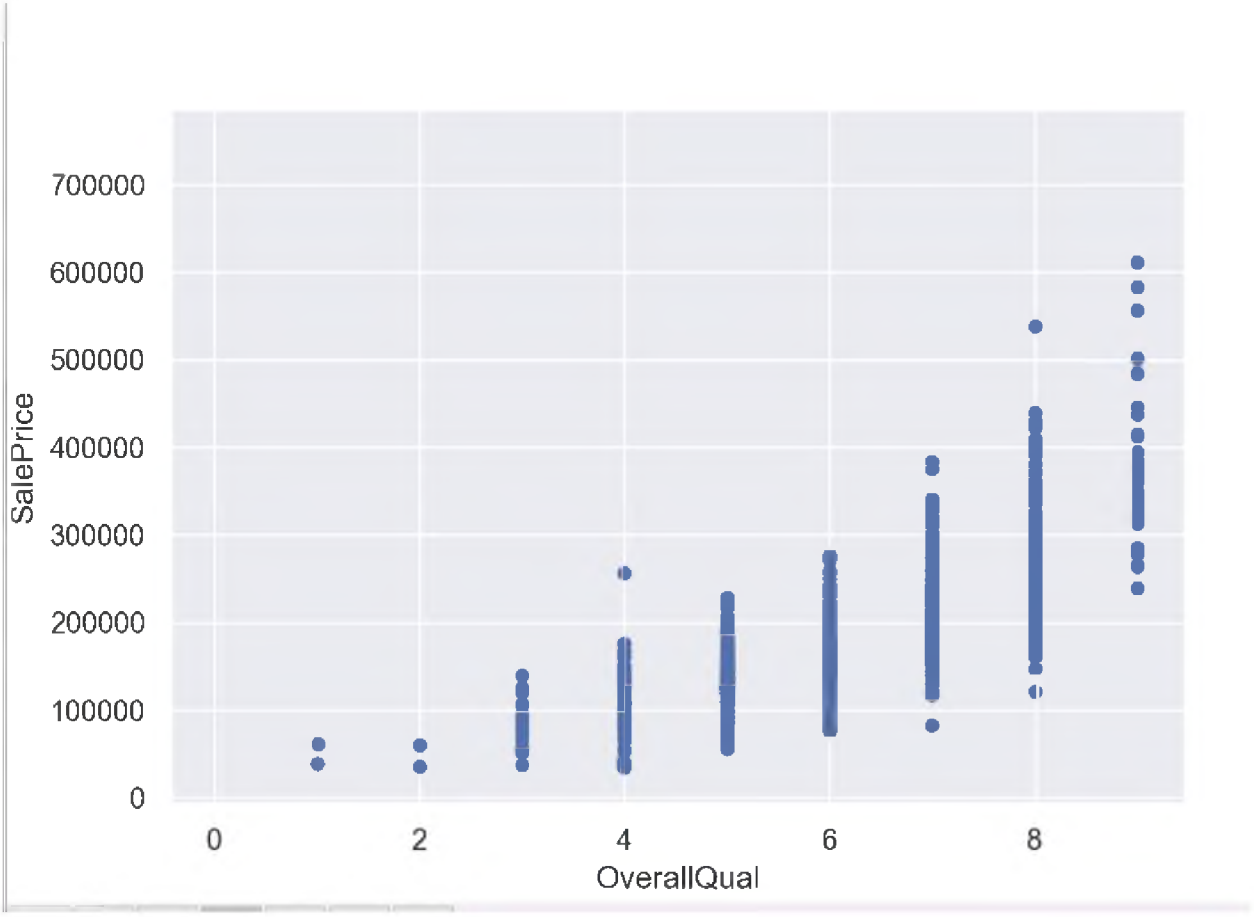

Рисунок 3.13 – Графік взаємодії найбільш передбачуваної та цільової змінних

Рисунок 3.14 – Детальні графіки аналізу усіх змінних, які представляють інтерес

Даний аналіз дає певні підказки про типи різних змінних. Виходячи з отриманих результатів можна сказати, що присутні декілька дискретних змінних та певна кількість безперервних змінних. Далі можна збільшити кореляційну матрицю зображену на рисунку 3.9, але показуючи на ній лише змінні, які представляють інтерес для прогнозування. Такий підхід може виявити деякі приховані взаємозв'язки між змінними. На рисунку 3.15 ілюстровано обмежену кореляційну матрицю, яка показує лише ті змінні, які представляють інтерес.

