**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №5**

**з навчальної дисципліни «Технології Data Science»**

**Тема:**

**РЕАЛІЗАЦІЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ**

**(MACHINE LEARNING (ML))**

**Виконав:**

Студент 4 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

Виявити, дослідити та узагальнити особливості аналізу даних з використанням методів та технологій машинного навчання (Machine Learning (ML)).

## **ІІ. Завдання:**

Розробити програмний скрипт мовою Python що реалізує обчислювальний алгоритм машинного навчання (Machine Learning (ML)) відповідно до технічних умов:

**Група технічних вимог\_1:**

Реалізувати кластеризацію вхідних даних, отриманих Вами у ході виконання ЛР\_1, модельних та (або) реальних – на власний вибір. Методи Machine Learning з переліку: k-means (k-середніх); Support Vector Machine (машина опорних векторів); k-nearest neighbors (найближчих сусідів); ієрархічна кластеризація – для кластеризації обраних даних обрати самостійно. Провести аналіз та пояснення отриманих результатів, сформувати висновки.

**Група технічних вимог\_2:**

Реалізувати кластеризацію за кольоровою ознакою об’єктів на самостійно обраному цифровому зображенні. Методи Machine Learning з переліку: k-means (k-середніх); Support Vector Machine (машина опорних векторів); k-nearest neighbors (найближчих сусідів) – для кластеризації обраного зображення обрати самостійно.

За необхідності провести покращення якості зображення: зміна кольору; підвищення контрасту; фільтрація, тощо. Етапи покращення якості та кластерізації повинні забезпечувати виділення геометричних або кольорових ознак обраного на цифровому зображенні об’єкту для його подальшої ідентифікації. Провести аналіз отриманих результатів, сформувати висновки.

**Група технічних вимог\_3:**

Підрахувати кількість об’єктів на обраному цифровому зображенні. Об’єкти, що підлягають обрахунку обрати самостійно. Зміст етапів попередньої обробки зображень (корекція кольору, фільтрація, векторизація, кластеризація) має буди результатом R&D процесів, що конкретизується обраним зображенням і об’єктами для підрахунку. Провести аналіз отриманих результатів, сформувати висновки.

**Завдання ІІІ рівня складності (9 балів):** реалізувати ТРИ групи технічних вимог.

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичні моделі, що реалізують кластеризацію даних про індекс S&P 500, кластеризацію за кольоровою ознакою озер, здійснює підрахунок шматків кавуна на зображенні.

Кластеризація даних про індекс S&P 500 відбувається за допомогою алгоритму k-means. Він поділяє набір даних на k груп на основі схожості. Спочатку він випадковим чином обирає центроїди, які будуть центрами кластерів. Далі кожна точка призначається найближчому центроїду, і таким чином формуються кластери. Після цього центроїди оновлюються як середні значення всіх точок у кластері. Попередні два кроки повторюються, поки центроїди не перестануть змінюватися. Таким чином точки даних поділено на k кластерів. Оптимальну кількість кластерів можна визначити з допомогою методу Elbow. Для цього на проміжку можливих значень кластерів виконується алгоритм k-means, для якого обчислюється сума квадратів відстаней між кожною точкою і центроїдом. По цих сумах будується графік. Точка на графіку, де ця похибка різко перестає змінюватися (“лікоть”), і є оптимальною кількістю кластерів.

Кластеризація за кольоровою ознакою озер відбувається з допомогою алгоритму k-means. Проте кількість кластерів уже відома – 2 (колір озер та не озер). Перед кластеризацією до зображення застосовується обробка. Вона полягає у виділенні кольору озер серед інших кольорів на зображенні. Для цього спочатку виконується Гаусівське розмиття, виділяються темні області на зображенні, і на останок, застосовується медіанне згладжування.

Підрахунок кількості шматків кавуна на зображенні відбувається з допомогою методів бібліотеки OpenCV. Спершу зображення проходить обробку: збільшується контраст зображення, далі виконується Гаусівське розмиття, виділяються світлі області на зображенні, виконується згладжування, і, нарешті, знаходження кутів на зображенні та морфологічне закриття. Далі на зображенні знаходяться контури, які пізніше фільтруються за кількістю кутів та розмірами. Кількість контурів, які залишились, і є кількістю шматків кавуна на зображенні.

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до програмного скрипту 1:



Рисунок 1 – Блок схема алгоритму 1

Робота алгоритму розпочинається зі зчитування даних із файлу та їх попередня обробка. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 виконується вибір ознак для кластеризації (день та значення індексу S&P 500).

