**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №8**

**з навчальної дисципліни «Технології Data Science»**

**Тема:**

**МАКЕТ CRM СИСТЕМИ SCORING – АНАЛІЗУ**

**Виконав:**

Студент 4 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

Дослідити, виявити та узагальнити особливості реалізації проектного практикуму в галузі кредитного scoring-у.

## **ІІ. Завдання:**

Відділ мікрокредитування банківської установи замовив розробку Backend компоненту SRM банківської системи.

Компонента Backend повинна забезпечити:

1. Побудову скорингової оцінки;

2. Кластеризацію заявників за бінарної оцінкою.

Вихідні дані для опрацювання задачі представлені у формі 2-х файлів:

1. sample\_data.xlsx – файл реальних даних про позичальників та проміжних параметрів банківських індикаторів;

2. data\_description.xlsx – файл пояснень структури скорингової таблиці та тлумачення індикаторів,

Розробити програмний скрипт, що реалізує функціонал за обраним рівням складності:

**ІІІ рівень складності 10 балів.**

Розробити програмний скрипт, що реалізує: скоринговий аналіз позичальників за даними скорингової карти; виявлення шахрайства та фальсифікації даних – за методиками/алгоритмами, що є R&D – новацією.

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичну модель, що реалізує скоринговий аналіз позичальників, кластеризує позичальників за бінарної ознакою та виявляє шахрайство та фальсифікацію даних.

Скоринговий аналіз позичальників за даними скорингової карти реалізувається наступним чином. Спочатку дані нормалізуються з допомогою масштабування за формулою:

де – вихідне значення, – значення стовпця, – середнє значення стовпця, – стандартне відхилення по стовпцю.

Так як у таблиці багато стовпців, то їх кількість можна скоротити за допомогою PCA (Principal Component Analysis). Це метод зниження розмірності даних, який перетворює їх у новий простір меншої кількості змінних (головний компонент). PCA визначає нові ортогональні осі (компоненти), де перша головна компонента захоплює максимальну дисперсію даних, друга – наступну найбільшу дисперсію і так далі. Таким чином із 31 ознаки отримаємо 3, які пояснюватимуть більшу частину варіативності.

Кластеризація виконуватиметься на два кластери, або ж ризикові групи, де (перша отримає кредит, друга – ні) з використанням алгоритму k-means.

Виявлення шахрайства та фальсифікації даних відбувається з використанням Isolation Forest. Це алгоритм машинного навчання для виявлення аномалій. Його основна ідея полягає у тому, що аномальні точки легше ізолювати, ніж нормальні. Алгоритм створює набір дерев, де на кожному кроці обирається одна із ознак та поріг для поділу даних. Аномалії знаходитимуться ближче до кореня дерева, адже вони ізолюються за меншу кількість поділів. Це алгоритм навчання без вчителя, тому він чудово підходить для вирішення цієї задачі.

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до програмного скрипту:



Рисунок 1 – Блок схема алгоритму

Робота алгоритму розпочинається зі зчитування даних із файлів та перевірки їх структури. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 відбувається формування скорингової таблиці. Сюди входить очищення даних та видалення неважливих стовпців.

У блоці 3 виконується підготовка даних до створення скорингової моделі. Замість стовпця birth\_date формується стовпець age. Усі дані масштабуються з допомогою StandartScaler.

У блоці 4 визначаються три PCA компоненти.

У блоці 5 виконується кластеризація з поділом на два кластери (ризикові групи: перша отримає кредит, друга – ні) з допомогою методу k-means.

У блоці 6 відбувається виявлення шахрайства та фальсифікації даних з допомогою алгоритму Isolation Forest.

У блоці 7 візуалізуються та зберігаються результати. На цьому виконання алгоритму завершене.