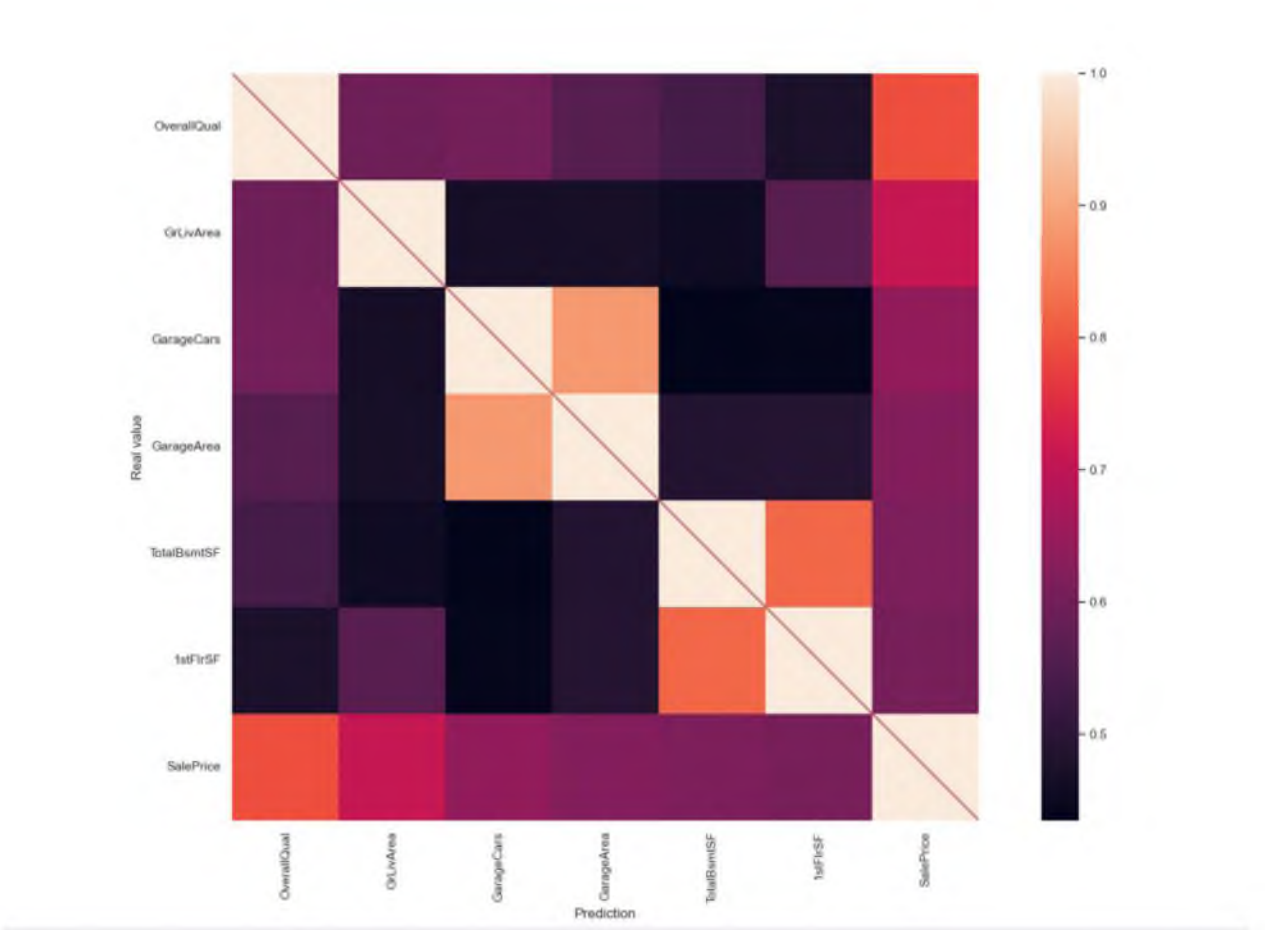

Рисунок 3.15 – Приклад обмеженої кореляційної матриці, яка показує лише змінні, які представляють інтерес

Виходячи з даного прикладу, можна знайти певні залежності, такі як кластери, що має велике значення при прогнозуванні. Також важливою деталлю є зображення взаємодії цільової змінної з іншими присутніми на цій

матриці. Процес аналізу можна вважати завершеним, тепер можна перейти до кінцевого етапу розробки програмного застосунку, а саме машинного навчання та прогнозування ціни нерухомості.

Критична відмінність між алгоритмами випадкового лісу та дерева рішень полягає в тому, що дерева рішень – це графіки, які ілюструють усі можливі результати рішення за допомогою підходу розгалуження, в той час як алгоритм випадкового лісу є набором дерев рішень, які працюють відповідно до результату.

Зазвичай використовується алгоритм випадкового лісу, адже він дуже точний та зважаючи на те, що сучасні комп'ютерні системи здатні обробляти великі набори даних, які раніше були некерованими.

Недоліком алгоритму випадкового лісу є те, що відсутня можливість остаточної візуалізації моделі при відсутності достатньої обчислювальної потужності.

Перевага простого дерева рішень полягає в тому, що модель легко інтерпретувати. При побудові дерева рішень, одразу зрозуміло, яка змінна та яке значення використовується змінною для розділення даних, що дозволяє швидко прогнозувати кінцевий результат. З іншого боку, моделі алгоритму випадкового лісу є більш складними, оскільки вони являють собою комбінації дерев рішень. Під час побудови моделі алгоритму випадкового лісу, треба визначити, скільки дерев потрібно створити та скільки змінних потрібно для кожного вузла.

Загалом, більше дерев рішень підвищить продуктивність та зробить прогнози більш стабільними, але в цей же час сповільнить швидкість обчислення. Для задач регресії за кінцевий результат береться середнє значення всіх дерев. Регресійна модель алгоритму випадкового лісу має два рівні середніх: спочатку вибірка в цільовій клітинці дерева, потім усі дерева. На відміну від лінійної регресії, вона використовує наявні спостереження для оцінки значень за межами спостережуваного діапазону.

Для більш точних прогнозів потрібно більше дерев, що призводить до повільніших моделей. Якби існував спосіб створити велику кількість дерев шляхом усереднення їхніх рішень, то кінцевий результат був би дуже близький до справжнього.

Деякі відмінності між моделями алгоритмів дерева рішень та випадкового лісу наведені у таблиці 3.1.

Таблиця. 3.1 – Відмінності між моделями алгоритмів дерева рішень та випадкового лісу.

Дерево рішень	**Випадковий ліс**
Деревопобідна модель рішень разом з можливими результатами на діаграмі	Алгоритм класифікації, що складається з багатьох дерев рішень, об'єднаних для отримання точнішого результату порівняно з одним деревом
Завжди є можливість для переобладнання, викликаного наявністю дисперсії	Алгоритм випадкового лісу уникає та запобігає переобладнанню за допомогою кількох дерев
Результати не точні	Надає точні та чіткі результати
Вимагає низьких обчислень, що скорочує час на реалізацію та забезпечує низьку точність	Потребує більше обчислень. Процес генерації та аналізу займає багато часу
Легко візуалізувати. Єдине завдання – підібрати модель дерева рішень	Має складну візуалізацію, оскільки визначає шаблон, що стоїть за даними

За основу використовується класифікатор Random Forest. Якщо більш детальніше розглядати, то фактично це регресія випадкового лісу, так як цільова змінна – це безперервне дійсне число. Основна реалізація полягає в розділенні наборів тренувальних даних на один цілий набір та тестовий набір,

