Міністерство освіти і науки України

Національний університет «Львівська політехніка»

лабораторної роботи №8

з дисципліни «Спеціалізовані мови програмування»

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

Виконав:

Кулявець В. Р.

Перевірив:

Щербак С.С.

Львів 2024

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

Git: <https://github.com/Pivinter/-.git>

Завдання 1: Вибір CSV-набору даних

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

Date,Sales\_Amount,Advertising\_Spend,Product

2023-01-01,237,293,Product C

2023-01-02,246,150,Product A

2023-01-03,292,159,Product A

2023-01-04,418,225,Product B

2023-01-05,306,171,Product C

2023-01-06,203,218,Product C

2023-01-07,256,263,Product C

2023-01-08,325,175,Product C

2023-01-09,251,186,Product C

2023-01-10,219,271,Product A

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.

import pandas as pd

Завдання 3: Дослідження даних

Визначте екстремальні значення по стовцям

class DataProcessor:

def \_\_init\_\_(self, file\_path):

self.data = pd.read\_csv(file\_path)

def preprocess\_data(self):

# Basic data cleaning and handling of NaN values

self.data.dropna(inplace=True)

def identify\_extremes(self, column):

max\_value = self.data[column].max()

min\_value = self.data[column].min()

return {'max': max\_value, 'min': min\_value}

def get\_data(self):

return self.data

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.

def save\_interactive\_chart(self, column, file\_name):

fig = px.line(self.data, y=column, title=f'{column} Over Time')

# Save as HTML if PNG export fails

try:

fig.write\_image(file\_name)

except ValueError:

print("PNG export failed, saving as HTML instead.")

html\_file\_name = file\_name.replace('.png', '.html')

fig.write\_html(html\_file\_name)

Завдання 5: Підготовка даних

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

def plot\_basic\_chart(self, column):

fig = px.line(self.data, y=column, title=f'{column} Over Time')

fig.show()

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.

class IVisualizer(ABC):

@abstractmethod

def plot\_basic\_chart(self, column):

pass

@abstractmethod

def save\_interactive\_chart(self, column, file\_name):

pass

Завдання 7: Розширені візуалізації

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

data\_processor = DataProcessor(CSV\_FILE\_PATH)

data\_processor.preprocess\_data()

Завдання 8: Декілька піддіаграм

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

visualizer = Visualizer(data)

Завдання 9: Експорт і обмін

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

visualizer.plot\_basic\_chart('Sales\_Amount')

visualizer.save\_interactive\_chart('Sales\_Amount', IMAGE\_FILE\_PATH)

Висновок: Виконавши ці завдання, ви створите багатофункціональний додаток для візуалізації CSV-наборів даних за допомогою Matplotlib. Цей проект покращить ваші навички візуалізації даних, дозволяючи досліджувати результати з різноманітними наборами даних