Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №3

з дисципліни «Вступ до інтелектуального аналізу даних»

Тема «Додаткові графічні методи в Python»

Варіант №19

Студента 3-го курсу НН ІАТЕ гр. ТР-12
Ковальова Олександра
Перевірив: д.т.н., проф. Путренко В. В.

КИЇВ 2024

Мета: Ознайомитись з прикладом створення додаткових графіків за допомогою бібліотек Pandas, StatsModels, WordCloud, MatPlotLib. Виконати поставлене завдання.

Хід роботи

Завдання: побудувати 5 графіків, а саме мозаїчний, графік паралельних координат, піксельний, діаграму розсіювання, і хмаринку слів.

Код до кожної діаграми знаходиться в окремій комірці Jupyter Notebook, що дозволяє модифікувати код до однієї діаграми не зачіпаючи інші.

Спочатку створимо мозаїчний графік:

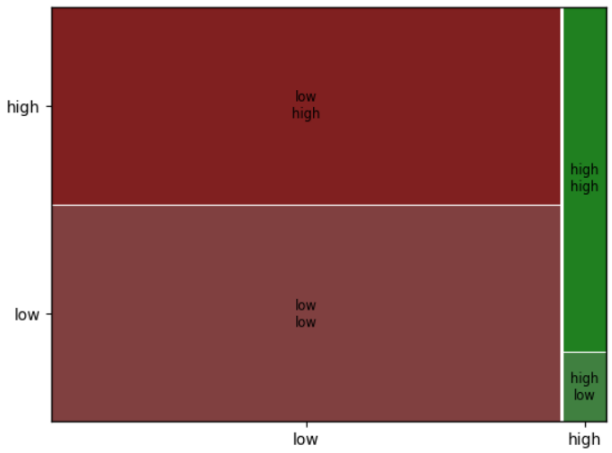
```
In [4]: import matplotlib.pyplot as plt
        import pandas
        from statsmodels.graphics.mosaicplot import mosaic

        df = pandas.read_csv('data/cereals_conf.csv', sep=",")

        mosaic(df, ['Fat', 'Protein'])
        plt.show()
```

У поданому фрагменті коду використовуються різні бібліотеки Python для аналізу та візуалізації даних в середовищі Jupyter Notebook. Починаючи з імпорту бібліотеки matplotlib.pyplot під псевдонімом plt, код готує середовище для створення графіків. Далі, бібліотека pandas використовується для зчитування даних з CSV-файлу за допомогою функції read_csv. Зчитані дані представлені у вигляді таблиці (DataFrame) та зберігаються в об’єкті з назвою df.

Останні два рядки коду використовують бібліотеку statsmodels graphics mosaicplot для створення мозаїчного графіку на основі даних з DataFrame df. Функція mosaic визначає, які колонки з DataFrame використовувати для побудови мозаїчного графіку, в даному випадку, це колонки ‘Fat’ та ‘Protein’.

Мозаїчний графік використовується для візуалізації категоріальних даних та дозволяє спостерігачу аналізувати співвідношення між двома категоріями. У цьому випадку, графік допомагає вивчити взаємозв’язок між кількістю жиру та білка в різних продуктах, надаючи швидкий та інтуїтивний спосіб візуалізації цих залежностей у датасеті "cereals_conf.csv".

Результат:



Графік паралельних координат:

```
In [8]: import pandas
        import matplotlib.pyplot as plt
        import pandas.plotting as plotting

        df = pandas.read_csv('data/cereals_num.csv', sep=",")

        plotting.parallel_coordinates(df, 'Cereal')
        plt.gca().legend_.remove()
        plt.show()
```

