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. ІТ-32
Шоха А.А.
Прийняв:
Щербак С.С.

Львів-2023

**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

Класи що виконують завдання лабораторної роботи:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import sys
import warnings
# Filter out UserWarnings related to figure layout changes
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
sys.path.append('/Users/admin/Desktop/lpnu/5 сем/Specialised programming
languages/source')
from config.paths_config import BOOKS_FILE_PATH, RATINGS_FILE_PATH,
USERS_FILE_PATH, CLEAN_DATA_FILE_PATH, PLOT_DIR
class PlotService:
    """A simple plot service class."""
    def __init__(self):
        """Initialize a PlotService object."""
                self.data_preprocessor = DataPreprocessor(BOOKS_FILE_PATH,
RATINGS_FILE_PATH, USERS_FILE_PATH)
        self.data_visualizer = None
    def preprocess_data(self):
        """Preprocess data"""
        self.data_preprocessor.preprocess_data(CLEAN_DATA_FILE_PATH)

    def visualize_data(self):
        """Visualize data"""
        self.data_visualizer = DataVisualizer(CLEAN_DATA_FILE_PATH)
        self.data_visualizer.visualize_data()


class DataPreprocessor:
    def __init__(self, BOOKS_FILE_PATH, RATINGS_FILE_PATH, USERS_FILE_PATH):
        self.books = pd.read_csv(BOOKS_FILE_PATH, low_memory=False)
        self.ratings = pd.read_csv(RATINGS_FILE_PATH)
        self.users = pd.read_csv(USERS_FILE_PATH)
        self.books_ratings = None
        self.books_ratings_users = None

