МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №8

з курсу

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Виконав студент

групи ІТ-21сп

**Резніченко Р.Д.**

Прийняв

**Щербак С.С.**

Львів - 2023

**Мета:** 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**

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).

Код виконаних завдань представлено нижче.

Клас DataProcessor:

import os

import seaborn as sns

import pandas as pd

from matplotlib import pyplot as plt

class DataProcessor:

# Constructor for the DataProcessor class

def \_\_init\_\_(self):

# Get the current file directory and the relative path to the data file

relative\_file\_path = os.path.join('..', 'Lb8', 'iris.csv')

# Load the data from the CSV file into a pandas DataFrame

self.df = pd.read\_csv(relative\_file\_path)

# Define the output directory for saving visualizations

self.output\_directory = os.path.join('..', 'Lb8')

# Method to explore the data

def explore\_data(self):

# Print the descriptive statistics of the DataFrame

print(self.df.describe())

# Print the maximum values of each column in the DataFrame

print("\nMaximum values:\n", self.df.max())

# Print the minimum values of each column in the DataFrame

print("\nMinimum values:\n", self.df.min())

# Method to visualize the data

def visualize\_data(self, save=False):

# Basic visualization: scatter plot of sepal length vs sepal width

plt.scatter(self.df['sepal.length'], self.df['sepal.width'])

plt.title('Basic Visualization')

plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')

if save:

plt.savefig(os.path.join(self.output\_directory, 'basic\_visualization.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'basic\_visualization.png'), format='png')

else:

plt.show()

# Extended visualizations: scatter plot with different colors for each variety

colors = {'Setosa': 'r', 'Versicolor': 'g', 'Virginica': 'b'}

plt.scatter(self.df['sepal.length'], self.df['sepal.width'], c=self.df['variety'].apply(lambda x: colors[x]))

plt.title('Extended Visualization')

plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')

if save:

plt.savefig(os.path.join(self.output\_directory, 'extended\_visualization.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'extended\_visualization.png'), format='png')

else:

plt.show()

# Multiple subplots: scatter plots of sepal length vs sepal width and petal length vs petal width

fig, axs = plt.subplots(2)

fig.suptitle('Vertically stacked subplots')

axs[0].scatter(self.df['sepal.length'], self.df['sepal.width'], c=self.df['variety'].apply(lambda x: colors[x]))

axs[0].set(xlabel='Sepal Length', ylabel='Sepal Width')

axs[1].scatter(self.df['petal.length'], self.df['petal.width'], c=self.df['variety'].apply(lambda x: colors[x]))

axs[1].set(xlabel='Petal Length', ylabel='Petal Width')

if save:

plt.savefig(os.path.join(self.output\_directory, 'multiple\_subplots.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'multiple\_subplots.png'), format='png')

else:

plt.show()

# Violin plot of sepal length distribution by variety

plt.figure(figsize=(10, 6))

sns.violinplot(x='variety', y='sepal.length', data=self.df, hue='variety', palette='Set1', legend=False)

plt.title('Sepal Length Distribution by Variety')

plt.xlabel('Flower Variety')

plt.ylabel('Sepal Length')

if save:

plt.savefig(os.path.join(self.output\_directory, 'violin\_plot.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'violin\_plot.png'), format='png')

else:

plt.show()

# Bar plot

plt.figure(figsize=(10, 6))

sns.barplot(x='variety', y='sepal.length', data=self.df)

plt.title('Bar Plot of Sepal Length by Variety')

plt.xlabel('Flower Variety')

plt.ylabel('Sepal Length')

if save:

plt.savefig(os.path.join(self.output\_directory, 'bar\_plot.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'bar\_plot.png'), format='png')

else:

plt.show()

# Histogram

plt.figure(figsize=(10, 6))

plt.hist(self.df['sepal.length'], bins=10)

plt.title('Histogram of Sepal Length')

plt.xlabel('Sepal Length')

plt.ylabel('Frequency')

if save:

plt.savefig(os.path.join(self.output\_directory, 'histogram.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'histogram.png'), format='png')

else:

plt.show()