оскільки немає зацікавленості у запуску аналізу лише на тестовому наборі даних. На рисунку 3.16 показана реалізація розділення набору та використання RandomForestClassifier для подальшого прогнозування ціни.

```
pred_vars = [v for v in interesting_variables.index.values if v != 'SalePrice']
target_var = 'SalePrice'

X = df_train[pred_vars]
y = df_train[target_var]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Рисунок 3.16 – Лістинг коду, який ілюструє використання RandomForestClassifier

Детальний результат у вигляді графіку на якому зображена ціна, яка була цільовою змінною, лінією ідеального прогнозування та прогнозованою ціною зображено на рисунку 3.17.

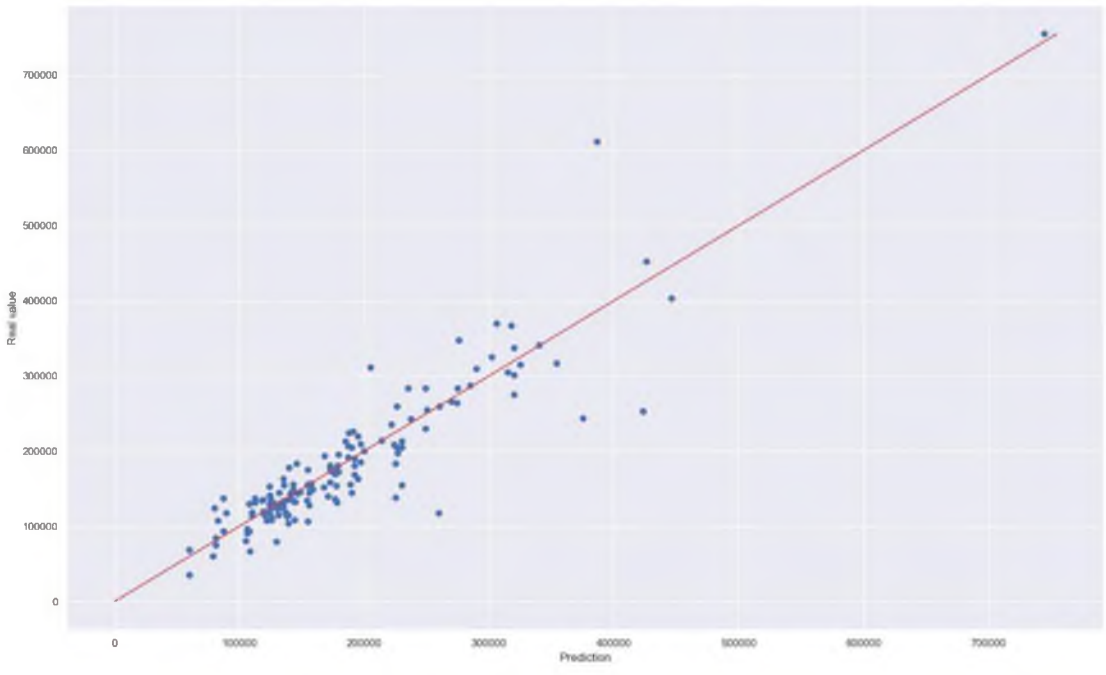

Рисунок 3.17 – Графік кінцевого прогнозування

Червона лінія показує ідеальні прогнози та якби прогноз відповідав реальному значенню, то всі точки належали б червоній лінії. Також можна побачити присутні деякі відхилення та кілька викидів, це стосується лише тих цін, які є надзвичайно високими. Останнім етапом є обчислення середньоквадратичної помилки. Обчислення даного показника ілюстровано на рисунку 3.18.