* 1. **Опис структури проекту програми**

Для реалізації розроблених алгоритмів мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

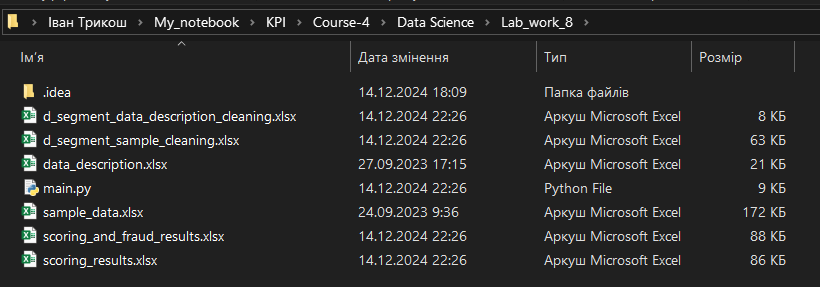


Рисунок 2 – Структура проєкту

Lab\_work\_8 – головний каталог проєкту

d\_segment\_data\_description\_cleaning.xlsx – очищені дані з описом індикаторів

d\_segment\_sample\_cleaning.xlsx – очищена скорингова таблиця

data\_description.xlsx – дані про індикатори у sample\_data.xlsx

main.py – файл програмного коду

sample\_data.xlsx – початкові дані

scoring\_and\_fraud\_results.xlsx – результати скорингу та виявлення шахрайства

scoring\_results.xlsx – результати скорингу

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. На рисунках 3-17 зображено результати роботи алгоритму:

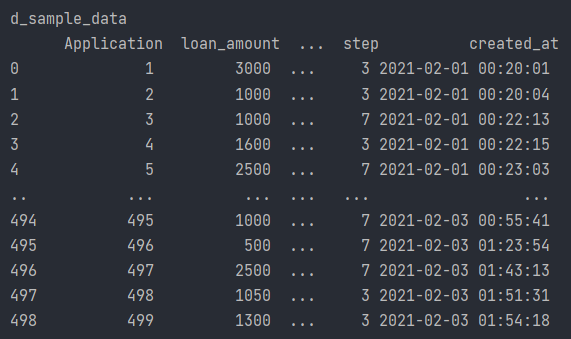


Рисунок 3 – Початкові дані

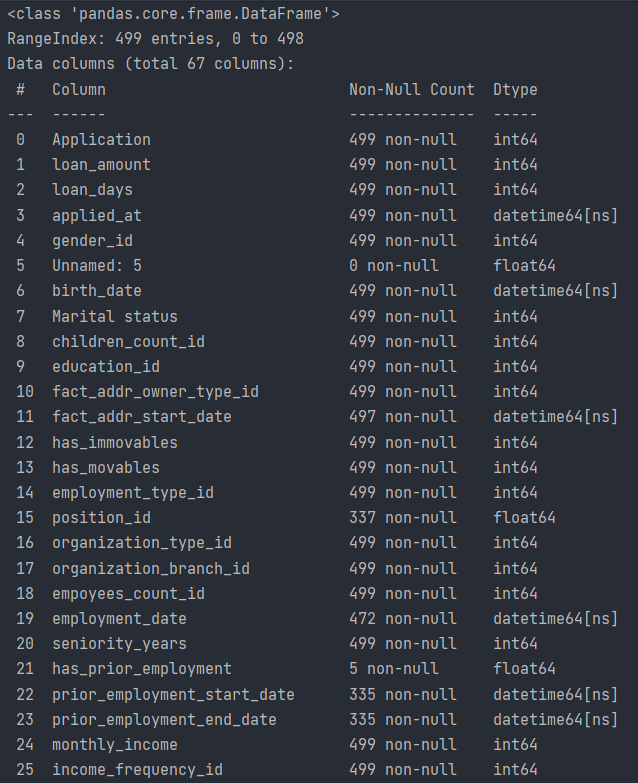


Рисунок 4 – Інформація про датафрейм (фрагмент)

Бачимо, що є доволі багато пропущених значень по певним стовпцям.

