

**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:

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ВИЗНАЧЕННЯ КРЕДИТНИХ РИЗИКІВ ДЛЯ
БАНКІВСЬКИХ CRM СИСТЕМ**

Виконав:

Студентка 2 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІТ-03
Цуканова М.С.

Перевірив:

Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2022

I. Мета:

виявити дослідити та узагальнити особливості застосування методів визначення кредитних ризиків для банківських CRM, ERP систем з використанням спеціалізованих пакетів мови програмування Python.

II. Завдання:

Завдання реалізувати у відповідності до пунктів:

- 1.1. Завантажити вихідні дані з файлів додатку Pr15_data_description.xlsx, Pr15_sample_data.xlsx;
- 1.2. Здійснити визначення кредитних ризиків для банківських CRM систем відповідно до технічних умов, заданих у таблицях 1,2 додатку Д1;
- 1.3. Здійснити візуалізацію результатів оцінювання і прогнозування (у формі таблиці та графіків);
- 1.4. Оцінити ефективність розробленого скрипта для різного складу індикаторів скорингової таблиці та за різними математичними моделями скорингового аналізу.

Результат представити у формі:

- 1.5. Результати архітектурного проектування скрипта, що реалізує технічні умови задачі.
- 1.6. Програмний скрипт, результати його функціонування.
- 1.7. Результати візуалізації визначення кредитних ризиків для банківських CRM систем.

Варіант (порядковий номер в списку групи)	Технічні умови завдання
1, 16	Розробити програмний скрипт, що реалізує: 1. Парсинг файлів параметрів: Pr15_data_description.xlsx, Pr15_sample_data.xlsx; 2. Вибір індикаторів скорингової таблиці (10 шт.); 3. Очищення даних; 4. Розрахунок інтегрованої оцінки Scor за самостійно обраною моделлю; 5. Кластеризацію позичальників за бінарною характеристикою надання / відмова у кредитування; 6. Візуалізація результатів розрахунків у формі графіку, файлів (таблиці).

Варіант (порядковий номер в списку групи)	Технічні умови завдання
1, 16	Розробити програмний скрипт, що реалізує: 1. Скоринговий аналіз позичальників за даними Pr15_data_description.xlsx, Pr15_sample_data.xlsx відповідно до моделі дискримінантного аналізу. 2. Передбачити чи буде кредит повернуто у форматі бінарної оцінки (0 або 1); 3. Виявлення шахрайства та фальсифікації даних.

III. Результати виконання лабораторної роботи.

3.1. Результати архітектурного проектування на їх опис

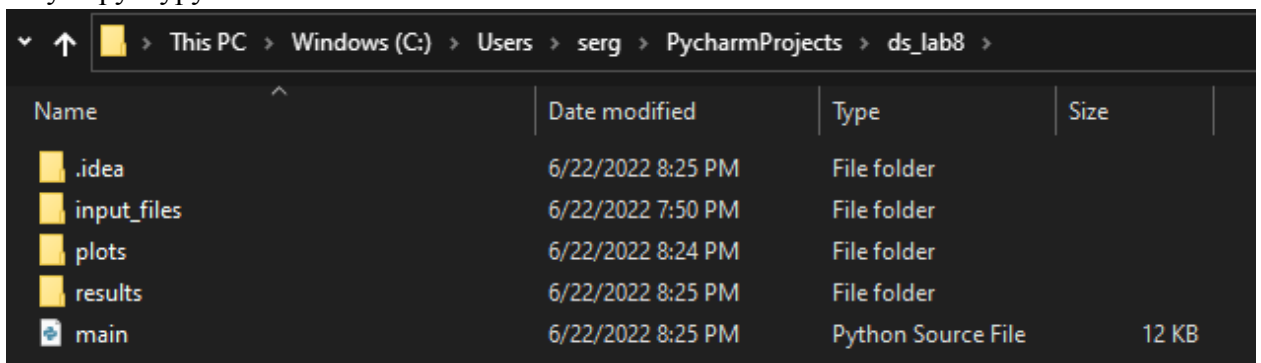
Після початкового парсингу та аналізу вхідного з інформацією про людей, що хочуть взяти кредит, відбувається парсинг файлу з скоринговими індикаторами та первинне формування скорингової таблиці. Наступний крок - аналіз перетину скорингових індикаторів та вхідних даних, отриманий датафрейм очищається від пропусків та створюється скорингова карта. В якості параметрів за якими оцінюється SCOR було обрано поля 'loan_amount', 'loan_days', 'education_id', 'has_immovables', 'has_movables', 'monthly_income', 'income_frequency_id', 'monthly_expenses', 'product_dpr',

