Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

(повна назва факультету)

Кафедра програмної інженерії

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

(назва освітнього ступеня)

на тему: Розробка системи для підтримки прийняття рішень щодо надання кредитів

Виконав: студент IV курсу, групи СП-41

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Бойко Д.А.

(підпис) (прізвище та ініціали)

Керівник Пастух О.А.

(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю.М.

(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М.Р.

(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

Тернопіль

2024

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

(повна назва факультету)

Кафедра програмної інженерії

(повна назва кафедри)

ЗАТВЕРДЖУЮ

Завідувач кафедри

Петрик М.Р.

(підпис) (прізвище та ініціали)

« » _____ 2024р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*

(назва освітнього ступеня)

за спеціальністю *121 Інженерія програмного забезпечення*

(шифр і назва спеціальності)

студенту *Бойко Данило Андрійович*

(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка системи для підтримки прийняття рішень щодо надання кредитів*

Керівник роботи *Пастух Олег Анатолійович, д.т.н., професор*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2024 року № _____.

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи *Числовий двовимірний масив даних – результати кредитних історій клієнтів банку; програмні засоби: Scikit-learn, Colloction, NumPy, Pandas, Matplotlib технології програмування Python*

4. Зміст роботи (перелік питань, які потрібно розробити) *Огляд, аналіз предметної області – кредитний скоринг. Огляд та аналіз програмних засобів: фреймворків, бібліотек, функцій для програм. автомат. кредитного скорингу. Проектування програмної системи для аналізу надання кредитів та прогнозу їх повернення клієнтами банку. Практичне конструювання програмної системи для аналізу та прогнозу у сфері кредитного скорингу.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів) *Проаналізов. актуальність теми, мету роботи, об'єкт роботи, предмет роботи, завдання роботи. Представлено результати проектування кредитно-скорингової програмної системи, а саме її архітектури на абстрактному рівні. Результати практично-прикладного конструювання кредитно-скорингової програмної системи. Представлено експериментальні результати роботи кредитно-скорингової програмної системи у вигляді коду та даних. Виконано повний метрологічний аналіз роботи даної кредитно-скорингової програмної системи: обчислені звичайна точність класифікації, метрика – точність f1 та метрика точності – roc_auc, які у сукупності вичерпно характеризують якість роботи розробленої кредитно-скорингової програмної системи. Забезпечено можливість зберігати розроблені програмні компоненти для повторного їх використання. Висновки.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Лазарюк В.В., к.т.н., доц., доц. каф. МТ		
Нормоконтроль	Стоянов Ю.М., к.т.н.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Вибір та затвердження теми		виконано
2.	Аналіз предметної області та технічного завдання		виконано
3.	Аналіз особливостей проектування та конструювання ПС		виконано
4.	Проектування програмної системи		виконано
5.	Практичне конструювання програмної системи		виконано
6.	Оформлення записки кваліфікаційної роботи		виконано
7.	Безпека життєдіяльності, основи охорони праці		виконано
8.	Підготовка презентації і доповіді		виконано
9.	Нормоконтроль		виконано
10.	Перевірка на плагіат		виконано
11.	Попередній захист кваліфікаційної роботи		виконано
12.	Захист кваліфікаційної роботи		виконано

Студент _____ *Бойко Д.А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Пастух О.А.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

У даній кваліфікаційній роботі на здобуття освітнього ступеня бакалавр виконано детальний і конструктивний огляд і аналіз сфери розробки програмного забезпечення для такої галузі фінансової інженерії, як кредитний скоринг. Потреба у розробці такого роду програмного забезпечення обумовлено вищою точністю у прийнятті рішень щодо надання чи не надання кредитних послуг клієнтам банку із мінімізацією ризику його не повернення клієнтом. Окрім того, програмне забезпечення зменшує витрати банку на проведення такого роду процедур. У роботі спочатку спроектовано кридитно-скорингову програмну систему, а потім її практично сконструйовано. В основу прийняття рішення використано алгоритм машинного навчання – дерево прийняття рішень. Виконано різностороннє дослідження точності роботи такого програмного забезпечення на основі звичайної метрики точності, f1 метрики точності та метрики точності у вигляді roc_auc. Для підвищення якості роботи даного програмного забезпечення проведено у вигляді програмної реалізації процедури гіпероптимізації алгоритму машинного навчання – дерева прийняття рішень на основі різних оптимізаційних програмних засобів. Програмно реалізовано можливість зберігати для повторного використання певних програмних компонент, які були навчені у дані програмній системі на даних, тобто реалізовано технологію повторновикористовуваних компонентів. Таким чином, розробка дає змогу зберігати класифікатора і імпортувати його зовні для можливості використання сторонніми програмами. Також у роботі приділена велика увага процедурі попередньої обробки даних для суттєвого підвищення якості роботи програмної розробки, що підтверджує її високу прицизійність і відповідно зниження ризиків для банку.

ANNOTATION

In this qualification work for obtaining a bachelor's degree, a detailed and constructive review and analysis of the field of software development for such a field of financial engineering as credit scoring is performed. The need for the development of this kind of software is due to higher accuracy in decision-making regarding the provision or non-provision of credit services to bank clients with the minimization of the risk of non-return by the client. In addition, the software reduces the bank's costs for conducting such procedures. The work first designed a credit-scoring software system, and then practically constructed it. A machine learning algorithm - a decision tree - was used as the basis for decision-making. A comprehensive study of the accuracy of the operation of such software was performed based on the usual accuracy metric, the f1 accuracy metric, and the accuracy metric in the form of roc_auc. To improve the quality of this software, a hyper-optimization procedure of the machine learning algorithm - a decision-making tree based on various optimization software tools was carried out in the form of a software implementation. Software has implemented the ability to save for reuse certain software components that were taught to the data software system on the data, that is, the technology of reusable components has been implemented. Thus, the development makes it possible to save the classifier and import it externally for use by third-party programs. The work also pays great attention to the data preprocessing procedure to significantly improve the quality of software development work, which confirms its high precision and, accordingly, the reduction of risks for the bank.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмний засіб

CART – дерева підтримки прийняття рішень для завдань класифікації та регресії

C4.5 – алгоритм генерування дерев підтримки прийняття рішень для завдань класифікації

ID3 – алгоритм генерування дерев підтримки прийняття рішень

Дерев. Підрим. Прийнят. Рішень – дерева підтримки прийняття рішень

CHAID – технологія дерева підтримки прийняття рішень на основі тестування значущості

ПС – програмна система

Itrule – дерево підтримки прийняття рішень, яке реалізує індукцію імовірнісних наборів правил

CN2 - дерево прийняття рішень для задач класифікації та регресії

Фін. інженерія – фінансова інженерія

Фін. інструменти – фінансові інструменти

ВСТУП

Однією із високомаржинальних сфер застосування нових програмних розробок є фін. інженерія. Розроблювальне різного роду програмне забезпечення для цієї сфери – це є фін. інструменти для ринку фінансів, а саме, його гравців: банків, трейдерів, тощо. Від такого програмного забезпечення вимагається висока точність для зниження ризиків втрати фін. гравцями та високого рівня реактивності в прийнятті рішень.

Фактично програмне забезпечення для фін. гравців представляє собою зброю на полі фінансового ринку. Чим точніше, реактивніше, стабільніше у роботі програмне забезпечення в руках фінансового гравця тим він більш успішний на фінансовому ринку.

Завжди, особливо важливим є питання зменшення ризиків втрати фін. ресурсів, тому банківські гравці гостро потребують програмного забезпечення для вирішення своїх потреб у сфері кредитування, тобто, їм дуже потрібні такого роду фін. інструметни.

Тому у цій роботі розглядається проектування та розробка програмного забезпечення і на його основі системи – фінансового інструменту для вирішення прикладних завдань кредитного скорингу. На основі ідеї щодо вирішення цієї прикладної проблеми фін. інженерії у кваліфікаційній роботі сформульовано мету та завдання.

РОЗДІЛ 1. ОГЛЯД СКОРИНГОВОГО БІЗНЕС ДОМЕНУ. АНАЛІЗ СКОРИНГОВОГО БІЗНЕС ДОМЕНУ

1.1 Огляд скорингового домену. Аналіз скорингового домену

Домен скорингових систем на сьогодні є одним із актуальних напрямів у багатьох сферах життєдіяльності людини. Проте, серед цих сфер є одна в якій надзвичайно часто використовуються скорингові підходи щодо вирішення широкого пула завдань у предметній області. До цієї сфери – домену відноситься скоринг у сфері фінансів, зокрема, у галузі інженерії фінансів (фінансовій інженерії). Особливо актуальність використання скорингових систем у галузі фінансів підсилилася з появою інтелектуалізованих технологій зі сфери штучного розуму – інтелекту. По причині того, що інтелектуалізовані технології на яких розробляються інтелектуалізовані системи підтримки прийняття рішень дають змогу оперативно, тобто швидко у часі приймати рішення повязані із фінансовими діями, тобто операціями. Серед сфери фінансів виділяється сфера фінансового кредитування клієнтських осіб наприклад клієнтів кредитних спілок, клієнтів банків чи довільних інших фінансових установ. Загалом, відійшовши в сторону хочиться відзначити корисність інформації для галузі скорингу, яка витікає із алгоритмів і підходів з предметних областей, що представлені в роботах [1-11]. Побудова, конструювання скорингових систем на основі інтелектуалізованих технологіях приводить до того, що здешевлюються витрати на проведення операцій кредитування, а особливо важливо – це зменшуються ризики при вадаванні кредитів банками на основі прийнятих рішень інтелектуалізованою скоринговою фінансовою системою.

Інтелектуалізовані скорингові фінансові системи в основі своєї програмної роботи використовують найрізноманітніші алгоритми, які використовуються в інтелектуалізованій переробці даних. Так наприклад вони використовують алгоритми навчання машин у вигляді рандомізованого лісу, можуть використовувати

довільні інші алгоритми логічної парадигми, наприклад, бегінг чи алгоритм підсиленг дерев, який ще називають бустинг дерев підтримк. прийнятт. рішень. Це дуже сильний пул алгоритмів логічної парадигми, оскільки вони дають високу точність прийнятих рішень, а відповідно суттєво знижують імовірність не повернення фінансових коштів. Але поряд з цим вони часто використовують значні заьрати процесорного часу чи памяті компютерів використання яких приводить до збільшення затрат фінансів. Тому прийняти рішення щодо вибору із пула алгоритмів конкретного алгоритму є не тривіальною задачею. Існує ціла батарея, пул інших алгоритмів – це алгоритми на основі нейромережевих технологій, якщо коротко то це нейронні мережі, що будуються на різних архітектурах. Сюди можна віднести багатоопераційний чи багатошаровий перцептрон, різновиди нейронних мереж із зворотнім звязком – рекурентні неронн. мережі та інші. Цей набір алгоритмів має також високу точність, але потребує свого апартного забезпечення і при цьому – графічних процесорів, а часто і тензорних процесорів. І хоч вони дають високу точність але затрати ними обчислювальних ресурсів є відчутно великими. Тому і у разі їх використання для фінансових інституцій будуть зростати витрати на впровадження ними скорингових фінансових систем – систем для підтримк. прийнятт. рішень для наданання кредитів. Тому завжди є актуальною задачею використання різних інших алгоритмів, які би давали високу точність і при цьому не давали великих затрат на їх використання на практиці. До наступних високоточних алгоритмів відносить ся алгоритм методу опорних векторів – це є високоточний алгоритмічний метод бінарної класифікації давати чи не давати кредит клієнтові, але і він потребує значимо великих затрат на його реалізацію на обчислювальних машинах. Можна багато перераховувати тут інших різних алгоритмів але це потребує використання багато сторінок для написання, тому перелік інших алгоритмів тут у роботі не будемо, але наведемо відразу той вид алгоритмів для навчання машин який використовуватиметься у цій роботі – алгоритм дерев підтримк. прийнятт. рішень щодо надавати чи не надавати клієнтам фінансовий кредит. Зауважимо, що розглядаються різновиди дерев підтримк. прийнятт. рішень,

наприклад, бінарні дерева та не бінарні дерева, які можуть виконувати нечітку чи мяку класифікацію та чітку класифікацію. Під нечіткою класифікацією розуміється класифікація яка повертає розподіл імовірностей заданий на мітках класів, тобто мітки класів представляють собою випадкову величину – бінарну рандомізовану величину, надавати кредит чи не надавати кредит.

Фінансові скорингові системи поділяються не лише по виду конкретно використовуваного алгоритму але загалом по архітектурі. Тому у у другому підрозділі першого розділу розглядається спочатку узагальнена архітектура інтелектуалізованої фінансової скорингової системи, а потім у подальшому буде перехід до конкретної архітектури, яка пропонується у роботі та прикладного конструювання скорингової системи на основі запропонованої конкретної архітектури.

1.2 Узагальнене архітектурне рішення кредитно-скорингової програмної системи

Перед тим, як навести узагальнену архітектуру інтелектуалізованої фінансової скорингової системи слушно розглянути базові блоки – елементи системи. Так, наприклад, у програмному конвеєрі узагальненої архітектури першим блоком є уведення даних клієнта, який претендує на одержання кредиту, він представляється негомогенними даними, тобто даними змінні мають різну інтерпритацію, а з еврістичних даних відомо, що логічна парадигма працює в середньому, як найкраще. В принципі з цієї причини і використовуються в основі роботи дерева підтримки прийняття рішень. Чим більше найрізноманітніших даних про клієнта є у системі тим збільшується імовірність правильго рішення системи щодо надавання кредиту

для клієнта. Потім йде блок апріорної переробки даних, загалом, це інженерія ознак, на основі тих чи інших підходів, тобто наприклад тих же дерев підтримки прийняття рішень, бустингових алгоритмів, методу опорних вектор., тощо, або на статистичних тестах Фішера або Пірсона, або інших. Тобто виконується селекція ознак, а крім того і створення, синтез нових ознак, наприклад на основі поліноміальної апроксимації. Далі виконуються інші апріорні операцій переробки – це відновлення пропущених даних, стандартизація даних, витирання викидів у даних максимальних значень або аномально мінімальних значень. Розгляньмо наведений рисунок узагальненої архітектури інтелектуалізованої фінансової скорингової системи.

