МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра інформаційних систем та мереж

Лабораторна робота №8

з дисципліни: «Спеціалізовані мови програмування»

на тему: «Візуалізація та обробка даних за допомогою спеціалізованих бібліотек Python»

Виконала:
студентка групи ІТ-32
Моляща Ю.А.
Прийняв:
Щербак С.С.

Львів - 2023

Лабораторна робота №8

«Візуалізація та обробка даних за допомогою спеціалізованих бібліотек Python»

Мета роботи: розробка додатка для візуалізації CSV-наборів даних за допомогою Matplotlib та базових принципів ООП (наслідування, інкапсуляція, поліморфізм)

Завдання на лабораторну роботу.

Завдання 1: Вибір CSV-набору даних

Оберіть CSV-набір даних, який ви хочете візуалізувати. Переконайтеся, що він містить відповідні дані для створення змістовних візуалізацій.

Завдання 2: Завантаження даних з CSV

Напишіть код для завантаження даних з CSV-файлу в ваш додаток Python. Використовуйте бібліотеки, такі як Pandas, для спрощення обробки даних.

Завдання 3: Дослідження даних

Визначте екстремальні значення по стовцям

Завдання 4: Вибір типів візуалізацій

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

Завдання 6: Базова візуалізація

Створіть базову візуалізацію набору даних, щоб переконатися, що ви можете відображати дані правильно за допомогою Matplotlib. Розпочніть з простої діаграми для візуалізації однієї змінної.

Завдання 7: Розширені візуалізації

Реалізуйте більш складні візуалізації, виходячи з характеристик набору. Поекспериментуйте з різними функціями Matplotlib та налаштуваннями.

Завдання 8: Декілька піддіаграм

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

Завдання 9: Експорт і обмін

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

Хід роботи.

Код програми:

data_preprocessing.py

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

# Filter out UserWarnings related to figure layout changes
warnings.filterwarnings("ignore", category=UserWarning)


class DataExploration():
    """Class for data exploration operations"""
      def __init__(self, csv_file, folder_path_plots, folder_path_datasets) -> None:
        if not os.path.isfile(csv_file):
             raise FileNotFoundError(f"The specified CSV file '{csv_file}' does not exist.")

        self.csv_file = pd.read_csv(csv_file)
        self.csv_file_name = os.path.splitext(os.path.basename(csv_file))[0]
        self.folder_path_plots = folder_path_plots
        self.folder_path_datasets = folder_path_datasets

    def data_exploration(self):
        """Perform data exploration"""
        print("\nData exploration:")

        print("\nFirst 5 rows of data:")
        print(self.explore_data_first())
        print("\nLast 5 rows of data:")
        print(self.explore_data_last())
```

```
        print("\nColumns of data:")
        print(self.explore_data_columns())
        print("\nShape of data:")
        print(self.explore_data_shape())
        print("\nExtreme values of data:")
        self.get_extreme_values()

    def explore_data_first(self):
        """Get first 5 rows of data"""
        return self.csv_file.head()

    def explore_data_last(self):
        """Get last 5 rows of data"""
        return self.csv_file.tail()

    def explore_data_columns(self):
        """Get columns of data"""
        return self.csv_file.columns

    def explore_data_shape(self):
        """Get shape of data"""
        return self.csv_file.shape

    def get_extreme_values(self):
        """Get extreme values of data"""
        for column in self.csv_file.columns:
            if self.csv_file[column].dtype == 'object':
                continue
            min_value = self.csv_file[column].min()
            max_value = self.csv_file[column].max()
            median = self.csv_file[column].median()
                 print(f"Column: {column}, min: {min_value}, max: {max_value},
median: {median}")
```

```
class DataCleaning(DataExploration):
    """Class for data cleaning operations"""
      def __init__(self, csv_file, folder_path_plots, folder_path_datasets) ->
None:
        super().__init__(csv_file, folder_path_plots, folder_path_datasets)

    def data_cleaning(self):
        """Perform data cleaning"""
        print("\nData cleaning:")

        if self.csv_file.isnull().values.any():
            print("Missing values found")
            cleaned_file_path = self.drop_missing_values()
            print(f"Missing values dropped and saved to {cleaned_file_path}")
        else:
            print("No further cleaning required.")