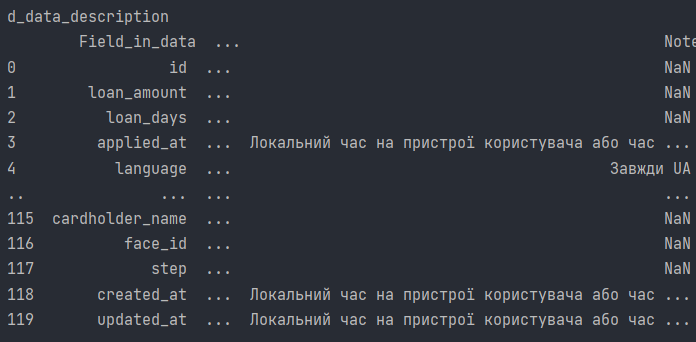


Рисунок 5 – Опис індикаторів

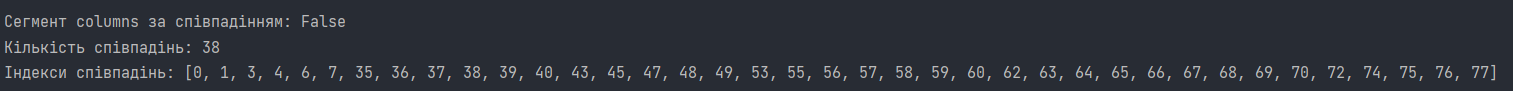


Рисунок 6 – Співпадіння між датафреймами по стовпцях

Тут помічаємо, що певні стовпці у цих таблицях відрізняються.

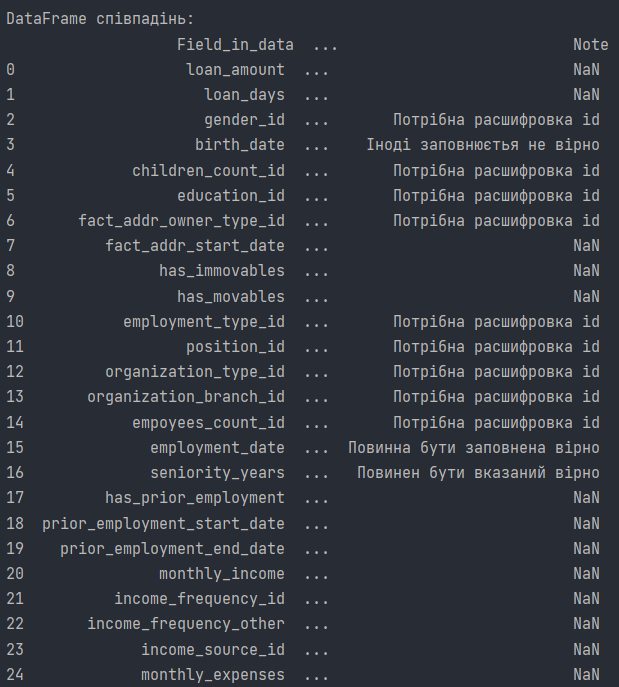


Рисунок 7 – Стовпці, які співпали (фрагмент)

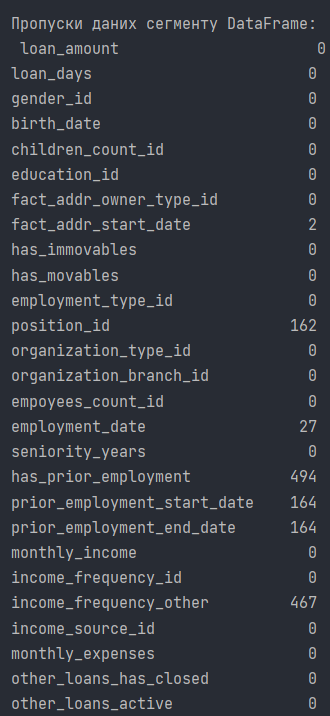


Рисунок 8 – Пропущені дані по стовпцях (фрагмент)

Бачимо, у деяких стовпцях дуже багато пропущених значень, тому їх можна видалити.



