ЛАБОРАТОРНА РОБОТА № 3

ВВЕДЕННЯ В АНАЛІЗ ДАНИХ НА PYTHON

Мета: вивчення основних етапів аналізу даних з використанням бібліотеки Pandas в середовищі Python.

Хід роботи:

Завдання 1. Прочитайте дані з файлу data.csv.

Лістинг програми:

```
import io
# завантаження файлу
from google.colab import files
uploaded = files.upload()
# зчитування даних
data = pd.read_csv(io.StringIO(uploaded['data.csv'].decode('utf-8')))
```

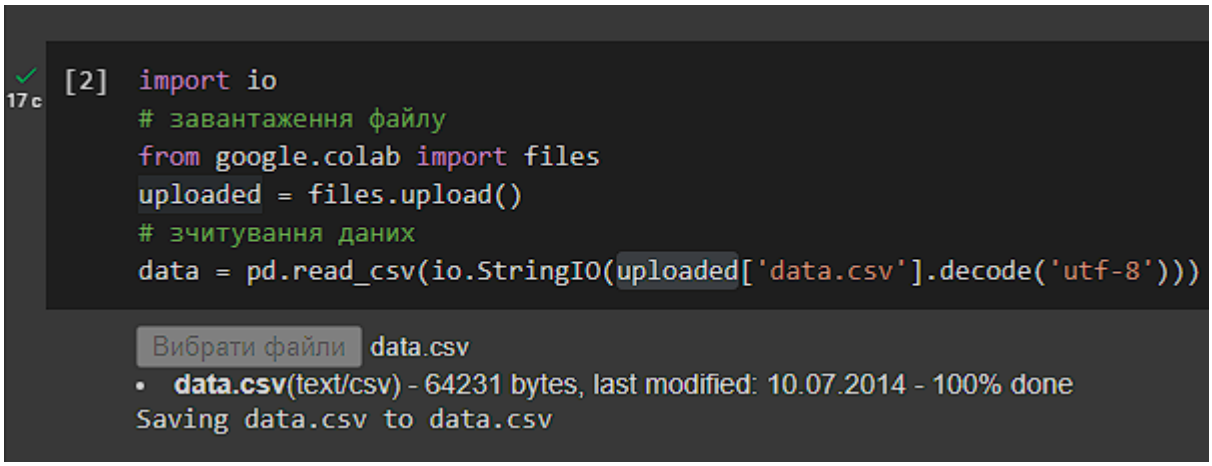

Рис. 1. Результат читання даних з файлу data.csv

Цей код завантажує бібліотеку io та модуль files з бібліотеки google.colab, щоб надати змогу завантажити файл з локального комп'ютера користувача в середовище Colab. Далі, за допомогою методу files.upload() користувач може завантажити файл data.csv. Отриманий файл зчитується у змінну data за допомогою ме

тоду read_csv() з бібліотеки pandas, який призначений для читання даних з файлу у форматі CSV.

Завдання 2. Виведіть опис даних, що було прочитано.

Лістинг програми:

```
data.describe()
```

[3] data.describe()

	Id	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	NumberOfOpe
count	1350.000000	1350.000000	1350.000000	1350.000000	1350.000000	1350.000000	1094.000000	
mean	675.500000	0.060000	3.577895	52.048889	0.257778	356.123363	6438.473492	
std	389.855743	0.237575	84.914699	15.009875	0.751718	1156.603074	7849.754675	
min	1.000000	0.000000	0.000000	22.000000	0.000000	0.000000	0.000000	
25%	338.250000	0.000000	0.031140	40.000000	0.000000	0.175125	3300.000000	
50%	675.500000	0.000000	0.156891	52.000000	0.000000	0.367049	5222.500000	
75%	1012.750000	0.000000	0.543145	63.000000	0.000000	0.807001	8055.250000	
max	1350.000000	1.000000	2340.000000	97.000000	10.000000	15466.000000	208333.000000	

Рис. 2. Результат виводу опису даних, що було прочитано

data.describe() є методом з бібліотеки pandas, який надає статистичний опис числових даних з DataFrame або Series. Виклик цього методу поверне деякі основні статистичні показники, такі як кількість, середнє значення, стандартне відхилення, мінімальне та максимальне значення, квартилі та медіана. Це може допомогти користувачеві отримати загальне уявлення про дані та їх розподіл, а також виявити потенційні проблеми, такі як відсутність даних або аномальні значення.

