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 №16

Виконав:

студент 3-го курсу, НН ІАТЕ гр. ТР-23

Ровний Григорій

Перевірив: проф. Путренко В.В.

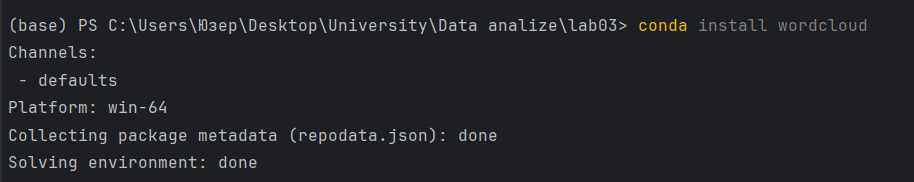
КИЇВ 2025

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

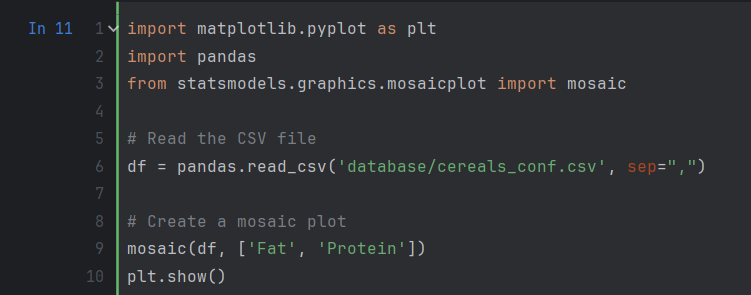
**Завдання:** побудувати 5 графіків, а саме мозаїчний, графік паралельних координат, піксельний, діаграму розсіювання, і хмаринку слів.

**Хід роботи**

Перед початком виконання лабораторної роботи потрібно додатково встановити бібліотеки pandas, matplotlib та wordcloud

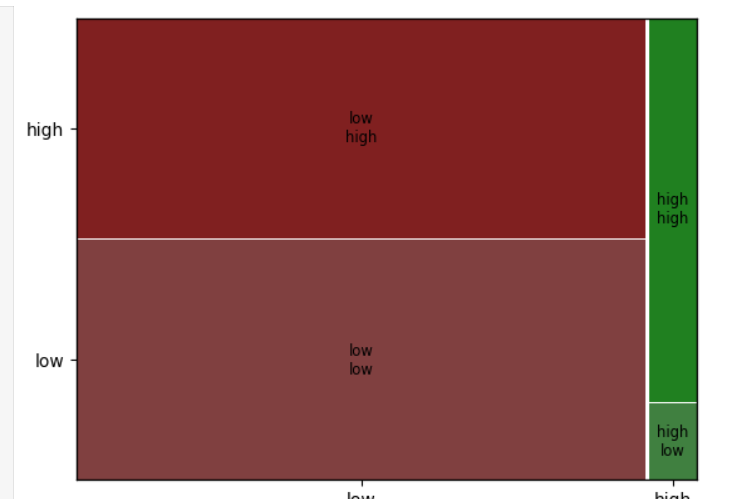


Спочатку створимо мозаїчний графік. Matplotlib.pyplot під псевдонімом plt готує середовище для створення графіків. Далі, бібліотека pandas використовується для зчитування даних з CSV-файлу за допомогою функції read\_csv. Зчитані дані представлені у вигляді DataFrame та зберігаються в об’єкті df. Останні 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’.

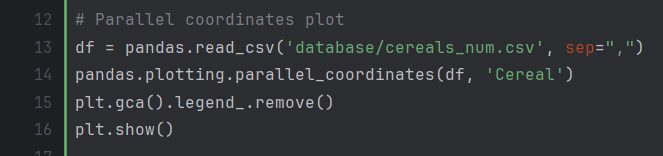


Графік допомагає вивчити взаємозв’язок між кількістю жиру та білка в різних продуктах, надаючи швидкий та інтуїтивний спосіб візуалізації цих залежностей у датасеті "cereals\_conf.csv".