'product_interest_min'. Скорингова модель багатокритеріального оцінювання починається з парсингу файлу критеріїв скорингової карти, далі відбувається відбір даних за критеріями та нормування критеріїв. Інтегрована багаторитеріальна оцінка SCOR додається окремою колонкою 'Score' до основного датафрейму. Для роботи з аномаліями використано z-score, його результати записані в поле 'Zscore', всі результати, що більше 3х були видалені з загального датафрейму. Було вирішено, що кредит було видано всім зі SCOR більше середнього, всіх іншим в ньому буде відмовлено. Таким же чином було спрогнозовано які люди повернуть кредит (з показником SCOR вище середнього серед тих, кому дали кредит). Також було зроблено кластеризацію за бінарним принципом для видачі кредитів та їх повернення.

3.2. Опис структури проекту програми в середовищі PyCharm.

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проект.

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



Name	Date modified	Type	Size
.idea	6/22/2022 8:25 PM	File folder	
input_files	6/22/2022 7:50 PM	File folder	
plots	6/22/2022 8:24 PM	File folder	
results	6/22/2022 8:25 PM	File folder	
main	6/22/2022 8:25 PM	Python Source File	12 KB

мал. 1, структура проекту

ds_lab8 - головна директорія проекту

main.py - головний файл проекту

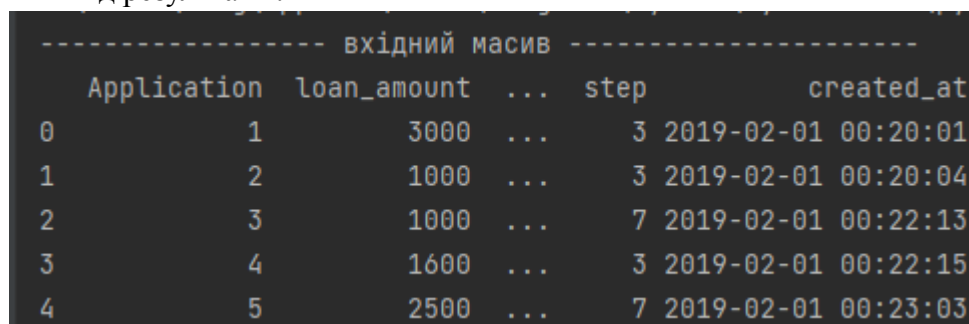
input_files - директорія з вхідними файлами для лабораторної роботи

plots - директорія зі збереженими результатами графічного аналізу

results - директорія з збереженими датафреймами, які були створені під час виконання лабораторної роботи

3.3. Результати роботи розробленого скрипта відповідно до завдання

Консольний вивід результатів:



```

----- вхідний масив -----
  Application  loan_amount  ...  step  created_at
0            1         3000  ...    3  2019-02-01 00:20:01
1            2         1000  ...    3  2019-02-01 00:20:04
2            3         1000  ...    7  2019-02-01 00:22:13
3            4         1600  ...    3  2019-02-01 00:22:15
4            5         2500  ...    7  2019-02-01 00:23:03
  
```

мал. 1, склад вхідного масиву

```

----- пропущені значення стовпців (суми) -----
Application          0
loan_amount          0
loan_days            0
applied_at           0
gender_id            0
...
prolongation_number   436
prolongation_total_days 436
wizard_type_id        0
step                 0
created_at            0

```

мал. 2, суми кількості пропущених значень стовпців

```

----- скорингові індикатори -----
Field_in_data  ...      Note
0      id      ...      NaN
1  loan_amount  ...      NaN
2   loan_days  ...      NaN
3  applied_at  ...  Локальное время на устройстве пользователя или...
4   language  ...      Всегда RU

```

мал. 3, вхідний масив скорингових індикаторів

```

----- первинна скор. таблиця -----
Field_in_data  ...      Note
0  loan_amount  ...      NaN
1   loan_days  ...      NaN
2 purpose_other  ...      NaN
3   gender_id  ...      Нужна расшифровка id
4  birth_date  ...  Иногда заполнена неправильно

```

мал. 4, первинна скорингова таблиця

```

1.5.1. Аналіз перетину скорингових індикаторів та сегменту вхідних даних
к-ть співпадінь: 38
Індекси співпадінь [ 0.  1.  3.  4.  6.  7. 35. 36. 37. 38. 39. 40. 43. 45. 47. 48. 49. 53.
55. 56. 57. 58. 59. 60. 62. 63. 64. 65. 66. 67. 68. 69. 70. 72. 74. 75.
76. 77.]