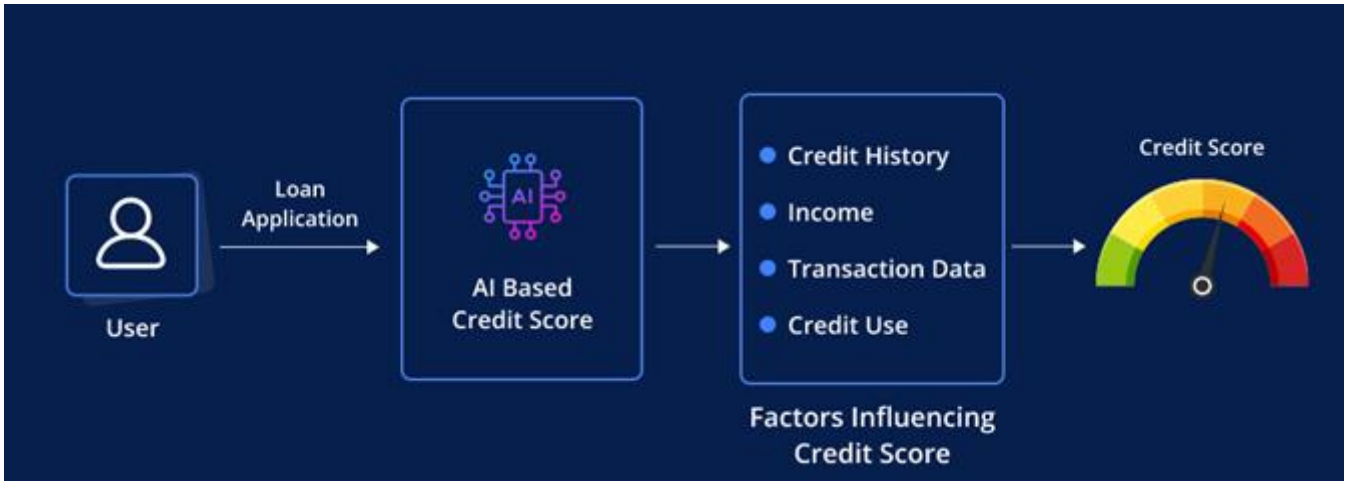

Рис.1.1 – Узагальнене архітектурне рішення кредитно-скорингової прогр. системи

З узагальненої архітектури інтелектуалізованої фінансової кредитно-скорингової системи видно, що вихідні рішення можуть мати велику цінність коли система повертає не чітко класифіковану відповідь надати кредит клієнтові чи не давати фінансовий кредит клієнтові, а надає нечіткий класифікаційне рішення із імовірнісною вагою для класової мітки бінарної класифікації. По-перше, можна задавати системі керування пороговий рівеь прийняття нею рішення, або у не стандартних моментах може бути підключена людина – банківський працівник до остаточного прийняття рішення щодо кредитування. Цікавим моментом може бути,

коли система примає кожне з бінарних рішень з імовірнісною вагою пятдеся на пятдесіт відсотків. У такому разі залучення працівника фінансової інституції є реально і однозначно потрібним. Перейдемо до конкретнішого опису інтелектуалізованих фінансоких скорингвих систем для надання кредитів у наступному підрозділі даного розділу.

1.3 Кредитно-скорингові програмні системи на основі дерев підтримки прийняття рішень

Як було описано вище у цій інтелектуалізованій фінансовій кредитно-скоринговій системі прийняття рішень шодо кредитування використовуватимуться дерева для підтримки прийняття рішень. Але це є не простою задачею вибір конкретного алгоритму на основі дерев підтримки рішень, оскільки їх існує велике сімейство. До цього сімейства відносяться наступні редева підтримки рішень:

CART – дер. підтримки рішень для завдань класифікації, регресії;

C4.5 – алгоритм генерування дер. підтримки рішень для завдань класифікац.;

ID3 – алгоритм генерування дер. підтримки рішень;

CHAID – технологія дер. підтримки рішень на основі тестування значущості;

Itrule – дер. підтримки рішень, яке реалізує індукцію імовірнісних множин правил;

CN2 – дер. прийняття рішень для задач класифікації, регресії.

Тобто існує велике сімейство дерев підтримки прийняття рішень для найрізноманітних завдань важливо тут буде навести конкретні різновиди таких дерев для найрізноманітніших завдань, наприклад для завдань класифікації, завдань регресії і навіть для завдань кластнризації, що демонструється наступними

рисунками, які демонструють різновиди дерев і візуалізують ряд їх суттєвих гіперпараметрів за допомогою яких ними можна керувати.

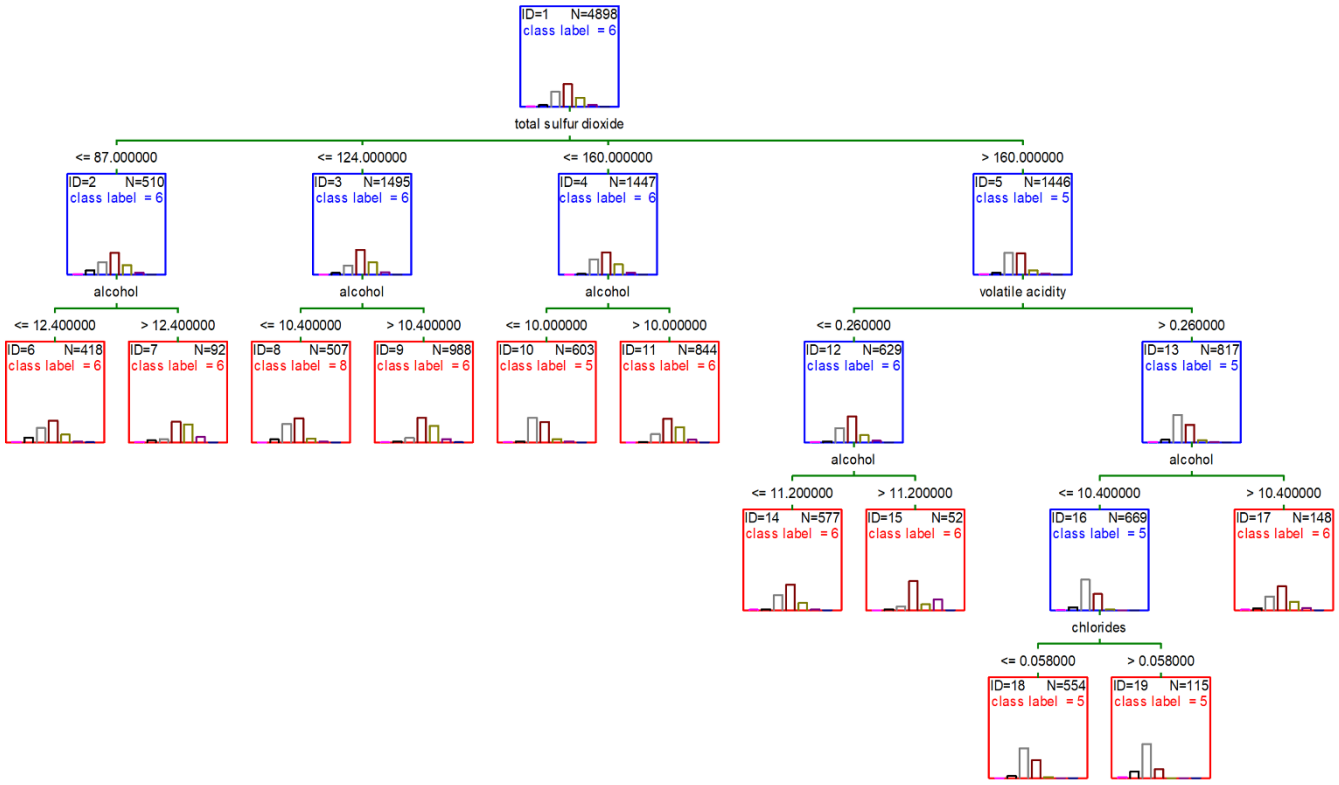

Рис.1.2 – Дерево підтримки прийняття рішень для класифікаційних завдань

Як видно з вище наведеного дерева воно може мати багато ярусів та багато нодів, тобто вершин. У середині кожної ноди є розподіл цільової змінної, яка має свій розподіл імовірностей на основі якого виводиться нечітка класифікація. Тобто таке дерево потенційно може повернути мітки класів, як значення випадкової величини і відповідні їм імовірності. Тому можна бачити розширену – диференціальну картину для підтримки прийняття рішень. У випадку чіткої класифікації використовується правило максимуму функції правдоподібності і для максимального значення імовірності повертається мітка класу і цим самим машина на основі методу прогнозування можк повернути цю мітку класу. Також на ребрах з візуалізації видно характеристику по якій відбувається розщеплення дерева і крім

того візуалізовано значення медіани діапазону варіювання значень даної характеристики, що утворює нейрівність і розбиває інтервал варіювання фіксованого атрибуту. У листках дерева видно кількість елементів датасету, які потрапили у фіксований листок дерева підтримки прийняття рішень. Також такого виду алгоритми використовуються і для регресійних завдань. Для цього можна розглянути малюнок такий.

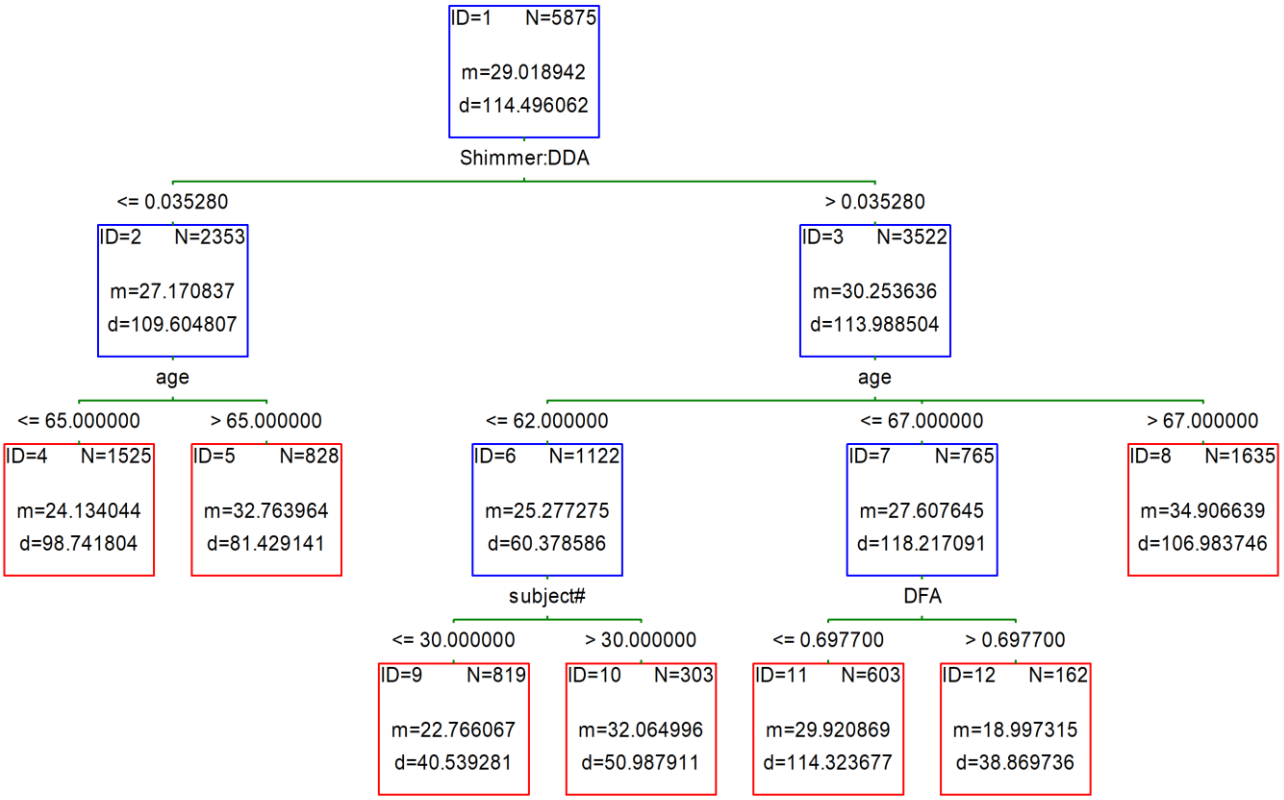

Рис.1.3 – Дерево підтримки прийняття рішень для регресійних завдань

З цього малюнку видно приклад застосування дерев для регресії, цікаво, що прийняте рішення вираховується на основі статистичних характеристик випадкої величини, якою є цільова змінна, а саме на основі оцінювання математичного сподівання. Ця статистична оцінка є вихідними даними такого виду деревовидного алгоритму. Також важливою величиною є оцінювання у листака такого регресійного дерevоводного алгоритму статистичної оцінки розсіювання значень цільової змінної

у листку, наприклад дисперсії чи її кореня кватратного середньоквадратичного відхилення. Така стандартна девіація може використовуватися для оцінювання величини похибки апроксимації у листку регресійного деревовидного алгоритму.

Окрім деревовидних алгоритмів, які наведені до цих пір окремо виділяють їх частинний вид – це бінарні дерева для рішень, візуалізацію можна бачити нище.