Рисунок 9 – Пропущені значення по стовпцях

Тепер пропущених значень немає і тому можна продовжити роботу.

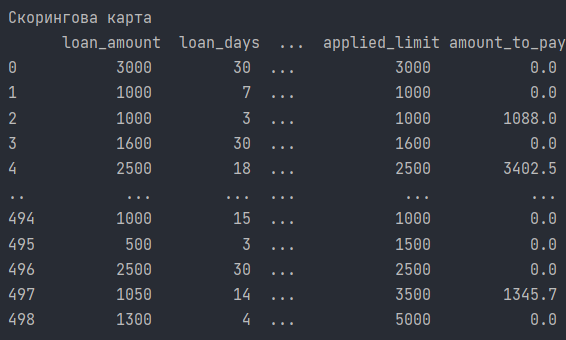


Рисунок 10 – Скорингова карта

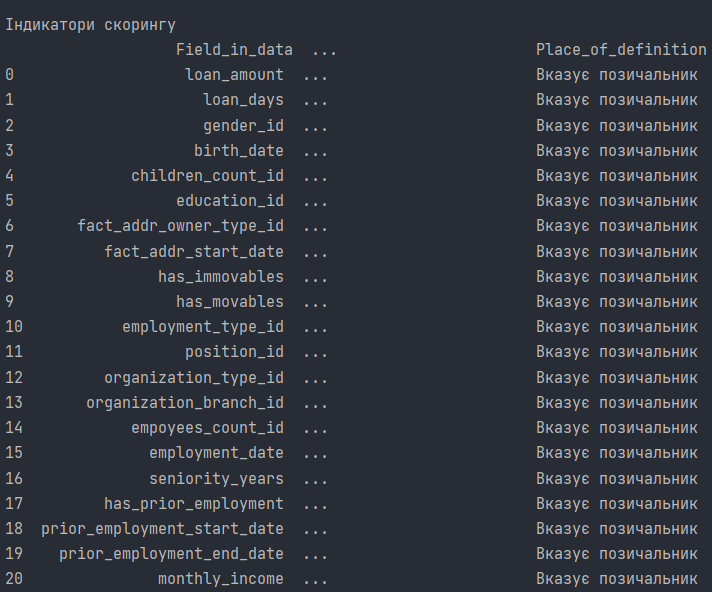


Рисунок 11 – Індикатори скорингу (фрагмент)

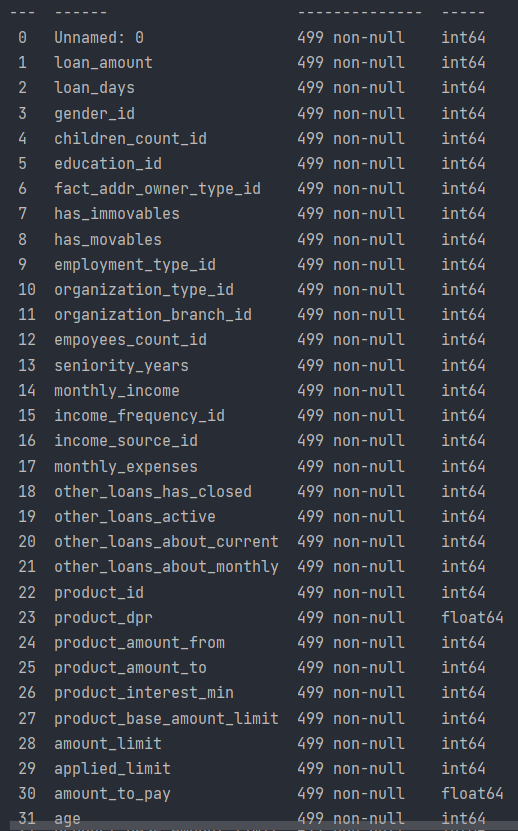


Рисунок 12 – Інформація про підготовлені дані (додався стовпець age)

Визначимо три компоненти, щоб зменшити кількість даних:



Рисунок 13 – Варіація по компонентах PCA

Бачимо, що кожна компонента пояснила все менше варіацій даних.