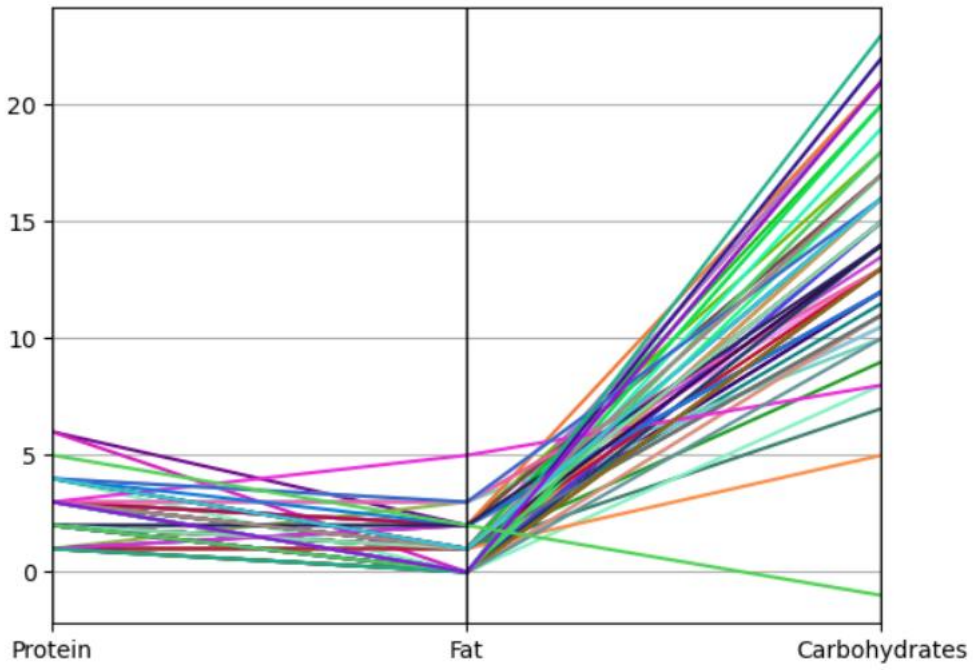
Спочатку, дані з CSV-файлу "cereals_num.csv" зчитуються за допомогою бібліотеки pandas та зберігаються у формі таблиці (DataFrame) під іменем df.

Далі, використовується бібліотека matplotlib.pyplot (під псевдонімом plt), а також модуль pandas.plotting під псевдонімом plotting. Функція parallel_coordinates з цього модулю використовується для створення графіку паралельних координат на основі числових даних у DataFrame df. Кожен рядок у таблиці представлений як лінія на графіку, а координатні вісі відображають числові характеристики.

Останні два рядки коду відповідають за відображення графіку. Виклик plt.gca().legend_.remove() видаляє легенду, оскільки для цього типу графіку інформація про конкретні дані не є обов’язковою. Завершальний рядок plt.show() відображає створений графік.

Графік паралельних координат використовується для візуалізації взаємозв’язків між числовими характеристиками у кожному рядку даних. У даному випадку, він допомагає аналізувати та порівнювати числові параметри для різних видів каш, представлених у датасеті "cereals_num.csv".

Результат:



Піксельний графік:

```
In [27]: import numpy as np
         import matplotlib.pyplot as plt
         import matplotlib.cm as cm

         inData = np.loadtxt('data/cereals_normed.csv',
                             delimiter=',',
                             dtype=float,
                             skiprows=1)
         tData = inData.transpose()

         plt.imshow(tData,
                    interpolation='nearest',
                    cmap=cm.inferno)
         plt.axis('off')
         plt.show()
```

У цьому коді використовуються бібліотеки NumPy та Matplotlib для обробки та візуалізації даних. На початку коду, бібліотека NumPy імпортується під псевдонімом np, а бібліотеки Matplotlib – plt та cm.

Далі, дані з CSV-файлу "cereals_normed.csv" завантажуються за допомогою функції np.loadtxt. Ця функція зчитує числові дані, використовуючи кому як роздільник та пропускаючи перший рядок файлу, який, містить заголовки стовпців. Після завантаження дані транспонуються для зручності подальшого використання.

Далі, використовується функція plt.imshow для створення зображення (зображення матриці) на основі транспонованих даних. Аргументи функції включають встановлення методу інтерполяції (interpolation), кольорової карти (cmap), яка визначає палітру для зображення (у даному випадку, використовується "inferno"), та вимкнення відображення осей за допомогою plt.axis('off').

Останній рядок plt.show() відображає створене зображення.

Застосування функції plt.imshow для візуалізації матриці дозволяє швидко аналізувати та порівнювати числові значення в датасеті, представленому у вигляді зображення. У даному випадку, це може бути особливо корисно для виявлення шаблонів та залежностей у нормованих даних про каші.