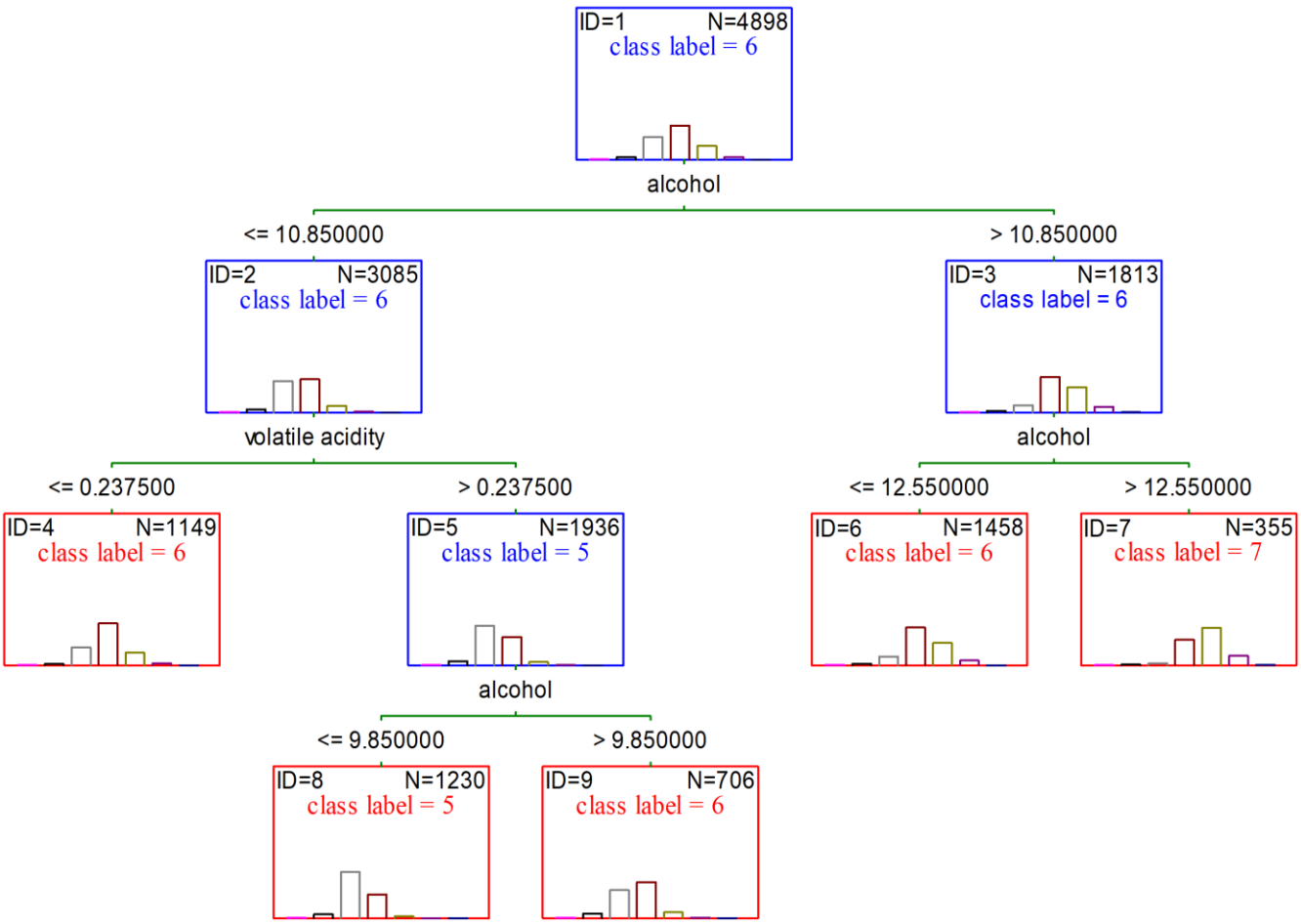

Рис.1.4 – Дихотомічне дерево підт. прийняття рішень для класифікаційних завдань

Це є, зокрема, бінарний деревовидний алгоритм для завдань класифікації у ньому, як і зазначалося є розщеплення у кожній ноді, але вже на дві гілки, оскільки воно є бінарним. Також у ньому розраховується значення в діапазоні варіації змінної для одержання нерівності у розщеплені на основі якої дані у ноді буду переміщатися

по правій гілці або по лівій гілці. Також у бінаризованих деревовидних алгоритмах у листак є розподів імовірностей для цільової рандомізованої змінної і знову ж тако розглядається нечітка та чітка класифікації, які дають змогу чітку визначити клас прийнятого рішення, наприклад дати кредит клієнтові чи інше чітке рішення не дати кредит клієнтові. А також на основі розгподілу імовірності визначити ранжований список значень цільової змінної.

Бінаризовані деревовидні алгоритми використовуються для регресійниз задач, їх вид можна побачити нище наведеним. І у них також багато з не бінаризованими деревовидними алгоритмами підтримки рішень. У кожній ноді визначається змінна – атрибут по якому буде розщеплюватися деревоводне рішення.

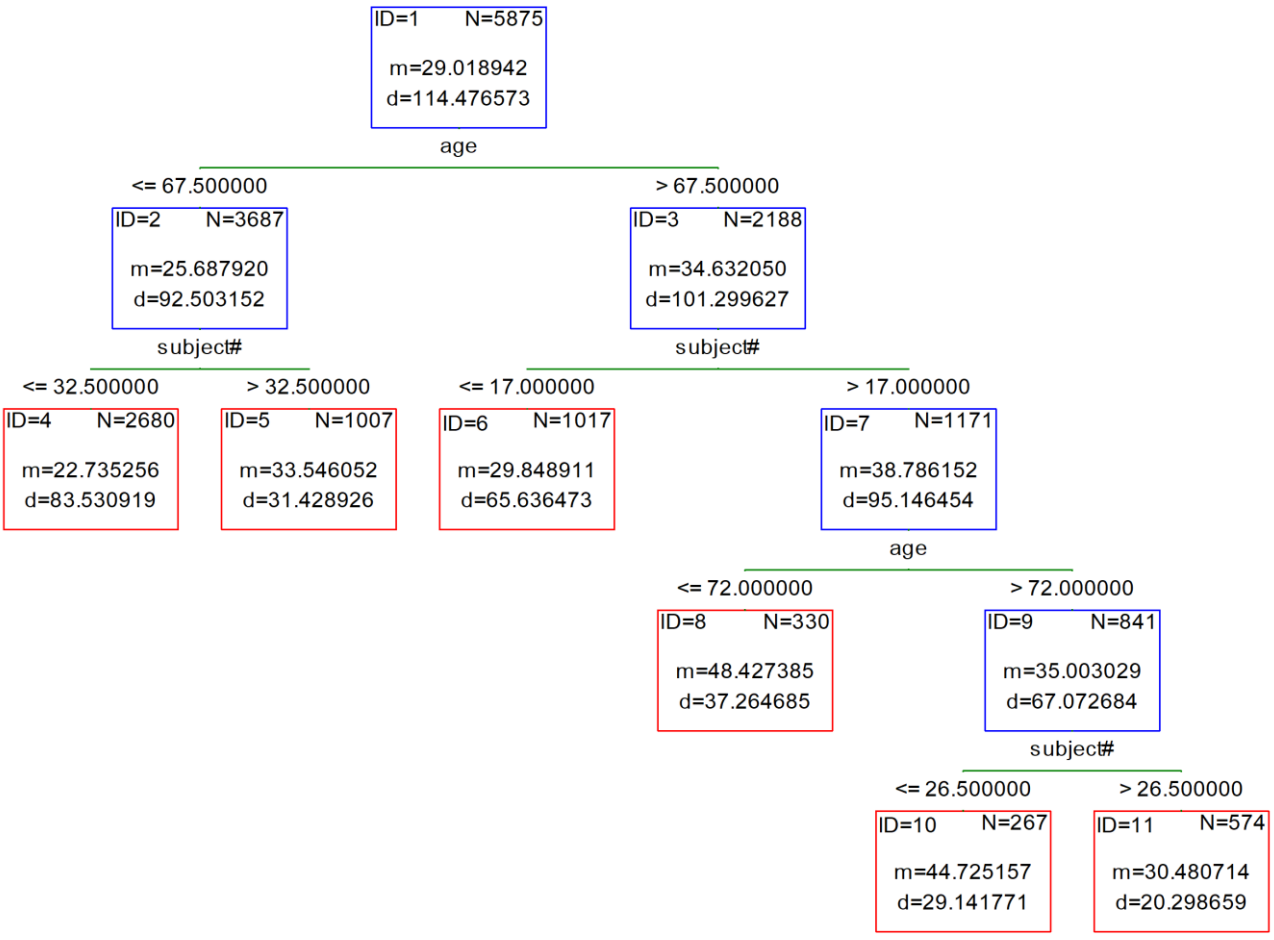

Рис.1.5 – Дихотомічне дерево підт. прийняття рішень для регресійних завдань

Слід зауважити, що атрибути можуть у нодах такого роду, як і іншого виду дерев повторюватися причому навіть багато разів, якщо даний атрибут є суттєво значимим для підтримки прийняття рішень фінансовою кредитно-скоринговою системою.

У листках – терміналах також візуалізується обєм вибірки даних, які дійшли до нього на основі цієї сукупності також обчислюються статистичні оцінки – математичного сподівання стандартної девіації – середньоквадратичного відхилення. Математичне сподівання є неперервним значенням системи, а середньоквадратичне відхилення характеризує абсолютно похибку прийнятого рішення із неперервним значенням.

На основі вище описаного та проганалізованого можна сформулювати мету роботи та завдання цієї бакалаврської роботи.

Мета кваліф. роботи: Спроектув. та сконструюв. програм. систему для підтримки прийнят. рішень щодо надання чи не надання банком кредиту для своїх клієнтів з метою зменшення ризиків їх не повернення.

Завдання кваліфікаційної роботи:

1. Спроектув. та сконструюв. програм. систему для підтримки прийнят. рішень банківським установам щодо надання чи не надання кредитів клієнтам банку.

2. Спроектув. та сконструюв. процеси гіпероптимізації прогр. забезпечення для покращення його метрологічних характеристик, накшталт, accuracy, f1-score, roc_auc-score, а також робастності функціонцвання.

РОЗДІЛ 2. ПРИКЛАДНЕ ПРОЕКТУВАННЯ КРЕДИТНО-СКОРИНГОВОЇ ПРОГРАМНОЇ СИСТЕМИ

В основі проектування фінансової інтелектуалізованої кредитно-скорингової програмної системи використовуються дерева підтримки прийняття рішень до тогош, як бінарні так і не бінарні деревовидні алгоритми. Вони є обчисльовально легкими використовують відносно не великі ресурси процесорного часу та не великі ресурси памяті для обчислень. Як було написано у першому розділі дані деревовидні алгоритми використовуються в режимі чіткої класифікації та нечіткої або мякої класифікації. Переваги мякої класифікації полягають в тому, що вона дає можливість одержувати значення імовірнісних мір, тобто значення функцій належності вектора характеристик клієнта на кредит до того чи іншого класу бінарної класифікації. Відповідно можна керувати рівнем – порогом границі класифікації фінансовій інституції надавати кредит чи не надавати кредит клієнту, тобто віднести до класу з міткою нуль чи до класу з міткою одиниця.

Дана розробка представляє собою проект, який містить ряд пайтонівських скриптів. Кожен скрипт програмується з використанням заданих бібліотек та фреймворків. Імпортування бібліотек та фреймворків для проектування такого проект виглядатиме так:

```
import pandas as pd
import numpy as np
from collections import Counter
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score, accuracy_score
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.metrics import plot_roc_curve
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.externals import joblib
```

Як видно використовуються наступні пайтонівські програмні засоби та інструменти. Бібліотека пандас призначатиметься для маніпуляції даними, зокрема, двовимірним масивом даних про клієнти, що претендують на кредит. Рядки набору даних інтерпритуються, як унікальний номер клієнта, а колонки інтерпритуються, як характеристики, атрибути клієнта. З пандасівської бібліотеки планується використати метод читання даних з файлу із розширенням csv, тобто текстового файлу, але із вказанням роздільника між даними, із вказуванням роздільника між цілою та дробовою частинами, а також із вказанням індексу назв змінних, які представляють собою атрибути клієнта, що бажає взяти кредит. Далі використовуватиметься бібіліотека нумпай також для маніпуляції даними, оскільки не завжди ті чи інші програмні обєкти різних пайтонівських бібліотек приймають на вхід обєкт даних пандасівської бібліотеки, оскільки зрештою пандас є обгорткою на нумпаївською бібліотекою, оскільки вона працює з числовими масивами даних. Тому уздовж всіх скриптів проекту між окремими операціями використовуються дані – обєкти нумпаївської бібліотеки.

Окремо важливим в такого роду систем, зокрема на стадіє їх проектування спланувати правильно зупинки виконання коду. Тому у скрипті проекту напочатку планується використати бібліотеку колекшин, зокрема з неї імпортувати обєкт Каунтер для визначення кількості клієнтів у даних, яким раніше надавали кредити і тих клієнтів, яким не надавали кредити, тобто визначити чи ці бінарні класи є збалансованими покількості елементів чи не збалансованими, оскільки відповідь на це питання вказує на те яку метрику якості використовувати для вирішення даної бінарної задачі.

Наступною для використання планується використовувати в основному скрипті проекту – це деревовидний алгоритм із класу алгоритмів, що працюють на основі логічної парадигми, накшталт, рандомний ліс, підсилення дерев, бегінг, тощо. Деревовидний алгоритм буде саме виконувати роль бінарного класифікатора. Він планується апріорі до навчання, а потім до тестування і лише тоді його використовуватиметься в режимі прийняття рішень. Розглядається навіть приклад

його зберігання, як програмного обєкта у файл, щоб його могли використовувати інші пайтонівські скрипти розроблювального проекту, тобто фінансової інтелектуалізованої кредитно-скорингової системи. Слід відмітити, що деревовидний алгоритм – бінарний класифікатор може приймати рішення як у чіткому форматі так і не чіткому форматі, оскільки він володіє двома методами – методом прогнозування, який повертає мітки чітких класів та методом імовірнісного прогнозування, який повертає і мітки класів і їх імовірності, тобто машина у цьому разі зможе, як чітко відповісти надати кредит клієнтові чи навпаки не надати кредит клієнтові. Також може відповісти з якою імовірністю вона впевнена щодо надання кредиту клієнтові.

Якість роботи основного бінарного класифікатора планується оцінити на основі використання функції перевірки (перехресної перевірки) із вказуванням у ній у вигляді гіперпараметра конкретної метрики. У цій функції планується керувати кількістю фолдів, щоб можливо було оцінити не лише обєктивну точність класифікації, але і визначити робастність роботи системи.

Окремо важливим є питання планування використання функції розщеплення на тренувальну сукупність та тестову сукупність. Також зауважується, що дана функція має гіперпараметр для керування, зокрема, обєм тестової сукупності, який можна вказувати в діапазоні від нуля до одиниці. З еврістичних знань відомо, а тому у проекті використовуватиметься або двадцять пять відсотків, або двадцять відсотків від усього набору даних для тестової сукупності.

Планується використовувати найрізноманітніші метрики якості класифікації, наприклад, проста точність, f1-точність та roc-точність, оскільки вони у сукупності можуть в цілому дати повну оцінку якості роботи кредитно-скорингової системи. Наприклад, перша є легкою для інтерпритації, друга володіє повнотою, але вже не стійка до незбалансованих даних, а третя метрика класифікації, тобто roc-точність хоч і не легка до інтерпритації, зате достатньо стійка до незбалансованих даних. Дані критерії оцінювання якості бінарної класифікації на основі вище записаних метрик

планують використовуватися шляхом їх імпортування із бібліотеки скітлерн. Також із цієї бібліотеки планується використовуватися обєкти, що надають можливість візуалізовувати точність бінарної чи загалом багатокласової класифікації, тобто матриці відносних похибок. Цінність використання такого роду матриць відносних похибок для бінарної класифікації визначається тим, що вони дають можливість візуалізувати відносні похибки бінарної класифікації в межах класів і це добре тим, що можна бачити вплив незбалансованості на процес класифікації.

Окремо важливими для імпортування в пайтонівські скрипти проекту обєктів для гіпероптимізації деревовидного алгоритму. Планується використання двох таких гіпероптимізаторів, перший на основі випадкового пошуку, а лругий на основі граткового пошуку. В одному і другому випадку розмірність фазового простору гіперпараметрів задається їх кількістю, а значення їх областями визначення. Дані гіпероптимізатори знаходять оптимальні значення гіперпараметрів для деревовидного алгоритму при яких значеннях він набуває максимальної точності. Ці гіпероптимізатори плануються братися із бібліотеки скітлерн.