```

мал. 5, перетин індикаторів та їх кількість

```

----- DataFrame співпадінь -----
Field_in_data  ...      Note
0  loan_amount  ...      NaN
1   loan_days  ...      NaN
2   gender_id  ...      Нужна расшифровка id
3  birth_date  ...  Иногда заполнена неправильно
4 children_count_id  ...      Нужна расшифровка id

```

мал. 6, датафрейм перетину індикаторів

```

----- DataFrame вхідних даних - скорингова карта -----
   loan_amount  loan_days  ...  applied_limit  amount_to_pay
0         3000         30  ...           3000          0.0
1         1000          7  ...           1000          0.0
2         1000          3  ...           1000         1088.0
3         1600         30  ...           1600          0.0
4         2500         18  ...           2500         3402.5

```

мал. 7, скорингова карта

```

----- DataFrame df_desc_minimax -----
   Unnamed: 0  Field_in_data  ...  Note Unnamed: 6
0          0  loan_amount  ...    NaN    NaN
1          1  loan_days  ...    NaN    NaN
2          5  education_id  ...  Нужна расшифровка id    NaN
3          7  has_immovables  ...    NaN    NaN
4          8  has_movables  ...    NaN    NaN

```

мал. 8, парсинг файлу критеріїв скорингової карти

```

----- DataFrame df_sample_minimax -----
   loan_amount  loan_days  ...  product_dpr  product_interest_min
0         3000         30  ...           1.80             50
1         1000          7  ...           0.00            120
2         1000          3  ...           1.90             50
3         1600         30  ...           1.80             50
4         2500         18  ...           1.90             50

```

мал. 9, датафрейм після відбору даних за критеріями

```

----- DataFrame: d_segment_sample_min -----
loan_amount      300.0
loan_days         3.0
education_id       1.0
has_immovables    0.0
has_movables      0.0
dtype: float64

```

мал. 10, датафрейм з полям з min значеннями

```

----- DataFrame: d_segment_sample_max -----
loan_amount      8000.0
loan_days        30.0
education_id       6.0
has_immovables     1.0
has_movables       1.0
dtype: float64

```

мал. 11, датафрейм з полям з max значеннями

```

----- Нормування критеріїв -----
[[0.37500937 0.99019608 0.0050005 ... 0.066671 0.84 0.00105679]
 [0.12502812 0.23856209 0.0207555 ... 0.0833375 0.12 0.00739754]
 [0.12502812 0.10784314 0.04591368 ... 0.0833375 0.88 0.00100876]

```

мал. 12, результат нормування критеріїв

```

----- SCOR -----
[1.16969697e+02 1.08995282e+01 1.82047482e+01 1.16600911e+02
 1.95096992e+01 1.16493770e+02 1.16624386e+02 1.87792778e+01
 1.42288887e+01 1.08800477e+01 1.16666094e+02 1.64331499e+01
 1.18917186e+02 1.13681223e+02 1.14063181e+02 1.16449542e+02
 1.60897173e+01 1.63885565e+01 1.16950900e+02 1.16701673e+02
 1.61802678e+01 1.80001325e+01 1.18703244e+02 1.82512513e+01

```

мал. 13, результати інтегрованої багатокритеріальної оцінки

```

-----
Shape до роботи з аномаліями: (499, 12)
Shape після роботи з аномаліями: (492, 12)

```

мал. 14, результати роботи за аномаліями

Графічний вивід:



мал. 15, графік до роботи з аномаліями



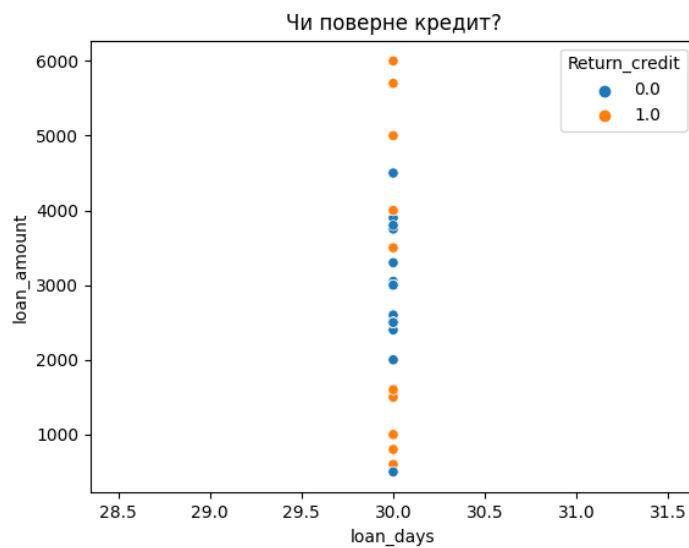
мал. 16, графік після роботи з аномаліями