Результат:

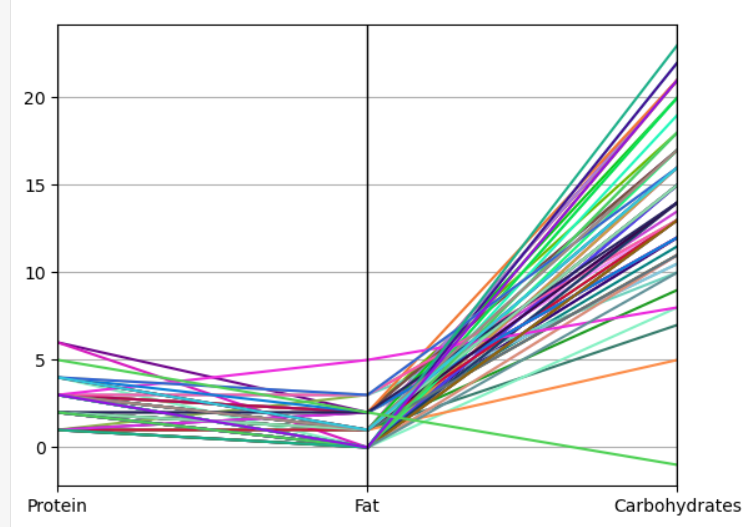


Графік паралельних координат. Дані з CSV-файлу "cereals\_num.csv" зчитуються за допомогою бібліотеки pandas та зберігаються у формі таблиці. Далі, використовується бібліотека matplotlib.pyplot а також модуль pandas.plotting під псевдонімом plotting. Функція parallel\_coordinates() з цього модулю використовується 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.



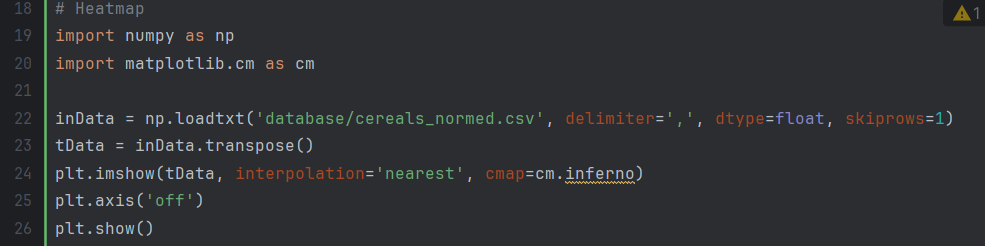
Такий графік використовується для візуалізації взаємозв’язків між числовими характеристиками у кожному рядку даних.

Результат:

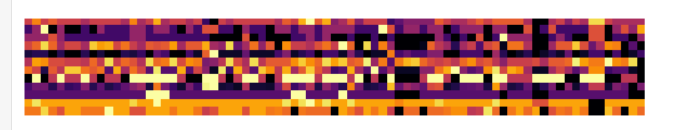


Піксельний графік:

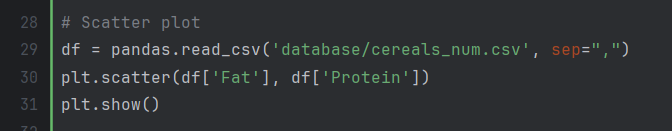
Тут вже використаємо бібліотеки NumPy та Matplotlib для обробки та візуалізації даних. На початку коду, бібліотека NumPy імпортується під псевдонімом np, а бібліотеки Matplotlib – plt та cm. Дані з CSV-файлу "cereals\_normed.csv" завантажуються за допомогою функції np.loadtxt. Після завантаження дані транспонуються для зручності подальшого використання. Далі, використовується функція plt.imshow для створення зображення матриці на основі транспонованих даних. Аргументи функції включають встановлення методу інтерполяції, cmap, яка визначає палітру для зображення та вимкнення відображення осей за допомогою plt.axis(‘off’).