Гратковий гіпероптимізатор переберає всі комбінації значень гіперпараметрів і тому знаходить фактично глобальний екстремум для деревовидного алгоритму, проте є і не долік у його роботі – він затрачає максимум ресурсів процесорного часу та памяті, оскільки перебирає всі комбінації значень гіперпараметрів.

Рандомний гіпероптимізатор для гіперпараметрів деревовидного алгоритму вже не перебирає сві комбінації значень гіперпараметрів і в тому його перевага, оскільки він затрачає менше ресурсів процесорного часу та памяті. Разом з тим він не може знайти глобального екстремуму – тобто мінімального значення відносної похибки для значень гіперпараметрів деревовидного алгоритму.

Останньою важливою бібліотекою для використання є джобліб, яка безпосередньо розприділяє, зокрема розпаралелює обчислення по процесорним ядрам обчислювальної машини, а також по всіх її потоках, щоб максимально раціонально використати її обчислювальні можливості.

Також дана бібліотека дає можливість розгорнути проект фінансової інтелектуалізованої кредитно-скорингової програмної системи на тих чи інших обчислювальних кластерних системах. Кожна з робочих станцій кластерної обчислювальної системи використовуватиме різну кількість фізичних процесорних ядер та їх потоків із вказанням ресурсів памяті які можуть використовуватися при обчисленнях. Ця бібліотека джобліб дає змогу використовувати бібліотеку даск для розподілених обчислень та паралельних обчислень. Таким чином джобліб лежить в основі можливостей використання високопродуктивних обчислень для практичної реалізації фінансової інтелектуалізованої кредитно-скорингової програмної системи.

2.1 Прикладне проектування введення даних та їх візуалізація у провіднику змінних

Введення даних планується реалізувати на пандасівському обєкті для чиатння даних. Зокрема використати метод для читаття текстових файлів із вказанням через гіперпараметр значення що це є текстовий файл. Метод читання даних у фрем даних пандасівської бібліотеки має велику низку гіперпараметрів для керування. Основними з яких звісноє є такі гіперпараметри. Гіперпараметр – роздільник цілої частини числа та його дробової частини, роздільник між числами у файлі даних, наприклад кома чи крапка з комою, чи табулятор або ж інший знаковий роздільник. Також важливим гіперпараметром є гіперпараметр із вказуванням голови, тобто вказуванням рядка у якому містяться назви змінних і які використовуватимуться у фреймі даних. отож розгляньмо даний пайндасівський програмний обєкт та його гіперпараметри для проектування процесу введення даних у основний

пайтонівський скрипт проекту для читання текстових даних в програмний обєкт біліотеки пандас і він має наступний вигляд.

```
pandas.read_csv(
filepath_or_buffer, *, sep=_NoDefault.no_default,
delimiter=None, header='infer',
names=_NoDefault.no_default,
index_col=None, usecols=None,
dtype=None, engine=None,
converters=None, true_values=None,
false_values=None,
skipinitialspace=False,
skiprows=None, skipfooter=0,
nrows=None, na_values=None,
keep_default_na=True,
na_filter=True,
verbose=_NoDefault.no_default,
skip_blank_lines=True, parse_dates=None,
infer_datetime_format=_NoDefault.no_default,
keep_date_col=_NoDefault.no_default,
date_parser=_NoDefault.no_default,
date_format=None, dayfirst=False,
cache_dates=True, iterator=False,
chunksize=None, compression='infer',
thousands=None, decimal='.',
lineterminator=None, quotechar='"',
quoting=0, doublequote=True,
escapechar=None, comment=None,
encoding=None, encoding_errors='strict',
dialect=None, on_bad_lines='error',
delim_whitespace=_NoDefault.no_default,
low_memory=True, memory_map=False,
float_precision=None, storage_options=None,
dtype_backend=_NoDefault.no_default
)
```

На цьому етапі далі більш детально немає необхідності, а тому є зміст перейти до наступних кроків проектування програмного забезпечення для розроблювальної фінансової інтелектуалізованої кредитно-скорингової програмної системи, що

приймає рішення, дотого ж надавати чи не надавати кредит кілєнтам фінансової інституції, а на основі мякої класифікації визначити поріг на рівні якого визначати надавати чи не надавати кредит клієнтові.

2.2 Первинне опрацювання введених даних

Далі послідовно розглянемо програмні обєкти для проектування процесу апріорної переробки даних, який представляє собою конвеєр ряду операцій. І розгляньмо для початку проектування операції видалення дублюючих рядків даних.

2.2.1 Видалення одинакових рядків у введених даних

Дана операція надзвичайно важливою у апріорній переробці даних, оскільки вона позбавляє виникнення моментів перекошення знань машини у одни або інший лкас щодо завдання кредитування клієнтів. Дана операція програмно реалізуватиметься на основі методу пандасівської бібліотеки, а саме використання методу витирання дублікатів. Серед сімейства гіперпараметрів важливим є гіперпараметр, що називається заміна місця, який приймає булеві значення Істина або Заперечення. Цей гіперпараметр може керувати процесом витирання дублікатів, а саме результат дії записувати у нову змінну через використання знаку дорівнює чи в цю ж змінну і цим самим економити простів змінник і память машини, а значить

зменшуються видатки на роботу даної системи. Вигляд даного методу має тау форму, яка наведена нище.

```
DataFrame.drop_duplicates(
subset=None,
*,
keep='first’,
inplace=False,
ignore_index=False
)
```

Зауважимо, що використання методу витирання дублікатів приводить до того, що змінюється індексація рядків у наборі даних, а тому необхідно після дії цієї операції в апріорній переробці даних задіяти операціє по перевстановлені індексів рядків, щоб можна було безпомилково застосовувати дію інших операцій до яких відноситься операція перевстановлення індексів рядків.

2.2.2 Перевстановлення індексів рядків у введених даних

Перейдемо до розгляду проектування операції перевстановлення індексів, що були видаленні підчас виконання попередньої операції – операції видалення дублікатів. Ця операція, як і всі інші операції апріорного перероблення даних має цілу низку власних гіперпараметрів. Розглянемо ряд основних із них. Витирання рядка чи колонки, для цього є окремий гіперпараметр, який вказує на витирання, якщо в режимі Істина та витирання рядка, якщо в режимі фальш. Окрім цього, гіперпараметра є також важливий гіперпараметр, який відповідає за заміну місця. Якщо його поставити в режимі істина, тоді результати будуть виконані над елементами фрейму даних і записані у змінну над якою виконувалася операція, якщо ж значення гіперпараметру заміни місця поставити в режимі фальш, то результати

переробки даних запишуться у нову змінну, яка повинна бути позначена після знаку дорівнює. У випадку, якщо цього не буде, тоді переробка даних відбудеться, але записування результату дії не буде. Важливими є також інші гіперпараметри даного методу, але у проектуванні системи, що розглядається у роботі не використовуються. Отож метод перевстановлення індексів має такий вид.

```
DataFrame.reset_index(
level=None,
*,
drop=False,
inplace=False,
col_level=0,
col_fill='',
allow_duplicates=_NoDefault.no_default,
names=None
)
```

Розглянемо наступну операцію апріорної переробки даних, яка використовується для оцінювання збалансованості даних і для цього використовується окрема бібліотека та її обєкти програмування.

2.2.3 Ідентифікація збалансованості у класах введених даних

Бібліотека колекшн містить програмний обєкт каунтер, який бере на вхід набір даних і по обєму словника унікальних значень, зокрема по значенням цільової змінної визначає кількість елементів у класі і цим самим визначається збалансовані дані чи не збалансовані дані. Якщо відношення між кількістю елементів довільних класів не перевищує порядку (це випливає з еврістичної інформації), то дані вважаються збалансованими і до завдання бінарної класифікації можна

використовувати різного роду критерії якості у іншому випадку, коли відношення перевищує хоча би один або більше порядків, тоді дані вважаються не збалансованими і до них застосовуються лише окремі метрики якості, зокрема roc-точність. Але розглянемо програмний обєкт каунтер бібліотеки коллекшин.

```
class collections.Counter([iterable-or-mapping])
```

Наступною операцією у апріорній переробці даних є операція розщеплення даних на вхідні і вихідні дані, тому перейдемо до наступного підрозділу.

2.3 Розщеплення введених даних на вхідні та вихідні

Операція розщеплення даних займає одне з важливих місць у апріорній переробці даних. Фактично вона розділяє колонки на незалежні змінні та залежні змінні. Ця операція має такий вигляд

```
"""Splitting independent 'x' and dependent 'y'  variables of dataframe 'df'"""
x = df[column_names[0:len(column_names)-1]].values
y = df[column_names[len(column_names)-1]].values
```

Перейдемо до проетування наступної операції апріорної переробки даних для бінарного класифікатора у вигляді деревовидного алгоритму, який є основним серцем прийняття рішення для проектованої фінансової інтелектуалізорваної кредитно-скорингової систем підтримки прийняття рішень. Для практичної реалізації на практиці даної операції планується над пандасівським обєктом даних виконати перетворення, тобто перетворити його у обєкт даних звичайного масиву нумпаївський обєкт даних. Трансформація пандасівського обєкту даних в нумпаївський обєкт даних буде реалізовуватися на основі атрибуту Значення

пандасівського обєкту даних. Це включається у загальне проектування з тієї причини, далі всі наступні програмні обєкти довільних бібліотек пайтона нумпаївські масиви беруть на вхід безпомилково.

2.4 Програмна ініціалізація дерев підтримки прийняття рішень та їх перехресне тестування

Перейдемо до розгляду в проектуванні наступного кроку апріворної переробки даних. І це є використання скітлернівського обєкту – деревовидного алгоритму, який має багато різного роду гіперпараметрів для керування його структурою.

До цієї структури відносяться такі гіперпараметри управління. Висота дерева, даний гіперпараметр дає можливість керувати кількістю ярусів у деревовидному алгоритмі і цим самим керувати точністю, але разорм із тим і обчислювальною складністю. Є також гіперпараметр кількість нодів у розщеплені, тобто їх кількість. Цей гіперпараметр також керує чутливістю девіації змінної у конкретній ноді. Окремим є гіперпараметр, який керує критерієм розщеплення у фіксованій ноді деревовидного алгоритму. Таким критерієм може бути, наприклад, ентропійний критерій, чи наприклад індек джині, чи критерій випадкового вибору, тощо. Можна керувати гіперпараметром, який задає максимальну кількість елементів із набору даних для конкретної ноди і цим самим машина визначається чи потрібно далі щоб дерево зростало чи ні. Також важливим гіперпараметром є величина точності при класифікації набору даних у конкретній ноді дерева. Якщо ця точність є достатньою, то дерево приминяє рости, якщо ж ні то дерево продовжує зростати. Також окремо важливою процедурою є використання так званого пронінгу дерев, тобто обрізання

дерев. Гіперпараметр прунінг – обрізка також дає можливості керувати точністю та складністю деревовидного алгоритму при вирішенні задач класифікації не залежно від претметної області застосування. Перейдемо до опису програмнох обєкта бібліотеки скітлерн, який представляє деревовидний алгорит.

2.4.1 Програмна ініціалізація дерев підтримки прийняття рішень

Скітлернівський деревовидний алгоритм, як буде видно нище має багато гіперпараметрів для керування ним при вирішенні задач класифікації.

```
class sklearn.tree.DecisionTreeClassifier(
*,
criterion='gini',
splitter='best',
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features=None,
random_state=None,
max_leaf_nodes=None,
min_impurity_decrease=0.0,
class_weight=None,
ccp_alpha=0.0,
monotonic_cst=None
)
```

У програмному проекті фінансової інтелектуалізованої кредитно-скорингової програмн. системи для підтримк. прийнятт. рішень він імпортується в основний скрипт із скітлернівської бібліотеки. Головними його методами є метод підгонки або просто метод навчання, коли деревовидний алгоритм навчається на навчальному

наборі даних. також важливими методами є метод прогнозування на основі якого алгоритм приймає рішення щодо повернення мітки класу для фіксованих даних. окрім цих методів важливим є метод прогнозування імовірності класу, який повертає мякі рішення, тобто значення цільової змінної із імовірностями.

2.4.2 Перехресне тестування дерев підтримки прийняття рішень

Особливо важливою процедурою при проектуванні є організація процедури перерхресної перевірки, яка дає змогу уникнути явища випадкового співпадання значень характеристик. Для цього використовуватиметься скітлернівська функція крос-валідації, вигляд якої має такий.

```
sklearn.model_selection.cross_val_score(
                                estimator,
                                X,
                                y=None,
                                *,
                                groups=None,
                                scoring=None,
                                cv=None,
                                n_jobs=None,
                                verbose=0,
                                fit_params=None,
                                params=None,
                                pre_dispatch='2*n_jobs’,
                                error_score=nan)
```

Дана функція дає змогу реалізовувати на практиці крос-валідацію. Вона має багато гіперпараметрів для керування серед яких є дуже різноманітні. Наприклад, вона дає можливість використати ефективніше високопродуктивні обчислення, а саме вона дає змогу розпаралелювати крос-валідацію по потоках на процесорних

ядрах. До її аргументів відносяться вхідні дані у форматі обєкті даних пандаса чи нумпай, але у проектованому проекті це обєкт нумпая. Також вихідні дані, тобто цільова змінна чи змінні у даній роботі тут лише одна бінарна змінна, яка вказує на прийняття рішення, щоб дати чи не давати кредит клієнтові. Одним із найважливіших її гіперпараметрів є мітка метрики на основі якої скітлернівська функція крос-валідації визначає точність. Програмний вид значень метрик якості є таким, як наведено по тексту нище.

```
sklearn.metrics.accuracy_score(
y_true,
y_pred,
*,
normalize=True,
sample_weight=None)

sklearn.metrics.f1_score(
y_true,
y_pred,
*,
labels=None,
pos_label=1,
average='binary’,
sample_weight=None,
zero_division='warn')

sklearn.metrics.roc_auc_score(
y_true,
y_score,
*,
average='macro’,
sample_weight=None,
max_fpr=None,
multi_class='raise’,
labels=None)
```

Вибір значення для гіперпараметра метрики визначається багатьма факторами, зокрема і вище визначеними збалансованістю чи не збалансованістю наборів даних у межах класів. Питаннями щодо інтепритації значень метрик якості

у випадку простої точності, яка наочно зрозуміла, як величина частки правильних відповідей машини (системи).