Завдання 3. Відобразіть декілька перших та декілька останніх записів.

Лістинг програми:

```
# відобразити перші 10 записів
data.head(10)

# відобразити останні 7 записів
data.tail(7)
```

```
[4]  # відобразити перші 10 записів
     data.head(10)

     # відобразити останні 7 записів
     #data.tail(7)
```

	Id	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	NumberOfOpenCreditLinesAndLoans
0	1	1	0.766127	45	2	0.802982	9120.0	13
1	2	0	0.957151	40	0	0.121876	2600.0	4
2	3	0	0.658180	38	1	0.085113	3042.0	2
3	4	0	0.233810	30	0	0.036050	3300.0	5
4	5	0	0.907239	49	1	0.024926	63588.0	7
5	6	0	0.213179	74	0	0.375607	3500.0	3
6	7	0	0.305682	57	0	5710.000000	NaN	8
7	8	0	0.754464	39	0	0.209940	3500.0	8
8	9	0	0.116951	27	0	46.000000	NaN	2
9	10	0	0.189169	57	0	0.606291	23684.0	9

Рис. 3. Результат відображення перших 10 записів

```
# відобразити перші 10 записів
#data.head(10)

# відобразити останні 7 записів
data.tail(7)
```

	Id	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	NumberOfOpenCreditLinesAndL
1343	1344	0	0.202775	59	0	6994.000000	NaN	
1344	1345	0	0.087406	32	0	0.288978	1750.0	
1345	1346	0	0.000000	39	0	0.055916	4166.0	
1346	1347	0	0.045694	49	0	0.300175	4000.0	
1347	1348	0	0.022780	53	0	0.323068	10000.0	
1348	1349	0	0.036934	56	0	0.287935	8362.0	
1349	1350	0	0.000000	62	0	1463.000000	NaN	

Рис. 4. Результат відображення останніх 7 записів

Обидві функції можуть приймати необов'язковий аргумент n, який вказує кількість рядків, які потрібно відобразити. Якщо цей аргумент не вказується, то за замовчуванням буде відображено 5 рядків. Наприклад, data.head(10) відображує перші 10 рядків, а data.tail(7) відображує останні 7 рядків.

Завдання 4. Прочитайте у файлі DataDictionary-ua.txt, що означають стовпчики матриці. Якому типу належить кожен стовпчик (дійсний, цілий, категоріальний)?

Лістинг програми:

```
import re

name_index = 0
description_index = 1
name_list = []
desc_list = []

chars_to_remove = " \n"

with open('DataDictionary-ua.txt', 'r') as f:
    for i, line in enumerate(f):
        if i == name_index:
            # extract column name
            name = line.strip()
            name_list.append(name)
            name_index += 3  # update index for next iteration
        elif i == description_index:
            # extract column description
            description = line.strip()
            desc_list.append(description)
            description_index += 3  # update index for next iteration

column_types = []

for line in desc_list:
    if re.search(r'\binteger\b', line):
        column_types.append('цілий')
    elif re.search(r'\breal\b', line) or re.search(r'%', line):
        column_types.append('дійсний')
    else:
        column_types.append('категоріальний')

for i in range(len(name_list)):
    print(f"{name_list[i]} - {column_types[i]}")
```

```
SeriousDlqin2yrs - категоріальний
RevolvingUtilizationOfUnsecuredLines - дійсний
age - цілий
NumberOfTime30-59DaysPastDueNotWorse - цілий
DebtRatio - дійсний
MonthlyIncome - дійсний
NumberOfOpenCreditLinesAndLoans - цілий
NumberOfTimes90DaysLate - цілий
NumberRealEstateLoansOrLines - цілий
NumberOfTime60-89DaysPastDueNotWorse - цілий
NumberOfDependents - цілий
```