# Pie chart

plt.figure(figsize=(10, 6))

variety\_counts = self.df['variety'].value\_counts()

plt.pie(variety\_counts, labels=variety\_counts.index, autopct='%1.1f%%')

plt.title('Pie Chart of Flower Varieties')

if save:

plt.savefig(os.path.join(self.output\_directory, 'pie\_chart.svg'), format='svg')

plt.savefig(os.path.join(self.output\_directory, 'pie\_chart.png'), format='png')

else:

plt.show()

# Facet plot: scatter plots of sepal length vs sepal width for each variety

g = sns.FacetGrid(self.df, col='variety', height=5, aspect=1)

g.map(plt.scatter, 'sepal.length', 'sepal.width', alpha=0.7)

g.set\_titles(col\_template="{col\_name}")

g.set\_axis\_labels('Sepal Length', 'Sepal Width')

if save:

g.savefig(os.path.join(self.output\_directory, 'facet\_plot.svg'))

g.savefig(os.path.join(self.output\_directory, 'facet\_plot.png'))

else:

plt.show()

Клас MenuLab8:

import json

import os

from DataProcessor import DataProcessor

from Validator import Validator

class MenuLab8:

# Constructor method for the MenuLab8 class

def \_\_init\_\_(self, data\_processor, validator, reader\_writer):

self.data\_processor = data\_processor

self.validator = validator

self.reader\_writer = reader\_writer

def run(self):

# Define the valid options

valid\_options = ['1', '2', '3', '4']

# Loop until the user chooses to exit

while True:

# Display the menu options

# Get the user's choice and validate it

choice = self.validator.validate\_input("Choose an option (1/2/3/4): ", valid\_options)

# Perform the action corresponding to the user's choice

if choice == '1':

# If the user chooses to explore data, ask for confirmation and then explore data

if self.validator.validate\_input("Explore data? (yes/no): ", ['yes', 'no']) == 'yes':

print("Exploring data...")

self.data\_processor.explore\_data()

elif choice == '2':

# If the user chooses to generate visualizations, ask for confirmation and then generate visualizations

if self.validator.validate\_input("Generate visualizations? (yes/no): ", ['yes', 'no']) == 'yes':

print("Generating visualizations...")

self.data\_processor.visualize\_data()

elif choice == '3':

# If the user chooses to save visualizations, ask for confirmation and then save visualizations

if self.validator.validate\_input("Save visualizations? (yes/no): ", ['yes', 'no']) == 'yes':

print("Saving visualizations...")

self.data\_processor.visualize\_data(save=True)

elif choice == '4':

# If the user chooses to exit, print a goodbye message and break the loop

print("Goodbye!")

break

Клас RunnerLab8:

from MenuLab8 import MenuLab8

from DataProcessor import DataProcessor

from Validator import Validator

class RunnerLab8:

@staticmethod

def run():

data\_processor = DataProcessor()

validator = Validator(None)

reader\_writer = None

menu = MenuLab8(data\_processor, validator, reader\_writer)

menu.run()

if \_\_name\_\_ == "\_\_main\_\_":

RunnerLab8.run()

Клас Validator:

class Validator:

def \_\_init\_\_(self, initial\_value=None):

# Your initialization code goes here if needed

self.initial\_value = initial\_value

def validate\_input(self, prompt, choices):

# Utility for validating user input against predefined choices

while True:

user\_input = input(prompt)

if choices:

if user\_input in choices:

return user\_input

else:

print("Invalid choice.")

else:

try:

return float(user\_input)

except ValueError:

print("Invalid input. Please enter a number.")

