МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Лабораторна робота №8

з дисципліни

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

на тему

ВІЗУАЛІЗАЦІЯ ТА ОБРОБКА ДАНИХ ЗА ДОПОМОГОЮ СПЕЦІАЛІЗОВАНИХ БІБЛІОТЕК PYTHON

Виконав:

ст. гр. PI-31

ЛАЗАР В.С.

Прийняв:

ЩЕРБАК С.С.

Львів-2024

Мета роботи:

Розробка додатка для візуалізації CSV-наборів даних за допомогою Matplotlib та базових принципів ООП (наслідування, інкапсуляція, поліморфізм).

Хід роботи:

Завдання 1: Вибір CSV-набору даних

Обрати CSV-набір даних, який треба візуалізувати. Переконатися, що він містить відповідні дані для створення змістовних візуалізацій.

Завдання 2: Завантаження даних з CSV

Написати код для завантаження даних з CSV-файлу в додаток Python. Використовувати бібліотеки, такі як Pandas, для спрощення обробки даних**.**

Завдання 3: Дослідження даних

Визначити екстремальні значення по стовпцям.

Завдання 4: Вибір типів візуалізацій

Визначити, які типи візуалізацій підходять для представлення вибраних наборів даних. Зазвичай це може бути лінійні графіки, стовпчикові діаграми, діаграми розсіювання, гістограми та секторні діаграми.

Завдання 5: Підготовка даних

Попередньо обробити набір даних за необхідністю для візуалізації. Це може включати виправлення даних, фільтрацію, агрегацію або трансформацію.

Завдання 6: Базова візуалізація

Створити базову візуалізацію набору даних, щоб переконатися, що можна відображати дані правильно за допомогою Matplotlib. Розпочати з простої діаграми для візуалізації однієї змінної.

Завдання 7: Розширені візуалізації

Реалізувати більш складні візуалізації, виходячи з характеристик набору. Поекспериментувати з різними функціями Matplotlib та налаштуваннями.

Завдання 8: Декілька піддіаграм

Навчитися створювати кілька піддіаграм в межах одного малюнка для відображення декількох візуалізацій поруч для кращого порівняння.

Завдання 9: Експорт і обмін

Реалізувати функціональність для експорту візуалізацій як зображень (наприклад, PNG, SVG) або інтерактивних веб-додатків (наприклад, HTML).