Рис. 5. Результат визначення типу кожного стовпчика

За допомогою open(), відкриваємо файл та проходимо через кожний його рядок. Якщо змінна і дорівнює name_index, тоді поточний рядок містить ім'я стовпчика, додаємо до списку name_list. Якщо ж змінна і дорівнює description_index, то поточний рядок містить опис стовпчика, додаємо до списку desc_list. Після цього оновлюється name_index та description_index, щоб вони вказували на наступні імена та описи стовпчиків. Далі використовується список desc_list, щоб визначити тип кожного стовпчика: якщо знайдено слово "integer" - це цілий тип, якщо знайдено слово "real" або знайдено символ "%" - це дійсний тип, у всіх інших випадках це категоріальний тип. Знайдені типи зберігаються в списку column_types. В кінці, проходимось по всім елементам списку name_list та column_types, і виводимо ім'я та тип кожного стовпця за допомогою команди print().У останньому циклі for відображається повний опис кожного стовпця, який складається з назви та його типу.

Завдання 5. Зверніть увагу, що стовпчик DebtRatio вміщує неправдоподібні дані. Тільки значення, що відповідають відомому доходу за місяць, є відношеннями. Всі інші - абсолютні значення виплат відсотків за місяць. Виправте дані, зробивши всі значення стовпчика DebtRatio абсолютними (помножте їх на MonthlyIncome). Щоб ваша програма працювала швидко на повних даних, спробуйте не використовувати цикл.

Лістинг програми:

```
import numpy as np

i = data['MonthlyIncome'].notnull()
data.loc[i, 'DebtRatio'] = np.where(data.loc[i, 'DebtRatio'] <= 1,
                                    data.loc[i, 'DebtRatio']*data.loc[i, 'Mont
hlyIncome'],
                                    data.loc[i, 'DebtRatio'])

print(data['DebtRatio'])
```

```
[60] import numpy as np

     i = data['MonthlyIncome'].notnull()
     data.loc[i, 'DebtRatio'] = np.where(data.loc[i, 'DebtRatio'] <= 1,
                                         data.loc[i, 'DebtRatio']*data.loc[i, 'MonthlyIncome'],
                                         data.loc[i, 'DebtRatio'])

     print(data['DebtRatio'])
```

```
0       7323.197016
1        316.878123
2        258.914887
3        118.963951
4       1584.975094
           ...
1345     232.944085
1346    1200.699824
1347    3230.676930
1348    2407.712069
1349    1463.000000
Name: DebtRatio, Length: 1350, dtype: float64
```

Рис. 6. Результат виправлення стовпчика стовпчик DebtRatio

Спочатку створюється масив i, що містить булеві значення True або False в залежності від того, чи є значення в стовпці MonthlyIncome відсутніми (NaN). Далі використовується метод loc для вибору рядків, для яких значення в стовпці MonthlyIncome не є пропущеними, та стовпця DebtRatio. Вибрані значення стовпця DebtRatio замінюються за допомогою умовного присвоєння: якщо значення в стовпці DebtRatio менше або дорівнює 1, то нове значення буде дорівнювати добутку стовпців DebtRatio та MonthlyIncome. Якщо ж значення в стовпці DebtRatio більше 1, то нове значення буде дорівнювати поточному значенню стовпця DebtRatio. В кінці, за допомогою команди print(), виводиться стовпець DebtRatio зі зміненими значеннями.

Завдання 6. Змініть ім'я стовпчика на Debt.

Лістинг програми:

```
df = pd.DataFrame(data)
df.rename(columns={'DebtRatio': 'Debt'}, inplace=True)

data.head()
```

```
[63] df = pd.DataFrame(data)
     df.rename(columns={'DebtRatio': 'Debt'}, inplace=True)

     data.head()
```

	Id	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	Debt	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberOfTimes90Day
0	1	1	0.766127	45	2	7323.197016	9120.0	13	
1	2	0	0.957151	40	0	316.878123	2600.0	4	
2	3	0	0.658180	38	1	258.914887	3042.0	2	
3	4	0	0.233810	30	0	118.963951	3300.0	5	
4	5	0	0.907239	49	1	1584.975094	63588.0	7	

Рис. 7. Результат заміни імені стовпчика на Debt

У цьому коді створюється новий DataFrame df, який містить копію даних з вихідного DataFrame data. Потім за допомогою методу rename() виконується перейменування стовпця DebtRatio на Debt. Для цього використовується словник, де ключ - це поточна назва стовпця, а значення - нова назва стовпця. В результаті стовпець DebtRatio у DataFrame data не змінюється, оскільки ми змінюємо лише його копію у DataFrame df. Функція head() виводить перші 5 рядків DataFrame data.