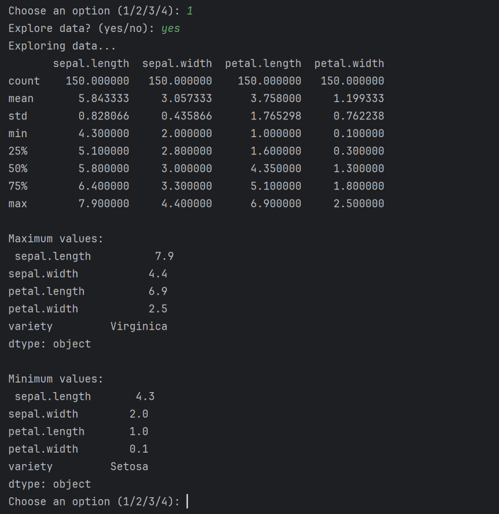
if \_\_name\_\_ == "\_\_main\_\_":

# Instantiate Validator with an initial value

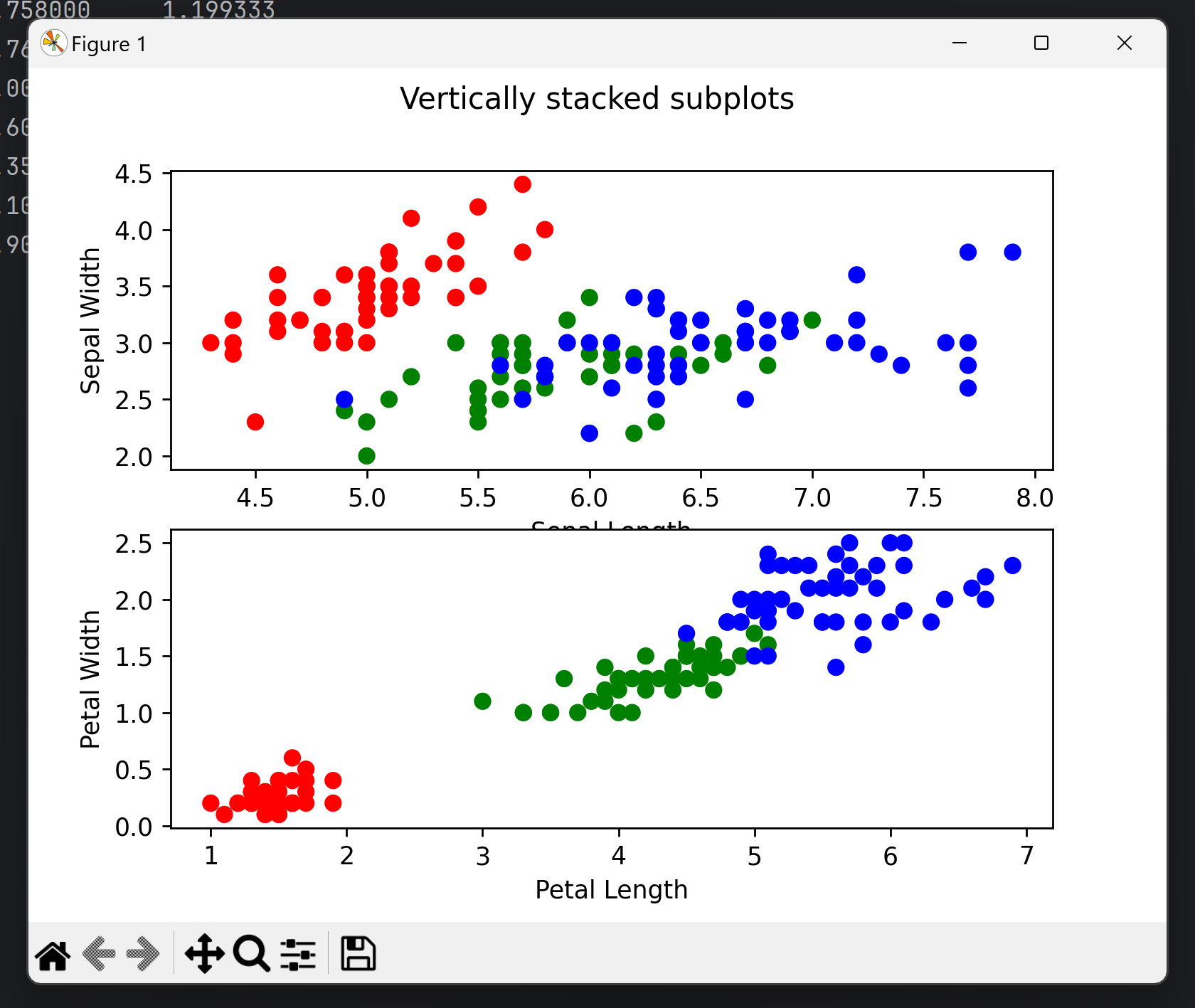
validator = Validator(None) # You can provide a specific value instead of None