У блоці 3 визначається оптимальна кількість кластерів з допомогою методу Elbow.

У блоці 4 виконується кластеризація з допомогою k-means.

У блоці 5 відбувається візуалізація результатів кластеризації. На цьому виконання алгоритму завершене.

На рис. 2 зображена блок-схема до програмного скрипту 2:

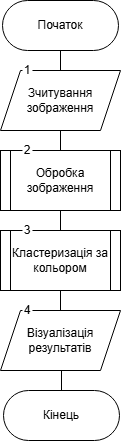


Рисунок 2 – Блок схема алгоритму 2

Робота алгоритму розпочинається зі зчитування зображення із файлу. Ця дія виконується у блоці 1 блок-схеми алгоритму рис.2.

У блоці 2 виконується обробка зображення, а саме: застосування Гаусівського розмиття, виділення темних областей на зображенні та застосування медіанного згладжування.

У блоці 3 виконується розподіл кольорів на два кластери з допомогою k-means.

У блоці 4 відбувається візуалізація результатів кластеризації. На цьому виконання алгоритму завершене.

На рис. 3 зображена блок-схема до програмного скрипту 3:



Рисунок 3 – Блок схема алгоритму 3

Робота алгоритму розпочинається зі зчитування зображення із файлу. Ця дія виконується у блоці 1 блок-схеми алгоритму рис.3.

У блоці 2 виконується обробка зображення, а саме: збільшення контрасту, застосування Гаусівського розмиття, виділення світлих областей на зображенні, застосування медіанного згладжування, визначення кутів на зображенні з допомогою Canny та морфологічне закриття.

У блоці 3 виконується пошук та виділення контурів об’єктів на зображенні.

У блоці 4 контури фільтруються за кількістю кутів та площею. Також виконується їх підрахунок.

У блоці 5 відбувається візуалізація результатів. На цьому виконання алгоритму завершене.

* 1. **Опис структури проекту програми**

Для реалізації розроблених алгоритмів мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

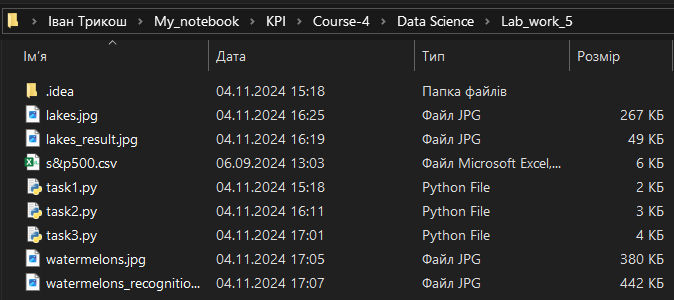


Рисунок 4 – Структура проєкту

Lab\_work\_5 – головний каталог проєкту

task1.py – файл програмного коду 1

task2.py – файл програмного коду 2

task3.py – файл програмного коду 3

s&p500.csv – дані за рік про значення індексу S&P 500

lakes.jpg, lakes\_result.jpg – зображення озер та результати кластеризації відповідно

watermelons.jpg, watermelongs\_recognition.jpg – зображення шматків кавуна та їх підрахунок відповідно

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. На рисунках 5-7 зображено результати роботи алгоритму 1:

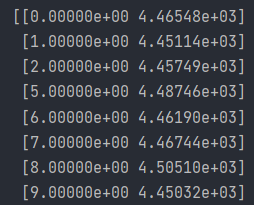


Рисунок 5 – Обрані ознаки для кластеризації (перший стовпчик – кількість днів від початку спостереження, другий – значення індексу в цей день)

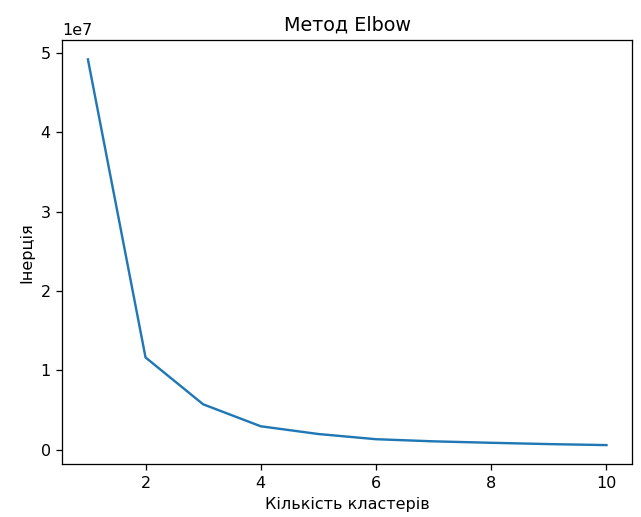


Рисунок 6 – Графік залежності інерції (суми квадратів відстаней між точками та центрами кластерів) від кількості кластерів

За цим графіком бачимо, що після 4 кластерів значення інерції падає повільно, а тому 4 – це оптимальна кількість кластерів.