Код програми:

```
"""The user interface of the lab work"""
import logging
import seaborn as sns
import matplotlib.pyplot as plt
from Data.Shared.classes.data_io import DataIO
from Data.Lab8.BLL.classes.data_processor import DataProcessor



logger = logging.getLogger(__name__)
logging.basicConfig(filename='Logs/logs.log', encoding='utf-8', level=logging.DEBUG)

class Console:
    """The console class of this lab work"""
    instance = None

    def __new__(cls):
        if cls.instance is None:
            cls.instance = super(Console, cls).__new__(cls)
        return cls.instance

    def __init__(self, file_path='data.csv', group_column='Category',
```

```
            value_column='Purchase Amount (USD)'):
        self.file_path = f"Data/Lab8/Imports/{file_path}"
        self.data = None
        self.group_column = group_column
        self.value_column = value_column
        self.main()

    @staticmethod
    def annotate(ax, group_column, data, is_boxplot=False):
        """Creates the annotations for the plots"""
        unique_categories = data[group_column].unique()
        for i, element in enumerate(ax.patches[:len(unique_categories)]):
            category_name = unique_categories[i]
            center_x = element.get_x() + element.get_width() / 2  # Центр стовпця
            height_y = element.get_height()  # Висота стовпця
            if is_boxplot:
                center_x = element.get_x() + element.get_width() / 2
                height_y = element.get_y() + element.get_height()
            ax.text(center_x, height_y + 0.5, category_name, ha='center', fontsize=10)

    def basic_visualization(self):
        """Outputs basic visualization of the imported data"""
                    prepared_data = DataProcessor.prepare_data(self.data, self.group_column,
self.value_column)
        fig, ax = plt.subplots(figsize=(10, 6))
                    ax.bar(prepared_data[self.group_column], prepared_data[self.value_column],
color='skyblue')
        ax.set_xlabel(self.group_column)
        ax.set_ylabel(self.value_column)
        ax.set_title(f"Average {self.value_column} by {self.group_column}")
        ax.tick_params(axis='x', rotation=45)
        # noinspection PyTypeChecker
        self.annotate(ax, self.group_column, prepared_data)
        DataIO.save_visualization(fig, "basic_visualization")
        plt.tight_layout()
```

```
        plt.show()
        logger.info("[Lab 8] Ended making basic graph")

    def advanced_visualization(self):
        """Outputs advanced histogram of the imported data"""
                    prepared_data = DataProcessor.prepare_data(self.data, self.group_column,
self.value_column)
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.histplot(prepared_data[self.value_column], kde=True, ax=ax, color='green')
        ax.set_title(f"Histogram of {self.value_column}")
        ax.set_xlabel(self.value_column)
        ax.set_ylabel('Frequency')
        DataIO.save_visualization(fig, "advanced_visualization_histogram")
        plt.tight_layout()
        plt.show()
        logger.info("[Lab 8] Ended making advanced graph")

    def multiple_subplots(self):
        """Outputs multiple subplots of the imported data"""
                    prepared_data = DataProcessor.prepare_data(self.data, self.group_column,
self.value_column)
        fig, axs = plt.subplots(1, 2, figsize=(14, 6))
        sns.barplot(x=self.group_column, y=self.value_column,
                data=prepared_data, ax=axs[0], color='skyblue')
        axs[0].set_title(f"Barplot of {self.value_column}")
        axs[0].set_xlabel(self.group_column)
        axs[0].set_ylabel(self.value_column)
        axs[0].tick_params(axis='x', rotation=45)
        self.annotate(axs[0], self.group_column, prepared_data)
        sns.histplot(prepared_data[self.value_column], kde=True, ax=axs[1], color='green')
        axs[1].set_title(f"Histogram of {self.value_column}")
        axs[1].set_xlabel(self.value_column)
        axs[1].set_ylabel('Frequency')
        DataIO.save_visualization(fig, "multiple_subplots_with_categories")
        plt.tight_layout()
```

```
    plt.show()
    logger.info("[Lab 8] Ended making multiple graphs")

def main(self):
    """The main menu of this lab work"""
    try:
        self.data = DataIO.load_data(self.file_path)
    except FileNotFoundError as e:
        print(e)
    while True:
        prompt = input("1 - Extreme values\n"
                       "2 - Display basic graph\n"
                       "3 - Display advanced graph\n"
                       "4 - Display multiple graphs\n"
                       "Your choice: ")
        match prompt:
            case '1':
                logger.info("[Lab 8] Found extreme values")
                DataProcessor.find_extreme_values(self.data)
            case '2':
                logger.info("[Lab 8] Started making basic graph")
                self.basic_visualization()
            case '3':
                logger.info("[Lab 8] Started making advanced graph")
                self.advanced_visualization()
            case '4':
                logger.info("[Lab 8] Started making multiple graphs")
                self.multiple_subplots()
            case _:
                return
```

На рис. 1-4 зображено результат виконання програми:

```
Data loaded successfully!
1 - Extreme values
2 - Display basic graph
3 - Display advanced graph
4 - Display multiple graphs
Your choice: 1
Column: Customer ID
Minimum value: 1
Maximum value: 3900
Column: Age
Minimum value: 18
Maximum value: 70
Column: Purchase Amount (USD)
Minimum value: 20
Maximum value: 100
Column: Review Rating
Minimum value: 2.5
Maximum value: 5.0
Column: Previous Purchases
Minimum value: 1
Maximum value: 50
```