```
print('MAE:\t%.2f' % mean_absolute_error(y_test, y_pred))
print('MSLE:\t%.5f' % mean_squared_log_error(y_test, y_pred))
```

Рисунок 3.18 – Приклад обчислення середньоквадратичної помилки

Помилка в даному випадку може здаватися досить великою, цьому сприяє екстремальні відхилення при прогнозуванні, це варто враховувати.

У висновку можна сказати, що класифікатор випадкового лісу досить добре справляється з поставленим завданням прогнозування ціни нерухомості. Використовуючи досить зручні та швидкі бібліотеки мови програмування Python, можна отримати досить точний та якісний прогноз на великих об’ємах даних.

3.3 Інструкція користувача

Для вдалого використання застосунку для прогнозування цін нерухомості потрібно виконати наступні кроки:

Крок 1. Потрібно авторизуватися в організації Salesforce. Для цього потрібно мати зареєстрованого юзера, який має підключення до Python проєкт, перейти до сторінки логіну та ввести імейл та пароль. На рисунку 3.19 зображено ілюстрація логіну до Salesforce організації.

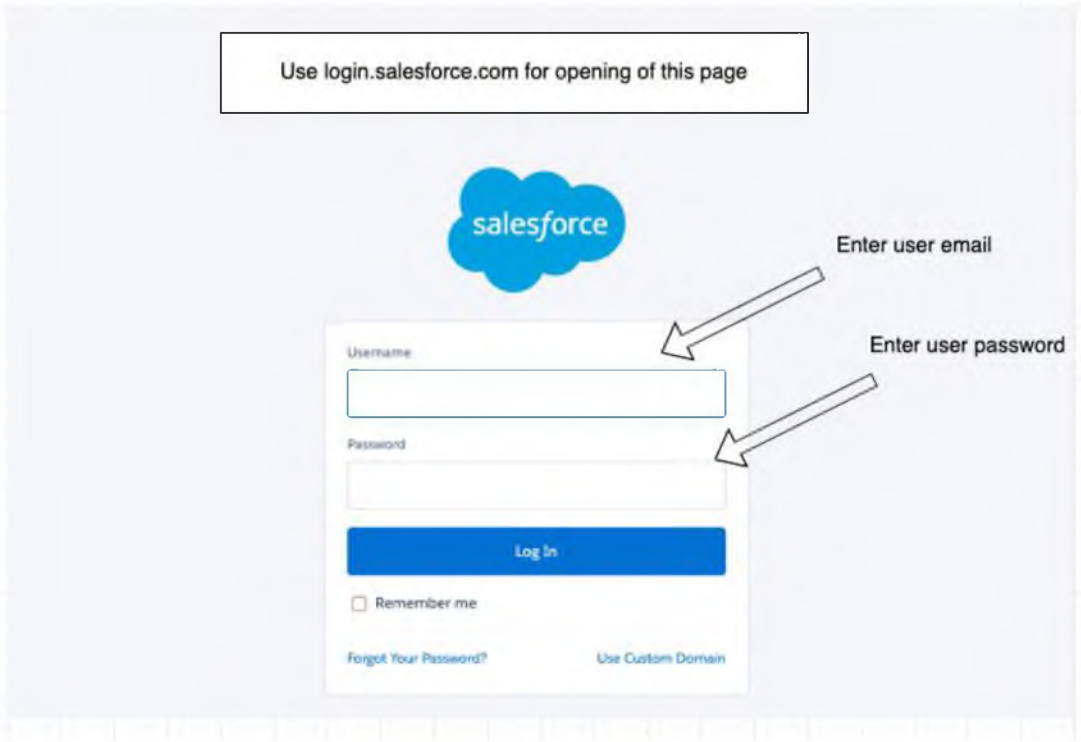

Рисунок 3.19 – Приклад ілюстрації логіну до Salesforce організації

Крок 2. При потраплянні на головну панель у верхній частині якої будуть зображені різні вкладки. Потрібно обрати вкладку, яка має назву «Deals», після кліку на цю вкладку буде показаний список доступних справ, потрібно обрати одну з них для того щоб відкрити сторінку одного запису нерухомості. На рисунку 3.20, зображено приклад початкової сторінки після входу до системи. На рисунку 3.21 зображено перехід по вкладці «Deals» та сам список доступних справ.

Рисунок 3.20 – Приклад початкового екрану після входу до системи

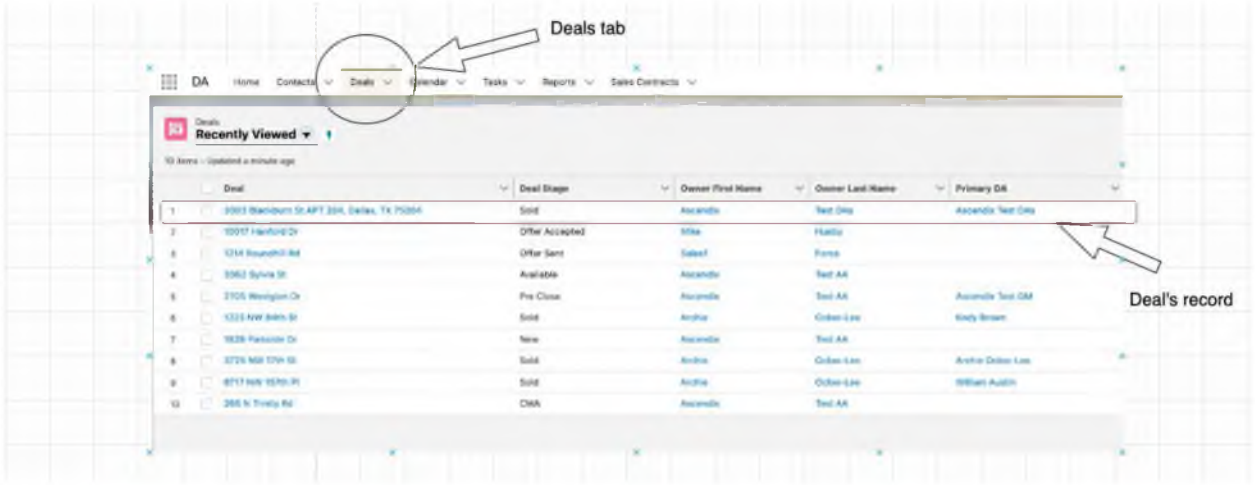

Рисунок 3.21 – Приклад екрану зі списком активних справ

Крок 3. Далі потрібно обрати будь-яку справу, для цього потрібно натиснути на будь-який надпис у колонці «Deal». На рисунку 3.22 ілюстровано приклад персональної сторінки одного запису нерухомості.

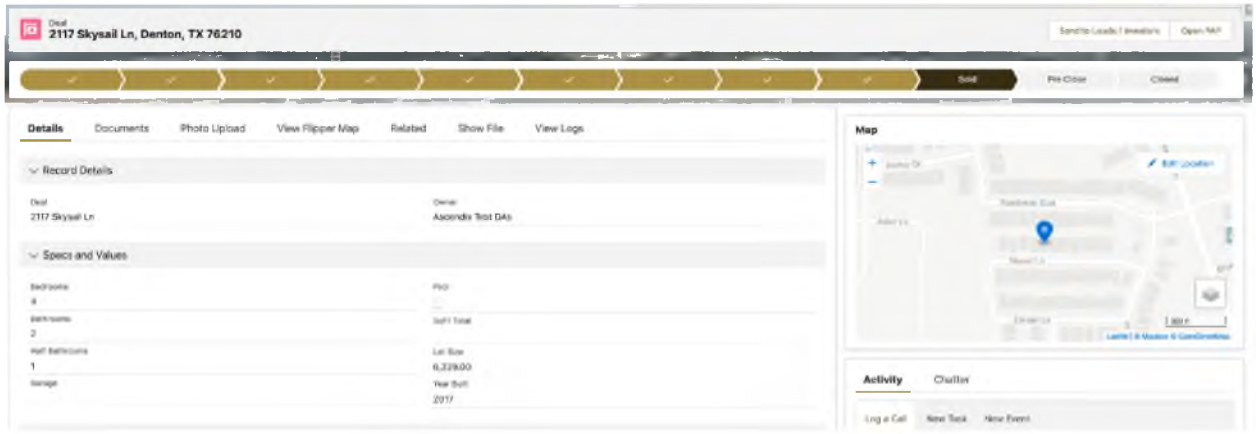

Рисунок 3.22 – Приклад персональної сторінки одного запису нерухомості

Крок 4. Перейти до панелі швидких дій та натиснути кнопку, яка має назву «House Prediction» та розташована у правій верхній частині екрану як це показано на рисунку 3.23.

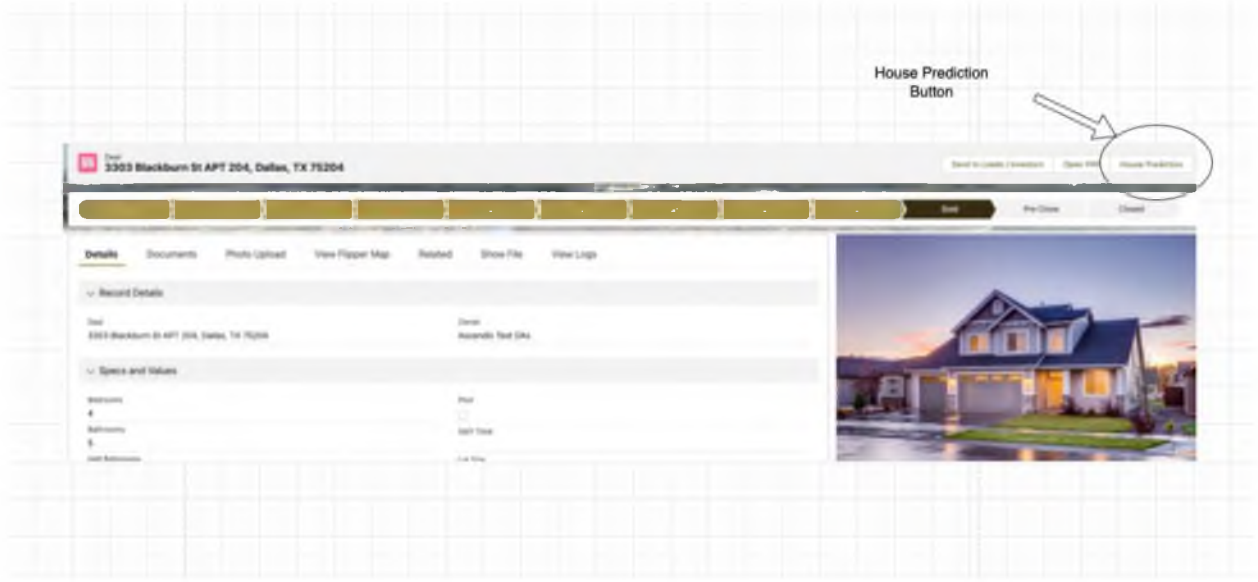

Рисунок 3.23 – Розташування потрібної кнопки

Крок 5. Натиснути на кнопку «House Prediction» та обрати потрібний місяць. Після цього повинна з’явитися таблиця з прогнозуванням по всім справам за обраний місяць. На рисунку 3.24 зображений приклад форми з опціями вибору місяця.

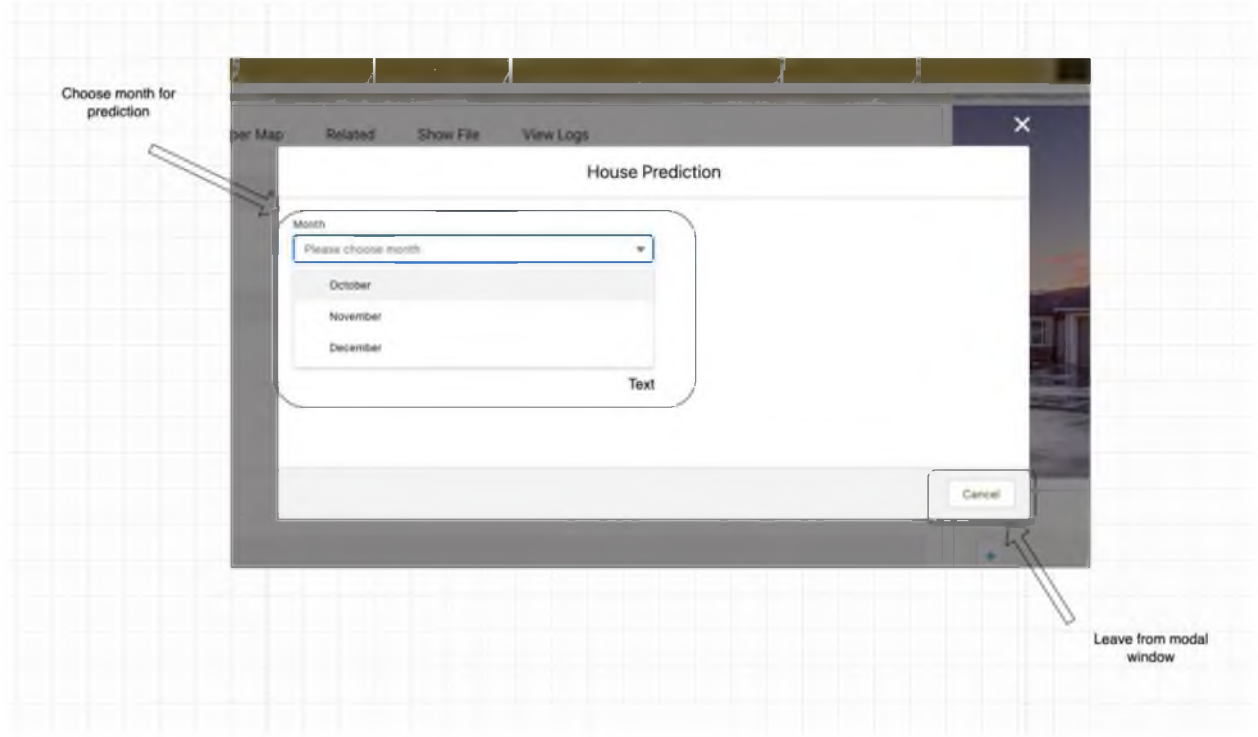

Рисунок 3.24 – Процес вибору місяця для прогнозування

Це був останній крок інструкції користувача, після закінчення перегляду кінцевої таблиці прогнозування, щоб вийти з модального вікна, потрібно натиснути «хрестик» або кнопку «Cancel».