Завдання 7. Обчисліть щомісячний дохід та привласніть всім клієнтам з невідомим доходом отримане число.

Лістинг програми:

```
mean_income = df['MonthlyIncome'].mean()
df.loc[data['MonthlyIncome'].isnull(), 'MonthlyIncome'] = mean_income
df.head()
```

```
[70] mean_income = df['MonthlyIncome'].mean()
     df.loc[data['MonthlyIncome'].isnull(), 'MonthlyIncome'] = mean_income
     df.head()
```

	Id	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	Debt	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberOfTimes90Day
0	1	1	0.766127	45	2	7323.197016	9120.0	13	
1	2	0	0.957151	40	0	316.878123	2600.0	4	
2	3	0	0.658180	38	1	258.914887	3042.0	2	
3	4	0	0.233810	30	0	118.963951	3300.0	5	
4	5	0	0.907239	49	1	1584.975094	63588.0	7	

Рис. 8. Результат обчислення щомісячного доходу клієнтів

У цьому коді обчислюється середнє значення стовпця MonthlyIncome за допомогою методу mean() з Pandas, і результат зберігається у змінну mean_income. Далі, за допомогою методу loc[] виконується вибірка тих рядків, де значення стовпця MonthlyIncome є пропущеними (null), та виконується їх заміна на середнє значення mean_income. На останок, функція head() виводить перші 5 рядків зміненого DataFrame df.

Завдання 8. Використовуйте метод groupby, оцініть ймовірність неповернення кредиту (SeriousDlqin2yrs=1) для різних кількості утриманців (NumberOfDependents). Проробіть аналогічну процедуру для різних значень стовпчика NumberRealEstateLoansOrLines.

Лістинг програми:

```
df['NumberOfDependents'].fillna(0, inplace=True)
df['NumberRealEstateLoansOrLines'].fillna(0, inplace=True)

dependents_prob = df['SeriousDlqin2yrs'].groupby(df['NumberOfDependents']).mea
n()
print(dependents_prob)

realestate_prob = df['SeriousDlqin2yrs'].groupby(df['NumberRealEstateLoansOrLi
nes']).mean()
print(realestate_prob)
```

```
NumberOfDependents
0.0    0.041667
1.0    0.089844
2.0    0.110465
3.0    0.057143
4.0    0.033333
5.0    0.000000
6.0    0.000000
8.0    0.000000
Name: SeriousDlqin2yrs, dtype: float64
NumberRealEstateLoansOrLines
0    0.056863
1    0.048729
2    0.063158
3    0.145455
4    0.105263
5    0.000000
6    1.000000
8    0.000000
Name: SeriousDlqin2yrs, dtype: float64
```

Рис. 9. Результат оцінки ймовірності неповернення кредиту

У цьому коді виконується заміна пропущених значень (NaN) у стовпцях NumberOfDependents та NumberRealEstateLoansOrLines на 0. Це робиться за допомогою методу fillna() з Pandas. Потім, за допомогою методу groupby() виконується групування рядків за значеннями відповідних стовпців, а потім за допомогою методу mean() визначається середня ймовірність неповернення кредиту (SeriousDlqin2yrs) для різних категорій кількості утриманців та кількості кредитів на нерухомість. Результати виводяться на екран за допомогою функції print().

Завдання 9а. Побудуйте графік розсіювання на вісях age та Debt. Синім відміттє клієнтів без серйозних заборгованостей (SeriousDlqin2yrs = 0) та червоним - боржників (SeriousDlqin2yrs = 1).