Виконаємо кластеризацію, де кластер – це група ризику. Кластерів буде лише два, тому перший кластер містить людей, які отримають кредит, а другий – людей, які не отримають кредит:

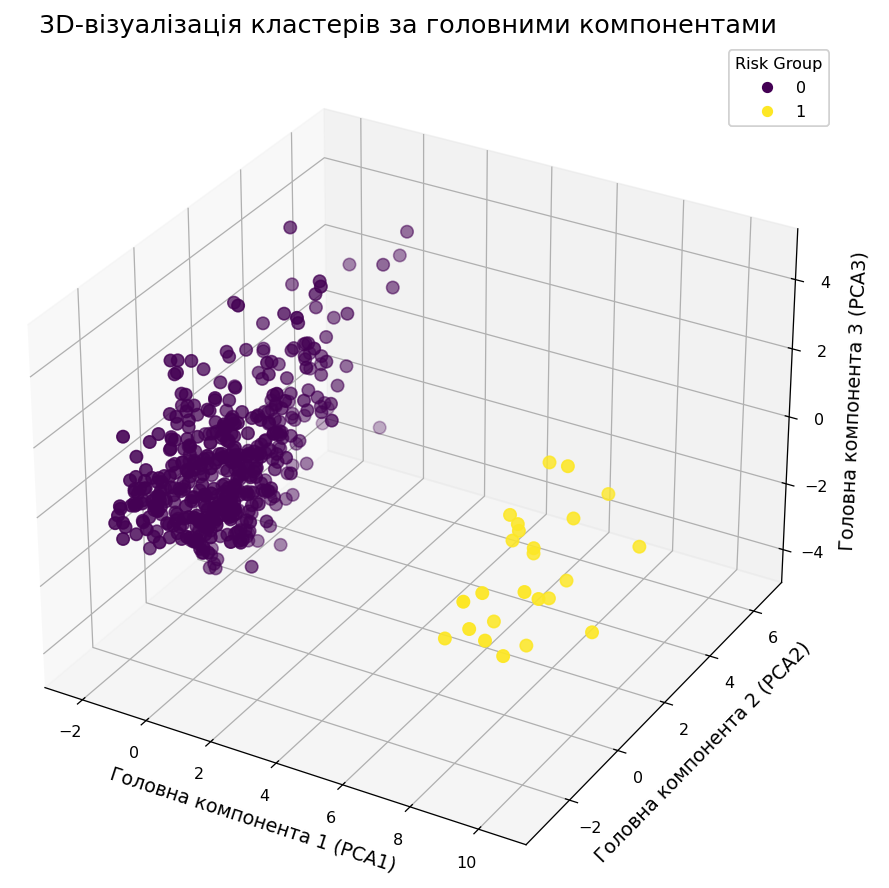


Рисунок 14 – Кластеризація даних

Бачимо, що переважна більшість людей отримають кредит. Тепер виведемо середні значення по кластеру для певних стовпців:

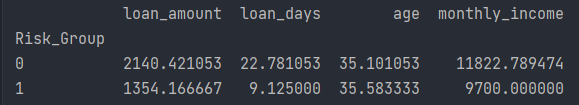


Рисунок 15 – Середні значення по кластерах

Бачимо, що люди з більшим доходом, часом для погашення та розміром ймовірно отримають кредит.