мал. 17, кластеризація за принципом 'Чи дадуть кредит?'



мал. 18, графік SCOR людей, яким одобрять кредит



мал. 19, кластеризація за принципом 'Чи поверне кредит?'

3.4. Програмний код, що забезпечує отримання результату

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
```

мал. 20, імпорт необхідних бібліотек

```
# ----- I. ПІДГОТОВКА ВХІДНИХ ДАНИХ -----
# 1.1. Парсинг файлу вхідних даних
input_dataframe = pd.read_excel('input_files/Pr15_sample_data.xlsx', parse_dates=['birth_date'])
print('----- вхідний масив -----')
print(input_dataframe.head(5))
print('-----')

# 1.2. Аналіз структури вхідних даних
print('----- пропущені значення стовпців (суми) -----')
print(input_dataframe.isnull().sum())

# 1.3. Парсинг файлу та аналіз структури скорингових індикаторів
score_ind_input = pd.read_excel('input_files/Pr15_data_description.xlsx')
print('----- скорингові індикатори -----')
print(score_ind_input.head(5))
print('-----')
```

```
# 1.4. Первинне формування скорингової таблиці
score_table_v1 = score_ind_input[(score_ind_input.Place_of_definition == 'Указывает заемщик')
                                | (score_ind_input.Place_of_definition == 'параметры связанные с выданным продуктом')]
n_client_bank = score_table_v1['Place_of_definition'].size
score_table_v1.index = range(0, len(score_table_v1))
print('----- первинна скор. таблиця -----')
print(score_table_v1.head(5))
score_table_v1.to_csv('results/1_score_table_v1.csv')
print('-----')
```

```
# 1.5. Очищення даних
print('1.5.1. Аналіз перетину скорингових індикаторів та сегменту вхідних даних')
b = score_table_v1['Field_in_data']
if set(b).issubset(input_dataframe.columns):
    Flag_b = 'Flag_True'
else:
    Flag_b = 'Flag_False'
# ----- кількість співпадінь
n_columns = score_table_v1['Field_in_data'].size
j = 0
for i in range(0, n_columns):
    a = score_table_v1['Field_in_data'][i]
    if set([a]).issubset(input_dataframe.columns):
        j = j + 1
print('к-ть співпадінь: ', j)
Columns_Flag_True = np.zeros((j))
j = 0
for i in range(0, n_columns):
    a = score_table_v1['Field_in_data'][i]
    if set([a]).issubset(input_dataframe.columns):
        Flag = 'Flag_True'
        Columns_Flag_True[j] = i
        j = j + 1
    else:
        Flag = 'Flag_False'
print('Індекси співпадінь', Columns_Flag_True)
```

мал. 21-23, обробка вхідних даних


```

# 1.5.2. Формування DataFrame даних з урахуванням відсутніх індикаторів скорингової таблиці
score_table_wout_ind = score_table_v1.iloc[Columns_Flag_True]
score_table_wout_ind.index = range(0, len(score_table_wout_ind))
print('----- DataFrame снівпадінь -----')
print(score_table_wout_ind.head(5))
score_table_wout_ind.to_csv('results/2_score_table_wout_ind.csv')
print('-----')

# 1.5.3. Очищення скорингової таблиці від пропусків
b = score_table_wout_ind['Field_in_data']
d_data_client_bank = input_dataframe[b]
print('---- пропуски даних сегменту DataFrame -----')
print(d_data_client_bank.isnull().sum())
print('-----')

```

мал. 24, формування скорингової таблиці