3.4 Тестування розробленої моделі

Для тестування програмного застосунку перш за все потрібно запустити проєкт алгоритму написаний мовою програмування Python. Для цього потрібно відкрити проєкт за допомогою середовища PyCharm та запустити проєкт за допомогою кнопки «старт».

Далі потрібно підготувати відповідний набір даних для використання, для цього перейшовши у платформу Salesforce та авторизувавшись під користувачем, який зв'язаний з Python проєктом.

Після успішною авторизацію, для успішного тестування потрібно підготувати тестовий набір даних та завантажити його до платформи за допомогою Data Import Wizard. Для того щоб скористатися даним застосунком потрібно перейти в Setup і пошуковій стрічці написати Import Wizard, далі натиснути на назву та перейти до влаштованого застосунку. На рисунку 3.25 зображено місцезнаходження даного застосунку.

Рисунок 3.25 – Місцезнаходження Data Import Wizard

Після натискання відкривається вікно даного застосунку, для того, щоб перейти до імпорту потрібно натиснути на кнопку «Launch Wizard!». Імпорт тестових даних поділяється на декілька етапів:

– обрати об’єкт в який буде відбуватися імпорт даних;

– обрати опцію, яка буде означати кінцеву мету імпорту даних. В даному випадку створюється нові записи, тому потрібно обрати опцію «додати новий запис»;

– завантажити файл з тестовими даними та натиснути кнопку «далі».

Цей процес показано на рисунку 3.26.

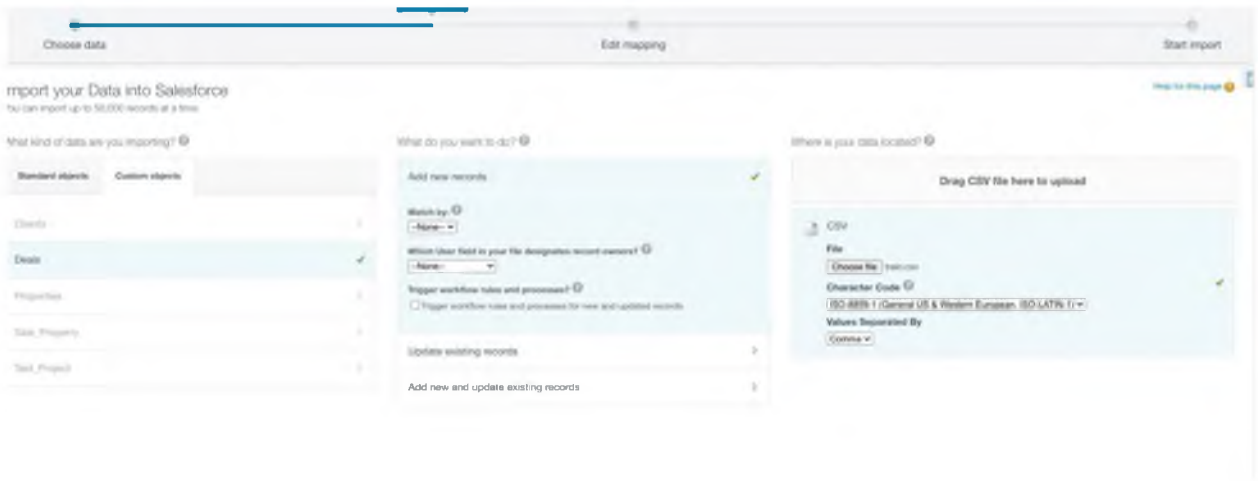

Рисунок 3.26 – Процес завантаження тестових даних в Salesforce платформу

Наступним етапом є перевірка використовуваних полів, а саме щоб назва колонки набору співпадала з полем об’єкта платформи Salesforce. Після успішної перевірки треба натиснути кнопку «далі». На рисунку 3.27 зображено процес перевірки полів вхідних даних.

На останньому екрані зображується вибір опцій, кількість полів, які пройшли валідацію та кількість полів, які не пройшли валідацію. Після перевірки потрібно натиснути кнопку «старт імпорт» та трохи почекати поки дані завантажаться до платформи Salesforce.

На рисунку 3.28 зображено приклад кінцевої перевірки всіх налаштувань.

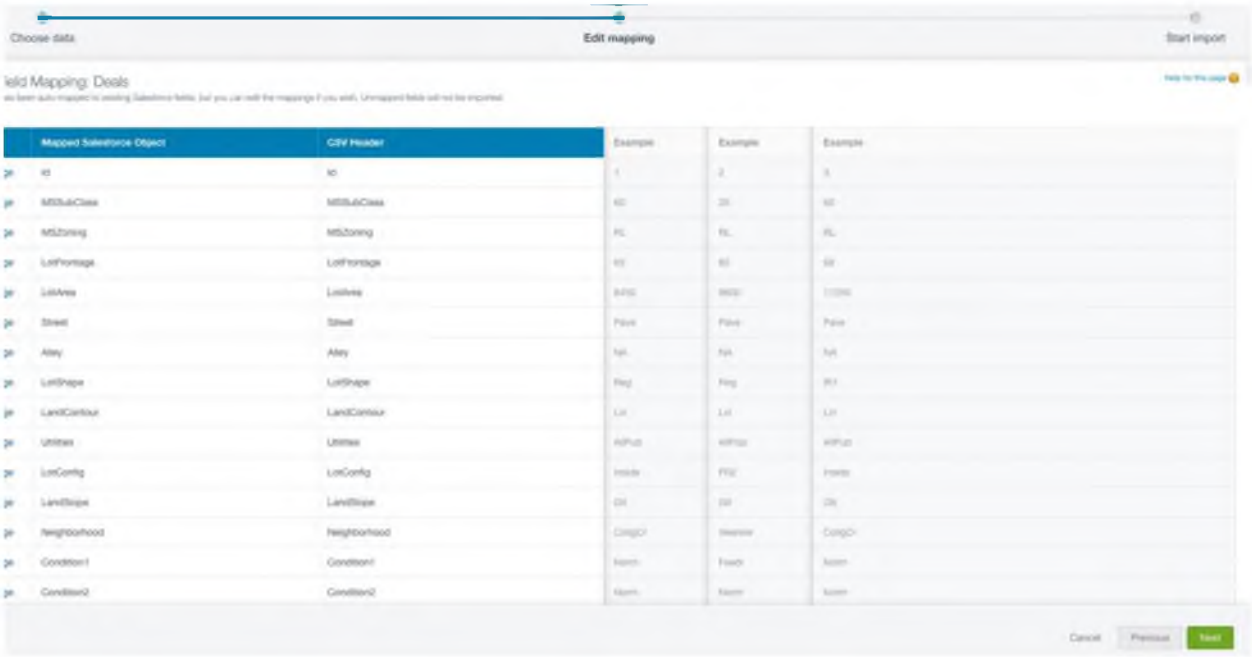

Рисунок 3.27 – Процес перевірки полів вхідних даних

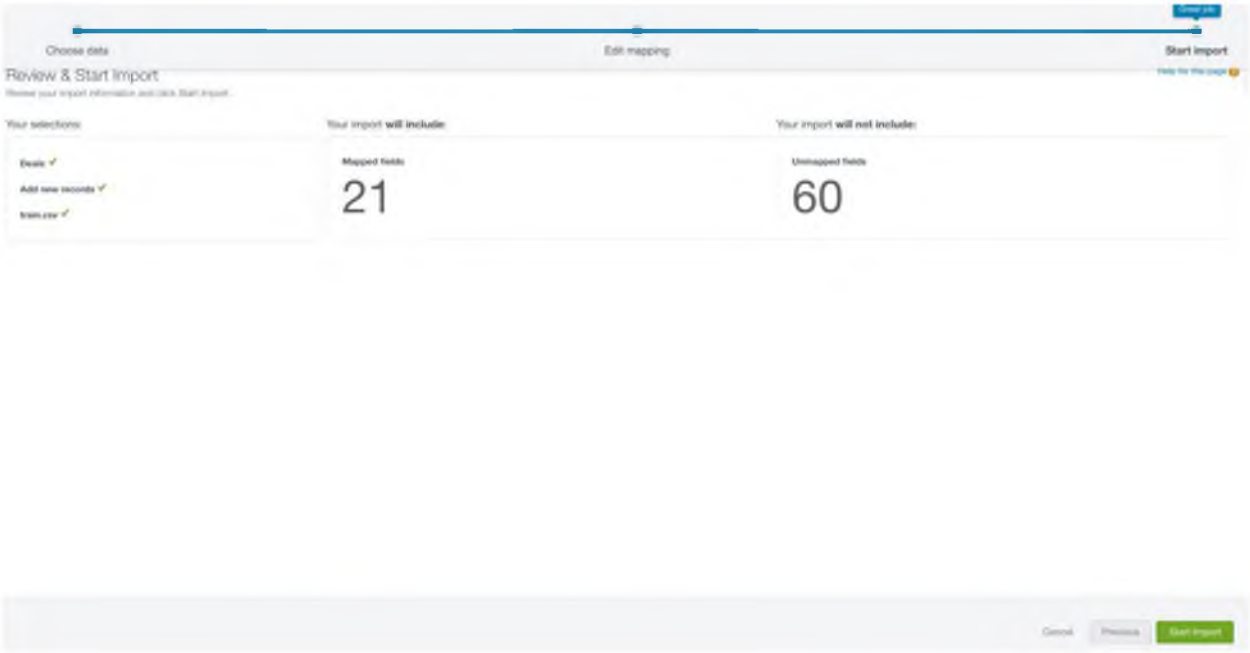

Рисунок 3.28 – Кінцева перевірка всіх налаштувань

Коли тестові дані підготовлені, можна переходити за інструкцією користувача до одного з записів, далі запустити застосунок натиснувши на кнопку «House Prediction».

Після відкриття модального вікна, потрібно обрати місяць листопад та очікувати на появлення таблиці з результатами прогнозування.

На рисунку 3.29 показаний приклад кінцевого результату прогнозування. Щоб побачити прогнозування за якийсь інший місяць, потрібно обрати в списку інший місяць, після цього таблиця автоматично перегенерується.