Лістинг програми:

```
# фільтруємо дані за значенням SeriousDlqin2yrs
no_default = df[df['SeriousDlqin2yrs'] == 0]
default = df[df['SeriousDlqin2yrs'] == 1]

# побудова графіку розсіювання
plt.scatter(no_default['age'], no_default['Debt'], c='blue', label='No Default
')
plt.scatter(default['age'], default['Debt'], c='red', label='Default')
plt.xlabel('Age')
plt.ylabel('Debt')
plt.legend()
plt.show()
```

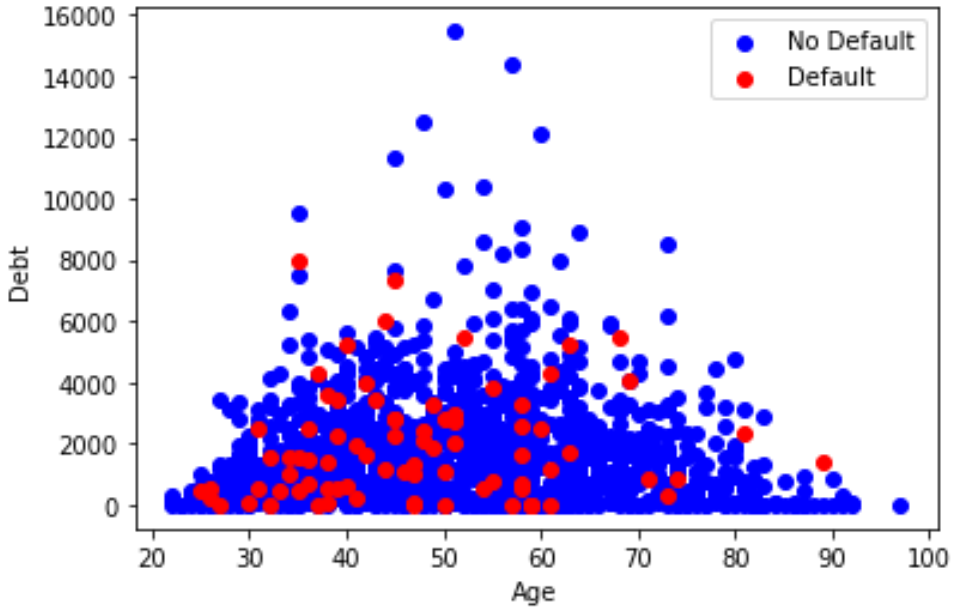

Рис. 10. Результат побудови графіка розсіювання на вісях age та Debt

Цей код фільтрує дані у два окремі дата фрейми на основі значення стовпця «SeriousDlqin2yrs»: «no_default» містить усі рядки, де це значення дорівнює 0, а «default» містить усі рядки, де це значення дорівнює 1. Потім створюється графік за допомогою бібліотеки matplotlib, де вісь x представляє стовпець «age», вісь y — стовпець «Debt», а два набори точок наносяться різними кольорами: синій для дата фрейму «no_default», і червоний для дата фрейму «default».

Завдання 9b. Побудуйте на одному графіку дві нормовані щільності розподілення: червону – для місяіного доходу клієнтів з заборгованостями, синю - для місяіного доходу клієнтів без заборгованостей. По вісі абсцис відобразіть значення до 25000.

Лістинг програми:

```
# Вибираємо клієнтів з заборгованістю та без заборгованостей
with_debt = df.loc[df['SeriousDlqin2yrs'] == 1, 'MonthlyIncome']
without_debt = df.loc[df['SeriousDlqin2yrs'] == 0, 'MonthlyIncome']

# Обчислюємо параметри нормального розподілу
mean_with_debt, std_with_debt = with_debt.mean(), with_debt.std()
mean_without_debt, std_without_debt = without_debt.mean(), without_debt.std()

# Задаємо діапазон значень для графіка
x = range(25000)

# Рахуємо нормовані щільності
pdf_with_debt = (1/(std_with_debt * np.sqrt(2 * np.pi))) * np.exp(-
0.5 * ((x - mean_with_debt) / std_with_debt)**2)
pdf_without_debt = (1/(std_without_debt * np.sqrt(2 * np.pi))) * np.exp(-
0.5 * ((x - mean_without_debt) / std_without_debt)**2)

# Побудова графіку
plt.plot(x, pdf_with_debt, color='red', label='Debt')
plt.plot(x, pdf_without_debt, color='blue', label='No Debt')
plt.xlabel('Monthly Income')
plt.ylabel('Normalized Density')
plt.title('Monthly Income Distribution')
plt.legend()
plt.show()
```