Тепер виконаємо виявлення шахрайства з допомогою Isolation Forest:

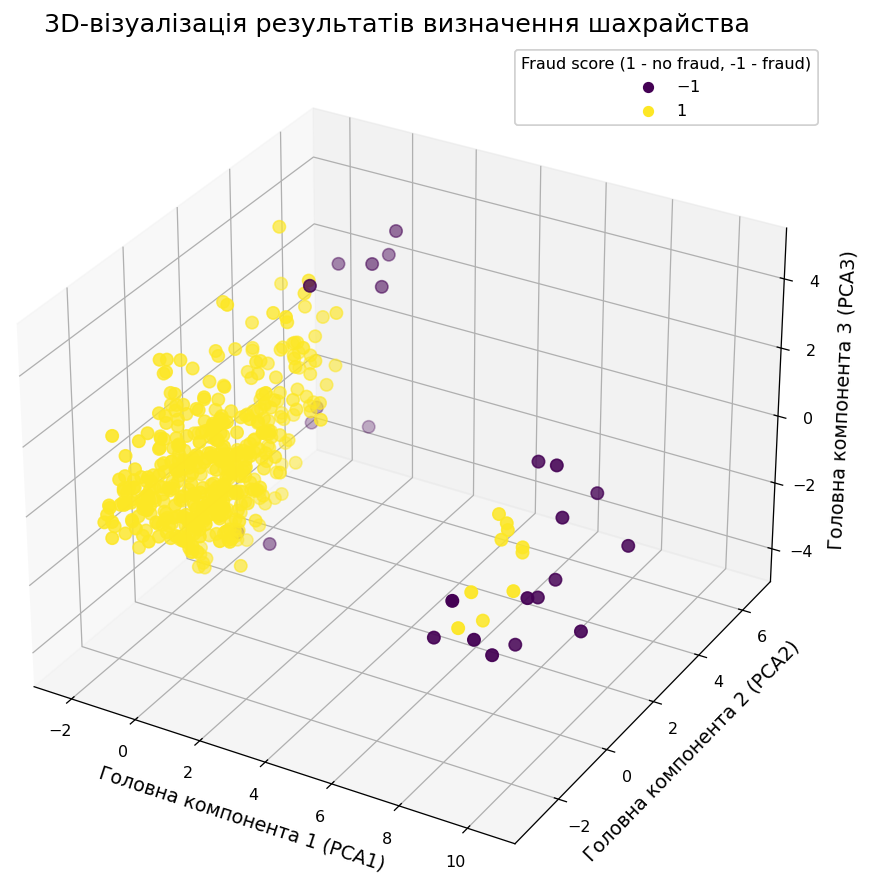


Рисунок 16 – Візуалізація результатів визначення шахрайства

Бачимо, що переважна кількість людей з другого кластера – шахраї.

Тепер визначимо кількість шахраїв (1 – не шахрайство, -1 – шахрайство):

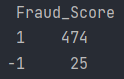


Рисунок 17 – Кількість шахраїв

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритм на рис.1 та спрямований на отримання результатів, поданих на рис.3-17.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: datetime, pandas, matplotlib, sklearn.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

main.py

*from* datetime *import* datetime  
*import* pandas *as* pd  
*import* matplotlib.pyplot *as* plt  
*from* sklearn.ensemble *import* IsolationForest  
*from* sklearn.preprocessing *import* StandardScaler  
*from* sklearn.decomposition *import* PCA  
*from* sklearn.cluster *import* KMeans  
  
*# ----------------------------------- ПІДГОТОВКА ВХІДНИХ ДАНИХ -----------------------------------  
# Зчитування файлу даних*d\_sample\_data = pd.read\_excel('sample\_data.xlsx', parse\_dates=['birth\_date'])  
print('\nd\_sample\_data\n', d\_sample\_data)  
title\_d\_sample\_data = d\_sample\_data.columns  
  
*# Аналіз структури вхідних даних*print('\nInfo:')  
d\_sample\_data.info()  
  
*# Зчитування файлу з інформацією про стовпці даних*d\_data\_description = pd.read\_excel('data\_description.xlsx')  
print('\nd\_data\_description\n', d\_data\_description)  
  
*# Первинне формування скорингової таблиці*d\_segment\_data\_description\_client\_bank = d\_data\_description[  
 (d\_data\_description.Place\_of\_definition == 'Вказує позичальник')  
 | (d\_data\_description.Place\_of\_definition == 'параметри, повязані з виданим продуктом')  
]  
d\_segment\_data\_description\_client\_bank.index = range(0, len(d\_segment\_data\_description\_client\_bank))  
print('\nd\_segment\_data\_description\_client\_bank\n', d\_segment\_data\_description\_client\_bank)  
  