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

```
    def remove_year_of_publication_with_string_value(self):
        """Remove year of publication with string value"""
         temp = (self.books['Year-Of-Publication'] == 'DK Publishing Inc') |
(self.books['Year-Of-Publication'] == 'Gallimard')
        self.books = self.books.drop(self.books[temp].index)
        self.books[(self.books['Year-Of-Publication'] == 'DK Publishing Inc')
| (self.books['Year-Of-Publication'] == 'Gallimard')]

    def convert_year_of_publication_to_int(self):
        """Convert year of publication to int"""
                                      self.books['Year-Of-Publication']    =
self.books['Year-Of-Publication'].astype(int)

    def remove_image_url_column(self):
        """Removing Image-URL column of all sizes"""
        self.books.drop(labels=['Image-URL-S', 'Image-URL-M', 'Image-URL-L'],
axis=1, inplace=True)

    def get_number_of_unique_values(self):
        """Get number of unique values"""
                              print("Number    of    Book    ISBN    numbers:",
len(self.books['ISBN'].unique()))
                                    print("Number    of    book    titles:",
len(self.books['Book-Title'].unique()))
                                   print('Number    of    book    authors:',
len(self.books['Book-Author'].unique()))
                                 print('Number    of    Publication    Years:',
len(self.books['Year-Of-Publication'].unique()))
                                  print('Number    of    publisher    names:',
len(self.books['Publisher'].unique()))

    def merge_books_ratings_users(self):
        """Merge books, ratings, and users data"""
        self.merge_books_ratings()
          self.books_ratings_users = pd.merge(self.books_ratings, self.users,
on='User-ID').dropna()

    def merge_books_ratings(self):
        """Merge books and ratings data"""
          self.books_ratings = pd.merge(self.books, self.ratings, on='ISBN',
how='left').dropna()

    def drop_nan_values(self):
        """Drop NaN values"""
        self.books.dropna()
        self.ratings.dropna()
        self.users.dropna()

    def preprocess_data(self, clean_data_file_path):
        """Preprocess data"""
        try:
            self.remove_year_of_publication_with_string_value()
            self.convert_year_of_publication_to_int()
```

```
            self.remove_image_url_column()
            self.drop_nan_values()
            self.merge_books_ratings()
            self.merge_books_ratings_users()
                        self.books_ratings_users.to_csv(clean_data_file_path,
index=False)
            self.get_number_of_unique_values()
            self.get_extreme_values()
        except Exception as e:
            print(f"An error occurred during data preprocessing: {str(e)}")
class DataVisualizer:
    def __init__(self, data_file_path, save_dir=PLOT_DIR):
        self.data = pd.read_csv(data_file_path)
        self.save_dir = save_dir

    def plot_ratings_distribution(self):
        """Plot ratings distribution"""
        sns.countplot(x='Book-Rating', data=self.data)
        plt.title('Distribution of Book Ratings')
                                        plt.savefig(os.path.join(self.save_dir,
'book_ratings_distribution.png'))
        plt.close()

    def plot_publication_year_distribution(self):
        """Plot publication year distribution"""
                                                        yearly_ratings       =
self.data.groupby('Year-Of-Publication')['Book-Rating'].mean()
        plt.figure(figsize=(10, 6))
        plt.xlim(1800, 2020)
        yearly_ratings.plot()
        plt.title('Average Ratings per Year of Publication')
        plt.xlabel('Year of Publication')
        plt.ylabel('Average Rating')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.savefig(os.path.join(self.save_dir, 'ratings_per_year_line.png'))
        plt.close()

    def plot_user_age_distribution(self):
        """Plot user age distribution"""
        plt.figure(figsize=(10, 6))
        sns.histplot(self.data['Age'].dropna(), bins=50, kde=False)
        plt.title('User Age Distribution')
        plt.xlabel('Age')
        plt.ylabel('Count')
        plt.xlim(0, 100)
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'user_age_distribution_adjusted.png'))
        plt.close()

    def plot_ratings_by_top_publishers(self):
        """Plot ratings by top publishers"""
```

```
        top_publishers = self.data['Publisher'].value_counts().head(10).index
                                                        publisher_ratings      =
self.data[self.data['Publisher'].isin(top_publishers)].groupby('Publisher')['
Book-Rating'].mean()
        plt.figure(figsize=(10, 6))
        publisher_ratings.plot(kind='bar')
        plt.title('Average Ratings of Top Publishers')
        plt.ylabel('Average Rating')
        plt.xticks(rotation=45)
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'ratings_by_top_publishers.png'))
        plt.close()

    def plot_books_published_each_year(self):
        """Plot books published each year"""
                                                          books_per_year       =
self.data['Year-Of-Publication'].value_counts().sort_index()
        plt.figure(figsize=(10, 6))
        books_per_year.plot()
        plt.title('Number of Books Published Each Year')
        plt.xlabel('Year of Publication')
        plt.ylabel('Number of Books')
        plt.xlim(1950, 2020)
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'books_published_each_year.png'))
        plt.close()

    def plot_book_counts_by_authors(self):
        """Plot book counts by authors"""
        books_by_author = self.data['Book-Author'].value_counts()
        plt.figure(figsize=(10, 6))
        sns.histplot(books_by_author, bins=10000, kde=False)
        plt.title('Distribution of Book Counts by Authors')
        plt.xlabel('Number of Books')
        plt.ylabel('Count of Authors')
        plt.xlim(0, 20)
        plt.xticks(np.arange(0, 20, 1))
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'book_counts_by_authors.png'))
        plt.close()

    def plot_average_rating_by_age_group(self):
        """Plot average rating by age group"""
          self.data['Age Group'] = pd.cut(self.data['Age'], bins=[0, 18, 30,
50, 100], labels=['0-18', '19-30', '31-50', '51-100'])
        age_rating = self.data.groupby('Age Group')['Book-Rating'].mean()
        plt.figure(figsize=(10, 6))
        age_rating.plot(kind='bar')
        plt.title('Average Rating by Age Group')
        plt.ylabel('Average Rating')
```

```
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'average_rating_by_age_group.png'))
        plt.close()

    def plot_lineplot_avg_ratings_over_years(self):
        """Plot line plot of average ratings over years"""
                                                    avg_ratings_over_years      =
self.data.groupby('Year-Of-Publication')['Book-Rating'].mean()
        plt.figure(figsize=(12, 6))
        avg_ratings_over_years.plot()
        plt.title('Average Ratings Over Years')
        plt.xlabel('Year of Publication')
        plt.ylabel('Average Rating')
        plt.xlim(1800, 2020)
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'lineplot_avg_ratings_over_years.png'))
        plt.close()

    def plot_piechart_books_by_top_authors(self):
        """Plot pie chart of books by top authors"""
            top_authors = self.data['Book-Author'].value_counts().head(5)   #
Taking the top 5 authors
        plt.figure(figsize=(8, 8))
        top_authors.plot(kind='pie', autopct='%1.1f%%')
        plt.title('Distribution of Books by Top Authors')
        plt.ylabel('')  # Hide the y-label
        plt.tight_layout()
                                        plt.savefig(os.path.join(self.save_dir,
'piechart_books_by_top_authors.png'))
        plt.close()
    def plot_ratings_per_year_line(self):
        """Plot ratings per year line"""
                                                      yearly_ratings       =
self.data.groupby('Year-Of-Publication')['Book-Rating'].mean()
        plt.figure(figsize=(10, 6))
        plt.xlim(1800, 2020)
        yearly_ratings.plot()
        plt.title('Average Ratings per Year of Publication')
        plt.xlabel('Year of Publication')
        plt.ylabel('Average Rating')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.savefig(os.path.join(self.save_dir, 'ratings_per_year_line.png'))
        plt.close()

    def plot_subplots(self):
        """Create a figure and a set of subplots"""
        df = self.data
        fig, axs = plt.subplots(2, 2, figsize=(12, 8))
        # Subplot 1: Box Plot of Ratings by Top 5 Publishers
        top_publishers = df['Publisher'].value_counts().head(5).index
```

```
                                sns.boxplot(x='Publisher',    y='Book-Rating',
data=df[df['Publisher'].isin(top_publishers)], ax=axs[0, 0])
        axs[0, 0].set_title('Ratings by Top 5 Publishers')
        axs[0, 0].set_xticklabels(axs[0, 0].get_xticklabels(), rotation=45)
        # Subplot 2: Scatter Plot of Book Ratings vs. Publication Year
           sns.scatterplot(x='Year-Of-Publication', y='Book-Rating', data=df,
ax=axs[0, 1])
        axs[0, 1].set_xlim(1900, 2020)
        axs[0, 1].set_title('Book Ratings vs. Publication Year')
         # Subplot 3: Line Plot of Number of Books Published Each Year (Last
200 Years)
                                                          books_per_year       =
df['Year-Of-Publication'].value_counts().sort_index()
        axs[1, 0].plot(books_per_year)
        axs[1, 0].set_xlim(1800, 2020)
          axs[1, 0].set_title('Number of Books Published Each Year (Last 200
Years)')
        axs[1, 0].set_xlabel('Year of Publication')
        axs[1, 0].set_ylabel('Number of Books')
        # Subplot 4: Pie Chart of Book Counts by Top 5 Authors
        top_authors = df['Book-Author'].value_counts().head(5)
                      axs[1,   1].pie(top_authors,   labels=top_authors.index,
autopct='%1.1f%%')
        axs[1, 1].set_title('Book Counts by Top 5 Authors')
        plt.tight_layout()
        plt.savefig(os.path.join(self.save_dir, 'different_subplots.png'))
        plt.close()

    def visualize_data(self):
        """Visualize data"""
        self.plot_ratings_distribution()
        self.plot_publication_year_distribution()
        self.plot_user_age_distribution()
        self.plot_ratings_by_top_publishers()
        self.plot_books_published_each_year()
        self.plot_book_counts_by_authors()
        self.plot_average_rating_by_age_group()
        self.plot_lineplot_avg_ratings_over_years()
        self.plot_piechart_books_by_top_authors()
        self.plot_ratings_per_year_line()
        self.plot_subplots()
```

