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 «Вступ до технології Data Science»

Тема:

МАКЕТ CRM СИСТЕМИ SCORING – АНАЛІЗУ
(міні проекти в банківській сфері аналізу даних)

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-11
Сідак К.І.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2023

І. Мета роботи:

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 скорінгової оцінку;
2. Кластерізацію 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:

ІІ рівень складності 9 балів.

Відповідно до технічних умов, табл.2 додатку.

12	Розробити програмний скрипт, що реалізує: 1. Скоринговий аналіз позичальників за даними Data_description.xlsx, Sample_data.xlsx відповідно до моделі дерева прийняття рішень. 2. Передбачити чи буде кредит повернено у форматі бінарної оцінки (0 або 1). 3. Виявлення шахрайства та фальсифікації даних.

ІІІ. Виконання лабораторної роботи.

Спочатку я імпортував усі необхідні бібліотеки для подальшої роботи.

```
In 1
import pandas as pd
from datetime import datetime
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
Executed at 2024.01.02 19:12:25 in 846ms
```

Рис. 3.1 – Імпортування необхідних бібліотек

Наступним кроком я зчитав дані з .xlsx файлу у датафрейм та вивів його перші п’ять рядків.

```
In 2
sample_data = pd.read_excel('sample_data.xlsx')
sample_data.head()
Executed at 2024.01.02 19:12:26 in 255ms
```

Out 2 — 5 rows × 67 columns pd.DataFrame

	Application	loan_amount	loan_days	applied_at	gender_id	Unn
0	1	3000	30	2021-02-01 00:21:56	1	
1	2	1000	7	2021-02-01 00:24:08	1	
2	3	1000	3	2021-02-01 00:36:35	2	
3	4	1600	30	2021-02-01 00:34:22	1	
4	5	2500	18	2021-02-01 23:22:57	2	

Рис. 3.2 – Зчитування даних у датарфеймі

Потім я перевірив датасет на наявність дублікатів та створив новий стовпець, що містить вік кожної людини на основі стовпця, що містить дати народження, та видалив цей стовпець.

```
In 3
sample_data.duplicated().sum()
Executed at 2024.01.02 19:12:26 in 18ms
```

Out 3: 0

```
In 4
current_year = datetime.now().year
sample_data.loc[:, 'age'] = current_year - pd.DatetimeIndex
 (sample_data['birth_date']).year
sample_dat = sample_data.drop(columns='birth_date')
sample_data['age'].head()
Executed at 2024.01.02 19:12:26 in 2ms
```

Out 4 — 5 rows — Length: 5, dtype: int32 pd.Series

	age
0	29
1	40
2	30
3	32
4	27

Рис. 3.3 – Створення нового стовпця

В якості ознак для передбачення повернення кредиту я обрав суму кредиту, термін погашення, стать, кількість дітей, місячну зарплатню тощо, адже вони є основними ознаками, які можуть вплинути на повернення кредиту, та при цьому не містять пропущених значень.

```
In 5  1  features = sample_data[
      2      ['loan_amount', 'loan_days', 'gender_id', 'children_count_id',
      3       'monthly_income', 'has_immovables', 'other_loans_active',
               'income_frequency_id', 'seniority_years', 'age']].copy()
      4  features.head()
```

Executed at 2024.01.02 19:12:26 in 14ms

Out 5 — 5 rows × 10 columns pd.DataFrame

	loan_amount	loan_days	gender_id	children_count_id	monthly_income
0	3000	30	1	1	15000
1	1000	7	1	2	11000
2	1000	3	2	1	10000
3	1600	30	1	1	8000
4	2500	18	2	1	9000

Рис. 3.4 – Створення датасету ознак

```
In 6  1  features.isna().sum()
```

Executed at 2024.01.02 19:12:26 in 31ms

Out 6 — Length: 10, dtype: int64 pd.Series

	<unnamed>
loan_amount	0
loan_days	0
gender_id	0
children_count_id	0
monthly_income	0
has_immovables	0
other_loans_active	0
income_frequency_id	0
seniority_years	0
age	0