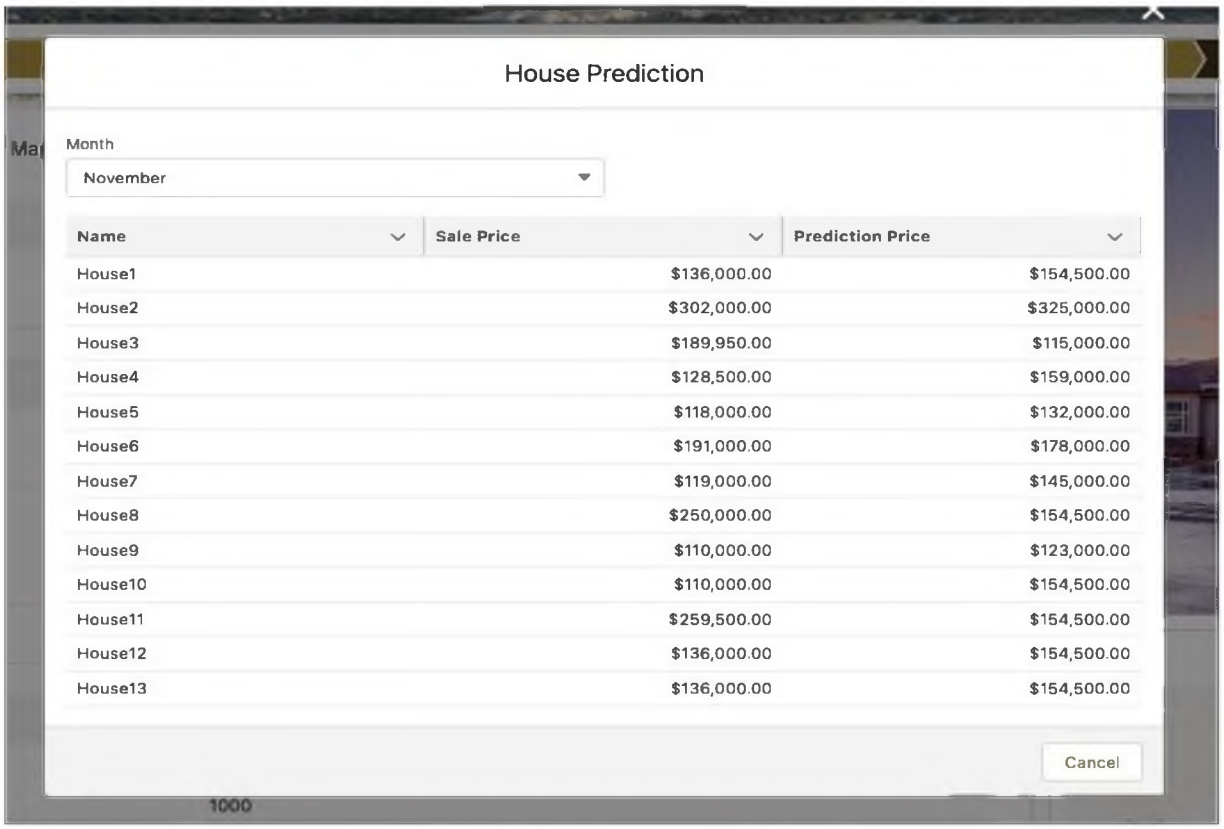

Рисунок 3.29 – Кінцевий результат прогнозування

Після завершення тестування потрібно натиснути «хрестик» або кнопку «Cancel».

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений та реалізований метод класифікації з метою прогнозування цін нерухомості.

У ході роботи були вирішені наступні завдання:

– проаналізовано усі методи класифікації, які виконують завдання прогнозування даних;

– детально проаналізовано метод класифікації даних дерево рішень;

– детально проаналізовано метод класифікації даних випадковий ліс;

– проаналізована математична модель методу класифікації даних дерево рішень;

– проаналізована математична модель методу класифікації випадковий ліс;

– розроблений універсальний алгоритм прогнозування цін нерухомості;

– виконана комп'ютерна модель алгоритму прогнозування цін нерухомості з використанням платформи Salesforce та мови програмування Python;

– розроблена документація отриманого програмного продукту;

– виконане тестування отриманого програмного продукту;

– проаналізований подальший розвиток отриманого програмного застосунку;

– виконана оптимізація отриманого програмного продукту;

– зроблено висновки, щодо виконаної роботи.

Результати дослідження опубліковано у вигляді тез доповіді під час «XXXVII Міжнародна науково-практична конференція «Modern ways of solving the latest problems in science» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

2. Работягов, А. В., Ляшенко, В. В., & Кобылин, О. А. (2016). Сегментация сложных изображений цитологических препаратов.

3. Lyashenko, V., Mohammad, A., & Kobylin, O. (2015). Experiments with Fusion of Images with Use of Wavelet Transformation in Problems of the Text Information Analysis.

4. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. Системи управління, навігації та зв'язку. Збірник наукових праць, 5(57).

5. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In Proceedings of the 9th International Conference on Information Management and Engineering (pp. 60-63). ACM.

6. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative Based Clustering of Long Multivariate Sequences with Different Lengths. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 545-548). IEEE.

7. Bodyanskiy, Y., Kobylin, I., Rashkevych, Y., Vynokurova, O., & Peleshko, D. (2018, February). Hybrid fuzzy-clustering algorithm of unevenly and asynchronously spaced time series in computer engineering. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 930-935). IEEE.

8. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. Information Technology and Management Science, 19(1), 23-28.

9. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

10. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

11. 58. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, Computers, Materials & Continua, 73 (3), pp. 6069–6084..