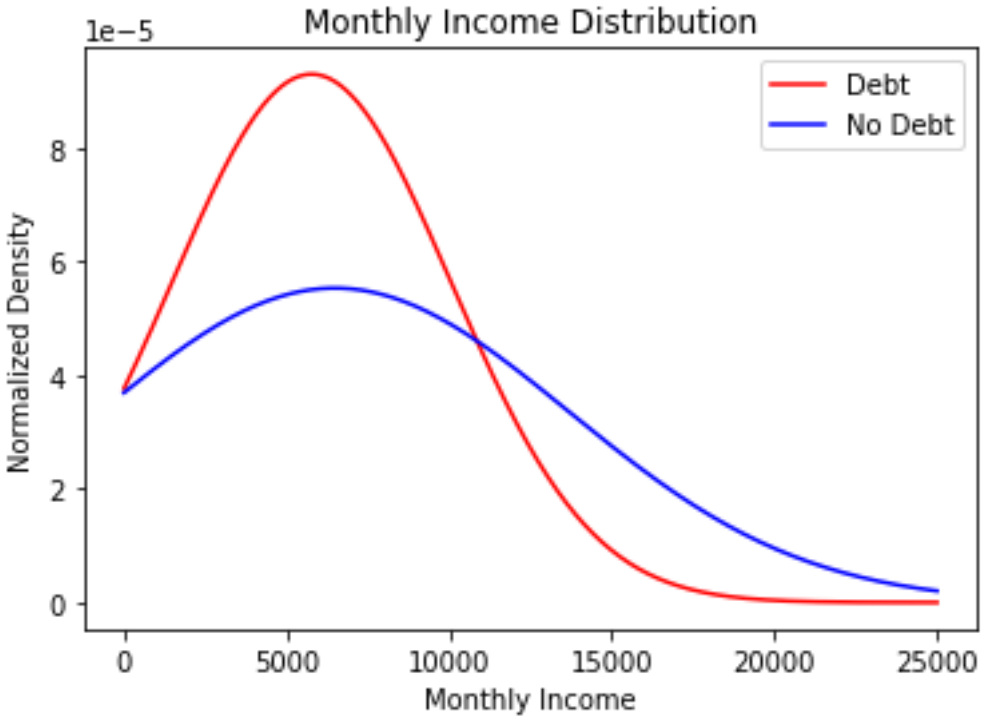

Рис. 11. Результат побудови графіка з двома нормованими щільностями розподілення

Цей код обчислює та порівнює густини розподілу місячного доходу між клієнтами, які мають заборгованості та клієнтами, які їх не мають. Для цього спочатку фільтрується оригінальний датафрейм за значенням SeriousDlqin2yrs, потім обчислюються середнє значення та стандартне відхилення місячного доходу для кожної групи. Наступним кроком задається діапазон значень для графіка та обчислюються нормовані щільності для кожної групи з використанням зазначених середнього значення та стандартного відхилення.

Завдання 9с. Візуалізуйте попарні залежності між небінарними ознаками 'age', 'MonthlyIncome', 'NumberOfDependents'. Обмежте при цьому місячний дохід значенням 25000.

Які закономірності ви можете спостерігати на отриманих графіках?

Лістинг програми:

```
pd.plotting.scatter_matrix(df.loc[df['MonthlyIncome'] <= 25000, ['age', 'Month
lyIncome', 'NumberOfDependents']], alpha=0.2, figsize=(10, 10))
plt.show()
```