Результат:

Діаграма розсіювання:

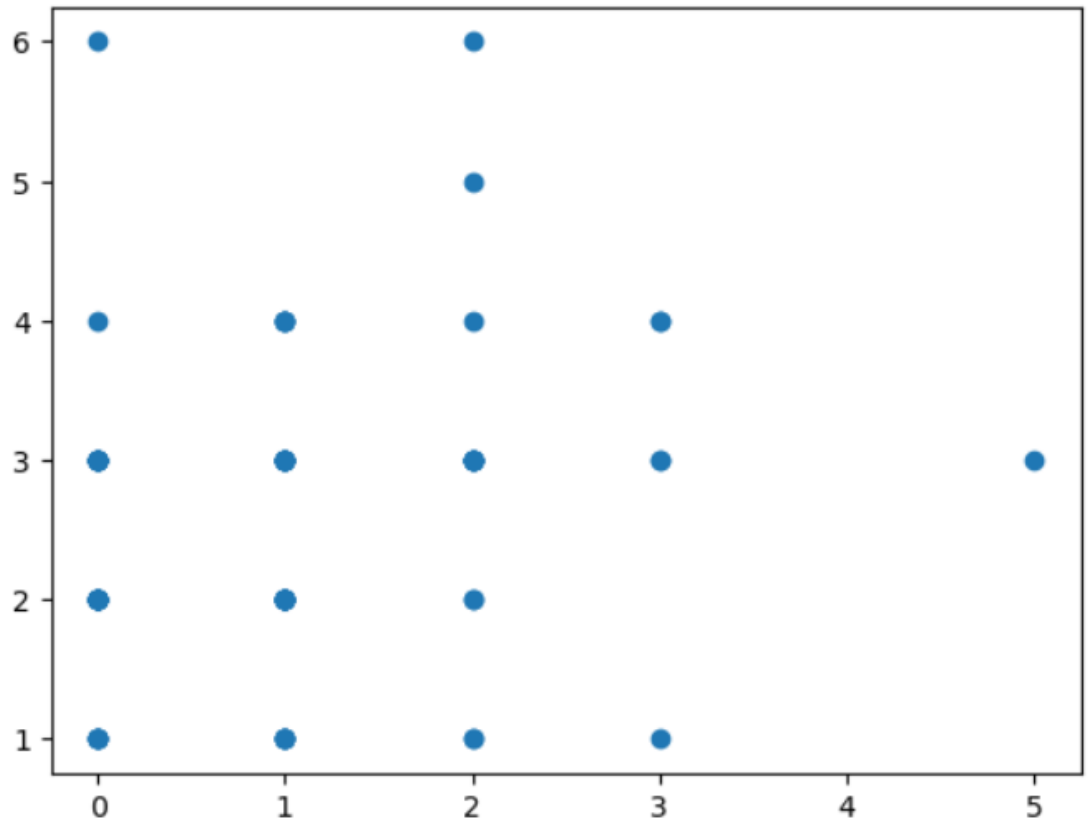
```
In [12]: import pandas
         import matplotlib.pyplot as plt

         df = pandas.read_csv('data/cereals_num.csv', sep=",")

         plt.scatter(df['Fat'], df['Protein'])
         plt.show()
```

У цьому фрагменті коду використовуються бібліотеки Pandas та Matplotlib для обробки та візуалізації даних. Спочатку, дані з CSV-файлу "cereals_num.csv" зчитуються за допомогою функції pandas.read_csv та зберігаються у формі таблиці (DataFrame) під іменем df.

Далі, використовується функція plt.scatter, яка створює діаграму розсіювання. У цьому випадку, на графіку представлені точки, де одна вісь відповідає значенням з колонки ‘Fat’, а інша - значенням з колонки ‘Protein’ для кожного рядка даних.

Діаграма розсіювання використовується для візуалізації взаємозв’язку між двома числовими змінними. У цьому випадку, вона дозволяє аналізувати розподіл кількості жиру та білка в різних продуктах каш, а також виявляти можливі кореляції між цими характеристиками. Такий тип графіку особливо корисний для виявлення можливих взаємозв’язків та варіацій в числових даних.