Рис. 1. Приклад роботи функції пошуку екстремальних значень таблиці

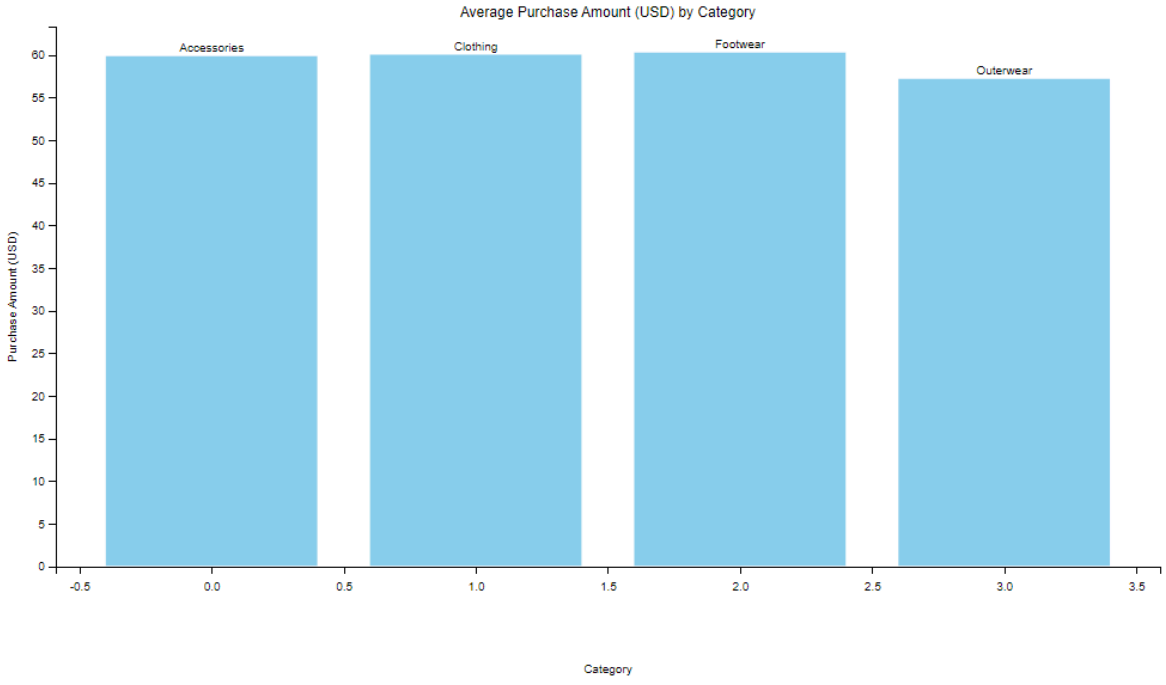

Рис. 2. Приклад базової візуалізації даних таблиці

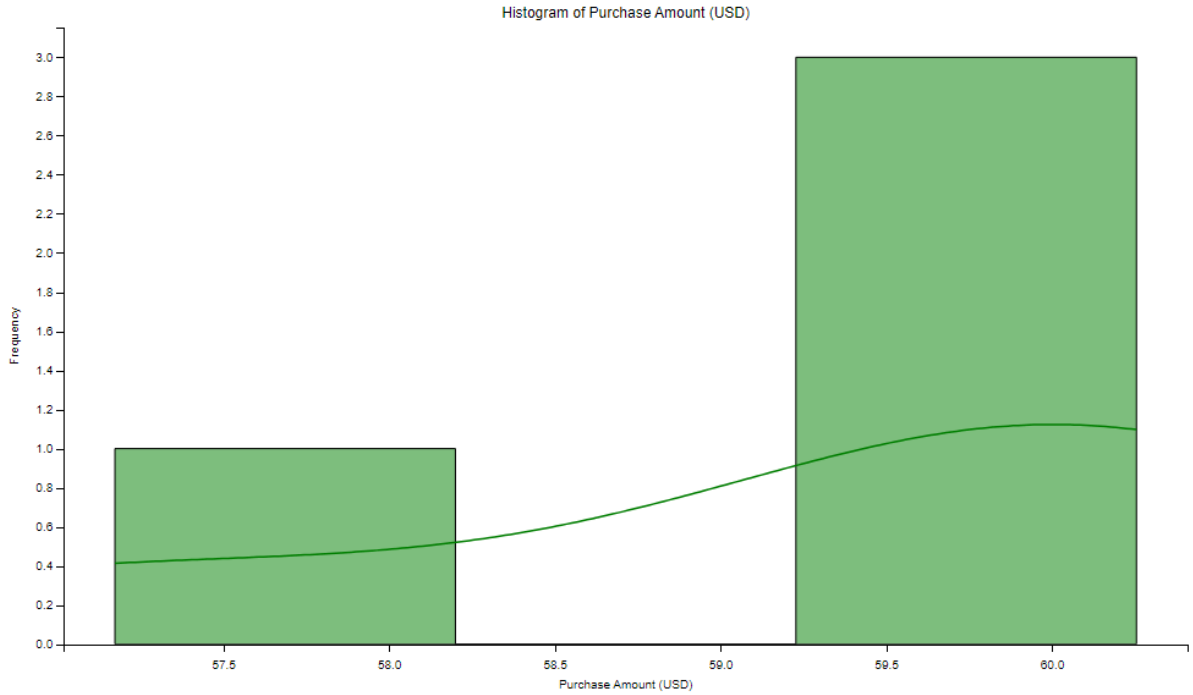

Рис. 3. Приклад розширеної візуалізації даних таблиці

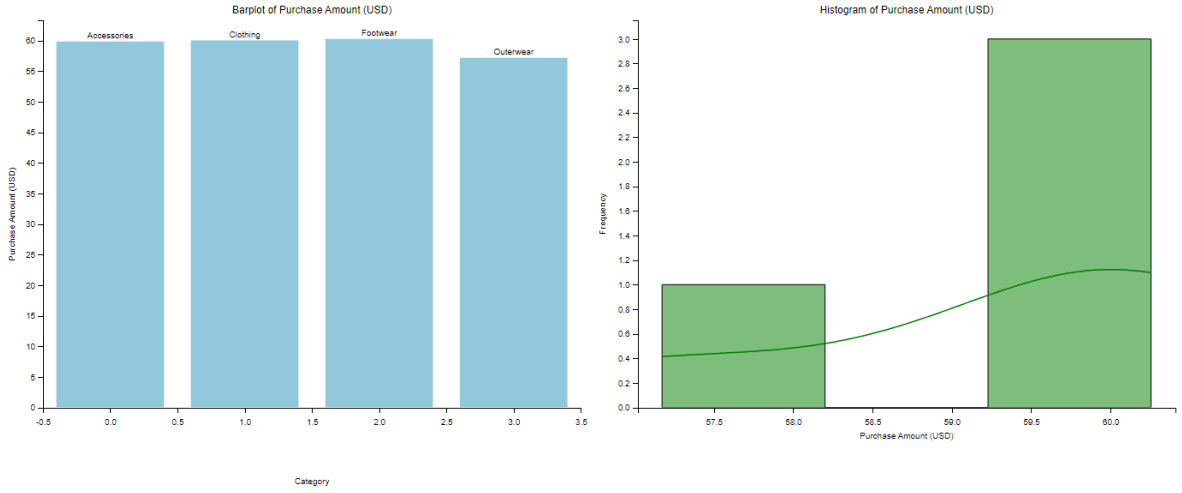

Рис. 4. Приклад візуалізації кількох піддіаграм даних таблиці

Посилання на Github: [PaperGlit/Python_Lab_8]

Висновок:

Виконавши ці завдання, я створив багатофункціональний додаток для візуалізації CSV-наборів даних за допомогою Matplotlib. Цей проект покращив мої навички візуалізації даних, дозволяючи досліджувати результати з різноманітними наборами даних