Приклад візуалізації даних:

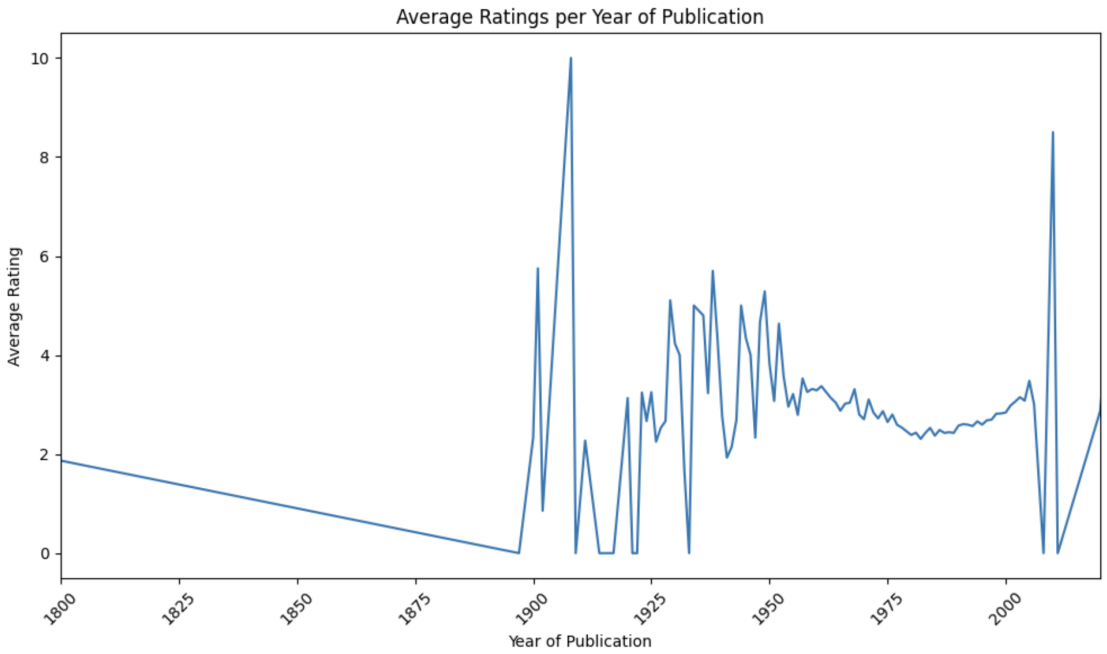


*Рис. 1. Приклад візуалізації даних*

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