2.5 Розщеплення введених даних на навчальний набір та тестовий набір

Черговою операцією при проектуванні апріорної переробки даниї є розщеплення набору даних на тренувальний набір та тестовий набори даних. Вигляд програмного обєкта розщеплення є таким

```
sklearn.model_selection.train_test_split(
*arrays,
test_size=None,
train_size=None,
random_state=None,
shuffle=True,
stratify=None)
```

Даний програмний обєкт має цілу низку для свого програмного управління гіперпараметрів. Серед них тут відзначається лише гіперпараметри, які визначають величину обєму тренувальног чи тестового набору даних.

2.6 Проектування оцінювання та візуалізації нормованої та не нормованої матриць похибок

Оскільки девелопери при розробці оперують, як і інші люди не лише логічними правилами, але і знаковими образами, якщо наочніше, то візуальним

образами, то при проетування програмного забезпечення для фінансової інтелектуалізованої кредитно-скорингової програмної системи використовуються нормовані та не нормовані матриці відносних похибок, які у своїх елементах обчислюють відносні похибки всередині двох класів – надавати кредит клієнтові чи не надавати кредит клієнтові. Таким, чином не нормованна матриця віндосних похибок бінарної класифікації програмно має вид такий.

```
sklearn.metrics.confusion_matrix(
y_true, y_pred,
*,
labels=None,
sample_weight=None,
normalize=None)
```

А нормована матриця відносних похибок в середині класів задачі бінарної класифікації програмується на основі програмного виду такого, який наведено далі по даному тексту.

```
sklearn.metrics.plot_confusion_matrix(
estimator,
X,
y_true,
*,
labels=None,
sample_weight=None,
normalize=None,
display_labels=None,
include_values=True,
xticks_rotation='horizontal’,
values_format=None,
cmap='viridis’,
ax=None,
colorbar=True)
```

Дані матриці обчислення відносних похибок в середині класів задачі бінарної класифікації лише кількісно констатують факт величини метрологічних характеристик розроблювальної програмної системи. Але перейдемо до розгляду наступних операцій, які входять в основне тіло скрипта програмного проекту, що з

точки зору програмної інженерії представляє суть фінансової інтелектуалізованої кредитно-скорингової програмної системи для підтримки прийняття рішень.

2.7 Проектування гіпероптимізації дерев підтремки прийнятя рішень

Проетування гіпероптимізації є надзвичайно важливим етапом у проектуванні програмного забезпечення, оякільки у дані області теорія не достатньо напрацьована і на практиці при програмуванні використовують виключно еврістичні знання. А еврістичні знання не дають змоги задати точно обгрунтовані, оптимальні значення для гіперпараметрів загалом, алгоритмів різного виду, але і зокрема використовуваного у роботі деревовидного алгоритму. Фактично гіперопттимізація – це є канонічна оптимізація функції, аргументами якої є гіперпараметри деревовидного алгоритму. Вона представляє собою дослідження функції помилки на мінімум. Тобто вона задаючи різні значення гіперпараметрів задає цілий простір деревовидних алгоритмів і для кожного з них виконує крос-валідаційну перевірку. І в решті повертає ті значення гіперпараметрів для яких було додсягнуто мінімальне значення похибки, яка зададана, як значення гіперпараметрі гіпероптимізаційного програмного обєкта.

Для гіпероптимізації можуть використовуватися самі різні програмні обєкти, проте одними із найчастіше використовуваних на практиці є гіпероптимізатори на основі випадкового пошуку та пошуку по гратці. Кожен із них має. Як свої переваги так і недоліки, наприклад випадковий пошук, завдяки тому, що псевдовипадково рухається по гратці декартової системи координат гіперпараметрів затрачає відносно не багато ресурсів памяті та процесорного часу процесорів, але і не гарантує досягнення глобального екстремуму. На противагу йому програмний обєкт у вигляді

гіпероптимізатора граткового пошуку. Перевага його, що він перебирає всі можливі комбінації значень у гратці гіперпараметрів і точно знаходить глобальний мінімум похибки, але разом з тим він затрачає надзвичайно багато обчислювальних ресурсів памяті та процесорного часу. Перейдемо до розгляду першого із програмних обєктів, зокрема, гіпероптимізаторів, і першим буде гратковий гіпероптимізатор.

2.7.1 Проектування гіпероптимізації дерев підтримки прийняття рішень на основі GridSearch

Програмний вид даного програмного гіпероптимізатора має такий вигляд, який наведено нище по тексту.

```
class sklearn.model_selection.GridSearchCV(
estimator,
param_grid,
*,
scoring=None,
n_jobs=None,
refit=True,
cv=None,
verbose=0,
pre_dispatch='2*n_jobs',
error_score=nan,
return_train_score=False
)
```

Як і всі програмні обєкти пайтона у нього є багато гіперпараметрів для управління ним, оскільки, як і всі пайтонівські обєкти вони відносяться до високого рівня абстракції відповідно у них у всіх є багато гіперпараметрів для керування. І програмний гіпероптимізатор є не виключенням з правил. Розглянемо ряд основних

його гіперпараметрів серед яких варто відзначити можливіть розпаралелювати обчислення по потокам процесорних ядер і це надзвичайно серйозна перевага на практиці програмування, а тому тут він і зазначається у процесі проетування програмного забезпечення у вигляді проекту. Також є у ньому гіперпараметр для керування конкретною міткою метрики якості, тобто можна вибирати метрику якості для обчислення похибки.

2.7.2 Проектування гіпероптимізації дерев підтримки прийняття рішень на основі RandomSearch

Програмний вигляд наступного гіпероптимізатора є таким, як наведено нище.

```
class sklearn.model_selection.RandomizedSearchCV(
estimator,
param_distributions,
*,
n_iter=10,
scoring=None,
n_jobs=None,
refit=True,
cv=None,
verbose=0,
pre_dispatch='2*n_jobs',
random_state=None,
error_score=nan,
return_train_score=False
)
```

Він, як і гратковий гіпероптимізатор призначений для знаходження оптимального виду формату алгоритму, оскільки теорії немає і необхідно прийняті рішення грунтвати виключно на еврістично-практичних знаннях, а тому виконувати

пошук у просторі гіперпараметрів оптимальних їх значень. Рандомний гіпероптимізатор є в обчислювальному розумінні набагато оптимальнішим із гратковим гіпероптимізатором, але і той який не гарантує досягнення глобального мінімуму відносної похибки.

Даний рандомний гіпероптимізатор дає змогу також розпаралелювати обчислювальні процеси по потокам, керувати імовірнісними розподілами при випадковому русі по гратці гіперпараметрів.

Після того, як виконані процеси гіпероптимізації, тобто знайдені оптимальні значення відносних похибок банарної класифікації, тобто знайдені значення гіперпараметрів деревовидного алгоритму цей класифікатор повністю готовий до його використання у програмні практичній реалізації для функцілнування фінансової інтелектуалізованої кредитно-скорингової прогарамної системи для підтримки прийняття рішень щодо того надавати кліжнтам кредити чи ні.

Для того, щоб цей класифікатор був доступній усім пайтонівським скриптам розроблювального проету та загалом іншим програмам, то необхідно його окремо зберегти, як програмний обєкт у файл і для цього процесу розглядається наступний підрозділ даного розділу цієї кваліфікаційної роботи.

2.8 Проектування експорту та імпорту попередньо навченого дерева підримки прийняття рішень

Для зберігання бінарного класифікатора – деревовидного алгоритму, який є верцем програмного проекту фінансової інтелектуалізованої кредитно-скорингової програмної системи для підтримки прийнятті рішень у файлі, як програмного обєкта

зі своєю структурою використовуватиметься пайтонівська бібліотека джоблшіб, конкретно зі своєю функцією дамп, тобто функцією зберегання програмного обєкта та його гіперпараметрів, як окремої програмної компоненти. Програмна реалізація наведена нище по тексту роботи.

```
joblib.dump(
value,
filename,
compress=0,
protocol=None,
cache_size=None
)
```

Як видно із сукупності його гіперпараметрів, можна при збкріганні бінарного класифікатора у вигляді програмної компоненти керувати багатьма супроводжуючими процесами, наприклад, можна керувати місце зберігання, тобто адресою зберігання створеного програмного обєкта. Величиною стискання та багатьма іншими гіперпараметрами.

Найтонівська бібліотека джобліб нає можливості не лише для формування та зберігання програмних компонент, як програмних інструментів, але і їх завантажувати у подальшому в ті чи інші проекти, програмні забезпечення. Процес програмного завантаження має наступний вигляд.

```
joblib.load(
filename,
mmap_mode=None
)
```

Таким чином метод лоад дає млгу завантажитити програмну компоненту із своєю структурою до одного із пайтонівських скриптів розроблювального проекту і використовувати знову бінарний класифікатор- деревовидний алгоритм, який вже є попередньо навченим в режимі прийняття рішень, тобто можна на його вхід подавати чергові дані від того чи іншого клієнта фінансової інституції, тобто банку чи кридитної спілки і користуватися цого виводом у вигляді чіткої мітки, тобто чіткої

класифікації давати чи не давати кредит, або не чіткох класиікації із значенням імовірнісної міри щодо надання чи не надання кредиту.

Тепер можна перейти до загальної програмної архітектури фінансової інтелектаілізованої кредитно-скорингової системи підтримки прийняття рішень, що і буде розглянуто у настпуному підрозділі цього ж розділу роботи.

2.9 Прикладне проектування архітектури кредитно-скорингової програмної системи

Таким чином, загальна програмна архітектура фінансової інтелектаілізованої кредитно-скорингової системи підтримки прийняття рішень щодо надання чи не надання кредитів клієнтам фінансових інституцій, накшталт, банків, кредитних спілок, тощо, має вигляд, який наведений на рисунку, що зображений нище і містить такі основіні блоки.

Першим етапом програмного забезпечення є етап імпортування всіх бібліотек та фреймворків у скрипти проекту фінансової інтелектаілізованої кредитно-скорингової системи підтримки прийняття рішень, зокрема, бібліотек для маніпуляції даними, їх обробки, тощо. Потім іде етап завантаження даних у систему та їх апріорна переробка цілим конвеєром операцій. Наступним етапом є єтап розщеплення даних, на цій основі навчання та тестування алгоритмів. Далі відбувається етап гіпероптимізації алгоритму підтримки прийняття рішень, а потім етап зберігання та експортування одержаних даних та програмних компонент у інші програмні забезпечення сторонніх інформаційних систем.

Таким чином узагальнена архітектура має вигляд конвеєра операцій.

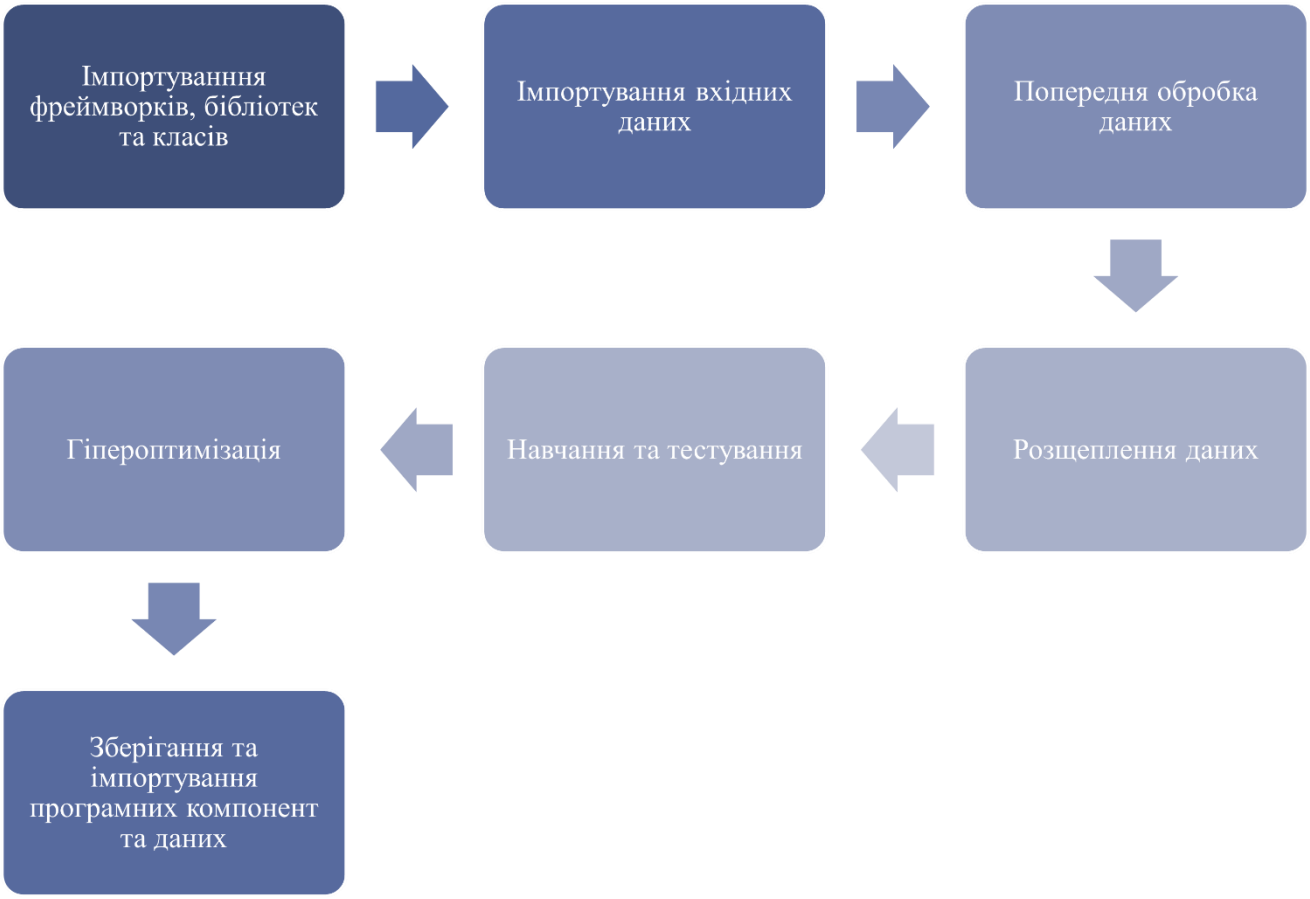

Рис.2.1 – Візуальне представлення прикладного проектування архітектури кредитно-скорингової програмної системи

На цьому етапі процес проетування даної розробки закінчується і у наступному розділі даної роботи буде розглянутий конвеєрний процес практичного програмного конструювання програмного забезпечення у вигляді програмного проекту для фінансової інтелектаілізованої кредитно-скорингової системи підтримки прийняття рішень

РОЗДІЛ 3. ПРАКТИЧНО-ПРИКЛАДНЕ КОНСТРУЮВАННЯ КРЕДИТНО-СКОРИНГОВОЇ ПРОГРАМНОЇ СИСТЕМИ

3.1 Імпортування пайтонівських фреймворків, бібліотек, модулів та функцій

Практична реалізація імпортування всіх необхідних пайтонівський фреймворків, бібліотек, функцій та модулів до скриптів розроблювального проекту у дані роботі практично у коді має вигляд такий.

```
import pandas as pd
import numpy as np
from collections import Counter
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score, accuracy_score
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.metrics import plot_roc_curve
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.externals import joblib
```

Зупинимося на детальнішому описі та обгрунтувані потреби імпортованих програмних пайтонівських засобів для практичної роботи скриптів проекту. Отож, використовуються дві пайтонівські бібліотеки для маніпулювання даними та процесу апріорної переробки даних – це бібліотеки пандас та нумпай. У пайтонівських скриптах проекту використовуються ряд методів до їх відповідних обєктів даних, наприклад, стандартизація всіх змінних, тобто колонок двовимірного нумпаївського масиву даних та двовимірного датафрейму – обєкту біліотеки пандас. Також використовуються методи нормування даних в евклідовому просторі даних в якому координатні вісі інтерпритуються, як колонки амсивів, тобто є змінними – атрибутами клієнтів, які хотять взяти кредит у банку чи кредитній спілці чи довільній іншій фінансовій інституції. Також використовується різні різновиди методу відновлення пропущених даних шляхом заміни пропусків статистичними

оцінками для кожної окремої колонки, накшталт, середнє арефметичне, мода набору даних, а також медіана набору даних. Апріорним кроком, операцією виконується видалення із масиву аутлаєрів, тобто викидів у даних, наприклад на основі статистичного тесту Грабба, якщо апріорі відомо, що колонка даних має гаусівський закон розподілу щільності імовірності, в іншому випадку використовуються інші підходи, наприклад на основі логічної парадигми навчання машин, типу ізольовані ліси, можуть використовуватися інші підходи, наприклад на основі кластерного аналізу, де визначається відстать від окремого елемента даних до сукупності даних на основі методу найближчих сусідів та багато інших підходів. Також у апріорній переробці даних використовуються інші нумпаївські та пандасівські операції для попередньої переробки даних.