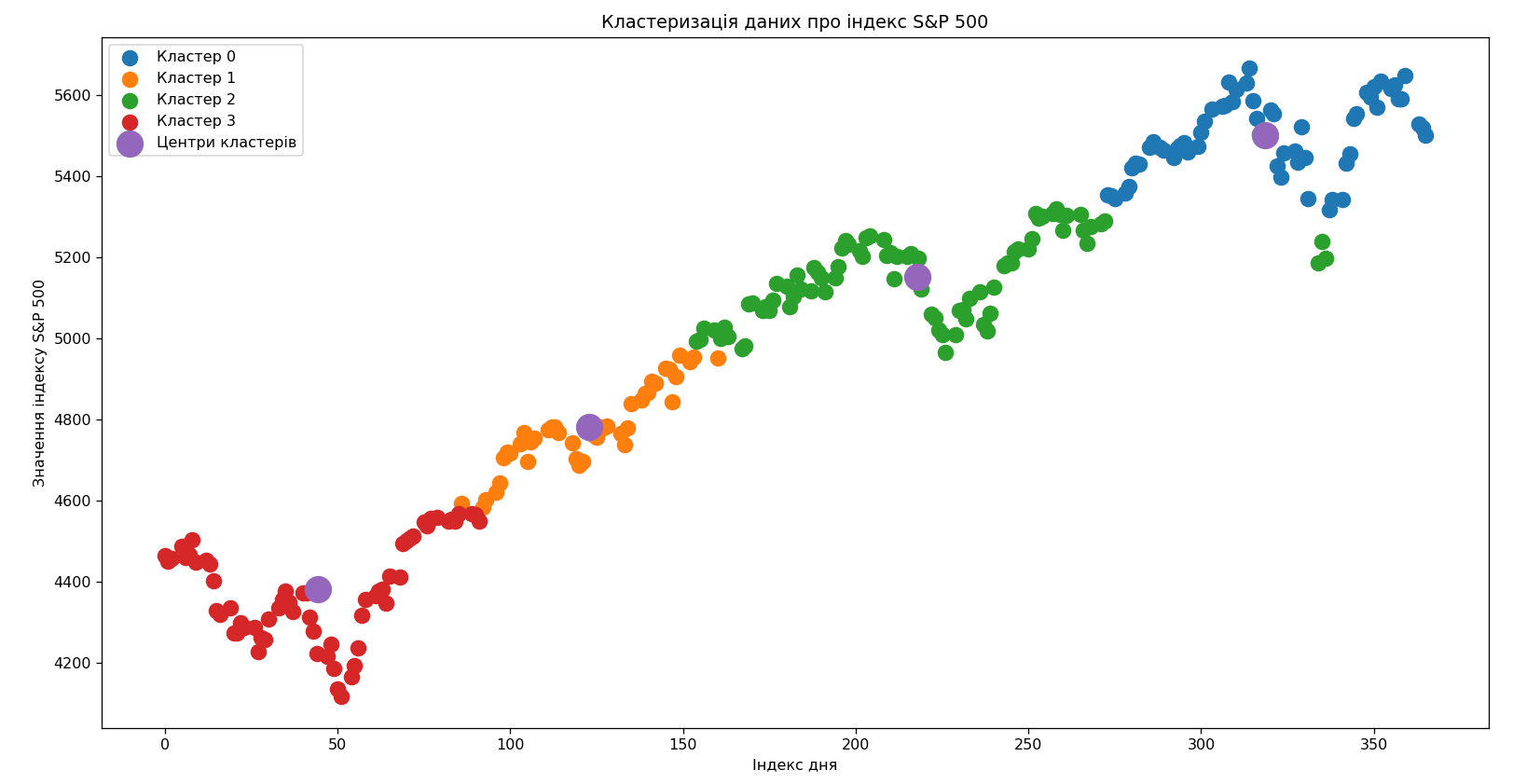


Рисунок 7 – Результати кластеризації даних про індекс S&P 500

Бачимо, що кластери по суті містять у собі точки, що попадають у певні проміжки значень індексу S&P 500 (наприклад, у кластер 3 попали точки, що знаходять у проміжку від 4000 до 4600) і значення дня на кластеризацію сильно не впливає.