        print("Data cleaning completed")

    def drop_missing_values(self):
        """Drop missing values and return cleaned file path"""
        if self.csv_file.isnull().values.any():
            self.csv_file.dropna(inplace=True)

            cleaned_file_path = os.path.join(
                self.folder_path_datasets,
                f"{self.csv_file_name}_cleaned.csv",
            )

            self.csv_file.to_csv(cleaned_file_path, index=False)
            return cleaned_file_path


class DataVisualization(DataExploration):
```

```
    """Class for data visualization operations"""
      def __init__(self, csv_file, folder_path_plots, folder_path_datasets) ->
None:
        super().__init__(csv_file, folder_path_plots, folder_path_datasets)

    def data_visualization(self):
        """Perform data visualization"""
        print("\nData visualization:")
        pairplot_path = self.plot_pairplot()
        print(f"Pairplot plotted and saved to {pairplot_path}")
        heatmap_path = self.plot_heatmap()
        if heatmap_path:
            print(f"Heatmap plotted and saved to {heatmap_path}")
        else:
            print("No numeric columns found for heatmap.")

    def plot_pairplot(self):
        """Plot pairplot"""
        sns.pairplot(self.csv_file)
        pairplot_path = f'{self.csv_file_name}_pairplot'
        pairplot_path = self.export_plot_to_png(plt, pairplot_path)
        return pairplot_path

    def plot_heatmap(self):
        """Plot heatmap for numeric columns"""
        numeric_df = self.csv_file.select_dtypes(include=[np.number])

        if not numeric_df.empty:
            plt.figure(figsize=(10, 8))
                    sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm',
fmt=".2f")
            plt.title("Correlation Heatmap")

            heatmap_path = f'{self.csv_file_name}_heatmap'
            heatmap_path = self.export_plot_to_png(plt, heatmap_path)
```

```
            return heatmap_path
        else:
            return None

    def export_plot_to_png(self, plt, file_name):
        """Export plot to PNG file with exception handling"""
        file_path = f"{self.folder_path_plots}{file_name}.png"
        try:
            plt.savefig(file_path, bbox_inches='tight')
        except Exception as e:
            print(f"Error exporting plot: {e}")
```

main.py

```
import sys
import os

# Include the parent directory in the system's import path
current_dir = os.path.dirname(os.path.abspath(__file__))
parent_dir = os.path.abspath(os.path.join(current_dir, '..'))
sys.path.append(parent_dir)

from lab8.data_preprocessing import DataExploration
from lab8.data_preprocessing import DataCleaning
from lab8.data_preprocessing import DataVisualization

FOLDER_PATH_PLOTS = 'source/lab8/plots/'
FOLDER_PATH_DATASETS = 'source/lab8/datasets/'

# Checking if directories exist and creating them if not
for directory in [FOLDER_PATH_PLOTS, FOLDER_PATH_DATASETS]:
    if not os.path.exists(directory):
        os.makedirs(directory)

def main():
```

```
    print("Provide file path or press enter to use the default file path.")
    user_input = input("File path: ")

    if user_input:
        csv_path = user_input
    else:
        csv_path = 'source/lab8/datasets/USA_Housing.csv'
        print(f"Using default file path: {csv_path}")

    try:
                    explorer = DataExploration(csv_path, FOLDER_PATH_PLOTS,
FOLDER_PATH_DATASETS)
                       cleaner = DataCleaning(csv_path, FOLDER_PATH_PLOTS,
FOLDER_PATH_DATASETS)
                  visualizer = DataVisualization(csv_path, FOLDER_PATH_PLOTS,
FOLDER_PATH_DATASETS)

        explorer.data_exploration()
        cleaner.data_cleaning()
        visualizer.data_visualization()

        print("Data processing completed successfully!")

    except FileNotFoundError:
        print(f"Error: The specified CSV file '{csv_path}' does not exist.")

    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

Висновок: під час виконання лабораторної роботи було розроблено додаток для візуалізації CSV-наборів даних за допомогою Matplotlib та базових принципів ООП (наслідування, інкапсуляція, поліморфізм)