3.2 Читання (ввід) з файлу вхідних даних з різними роздільниками

Для читання (вводу) з файлу даних практично використовувався клас із пандасівської бібліотеки мови програмування пайтон. Цей клас керувався пулом гіперпараметрів, які надали змогу розділити числові дані між собою та розділити їх цілі та дробові частини між собою. Практично кодова реалізація цього наведена нище по тексту.

```
"""Read dataframe"""
df = pd.read_csv("./CREDIT_SCORING/CREDIT_SCORING.csv",
                 delimiter=";", decimal=".", header=0)
```

Як видно з кодового представлення на практиці, для розділення числових даних при читатті їх із файлу використано гіперпараметр delimetr=”;” із значенням крапка з комою, а для розділення цілої частини від дробової частини в даних

використано гіперпараметр decimal=”.” зі своїм значенням крапки. Окремо виокремо, що назва змінних, яка знаходилася у нульовому рядку файлу з вхідними даними, практично була вказана за допомогою гіперпараметру header=0 із нульовим значення індексу рядка.

Вигляд прочитаних (введених) даних у пайтонівський скрипт має наступний вигляд, що візуалізовано на рисунку нище.

Index	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X
0	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	0	0	0	0	689	0	0	0
1	120000	2	2	2	26	-1	2	0	0	0	2	2682	1725	2682	3272	3455	3261	0	1000	1000	1000	0
2	90000	2	2	2	34	0	0	0	0	0	0	29239	14027	13559	14331	14948	15549	1518	1500	1000	1000	10
3	50000	2	2	1	37	0	0	0	0	0	0	46990	48233	49291	28314	28959	29547	2000	2019	1200	1100	10
4	50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670	35835	20940	19146	19131	2000	36681	10000	9000	68
5	50000	1	1	2	37	0	0	0	0	0	0	64400	57069	57608	19394	19619	20024	2500	1815	657	1000	10
6	500000	1	1	2	29	0	0	0	0	0	0	367965	412023	445007	542653	483003	473944	55000	40000	38000	20239	137
7	100000	2	2	2	23	0	-1	-1	0	0	-1	11876	380	601	221	-159	567	380	601	0	581	168
8	140000	2	3	1	28	0	0	2	0	0	0	11285	14096	12108	12211	11793	3719	3329	0	432	1000	10
9	20000	1	3	2	35	-2	-2	-2	-2	-1	-1	0	0	0	0	13007	13912	0	0	0	13007	112
10	200000	2	3	2	34	0	0	2	0	0	-1	11073	9787	5535	2513	1828	3731	2306	12	50	300	373
11	260000	2	1	2	51	-1	-1	-1	-1	-1	2	12261	21670	9966	8517	22287	13668	21818	9966	8583	22301	0
12	630000	2	2	2	41	-1	0	-1	-1	-1	-1	12137	6500	6500	6500	6500	2870	1000	6500	6500	6500	287
13	70000	1	2	2	30	1	2	2	0	0	2	65802	67369	65701	66782	36137	36894	3200	0	3000	3000	15
14	250000	1	1	2	29	0	0	0	0	0	0	70887	67060	63561	59696	56875	55512	3000	3000	3000	3000	30
15	50000	2	3	3	23	1	2	0	0	0	0	50614	29173	28116	28771	29531	30211	0	1500	1100	1200	13
16	20000	1	1	2	24	0	0	2	2	2	2	15376	18010	17428	18338	17905	19104	3200	0	1500	0	165
17	320000	1	1	1	49	0	0	0	-1	-1	-1	253286	246536	194663	70074	5856	195599	10358	10000	75940	20000	195
18	360000	2	1	1	49	1	-2	-2	-2	-2	-2	0	0	0	0	0	0	0	0	0	0	0
19	180000	2	1	2	29	1	-2	-2	-2	-2	-2	0	0	0	0	0	0	0	0	0	0	0

Рис.3.1 – Фрагмент вхідн. даних для завдання кредитног. скорингу

Оскільки на рисунку вище візуалізовано лише фрагмент вхідних даних та не очевидна їх інтерпритація, то формальний опис та інтерпритація ознак вхідних даних для програми, що приймає рішення надати чи не надати фінансову позику наводиться в описі нище.

Введемо символічні позначення для колонок даних, які представляють собою змінні, тобто атрибути клієнта кредитування, накшталт рівня його зарплати, сімейного стану, рівня його активів, тощо. Для позначення фрерйму даних застосовуватимемо символічне позначення df, а для атрибутів клієнтів

кредитування введемо символьні позначення такі: $f_1(\vec{x})$, $f_2(\vec{x})$, …, $f_{21}(\vec{x}) = y$, тут двадцять перша колонка даних є цільовою змінною, яка описується бінарними значеннями – один це надати кредит, а нуль це відмовити у кредитуванні клієнта.

Розглянемо деталювання інтерпритації кожної змінної двовимірного масиву, тобто дефініції атрибутів клієнтів, якими вони характеризуються і представляють собою вектори:

$f_1(\vec{x})$ – сума на кредиту, що надається (одиниці розмірностей тут є грошова одиниця долар: вона містить у собі, як індивідуальний користувацький кредит так і додатковий кредит для його суми;

$f_2(\vec{x})$ – стать (1 = чоловіча стать; 2 = жіноча стать;

$f_3(\vec{x})$ – освіта (1 = вища школа, 2 = університет, 3 = середня школа, 4 = довільна інша освіта (довільний інший рівень освіти);

$f_4(\vec{x})$ – сімейний стан (1 = одружений, 2 = не одружений, 3 = інший сімейний стан;

$f_5(\vec{x})$ – вік;

$f_6(\vec{x}) - f_{11}(\vec{x})$ – історія попередніх платежів клієнта. Відслідковувалися апріорні щомісячні звіти про виплати (з березня місяця до жовтня місяця) наступним чином): $f_6(\vec{x})$ – стан оплати платежу у березні місяці; $f_7(\vec{x})$ – стан оплати платижу у вересні місяці, ..., ; $f_{11}(\vec{x})$ – стан оплати платежу в травні місяці: (шкала вимірювання одержання: 1 = оплата своєчасна, 2 = ззатриска виплати кредиту протягом одного місяця, 3 = затримка виплати платежу протягом двох місяців, ..., 8 = затримка випдати платежу протягом восьми або девяти місяців, 9 = затримка виплати платежу протягом десяти місяців та більшої кількості місяців;

$f_{12}(\vec{x}) - f_{17}(\vec{x})$ – сума представленого рахунку (одиниця розмірностей у грошовій одиниці це долар: $f_{12}(\vec{x})$ – сума представленого платежу у вересні місяці,

$f_{13}(\vec{x})$ – сума представленого платежу у серпні місяці, ..., $f_{17}(\vec{x})$ – сума представленого платежу у квітні місяці;

$f_{18}(\vec{x}) - f_{23}(\vec{x})$ – сума попередньої оплати (у грошовій одиниці долар): $f_{18}(\vec{x})$ – сума оплати у вересні місяці, $f_{19}(\vec{x})$ – сума оплати в квітні місяці, ..., $f_{23}(\vec{x})$ – сума оплати у квітні місяці;

$f_{24}(\vec{x})$ – цільова змінна y. Сплата кредиту (так = 1, ні = 0).

Таким чином наведено дифініцію усіх атрибутів клієнта, що планує взяти фінансовий користувацький кредит.

3.3 Прикладна реалізація на практиці блоку попередньої обробки даних

Блок прикладної реалізації на практиці попередньої обробки даних складається з послідовності ряду пандасівських методів, оскільки обєкт даних є пандасівським фреймом даних. Також використано для кодингу пайтонівська collections бібліотека та обєкт із неї Counter. Рухатимемося послідовно уздовж даного модуля.

```
"""Preprocessing data"""
#Removing duplicate rows
df.drop_duplicates(subset=None, keep='first', inplace=True)
#Reset index of dataframe
df.reset_index(drop=True, inplace=True)
#Return column names of dataframe
column_names = list(df.columns)
#Checking data balance
target_var = df[column_names[-1]]
balance = Counter(target_var)
#Conclusion about metrics
print("Data set is not balance (volum of target variable values are",balance,
      "). Use 'roc_auc' metric and not use 'accuracy' or 'f1' metrics")
```

Перейдемо до коментування практичного конструювання у коді процесу апріорної переробки даних перед тим, як вони повинні подаватися на деревовидний алгоритм підтримки прийняття рішень.

На першому кроці видаляються усі дублікати рядків масиву даних, щоб знання деревовидного алгоритму не були точним на одних даних і менш точними на інших даних. На наступному етапі, тобто у наступному рядку перевстановлюються індекси рядків, оскільки мало місце порушення їх порядку після витирання дублікатів рядків і при потребі організувати той чи інший цикл програма буде видавти помилки. У наступному рядку коду створюється змінна назв змінних, тобто колонок і перетворюється до виду типу список. Потім формується цільова змінна шляхом звернення до останньої колонки масиву, використовуючи відємну індексацію. Наступним кроком виконується операція балансування з якої можна встановити чи збалансовані дані в класах чи не збалансовані – це визначається по величині відношення кількості елементів кожного із класів. Ну і у кінцевому випадку необхідна інформація виводиться на друк (в принципі ця операція може бути при потребі, тобто не є обовязковою).

Перейдемо до наступного підрозділу третього розділу роботи.

3.4 Розщеплення даних

Розщеплення набору введених даних виконується на практиці наступним чином, як наведено нище по тексту.

```
"""Splitting independent 'x' and dependent 'y'  variables of dataframe 'df'"""
x = df[column_names[0:len(column_names)-1]].values
y = df[column_names[len(column_names)-1]].values
```

Звернімо увагу, що для формування взідних і вихідних даних використовується змінна назв колоно і після їх формування вони за допомогою атрибуту датафрейму перетворюються в обєкт даних бібліотеки нумпай. Отож перейдемо до наступного підрозділу третього розділу цієї роботи.

3.5 Навчання, перевірка дерев підтримки прийняття рішень

Для початку виконується на практиці ініціалізація із біліотеки скітлерн дерева підтрим. прийнят рішень, всіх необхідних функцій, тощо. Реалізація (конструювання) у коді має наступний вигляд.

```
"""Training and testing model. Compute f1-score and
area under the receiver operating characteristic curve (ROC AUC)"""
#Initialization
dt = DecisionTreeClassifier(class_weight='balanced')
#Training
dt.fit(x_train, y_train)
#Prediction
y_pred = dt.predict(x_test)
#Predict class probabilities
y_pred_prob = dt.predict_proba(x_test)

#Compute the 'accuracy'
accuracy_score = accuracy_score(y_test, y_pred)
#Compute the 'f1-score'
f1_score = f1_score(y_test, y_pred)
#Compute the 'roc-auc'
roc_auc_score = roc_auc_score(y_test, y_pred)

#Compute confusion matrix to evaluate the accuracy of a classification
conf_matrix = confusion_matrix(y_test, y_pred)
#Compute normalized confusion matrix to evaluate the accuracy of a classification
norm_conf_matrix = confusion_matrix(y_test, y_pred, normalize='all')
```

Оскільки у попередньому розділі проектування було достатньо детально описаний процес навчання, тестування та крос-валідаційної перевірки деревоводного алгоритму для підтримки прийняття рішень щодо надання чи не

надання фінансовою інституцією кредиту для клієнта, то тут зупинятися не будемо і перейдемо до наступнього кроку, тобто наступного підрозділу третього розділу цієї роботи бакалавра.

3.6 Обчислення та зображення масивів похибок

Аналогічно, як було зазначено у попередньому підрозділі, тобто процес обчислення матриць відносних похибок та їх візуалізація були детально описані при проетуванні програмного забезпечення, то тут лише наведемо практичне конструювання цих моментів без коментарів.

```
"""Visualization of confusion matrix and receiver operating characteristic curve (roc-
curve)"""
#Plot non-normalized confusion matrix
disp = plot_confusion_matrix(dt, x_test, y_test, display_labels=np.unique(y))
disp.ax_.set_title('Confusion matrix, without normalization')
#Plot normalized confusion matrix
disp = plot_confusion_matrix(dt, x_test, y_test, normalize='all', display_labels=np.unique(y))
disp.ax_.set_title('Normalized confusion matrix')

#Plot roc-curve
disp = plot_roc_curve(dt, x_test, y_test)
disp.ax_.set_title('Receiver operating characteristic curve (roc-curve)')
```

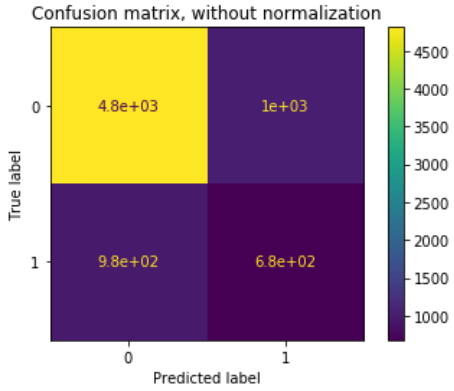

Рис.3.2 – Не нормована матриця похибок

Якщо на верхньому малюнку візуалізовано не нормовану матрицю відносних похибок в межіах класів, то на наступному малюнку наведено нормовану матрицю відносних похибок в межах бінарних класів.