Рис. 3.5 – Кількість пропущених значень у кожному стовпці

Оскільки в датасеті всі кредити є закритими, то навчати модель за такої цільової змінної немає сенсу, тому я створив стовпець, що містить бінарну змінну, чи людина закрила кредит вчасно, в якості цільової змінної.

```
In 7  1 sample_data['closed_in_time'] = ((sample_data['loan_closed'] == 1) &
        (sample_data['loan_overdue'] == 0)).astype(int)
      2 target = sample_data['closed_in_time']
      3 target.head()
```
Executed at 2024.01.02 19:12:26 in 54ms

Out 7 — 5 rows — Length: 5, dtype: int64 pd.Series — CSV

	closed_in_time
0	1
1	1
2	0
3	1
4	0

Рис. 3.6 – Кількість пропущених значень у кожному стовпці

На основі датасету ознак та масиву цільової змінної я створив навчальні та тестові набори. Оскільки датасет є відносно малим, то відношення розміру навчального до тестового набору я задав 70% до 30%. Крім того, я задав параметр stratify, щоб відношення значень цільової змінної було приблизно однаковим у навчальному та тестовому наборах.

```
In 8  1 X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0
        .3, random_state=42, stratify=target)
```
Executed at 2024.01.02 19:12:26 in 51ms

Рис. 3.7 – Розбиття на навчальні та тестові набори

Тепер можна перейти для навчання класифікатора, а саме дерева прийняття рішень. Окрім навчання, я також підібрав оптимальні гіперпараметри моделі серед заданих методом GridSearchCV, що перебирає всі можливі комбінації гіперпараметрів серед заданих та з використанням 5-згорткової крос-валідації обирає та навчає найкращу модель за значенням метрики точності (accuracy). Зокрема, найважливішим гіперпараметром тут є максимальна глибина дерева.

```
In 9  1 decision_tree_gcv = GridSearchCV(
      2     DecisionTreeClassifier(class_weight='balanced'),
      3     param_grid={
      4         'max_depth': range(5, 11),
      5         "min_samples_split": [2,5,7,10],
      6         "min_samples_leaf": [1,2,5]
      7     },
      8     n_jobs=-1,
      9     cv=5,
     10     verbose=1
     11 ).fit(X_train, y_train)
```
Executed at 2024.01.02 19:12:27 in 1s 310ms

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
```

```
In 10 1 decision_tree_gcv.best_params_
```
Executed at 2024.01.02 19:12:27 in 11ms

```
Out 10  {'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 10}
```

Рис. 3.8 – Навчання та пібдір гіперпараметрів для дерева рішень

Після навчання моделі я вивів звіт класифікації моделі для прогнозів на тестовому наборі та метрику точності для навчального набору. Можна побачити, що зокрема через малий розмір датасету (це скоріш за все є основним фактором, проте не єдиним) присутнє перенавчання, адже точність на навчальному наборі є суттєво вищою за точність на тестовому. Також незбалансованість датасету вплинула на метрики для різних класів, можемо, зокрема, бачити суттєву різницю в f-1 score (метрика, яка балансує precision та recall). Загалом можна зробити висновок, що для даного незбалансованого датасету варто було б обрати іншу модель, наприклад, SVM, яка є частим вибором для малих датасетів. Дерево рішень, яке треба було використати по завданню, є досить чутливим до незбалансованих датасетів.

```
In 11  1  predictions = decision_tree_gcv.predict(X_test)
       2  print(classification_report(y_test, predictions))
          Executed at 2024.01.02 19:12:27 in 8ms

                        precision    recall  f1-score   support

                     0       0.25      0.35      0.29        37
                     1       0.75      0.65      0.70       113

              accuracy                           0.57       150
             macro avg       0.50      0.50      0.49       150
          weighted avg       0.63      0.57      0.60       150

In 12  1  decision_tree_gcv.score(X_train, y_train)
          Executed at 2024.01.02 19:12:27 in 4ms

Out 12     0.7564469914040115
```

Рис. 3.9 – Метрики навченої моделі на навчальному та тестовому наборі

З метою виявлення шахрайства та фальсифікації даних я використав алгоритм кластеризації DBSCAN, який є досить ефективним для даної задачі, оскільки для нього не треба наперед визначати кількість кластерів, та він виділяє дані, які є шумом (викидами). Таким чином, за допомогою нього я й визначив аномалії.

Спершу перед навчанням цього методу я виділив ознаки, по яким будуть виявлятись шахрайство та фальсифікації даних, а саме: термін погашення кредиту, вік людини та відношення суми кредиту до місячного прибутку (я створив окремий стовпець для цієї змінної).