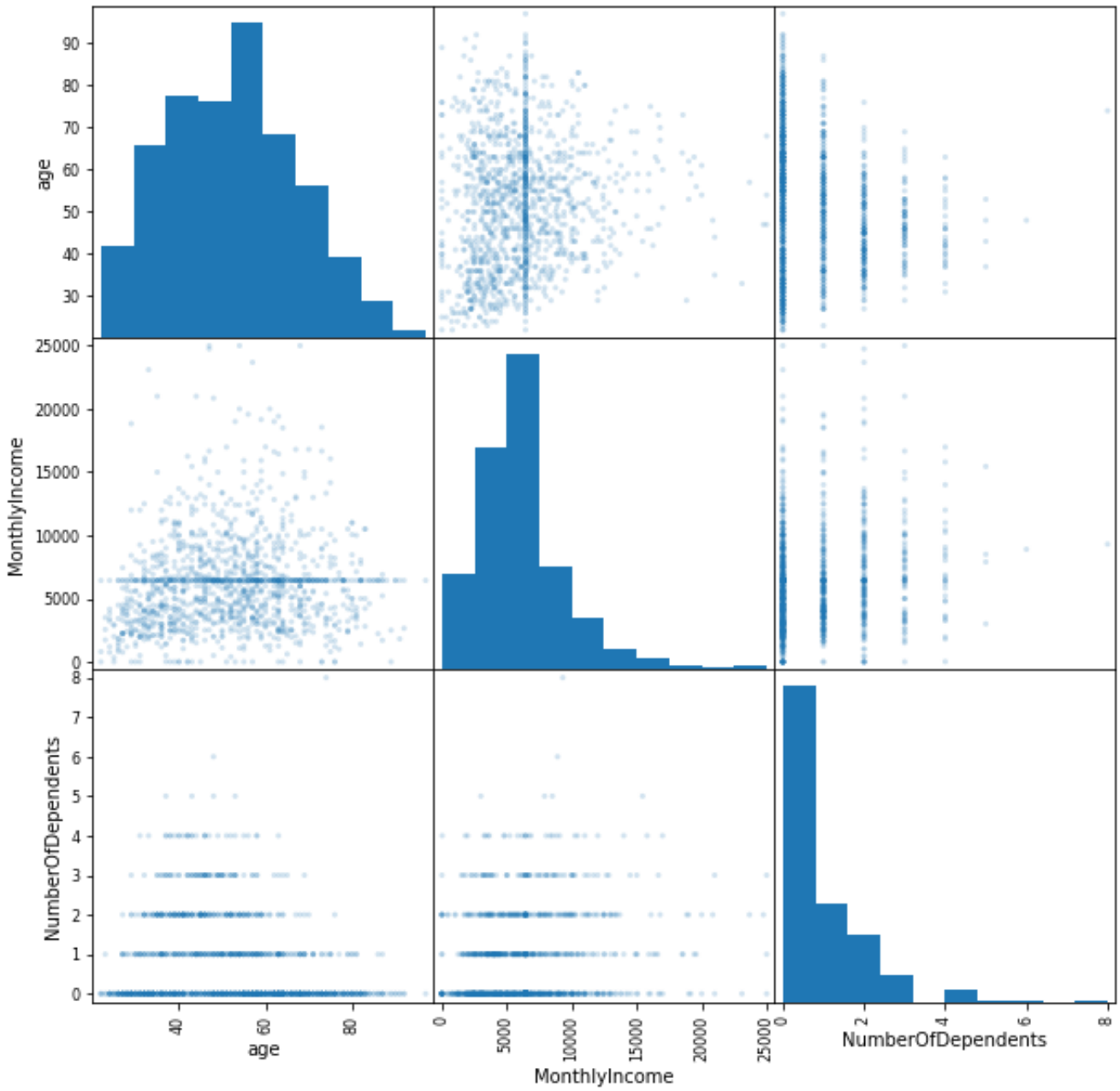

Рис. 12. Результат побудови попарних залежностей між небінарними ознаками

Цей код будує матрицю діаграм розсіювання для обраних змінних з дата-фрейму df. Вибір змінних здійснюється за допомогою індексування за назвами стовпців. У цьому випадку вибрані змінні age, MonthlyIncome та NumberOfDependents. Для зменшення впливу великих значень на графіки, дані з області з MonthlyIncome більше 25000 виключаються з аналізу. alpha встановлює прозорість графіків, а figsize визначає розмір графічного відображення.

На отриманих графіках можна помітити деякі закономірності:

- Зі зростанням віку кількість залежних осіб зазвичай зменшується.
- Між місячним доходом та кількістю залежних осіб може бути слабка залежність, але її важко оцінити, оскільки більшість даних зосереджена в малих значеннях доходу.
- Між місячним доходом та віком не спостерігається явної залежності, але можна помітити зосередження значень високого доходу в середньому віці.

Посилання на Google Colab:
https://drive.google.com/file/d/1QbVKRaxbYy2gnmH0Xl-95l_NGqD73wCl/view?usp=sharing

Висновки: в ході виконання лабораторної роботи було вивчено основні етапів аналізу даних з використанням бібліотеки Pandas в середовищі Python.