12. Tvoroshenko I.S., and Gorokhovatsky V.O. (2020) Effective tuning of membership function parameters in fuzzy systems based on multi-valued interval logic, *Telecommunications and Radio Engineering*, 79(2), pp. 149-163.

13. Tvoroshenko I.S., and Kramarenko O.O. (2019) Software determination of the optimal route by geoinformation technologies, *Radio Electronics Computer Science Control*, 3, pp. 131-142.

14. Gorokhovatskyi V., and Tvoroshenko I. (2020) Image Classification Based on the Kohonen Network and the Data Space Modification, *In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020)*, 2608, pp. 1013-1026.

15. Tvoroshenko I., and Tkachenko D. (2020) Mechanisms of image classification based on descriptors of local features, *Abstracts of IV International Scientific and Practical Conference «Integration of scientific bases into practice» (October 12-16, 2020). Stockholm, Sweden*, pp. 443-448.

16. Творошенко, І. С. (2018). Особливості застосування сучасних принципів штучного інтелекту до розробки ефективних механізмів моделювання складних систем. *Science and Technology of the Present Time: Priority Development Directions of Ukraine and Poland*, 118-121.

17. Gorokhovatskyi, V., Rusakova, N., and Tvoroshenko, I. (2020) The application of image analysis methods and predicate logic in applied problems of magnetic monitoring, *Telecommunications and Radio Engineering*, 79(20), pp. 1801-1811.

18. Гороховатський В.О., Творошенко І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

19. Tvoroshenko Irina, Ahmad M. Ayaz, Mustafa Syed Khalid, Lyashenko Vyacheslav, and Alharbi Adel R. (2020) Modification of Models Intensive Development Ontologies by Fuzzy Logic, *International Journal of Emerging Trends in Engineering Research*, 8(3), pp. 939-944.

20. Tvoroshenko I.S., and Gorokhovatsky V.O. (2020) Effective tuning of membership function parameters in fuzzy systems based on multi-valued interval logic, *Telecommunications and Radio Engineering*, 79(2), pp. 149-163.

21. Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28.

22. Lu, H., & Ma, X. (2020). Hybrid decision tree-based machine learning models for short-term water quality prediction. Chemosphere, 249, 126169.

23. Bui, X. N., Nguyen, H., Choi, Y., Nguyen-Thoi, T., Zhou, J., & Dou, J. (2020). Prediction of slope failure in open-pit mines using a novel hybrid artificial intelligence model based on decision tree and evolution algorithm. Scientific reports, 10(1), 1-17.

24. Jia, S., & Pang, Y. (2018). Teaching Quality Evaluation and Scheme Prediction Model Based on Improved Decision Tree Algorithm. International Journal of Emerging Technologies in Learning, 13(10).

25. Agrawal, L., & Adane, D. (2021). Improved decision tree model for prediction in equity market using heterogeneous data. IETE Journal of Research, 1-10.