```
In 17  1 sample_data['loan_to_income_ratio'] = sample_data['loan_amount'] /
          sample_data['monthly_income']
       Executed at 2024.01.02 19:14:22 in 10ms

In 18  1 clustering_features = ['loan_to_income_ratio', 'loan_days', 'age']
       2 clustering_data = sample_data[clustering_features].copy()
       3 clustering_data.head()
       Executed at 2024.01.02 19:14:22 in 4ms
```

Out 18 — 5 rows × 3 columns pd.DataFrame

	loan_to_income_ratio	loan_days	age
0	0.200000	30	29
1	0.090909	7	40
2	0.100000	3	30
3	0.200000	30	32
4	0.277778	18	27

Рис. 3.10 – Дані для кластеризації

Наступним кроком для навчання DBSCAN необхідно нормалізувати дані, щоб вони були в одній шкалі. Для цього я використав StandardScaler, що приводить дані до наступної шкали: середнє значення рівне 0, стандартне відхилення рівне 1, тобто усі значення виражені в кількості стандратних відхилень.

```
In 19  1 scaler = StandardScaler()
       2 clustering_data_scaled = scaler.fit_transform(clustering_data)
       Executed at 2024.01.02 19:14:22 in 4ms
```

Рис. 3.10 – Нормалізація даних

Потім вже я навчив DBSCAN на нормалізованих даних та створив окремий стовпець, що містить булеві значення на основі того, чи є даний рядок аномалією. DBSCAN позначає аномалії (шум) як кластер з номером -1.

```
In 20  1 dbscan = DBSCAN()
       2 clusters = dbscan.fit_predict(clustering_data_scaled)
       3 sample_data['suspicious'] = clusters == -1
       4 sample_data[sample_data['suspicious']].loc[:, clustering_features]
       Executed at 2024.01.02 19:14:23 in 6ms
```

Out 20 — 2 rows × 3 columns pd.DataFrame

	loan_to_income_ratio	loan_days	age
77	200.000000	15	62
111	0.188889	4	63

Рис. 3.11 – Виявлення аномалій

Як бачимо, лише 2 людини є підозрілими. Одна з них має дуже високе відношення суми кредиту до місячного прибутку, і при цьому термін погашення становить лише 15 днів. Щодо другої людини, то наразі складно казати про якесь шахрайство чи фальсифікацію даних, тому варто подивитись на значення описових статистик для обраних змінних.

In 21
```
clustering_data.describe()
```
Executed at 2024.01.02 19:14:30 in 13ms

Out 21

8 rows × 3 columns pd.DataFrame

	loan_to_income_ratio	loan_days	age
count	499.000000	499.000000	499.000000
mean	0.630140	22.124248	35.160321
std	8.945472	9.151590	8.291280
min	0.015789	3.000000	23.000000
25%	0.100000	15.000000	29.000000
50%	0.166667	28.000000	33.000000
75%	0.300000	30.000000	39.000000
max	200.000000	30.000000	65.000000

Рис. 3.12 – Значення описових статистик змінних, обраних для кластеризації

Можна побачити, що DBSCAN визначив цю людину як аномалію через те, що значення термін погашення в неї близький до мінімального у всьому датасеті, а вік – до масимального, тому загалом справді треба приділити додаткову увагу цій людині. У випадку першої людини можна досить впевнено казати про шахрайство та фальсифікацію даних, адже в неї найбільше відношення суми кредиту до місячного прибутку, яке справді є аномальним.

IV. Висновок

Отже, в ході даної лабораторної я узагальнив та застосував на практиці набуті навички реалізації проектного практикуму в галузі кредитного scoring-у, зокрема використовуючи заданий алгоритм машинного навчання (дерев рішень), я передбачив, чи закриє людина кредит вчасно та за допомогою потужного алгоритму кластеризації DBSCAN виявив потенційне шахрайство та фальсифікації даних (аномалії). DBSCAN показав досить якісні результати, проте дерево рішень за даного датасету не є оптимальним для задачі класифікації через незбалансованість датасету та малий об’єм датасету, що призвело до перенавчання. У випадку незбалансованого та малого датасету SVM серед методів машинного навчання зазвичай дає значно кращі результати інші алгоритми.