```

# ----- СКОРИНГОВА КАРТА -----
# Очищення індикаторів скорингової таблиці
scoring_map = score_table_wout_ind.loc[(score_table_wout_ind['Field_in_data'] != 'fact_addr_start_date')]
scoring_map = scoring_map.loc[(scoring_map['Field_in_data'] != 'position_id')]
scoring_map = scoring_map.loc[(scoring_map['Field_in_data'] != 'employment_date')]
scoring_map = scoring_map.loc[(scoring_map['Field_in_data'] != 'has_prior_employment')]
scoring_map = scoring_map.loc[(scoring_map['Field_in_data'] != 'prior_employment_start_date')]
scoring_map = scoring_map.loc[(scoring_map['Field_in_data'] != 'prior_employment_end_date')]
scoring_map = scoring_map.loc[(scoring_map['Field_in_data'] != 'income_frequency_other')]
scoring_map.index = range(0, len(scoring_map))
scoring_map.to_csv('results/3_scoring_map.csv')

useless_columns = ['gender_id', 'birth_date', 'children_count_id', 'fact_addr_owner_type_id', 'fact_addr_start_date',
                  'employment_type_id', 'position_id', 'organization_type_id', 'organization_branch_id',
                  'employees_count_id', 'employment_date', 'seniority_years', 'has_prior_employment',
                  'prior_employment_start_date', 'prior_employment_end_date', 'income_frequency_other',
                  'income_source_id', 'other_loans_has_closed', 'other_loans_active', 'other_loans_about_current',
                  'other_loans_about_monthly', 'product_amount_from', 'product_amount_to',
                  'product_base_amount_limit', 'amount_limit']

# Очищення вхідних даних
d_clean = d_data_client_bank.drop(columns=useless_columns)
d_clean.index = range(0, len(d_clean))
d_clean.to_csv('results/4_d_clean.csv')
print(d_clean.isnull().sum())
print('----- DataFrame вхідних даних - скорингова карта -----')
print(d_clean.head(5))
print('----- DataFrame індикатори скорингу -----')
print(scoring_map)
print('-----')

```

мал. 25, формування скорингової карти

```
# ----- II. ФОРМУВАННЯ СКОРИНГОВОЇ МОДЕЛІ -----
# 2.1. Парсінг файлу індикаторів (критеріїв) скорингової карти
data_desc_minimax = pd.read_excel('input_files/data_desc_cleaning_minimax.xlsx')
df_desc_minimax = data_desc_minimax.loc[(data_desc_minimax['Minimax'] == 'min')
                                         | (data_desc_minimax['Minimax'] == 'max')]
df_desc_minimax.index = range(0, len(df_desc_minimax))
print('----- DataFrame df_desc_minimax -----')
print(df_desc_minimax.head(5))
print('-----')
df_desc_minimax.to_csv('results/5_df_desc_minimax.csv')

# відбір даних за критеріями
d = df_desc_minimax['Field_in_data']
cols = d.values.tolist()
df_sample_minimax = d_clean[cols]
print('----- DataFrame df_sample_minimax -----')
print(df_sample_minimax)
print('-----')
df_sample_minimax.to_csv('results/6_df_sample_minimax.csv')
```

```
# 2.2. Парсінг файлу індикаторів (критеріїв) скорингової карти
d_sample_min = df_sample_minimax[cols].min()
d_sample_max = df_sample_minimax[cols].max()
print('----- DataFrame: d_segment_sample_min -----')
print(d_sample_min.head(5))
d_sample_min.to_csv('results/7_d_sample_min.csv')
print('----- DataFrame: d_segment_sample_max -----')
print(d_sample_max.head(5))
d_sample_min.to_csv('results/8_d_sample_max.csv')

# 2.3. Нормування критеріїв
m = df_sample_minimax['loan_amount'].size
n = df_desc_minimax['Field_in_data'].size
d_minimax_norm = np.zeros((m, n))
```

```
# 2.3. Нормування критеріїв
m = df_sample_minimax['loan_amount'].size
n = df_desc_minimax['Field_in_data'].size
d_minimax_norm = np.zeros((m, n))

delta_d = 0.3
for j in range(0, len(df_desc_minimax)):
    columns_d = df_desc_minimax['Minimax'][j]
    if columns_d == 'min':
        columns_m = df_desc_minimax['Field_in_data'][j]
        for i in range(0, len(df_sample_minimax)):
            max_max = d_sample_max[j] + (2 * delta_d)
            d_minimax_norm[i, j] = (delta_d + df_sample_minimax[columns_m][i]) / (max_max)
    else:
        for i in range(0, len(df_sample_minimax)):
            min_min = d_sample_max[j] + (2 * delta_d)
            d_minimax_norm[i, j] = (1 / (delta_d + df_sample_minimax[columns_m][i])) / (min_min)

print('----- Нормування критеріїв -----')
print(d_minimax_norm)
np.savetxt('results/9_d_minimax_norm.txt', d_minimax_norm)
print('-----')
```

мал. 26-28, формування скорингової моделі