*# Очищення даних  
# Аналіз перетину скорингових індикаторів та сегменту вхідних даних*b = d\_segment\_data\_description\_client\_bank['Field\_in\_data']  
flag\_b = set(b).issubset(d\_sample\_data.columns)  
print('\nСегмент columns за співпадінням:', flag\_b)  
  
matches = [field *for* field *in* b *if* field *in* d\_sample\_data.columns]  
num\_matches = len(matches)  
print('Кількість співпадінь:', num\_matches)  
  
indices\_matches = [i *for* i, field *in* enumerate(b) *if* field *in* d\_sample\_data.columns]  
print('Індекси співпадінь:', indices\_matches)  
  
*# Формування DataFrame даних з урахуванням відсутніх індикаторів скорингової таблиці*d\_segment\_data\_description\_client\_bank\_True = d\_segment\_data\_description\_client\_bank.iloc[indices\_matches]  
d\_segment\_data\_description\_client\_bank\_True.index = range(0, len(d\_segment\_data\_description\_client\_bank\_True))  
print('\nDataFrame співпадінь:\n', d\_segment\_data\_description\_client\_bank\_True)  
  
*# Очищення скорингової таблиці від пропусків (видалення стовпців з пропусками)*b = d\_segment\_data\_description\_client\_bank\_True['Field\_in\_data']  
d\_segment\_sample\_data\_client\_bank = d\_sample\_data[b]  
print('\nПропуски даних сегменту DataFrame:\n', d\_segment\_sample\_data\_client\_bank.isnull().sum())  
  
d\_segment\_data\_description\_cleaning = d\_segment\_data\_description\_client\_bank\_True.dropna(axis=1)  
d\_segment\_data\_description\_cleaning.index = range(0, len(d\_segment\_data\_description\_cleaning))  
d\_segment\_data\_description\_cleaning.to\_excel('d\_segment\_data\_description\_cleaning.xlsx')  
  
*# Очищення вхідних даних та збереження скорингової таблиці*d\_segment\_sample\_cleaning = d\_segment\_sample\_data\_client\_bank.drop(  
 columns=['fact\_addr\_start\_date', 'position\_id', 'employment\_date',  
 'has\_prior\_employment', 'prior\_employment\_start\_date',  
 'prior\_employment\_end\_date', 'income\_frequency\_other'])  
d\_segment\_sample\_cleaning.index = range(0, len(d\_segment\_sample\_cleaning))  
d\_segment\_sample\_cleaning.to\_excel('d\_segment\_sample\_cleaning.xlsx')  
print('\nПеревірка наявності пропусків у даних:\n', d\_segment\_sample\_cleaning.isnull().sum())  
print('\nСкорингова карта\n', d\_segment\_sample\_cleaning)  
print('\nІндикатори скорингу\n', d\_segment\_data\_description\_cleaning)  
  
  
*# ----------------------------------- ФОРМУВАННЯ СКОРИНГОВОЇ МОДЕЛІ -----------------------------------  
# Завантаження даних*data = pd.read\_excel('d\_segment\_sample\_cleaning.xlsx', sheet\_name='Sheet1')  
  
*# Підготовка даних. Формування стовпця age*today = datetime.now()  
data['age'] = data['birth\_date'].apply(  
 *lambda* date: today.year - date.year - ((today.month, today.day) < (date.month, date.day))  
)  
data.drop(columns=['birth\_date'], inplace=*True*)  
print('\nData info:')  
data.info()  
  
X = data  
  
*# Масштабування даних*scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
*# Визначення динамічних ваг (PCA), або ж компонентів*pca = PCA(n\_components=3)  
X\_pca = pca.fit\_transform(X\_scaled)  
data['PCA1'], data['PCA2'], data['PCA3'] = X\_pca[:, 0], X\_pca[:, 1], X\_pca[:, 2]  
  