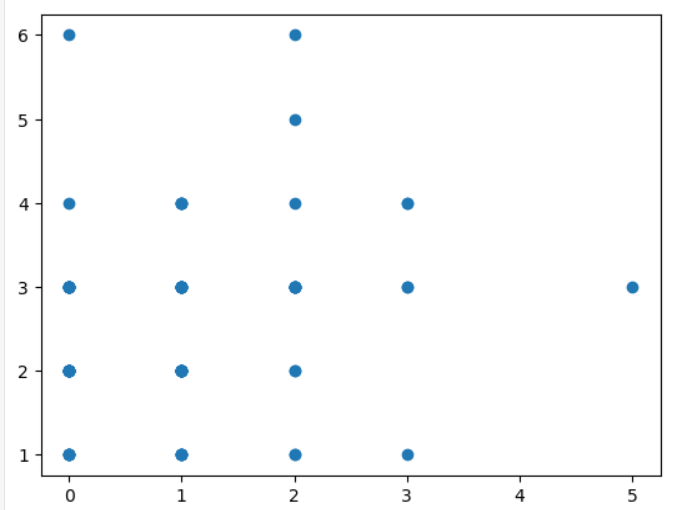
Результат:



Діаграма розсіювання. Процес аналогічний до попередніх діаграм. Але 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.



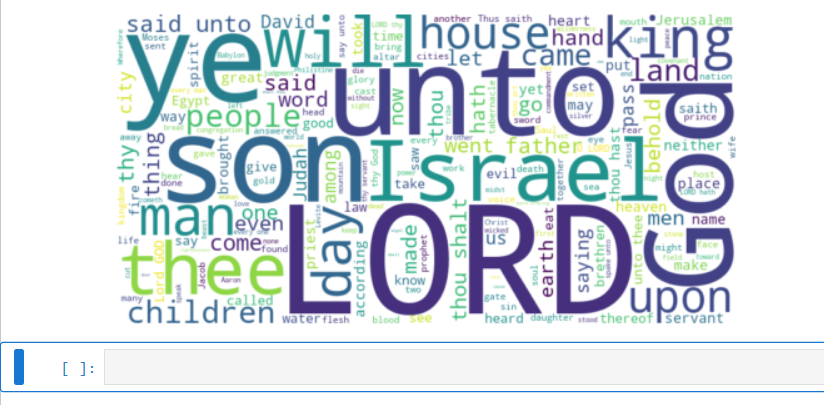
Результат:



Хмаринка слів. Хмара слів використовується для візуалізації частоти використання слів у тексті. 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.



Результат:



**Висновок:**

Під час виконання лабораторної роботи було набуто практичні навички роботи з діаграмами та графіками різних типів: розсіювання, піксельною, мозаїчним графіком, хмаринкою слів, графіком паралельних координат. Було виконано певні зразкові задачі з використанням бібліотек Pandas, MatPlotLib, NumPy, StatsModels та WordCloud.

**Програмний код:**

import matplotlib.pyplot as plt  
import pandas  
from statsmodels.graphics.mosaicplot import mosaic  
  
# Read the CSV file  
df = pandas.read\_csv('database/cereals\_conf.csv', sep=",")  
  
# Create a mosaic plot  
mosaic(df, ['Fat', 'Protein'])  
plt.show()  
  
# Parallel coordinates plot  
df = pandas.read\_csv('database/cereals\_num.csv', sep=",")  
pandas.plotting.parallel\_coordinates(df, 'Cereal')  
plt.gca().legend\_.remove()  
plt.show()  
  
# Heatmap  
import numpy as np  
import matplotlib.cm as cm  
  
inData = np.loadtxt('database/cereals\_normed.csv', delimiter=',', dtype=float, skiprows=1)  
tData = inData.transpose()  
plt.imshow(tData, interpolation='nearest', cmap=cm.inferno)  
plt.axis('off')  
plt.show()  
  
# Scatter plot  
df = pandas.read\_csv('database/cereals\_num.csv', sep=",")  
plt.scatter(df['Fat'], df['Protein'])  
plt.show()  
  
# Word cloud  
from wordcloud import WordCloud  
  
text = open('database/kjv.txt').read()  
wordcloud = WordCloud(background\_color='white', width=800, height=400)  
wordcloud.generate(text)  
plt.imshow(wordcloud, interpolation='bilinear')  
plt.axis('off')  
plt.show()