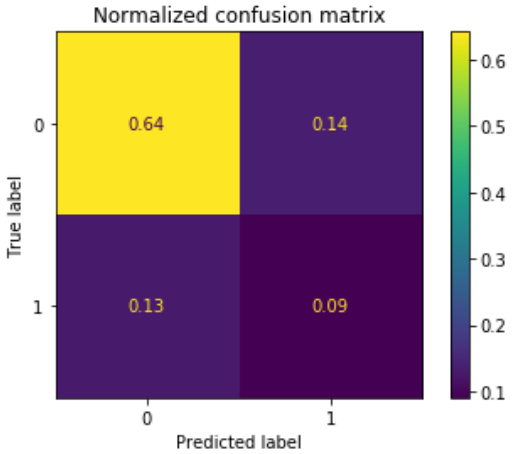

Рис.3.3 – Нормована матриця похибок

Для наочної візуалізації точності, яка є стійкою для незбалансованих даних, тобто roc-точність нище наводиться roc-крива, де візуально можна бачити, що площа під даною криволінійною трапецією дорівнює по модолю значенню roc-точності.

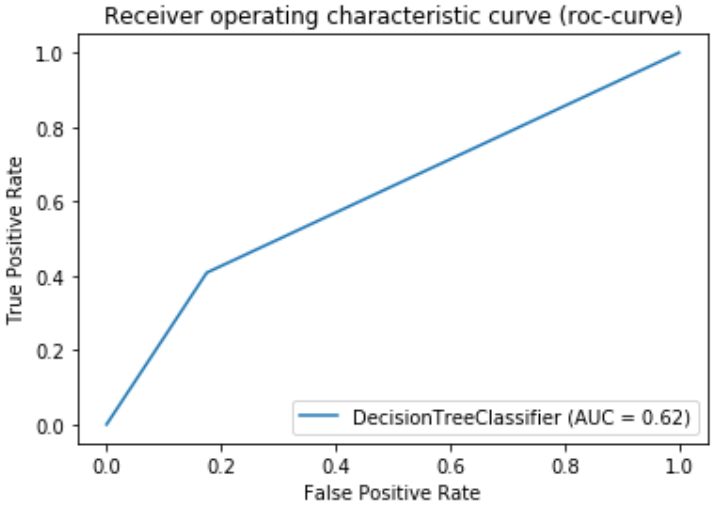

Рис.3.4 – Візуалізація площі криволінійної трапеції roc кривої, як точності

Перейдемо до опису прикладної реалізації процесів гіпероптимізації, які практично реалізовані на класах скітлернівської бібліотеки.

3.7 Гіпероптимізація дерева підтримки прийняття рішень

У роботі розглядається два гіпероптимізатори один із них – це гратковий гіпероптимізатор і його практична реалізація наведена далі у першому пункті.

3.7.1 Гіпероптимізація дерева підтримки прийняття рішень на основі GridSearchCV

```
"""Optimization of hyper-parameters based on GridSearchCV"""
#Label of the DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini', min_samples_split=2)

#Optimization
hyper_parameters = {'criterion':('gini', 'entropy'),
                    'min_samples_split':[2, 3, 4]}
grid_optimal_dt = GridSearchCV(dt, hyper_parameters, scoring='accuracy')
grid_optimal_dt.fit(x, y)

#Hyper-parameters of optimal estimator
grid_optimal_dt.best_params_

#Computing mean cross-validation accuracy for optimal estimator
grid_optimal_dt_cross_val_accuracy = cross_val_score(grid_optimal_dt, x, y,
                                                     cv=3, scoring='accuracy')
grid_optimal_dt_mean_accuracy = grid_optimal_dt_cross_val_accuracy.mean()

#Predict based on optimal estimator
y_pred_grid = grid_optimal_dt.predict(x_test)
```

У наступному другому пункті наведена практична реалізація гіпероптимізатора раномного пошуку.

3.7.2 Гіпероптимізація дерева підтримки прийняття рішень на основі RandomizedSearchCV

Практична реалізація гіпероптимізатора рандомного пошуку у гратці гіперпараметрів деревовидного алгоритму наведена нище у тексті.

```
"""Optimization of hyper-parameters based on RandomizedSearchCV"""
#Label of the DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini', min_samples_split=2)

#Optimization
hyper_parameters = {'criterion':('gini', 'entropy'),
                    'min_samples_split':[2, 3, 4]}
random_optimal_dt = RandomizedSearchCV(dt, hyper_parameters, scoring='accuracy')
random_optimal_dt.fit(x, y)

#Hyper-parameters of optimal estimator
random_optimal_dt.best_params_

#Computing mean cross-validation accuracy for optimal estimator
random_optimal_dt_cross_val_accuracy = cross_val_score(random_optimal_dt, x, y,
                                                       cv=3, scoring='accuracy')
random_optimal_dt_mean_accuracy = random_optimal_dt_cross_val_accuracy.mean()

#Predict based on optimal estimator
y_pred_random = random_optimal_dt.predict(x_test)
```

Як було зазаначено у розділі проектування фінансової інтелектуалізованої кредитно-скорингової системи на тезнології пайтон даний гіпероптимізатор володіє рядом переваг, а саме, він потребує у своїй роботі менше затрат компбтерного часу та менше затрат компютерної памяті на його практичну реалізацію, проте він не може гарантувати пошуку глобального екстремомк, зокрема, глобального мінімому для відносної похибки бінарної еласифікації.

3.8 Зберегіння попередньо навченого дерева підтримки прийняття рішень

Зберігання попередньо навченого деревовидного алгоритму підтримки рішень, як програмного обєкта у файлі для його виклику скриптами програмного проекту на практиці реалізується за допомогою бібліотеки джобліб. Також ця бібліотека використовується для його завантаження у скрипти пайтонівського проекту у цій роботі і може викликати бінарний класифікатор сторонніми програмами при необхіжності. Практична реалізація приводиться тут.

```
"""Saving trained model. Loading trained model. Decision making by trained model"""
#Saving trained model
joblib.dump(dt, 'clf.pkl')
#Loading trained model
dt = joblib.load('clf.pkl')
#Decision making by trained model
result = dt.predict(x_test)
```

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Критичні стани людини

В житті вистачає непередбачуваних ситуацій, коли життя від смерті відділяють хвилини або секунди. І навіть якщо не брати до уваги військові дії, в мирному житті існує багато небезпек, пов’язаних з надзвичайними ситуаціями, дорожньо-транспортними пригодами, виробничими й побутовими травмами різної складності, електрикою або хімічними речовинами тощо.

Домедична допомога – невідкладні дії та організаційні заходи, спрямовані на врятування та збереження життя людини у невідкладному стані та мінімізацію наслідків впливу такого стану на її здоров’я, що здійснюються на місці події особами, які не мають медичної освіти.

Першу медичну допомогу надають особи, які мають медичну освіту.

Екстрена медична допомога – медична допомога, яка полягає у здійсненні працівниками системи екстреної медичної допомоги невідкладних організаційних, діагностичних та лікувальних заходів, спрямованих на врятування і збереження життя людини у невідкладному стані та мінімізацію наслідків впливу такого стану на її здоров’я відповідно до Закону.

Невідкладний стан — патологічні зміни в організмі людини, котрі виникають внаслідок дії фізичних, хімічних і біологічних факторів, які призводять до різкого погіршення здоров'я, можуть загрожувати життю і потребують екстрених лікувальних дій (в межах хвилини/хвилин, рідше годин).

До них відносять патологічні стани, які: безпосередньо не загрожують життю, однак така загроза може стати реальною у будь-яку мить; під час яких відсутність своєчасної медичної допомоги може зумовити стійкі зміни в організмі; за яких в короткий термін треба полегшити страждання хворого чи постраждалого.

Розрізняють 5 видів невідкладних станів:

коли існує вірогідність смерті у випадку вчасного не надання медичної допомоги; не має безпосередньої загрози життю, але вона може виникнути в будь-який момент; не надання своєчасної медичної допомоги може призвести до стійких змін в організмі або в окремих органах; необхідно терміново полегшити страждання потерпілого; поведінка постраждалого становить небезпеку для оточуючих і потребує термінового медичного втручання.

Надання домедичної допомоги заключається в наступному:

1. Скласти загальне враження про травмованого - інформація, отримана під час наближення до постраждалого, на основі якої формується попередній висновок про механізм та обставини травми, попередній стан постраждалого.

2. Провести первинний огляд - швидка оцінка важливих життєвих функцій у постраждалого та за потреби усунення загрозливих для життя станів.

3. Виконати вторинний огляд - оцінка та визначення наявних травм у постраждалого, що проводиться за умови відсутності загрозливих для його життя станів.

Що для цього потрібно зробити:

1) Перед наданням допомоги переконатися у відсутності небезпеки для себе, оточуючих, постраждалого та тільки за її відсутності перейти до наступного кроку.

2) Первинне враження: під час наближення до постраждалого оцінити: вік, стать, вагу, положення, в тому числі по відношенню до оточуючих предметів; оцінити активність постраждалого: орієнтований в часі та просторі, виконує цілеспрямовані рухи, збуджений тощо; звернути увагу на ознаки зовнішньої масивної кровотечі; звернутися до постраждалого: «Ви мене чуєте? Мене звати (назвати своє ім’я). Я можу вам допомогти?»; наближатися до постраждалого зі сторони можливого візуального контакту, щоб обмежити його рухи головою.

У разі виявлення людини у невідкладному стані, яка не може особисто звернутися за наданням екстреної медичної допомоги та за відсутності медичних працівників на місці події громадянин України або будь-яка інша особа, які виявили таку людину, зобов’язані: негайно здійснити виклик екстреної медичної допомоги або повідомити про виявлену людину у невідкладному стані та про місце події працівників найближчого закладу охорони здоров’я чи будь-яку особу, яка зобов’язана надавати домедичну допомогу та знаходиться поблизу місця події; за можливості надати виявленій людині у невідкладному стані необхідну допомогу.

Як оцінити стан постраждалого та що робити.

Підійти до постраждалого (якщо немає загрози вашому здоров’ю та життю).

Покликати.

Якщо не реагує – обережно потрясти за плече.

Перевірити дихання: відновити прохідність дихальних шляхів, запрокинувши голову назад та піднявши підборіддя. Визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви чи є дихання, або воно ненормальне, вважати, що дихання відсутнє; за нормального дихання при відсутності свідомості слід перевести у стабільне бічне положення, здійснити виклик екстреної медичної допомоги та перевіряти кожні 3-5 хвилин дихання до моменту приїзду бригади екстреної (швидкої) медичної допомоги; якщо дихання відсутнє – перейти до серцево-легеневої реанімації.

4.2 Розрахунок екранів для захисту від електромагнітних випромінювань

Перш ніж дати відповідь на питання, як захиститись від електромагнітного випромінювання, згадаємо, що воно собою являє. Електромагнітне поле, окремим випадком якого є електромагнітне випромінювання, можна вважати особливою формою матерії, за допомогою якої здійснюється взаємодія між зарядженими частинками. Це взаємопов'язані між собою змінне електричне і магнітне поля. Цей взаємний зв'язок полягає в тому, що будь-яка зміна одного призводить до появи іншого: змінне електричне поле, яке індукується рухомими зарядами (джерелом поля), збуджує в суміжних областях простору змінне магнітне поле, яке, в свою чергу, збуджує в прилеглих до нього областях простору змінне електричне поле, і т. д. Таким чином, ЕМП поширюється від точки до точки простору у вигляді електромагнітних хвиль (випромінювань). Через скінченну швидкість поширення, що залежить від властивостей середовища, ЕМП може існувати автономно від джерела випромінювання і не зникає з усуненням джерела. ЕМВ являє собою змінне в часі ЕМП, що розповсюджується в просторі з кінцевою швидкістю.

Всі штучно згенеровані електричні, магнітні та електромагнітні поля складають електромагнітне забруднення навколишнього середовища, для якого віднедавна є спеціальний термін – електросмог . Невидимий, безшумний, без запаху, проте всюдисущий. Його джерелами є всі види електроприладів, електропроводка, причому, чим потужніший прилад, тим більш агресивне поле. Піднесіть мобільний телефон до працюючого радіо – почнуться збої, адже одні поля заважають іншим. Фактично те ж саме відбувається, якщо людина потрапляє в зону підвищеного рівня електромагнітного випромінювання – відбуваються збої в функціонуванні всіх органів і систем людського тіла. Негативного впливу зазнає нервова, ендокринна, імунна, сечостатева системи організму. Змінюється частота серцевих скорочень та артеріальний тиск.

Вплив зовнішнього електромагнітного поля на організм в цілому починається на молекулярному та клітинному рівні. Відбуваються коливання та рух заряджених частинок: електронів, протонів, йонів та молекул-диполів. Важко назвати молекули в клітині живого організму, що не мали б часткової поляризації на тих чи інших своїх

ділянках: це і білки, і фосфоліпіди (мають заряджені головки та нейтральні хвости), молекула ДНК в процесі свого функціонування може мати нескомпенсований заряд на різних своїх ділянках. Навіть найпростіша молекула води є диполем. А отже впливу зазнають всі біохімічні процеси в клітині.

Проте не можна сказати, що електромагнітне поле це завжди зло. Відомо, що його використовують у фізіотерапії для лікування багатьох захворювань, для загоєння ран, воно має протизапальний ефект, покращує клітинний метаболізм. Однак це тільки у разі строгого контролю з боку лікарів, у вибраних частотних діапазонах та в певних інтервалах інтенсивності електромагнітного поля.

На побутовому рівні найбільш небезпечними є мікрохвильові пічки, холодильники з системою No Frost, електроплити та мобільні телефони. Низькочастотне випромінювання від електропроводки вважається відносно безпечним, а за умови використання трижильної системи з заземленням такий вплив зводиться до мінімуму. А от високочастотне поле від телекомунікаційних систем, радарів, високовольтних ліній електропередач вважається найбільш небезпечним. Можна дискутувати про шкідливість чи безпечність впливу електромагнітного випромінювання, але в більшості Європейських країн, в яких турбуються про здоров’я свої громадян, вже запроваджено заборону встановлювати Wi-Fi системи в школах, громадських місцях, бібліотеках, лікарнях. Натомість пропонується повернутись до провідних пристроїв для підключення до інтернету.