```

# 2.4.Інтегрована багатокритеріальна оцінка - SCOR
def Voronin(d_minimax_norm, n):
    Integro = np.zeros(499)
    Scor = np.zeros(499)
    for i in range(0, 499):
        Sum_Voronin = 0
        for j in range(0, n):
            Sum_Voronin = Sum_Voronin + ((1 - d_minimax_norm[i, j]) ** (-1))
        Integro[i] = Sum_Voronin
        Scor[i] = 1000
        np.savetxt('results/10_integro_scor.txt', Integro)
    plt.title('Integro_Scor до роботи з аномаліями')
    plt.plot(Integro)
    plt.show()
    return Integro

integro_score = Voronin(d_minimax_norm, n)
print('----- SCOR -----')
print(integro_score)

```

мал. 29, формування інтегрованої багатокритеріальної оцінки

```

# ----- РОБОТА З АНОМАЛІЯМИ -----
zscore = stats.zscore(integro_score)
d_w_score = df_sample_minimax.copy()
d_w_score['Score'] = integro_score
d_w_score['Zscore'] = zscore
d_w_score.to_csv('results/11_with_score.csv')

d_wout_anom = d_w_score[d_w_score['Zscore'] < 2]
integro_score = d_wout_anom['Score'].tolist()
plt.title('Integro_Scor після роботи з аномаліями')
plt.plot(integro_score)
plt.show()
d_wout_anom.to_csv('results/12_without_anomalies.csv')

print('-----')
print('Shape до роботи з аномаліями: ', d_w_score.shape)
print('Shape після роботи з аномаліями: ', d_wout_anom.shape)

```

мал. 30, робота з аномаліями

```

# ----- ЧИ ДАВАТИ КРЕДИТ? (таблиця) -----
d_wout_anom.loc[d_wout_anom['Score'] > d_wout_anom['Score'].mean(), 'Give_credit'] = 1
d_wout_anom.loc[d_wout_anom['Score'] < d_wout_anom['Score'].mean(), 'Give_credit'] = 0
d_wout_anom.to_csv('results/13_give_loan.csv')

# ----- ЧИ ДАВАТИ КРЕДИТ? (візуалізація) -----
# кластеризація
sns.scatterplot(data=d_wout_anom, x='loan_days', y='loan_amount', hue='Give_credit',
                c=['blue', 'yellow'])
plt.title('Чи давати кредит?')
plt.show()
#видалення людей яким не дадуть кредит
d_give_loan = d_wout_anom[d_wout_anom['Score'] > d_wout_anom['Score'].mean()]
integro_score = d_give_loan['Score'].tolist()
plt.title('Integro_Score людей з одобреним кредитом')
plt.plot(integro_score)
plt.show()

# ----- ЧИ ПОВЕРНЕ КРЕДИТ? (таблиця, діаграма) -----
d_return_loan = d_give_loan[d_give_loan['Score'] > d_give_loan['Score'].mean()]
d_return_loan.loc[d_give_loan['Score'] > d_return_loan['Score'].mean(), 'Return_credit'] = 1
d_return_loan.loc[d_give_loan['Score'] < d_return_loan['Score'].mean(), 'Return_credit'] = 0
d_return_loan.to_csv('results/14_return_loan.csv')

sns.scatterplot(data=d_return_loan, x='loan_days', y='loan_amount', hue='Return_credit',
                c=['blue', 'yellow'])
plt.title('Чи поверне кредит?')
plt.show()

```

мал. 31, прийняття рішень та кластеризація

IV. Висновки.

У цій лабораторній роботі було реалізовано скрипт дослідження та узагальнення особливостей застосування методів визначення кредитних ризиків для банківських CRM, ERP систем з використанням спеціалізованих пакетів мови програмування Python. Було реалізовано скрипт для прийняття рішення про кредитування, спрогнозовано вірогідність повернення кредиту та виконано роботу по виявленню та видаленню випадків фальсифікації даних та шахрайства. В ході виконання лабораторної роботи було використано бібліотеки Pandas, Matplotlib, Numpy, Scipy та Seaborn мови Python. Серед розробки - PyCharm.

Виконав: студент Цуканова М.С.