# Rest of your code...

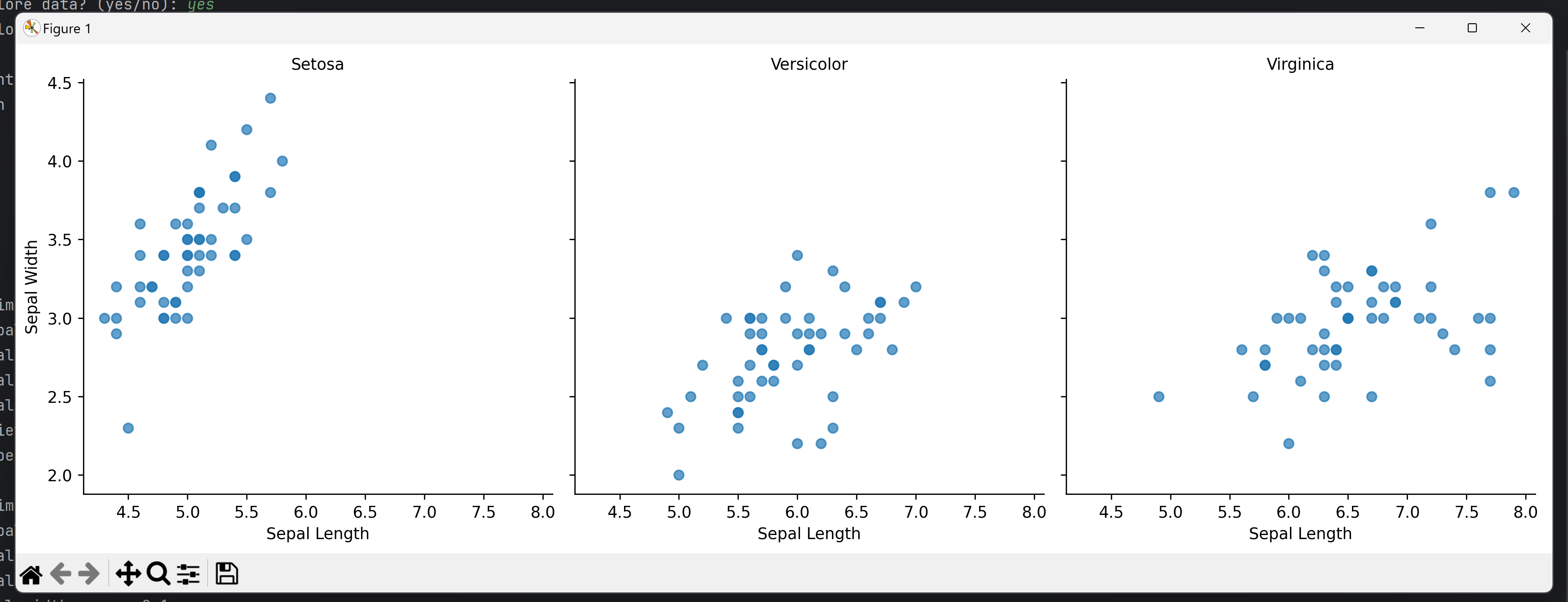
На рисунках 1-3 зображено результат виконання програми.



*Рис.1 Вивід програми*

**

*Рис.2 Вивід програми*

**

*Рис.3 Вивід програми*

**Висновок:** під час виконання лабораторної роботи я навчився створювати багатофункціональний додаток для візуалізації CSV-наборів даних за допомогою Matplotlib.