Негативний вплив електромагнітного випромінювання. Хто в групі ризику. Зважаючи на всеохоплююче поширення телекомунікаційних мереж, мобільного зв’язку, різноманітних технічних засобів у всіх їх проявах, у групі ризику знаходиться 99,9 % населення Земної кулі. Тому зрозуміло, що питання захисту від електромагнітного випромінювання стає життєво важливим. Хто ж в групі особливого ризику?

Це звичайно ж вагітні жінки і маленькі діти. Науково доведено, що саме на стадії розвитку і інтенсивного росту організм є найбільш вразливим. Сприятливу

атмосферу вдома, необхідну для відновлення сил та повноцінного відпочинку, забезпечать екрануючі фарби та екрануючі тканини. А поза межами дому у пригоді стане екрануючий одяг. Подбавши про захист від електромагнітного випромінювання своїх дітей зараз, Ви запобігаєте багатьом небезпекам, що пов'язані зі здоров'ям, у майбутньому.

Друга група особливого ризику – це люди, професійні обов'язки яких передбачають тривалий контакт з комп'ютерною технікою та електромагнітним випромінюванням (програмісти, системні адміністратори, оператори ПК, монтажники радіоелектронного обладнання і т. д.). Організовуйте своє робоче місце, скориставшись екрануючими матеріалами, запропонованими компанією SIMVOLT, а при роботі на відкритому просторі індивідуальні засоби захисту (екрануючий одяг) будуть просто необхідними.

Ще одну групу ризику складають люди – жителі районів навколо важливих державних та промислових об'єктів, де зосереджена найбільша кількість засобів радіолокації, а також жителі населених пунктів розміщених у безпосередній близькості від аеродромів (цивільних, військових) та інших військових об'єктів. Екрануючі фарби та екрануючі тканини повинні стати невід'ємною частиною вашого інтер'єру, а екрануючий одяг обов'язково повинен бути присутній у гардеробі.

Як захиститись від електромагнітного випромінювання. В загальному, методи і засоби захисту від електромагнітних полів можна умовно розділити на інженерно-технічні, організаційні та лікувально-профілактичні. Згідно зі встановленою процедурою, захист людини від такого небезпечного впливу повинен здійснюється такими способами: зменшення випромінювання від джерела; екранування джерела випромінювання та робочого місця; встановлення санітарно-захисної зони; поглинання або зменшення утворення зарядів статичної електрики; усунення зарядів статичної електрики; підтримання оптимальної відносної вологості (не нижче 60%),

іонного складу повітря робочих приміщень застосування засобів індивідуального захисту.

Отже, перший крок у цьому напрямку можна зробити не вдаючись до застосування спеціальних засобів захисту від електромагнітних полів та екрануючих матеріалів. Оптимально розмістіть електроприлади вдома, збільшуйте відстань від електроприладу до спального місця, видаліть штучні матеріали, зменшити навантаження, виконуйте елементарні правила техніки безпеки користування електроприладами, мобільними телефонами, засобами зв'язку. Однак цього не завжди достатньо. Якщо ж ви в групі ризику, якщо відчуваєте дискомфорт, пов'язаний з надмірним впливом ЕМВ на робочому місці чи вдома – застосування екрануючих матеріалів є життєво необхідним.

Перш ніж застосовувати будь-які засоби захисту від електромагнітних полів чи екрануючі матеріали, необхідно спершу провести дослідження використавши спеціальне обладнання. Якщо Вам самим це зробити важко з огляду на ряд причин, зверніться до спеціалістів.

Вплив такого фізичного явище як електромагнітне забруднення з усіма його різними специфічними проявами можна успішно зменшувати, скориставшись достовірно ефективними фізичними рішеннями та ефективними екрануючими матеріалами. Ми не пропонуємо такі «неоднозначні» продукти, як екрануючі чіпи для мобільних телефонів, хвильові гармонізатори і т. д. Всі екрануючі матеріали є результатом тривалих і кропітких наукових досліджень та мають під собою строгу фізичну основу.

ВИСНОВКИ

По результат. викон. роботи одерж. наступні висновки:

1. Спроектов. та сконструйов. прог. систему для підтримки прийняття рішень банківським установам щодо надання чи не надання кредитів клієнтам банку і цим самим дає змогу зменшити ризики виникнення ситуацій не погашення кредитів клієнтами, а також зменшити затрати людино-годин на обслуговування процесів надання кредитів.

2. Спроектовано та сконструйовано процеси гіпероптимізації програмного забезпечення, що покращило метрологічні характеристики системи, накшталт, accuracy, f1-score, roc_auc-score, а також робастність функціонцвання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yasniy O., Pastukh O., Didych I., Yatsyshyn V., Chykhira I. Application of machine learning for modeling of 6061-T651 aluminum alloy stress− strain diagram // Procedia Structural Integrity. 2023. №48. С. 183-189.

2. Пастух О.А., Яцишин В.В., Жаровський Р.О., Шаблій Н.Р. Software tool for productivity metrics measure of relational database management system // Математичне моделювання. 2023. №1 (48). С.7-17.

3. Pastukh O., Stefanyshyn V., Baran I., Yakymenko I., Vasylkiv V. Mathematics and software for controlling mobile software devices based on brain activity signals // The International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2023). 2023. №3628. С. 684–689.

4. Pastukh O., Petryk M., Bachynskiy M., Mudryk I., Stefanyshyn V. Processing of Cerebral Cortex Neurosignals from EEG Sensors and RecognizinSpecific Types of Mechanical Movements Elements of Pacient Limbs under the Cognitive Feedback Influenses // International Workshop on Computer Information Technologies in Industry 4.0. 2023. №3468. С. 61–70.

5. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. Technology of relational database management systems performance evaluation during computer systems design // Scientific Journal of TNTU. 2023. №1. С. 54–65.

6. Pastukh O., Yatsyshyn V. Brain-computer interaction neurointerface based on artificial intelligence and its parallel programming using high-performance calculation on cluster mobile devices // Scientific Journal of TNTU. 2023. №4. С. 26-31.

7. Pastukh O., Petryk M., Bachynskiy M., Mudryk I., Stefanyshyn V. Processing of Cerebral Cortex Neurosignals from EEG Sensors and RecognizinSpecific Types of Mechanical Movements Elements of Pacient Limbs under the Cognitive Feedback Influenses // International Workshop on Computer Information Technologies in Industry 4.0. 2023. №3468. С. 61–70.

8. Pandas documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 07.03.2024)

9. NumPy documentation. URL: https://numpy.org/devdocs/ (дата звернення: 07.03.2024)

10. Matplotlib 3.8.3 documentation. URL: https://matplotlib.org/stable/index.html (дата звернення: 07.03.2024)

11. Мелех Л.В. Безпека життєдіяльності та охорона праці : Навчальний посібник. Львів: Львівський державний університет внутрішніх справ. 2022. 219 с.

12. Сокуренко В.В. Безпека життєдіяльності та охорона праці : Підручник. Харків: Харків. нац. ун-т внутр. справ. 2021. 308 с.

ДОДАТКИ

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПІ
_____ Петрик М.Р.
"___" _____ 2024 р

РОЗРОБКА СИСТЕМИ ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ
ЩОДО НАДАННЯ КРЕДИТІВ

ТЕХНІЧНЕ ЗАВДАННЯ

Вид робіт: Кваліфікаційна робота
на здобуття освітнього ступеня "Бакалавр"
спеціальність 121 "Інженерія програмного забезпесення"

"УЗГОДЖЕНО"
керівник кваліфікаційної роботи
_____ д.т.н., проф. Пастух О.А.
"____" _____ 2024 р.

"ВИКОНАВЕЦ"
студент групи СП-41
_____ Бойко Д.А.
"____" _____ 2024 р.

Тернопіль 2024

ЗМІСТ

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка програмної системи виконується у відповідності до графіку навчального плану на 2023 – 2024 н.р. та згідно наказу ректора Тернопільського національного технічного університету імені Івана Пулюя на виконання кваліфікаційної роботи на здобуття освітнього ступеня бакалавр.

Тема кваліфікаційної роботи на здобуття освітнього ступеня бакалавр: ”Розробка системи для підтримки прийняття рішень щодо надання кредитів” для аналітичних відділів банків для ідентифікації клієнтів для надання кредитів та зменшення ризиків виникнення ситуацій їх не погашення.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмна система – ”Розробка системи для підтримки прийняття рішень щодо надання кредитів” дає змогу оптимізувати організацію роботи, здешевити затрати, зменшини ризики втрати фінансів аналітичних відділів банків. Фактично такого роду програмне забезпечення дає змогу оптимізувати крединтно-скорингові програми, визначити основні ознаки, що впливають на можливість погашення кредитів клієнтами банків. Програмне забезпечення дає змогу швидше, тобто реактивніше організувати роботу працівникам банку щодо надання кредитів, тобто кредитні програми ефективніше розробляються, оскільки інтелектуальний аналіз даних, що грунтується на переробці ретроданих клієнтських історій дає змогу більш точно виконувати такого роду операції. Одним із головних параметрів який при цьому досягається і ж крайнє важливим для самого інституційного банку – це зменшення ризиків щодо не повернення клієнтами коштів, тобто не погашення кредитів клієнтами. Також таке програмне забезпечення дає змгогу банківським

інституціям прогнозувати у часі якість вирішення ними інших завдань із портфеля фінансових інструментів.

Розроблювальна програмна система повинна володіти високими занченнями метрологічних характеристик, накшталт, точність, f1 метрика точності, roc_auc метрика точності. Окрім цього, система повинна володіти високою стійкістю роботи, тобто видавати результат не залежно від сторонніх шумових факторів.

3. ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ

3.1. Функціональні характеристики

- формувати та візуалізувати поточні дані кредитування
- формувати та візуалізувати поточні дані кредитування
- оцінювати точність одержаного прогнозного результату кредитування
- прогнозувати майбутні значення погашення чи не погашення кредиту
- видавати метрологічні характеристики роботи системи

3.2. Склад та параметри технічних засобів

1) персональний комп’ютер з 1024 Мб оперативної пам‘яті, встановленою системою Windows 8, 8.1, 10. Не менше 200 Мб вільного місця на жорсткому диску. Двоядерний процесор із двома потоками та тактовою частотою від 1.2 GHz і більше.

2) наявність встановленого дистрибутиву Анаконди та ядра мови програмування Python із біліотеками Collection, Pandas, Scikit-learn, Numpy.

3.3. Інформаційна та програмна сполучність

Програмна система повинна коректно коректно функціонувати в операційній системі Windows різних поколінь. Програмні модулі програмної системи повинні працювати з високою степінню надійності та точності. Розробку виконувати з використанням бібліотек та технологій мови Python в середовищі програмування Spyder.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи програмна система повинна пройти крізь стадії життєвого циклу розробки програмного забезпечення:

- аналіз предметної області
- проектування архітектури програмної системи
- практичне конструювання програмної системи
- реалізація певних елементів візуалізації даних
- тестування якості роботи окремих програмних модулів програмної системи
- оформлення супровідної документації
- здача роботи

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для розробленої системи для підтримки прийняття рішень щодо надання кредитів повинні бути розроблені наступні документи:

- пояснювальна записка

- технічне завдання

- презентаційний матеріал

- додатки

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблена програмна система має виконувати всі вимоги, що складаються з перерахованих у даному технічному завданні.

Приймання проводиться спеціально створеною екзаменаційною комісією у відведені терміни, що вказані у календарному плані завдання кваліфікаційної роботи на здобуття освітнього ступеня бакалавр.

Додаток Б

Тези доповіді на конференції

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2024 р.

(збірник тез конференції)

Тернопіль 2024

УДК 629.735.072.8.08:004(043.2)
Бойко Д. – ст. гр. СП-41, Пастух О.
Тернопільський національний технічних університет імені Івана Пулюя

ПРОГРАМА ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ ЩОДО КРЕДИТУВАННЯ

Науковий керівник: д.т.н., професор Пастух О.А.

Boyko D., Pastukh O.
Ternopil Ivan Puluj National Technical University

PROGRAM FOR MAKING DECISIONS REGARDING LENDING

Supervisor: Dr.Sc., Professor Pastukh O.A.

Однією із сфер застосування інтелектуалізованих алгоритмів є фінансова інженерія. Практично важливим завданням фінансової інженерії є завдання зменшення ризиків при кредитуванні клієнтів. Тому для ефективного вирішення даного завдання використовуються інформаційні технології із програмним забезпеченням, що використовує інтелектуалізовані алгоритми. Створення програмного забезпечення виконується на послідовних кроках: перший – це його проектування, другий – це практичне конструювання. Нище на рис.1. наведено спроектовану архітектуру програмного забезпечення.

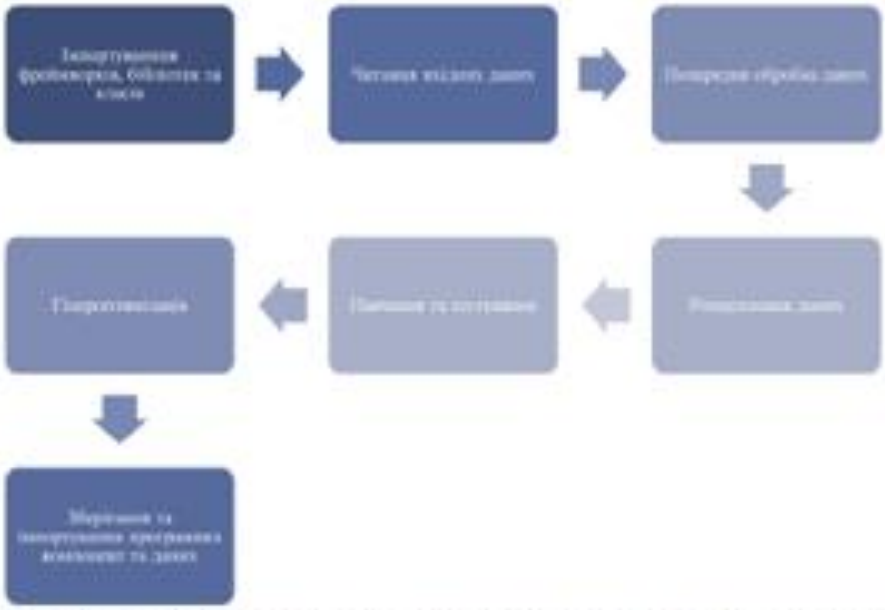

Рис.1 – Блок-схемне проектування архітектури підтримки прийняття рішень щодо надання кредитів

До основних блоків, що наведені на рис.1 відносяться блоки попередньої обробки даних, блок навчання алгоритму машини, блок гіпероптимізації та блок зберігання програмних компонент, які попередньо навчені.