26. Narayanan, H., Sokolov, M., Butté, A., & Morbidelli, M. (2019). Decision Tree-PLS (DT-PLS) algorithm for the development of process: Specific local prediction models. Biotechnology progress, 35(4), e2818.

27. Shaikhina, T., Lowe, D., Daga, S., Briggs, D., Higgins, R., & Khovanova, N. (2019). Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. Biomedical Signal Processing and Control, 52, 456-462.

28. Jackins, V., Vimal, S., Kaliappan, M., & Lee, M. Y. (2021). AI-based smart prediction of clinical disease using random forest classifier and Naive Bayes. The Journal of Supercomputing, 77(5), 5198-5219.

29. Chowdhury, A. R., Chatterjee, T., & Banerjee, S. (2019). A Random Forest classifier-based approach in the detection of abnormalities in the retina. Medical & biological engineering & computing, 57(1), 193-203.

30. Munasinghe, M. I. N. P. (2018, June). Facial expression recognition using facial landmarks and random forest classifier. In 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS) (pp. 423-427). IEEE.

31. Tariq, A., Yan, J., Gagnon, A. S., Riaz Khan, M., & Mumtaz, F. (2022). Mapping of cropland, cropping patterns and crop types by combining optical remote sensing images with decision tree classifier and random forest. Geo-spatial Information Science, 1-19.

32. Lv, Z., Jin, S., Ding, H., & Zou, Q. (2019). A random forest sub-Golgi protein classifier optimized via dipeptide and amino acid composition features. Frontiers in bioengineering and biotechnology, 7, 215.

33. Alzubi, J., Nayyar, A., & Kumar, A. (2018, November). Machine learning from theory to algorithms: an overview. In Journal of physics: conference series (Vol. 1142, No. 1, p. 012012). IOP Publishing.

34. Bisong, E. (2019). Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners. Apress.

35. Fatima, Z. (2021). Salesforce control systems: a review of studies. International Journal of Business Excellence, 23(2), 188-225.

36. Salesforce, C. P. Q. (2020). Salesforce. Cloud CRM Solutions [online].

37. Xiao, B., & Xiao, W. (2020). Optimal salesforce compensation with supply–demand mismatch costs. Production and Operations Management, 29(1), 62-71.

38. Sneh, M. S. (2018). Analysis of Business Strategies of Salesforce. com Inc. Sneha, MS & Krishna Prasad, K.(2018). Analysis of Business Strategies of Salesforce. com Inc. International Journal of Case Studies in Business, IT and Education (IJCSBE), 2(1), 37-44.

39. Zhao, Y., Nasrullah, Z., & Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. arXiv preprint arXiv:1901.01588.

40. Іващенко О. (2022) Огляд методів класифікації для прогнозування цін нерухомості, Abstracts of XXXVII International Scientific and Practical Conference «Modern ways of solving the latest problems in science» (September 20 – 23, 2022). Varna, Bulgaria, pp. 423-425.