Результат:



Хмаринка слів:

```
In [1]: from wordcloud import WordCloud
        import matplotlib.pyplot as plt

        text = open('data/kjb.txt').read()

        wordcloud = WordCloud(background_color='white', width=800, height=400)
        wordcloud.generate(text)

        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis('off')
        plt.show()
```

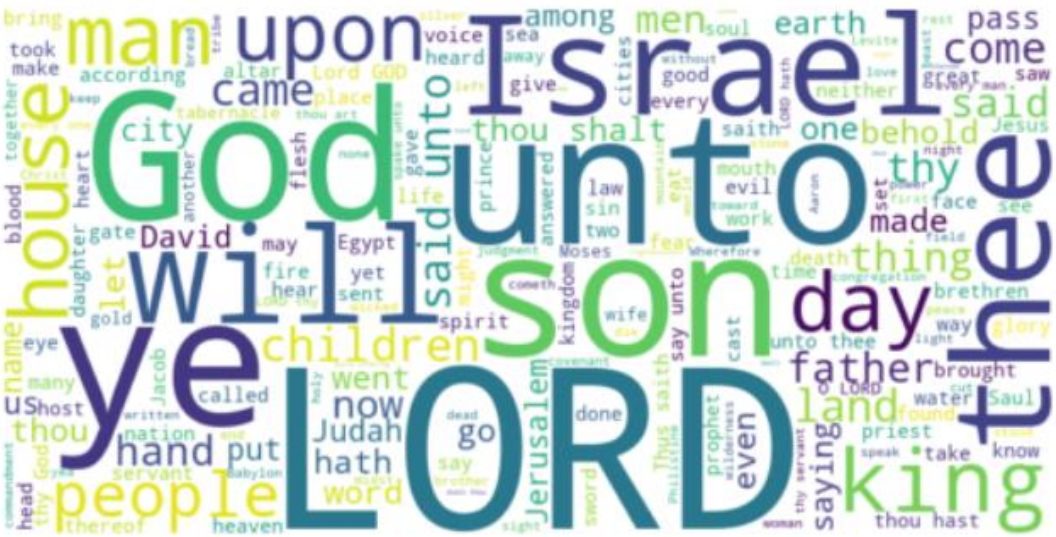
У цьому коді використовуються бібліотеки WordCloud та Matplotlib для створення та візуалізації хмари слів з текстового файлу "kjb.txt". Спочатку, імпортується клас WordCloud з бібліотеки wordcloud та бібліотека Matplotlib під псевдонімом plt. Далі, відбувається зчитування текстового вмісту з файлу "kjb.txt" за допомогою open('data/kjb.txt').read(). Текст зберігається у змінній text.

Після цього створюється об'єкт WordCloud з параметрами, які визначають вигляд хмари слів: білий фон (background_color='white'), ширина та висота хмари (width=800, height=400). Потім, згенерована хмара слів застосовується до тексту методом generate(text).

Останні три рядки коду відповідають за візуалізацію хмари слів. plt.imshow(wordcloud, interpolation='bilinear') відображає згенеровану хмару слів, використовуючи метод білінійної інтерполяції для згладжування зображення. plt.axis('off') вимикає відображення координатних осей, а plt.show() виводить створену візуалізацію.

Хмара слів використовується для візуалізації частоти використання слів у тексті. У даному випадку, вона може використовуватися для визначення ключових тем або слів у текстовому файлі "kjb.txt" (Біблія Короля Якова).

Результат:



Висновок: Під час виконання лабораторної роботи були набуті практичні навички роботи з діаграмами та графіками різних типів: діаграмою розсіювання, піксельним, мозаїчним графіком, хмаринкою слів, графіком паралельних координат. Були виконані певні зразкові задачі з використанням бібліотек Pandas, MatPlotLib, NumPy, StatsModels та WordCloud.

Програмний код

Notebook.ipynb:

```
import matplotlib.pyplot as plt
import pandas
from statsmodels.graphics.mosaicplot import mosaic

df = pandas.read_csv('data/cereals_conf.csv', sep=",")

mosaic(df, ['Fat', 'Protein'])
plt.show()

import pandas
import matplotlib.pyplot as plt
import pandas.plotting as plotting

df = pandas.read_csv('data/cereals_num.csv', sep=",")

plotting.parallel_coordinates(df, 'Cereal')
plt.gca().legend_.remove()
plt.show()

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

inData = np.loadtxt('data/cereals_normed.csv',
                    delimiter=',',
                    dtype=float,
                    skiprows=1)
tData = inData.transpose()

plt.imshow(tData,
           interpolation='nearest',
           cmap=cm.inferno)
plt.axis('off')
plt.show()

import pandas
import matplotlib.pyplot as plt

df = pandas.read_csv('data/cereals_num.csv', sep=",")
plt.scatter(df['Fat'], df['Protein'])
plt.show()

from wordcloud import WordCloud
import matplotlib.pyplot as plt

text = open('data/kjb.txt').read()

wordcloud = WordCloud(background_color='white', width=800, height=400)
wordcloud.generate(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```