*# Доля варіації, пояснена кожною компонентою*explained\_variance\_ratio = pca.explained\_variance\_ratio\_  
print('\nДоля варіації, пояснена кожною компонентою:\n', explained\_variance\_ratio)  
  
*# Кластеризація для ризикових груп (буде дві групи ризику - одній буде надано кредит, а іншій - ні)*kmeans = KMeans(n\_clusters=2, random\_state=42)  
data['Risk\_Group'] = kmeans.fit\_predict(X\_pca)  
  
*# Збереження результатів*data.to\_excel('scoring\_results.xlsx', index=*False*)  
  
*# Аналіз середніх значень деяких характеристик для кожної групи ризику*group\_analysis = data.groupby('Risk\_Group')[['loan\_amount', 'loan\_days', 'age', 'monthly\_income']].mean()  
print(group\_analysis)  
  
*# Візуалізація кластеризації  
# Створення 3D-графіка*fig = plt.figure(figsize=(12, 8))  
ax = fig.add\_subplot(111, projection='3d')  
  
*# Розподіл точок за трьома компонентами*scatter = ax.scatter(data['PCA1'], data['PCA2'], data['PCA3'], c=data['Risk\_Group'], cmap='viridis', s=60)  
  
*# Додавання легенди*legend1 = ax.legend(\*scatter.legend\_elements(), title='Risk Group', fontsize=10)  
ax.add\_artist(legend1)  
  
*# Налаштування осей*ax.set\_title('3D-візуалізація кластерів за головними компонентами', fontsize=16)  
ax.set\_xlabel('Головна компонента 1 (PCA1)', fontsize=12)  
ax.set\_ylabel('Головна компонента 2 (PCA2)', fontsize=12)  
ax.set\_zlabel('Головна компонента 3 (PCA3)', fontsize=12)  
  
*# Показ графіка*plt.tight\_layout()  
plt.show()  
  
*# Виявлення шахрайства з допомогою Isolation Forest  
# Ініціалізація Isolation Forest*iso\_forest = IsolationForest(n\_estimators=50, contamination=0.05, random\_state=42)  
data['Fraud\_Score'] = iso\_forest.fit\_predict(X\_pca)  
  
*# Аналіз результатів (1 - шайхраство не виявлено, -1 - шахрайство виявлено)*print('\n', data['Fraud\_Score'].value\_counts())  
*# Збереження результатів*data.to\_excel('scoring\_and\_fraud\_results.xlsx', index=*False*)  
  
*# Візуалізація результатів  
# Створення 3D-графіка*fig = plt.figure(figsize=(12, 8))  
ax = fig.add\_subplot(111, projection='3d')  
  
*# Розподіл точок за трьома компонентами*scatter = ax.scatter(data['PCA1'], data['PCA2'], data['PCA3'], c=data['Fraud\_Score'], cmap='viridis', s=60)  
  
*# Додавання легенди*legend1 = ax.legend(\*scatter.legend\_elements(), title='Fraud score (1 - no fraud, -1 - fraud)', fontsize=10)  
ax.add\_artist(legend1)  
  
*# Налаштування осей*ax.set\_title('3D-візуалізація результатів визначення шахрайства', fontsize=16)  
ax.set\_xlabel('Головна компонента 1 (PCA1)', fontsize=12)  
ax.set\_ylabel('Головна компонента 2 (PCA2)', fontsize=12)  
ax.set\_zlabel('Головна компонента 3 (PCA3)', fontsize=12)  
  
*# Показ графіка*plt.tight\_layout()  
plt.show()

## **IV. Висновки.**

У ході виконання лабораторної роботи було досліджено, виявлено та узагальнено особливості реалізації проектного практикуму в галузі кредитного scoring-у. Було побудовано скорингову модель, кластеризовано дані та визначено шахрайство.

Виконав: студент Трикош І. В.