1. На рисунках 8-10 зображено кластеризацію за кольором на зображенні. Зображення – Шацькі озера:

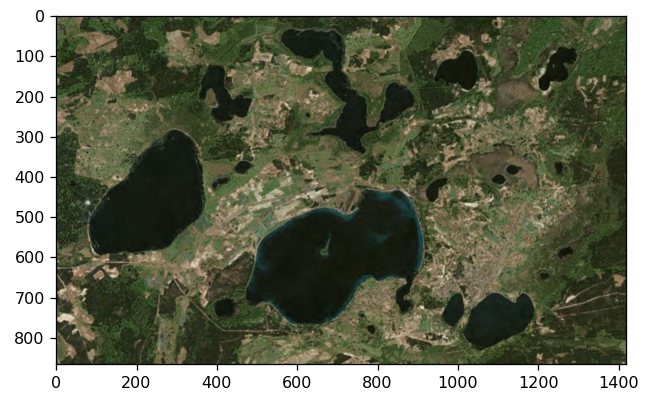


Рисунок 8 – Початкове зображення

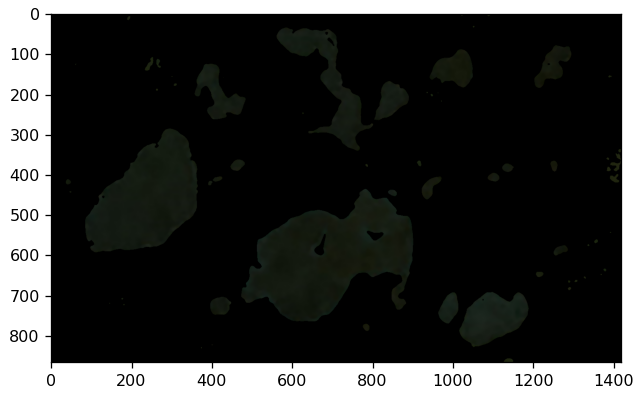


Рисунок 9 – Зображення після обробки

Бачимо, що обробка непогана, озера добре виділилися. Тепер треба застосувати кластеризацію, щоб озера виділились лише одним кольором.

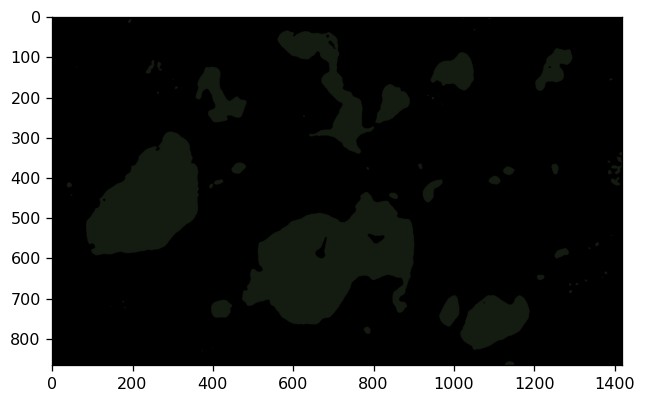


Рисунок 10 – Кластеризація за кольором

Тепер на зображенні лише два кольори. Чорним кольором виділився фон, а трохи світлішим – озера. У подальшому це зображення можна використати для підрахунку кількості озер або ж для обрахунку загальної площі озер.

1. На рисунках 11-15 зображено підрахунок кількості шматків кавуна на зображенні:

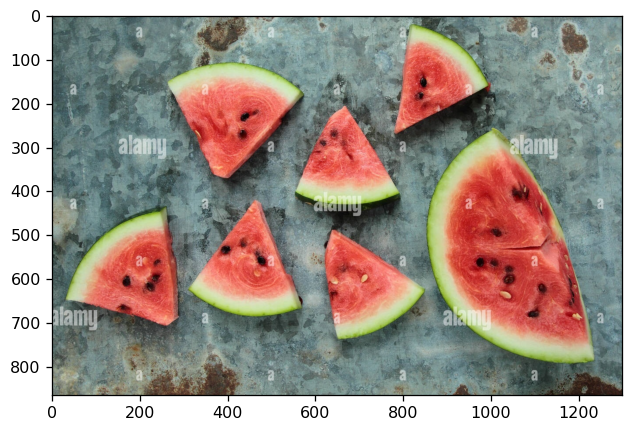


Рисунок 11 – Початкове зображення

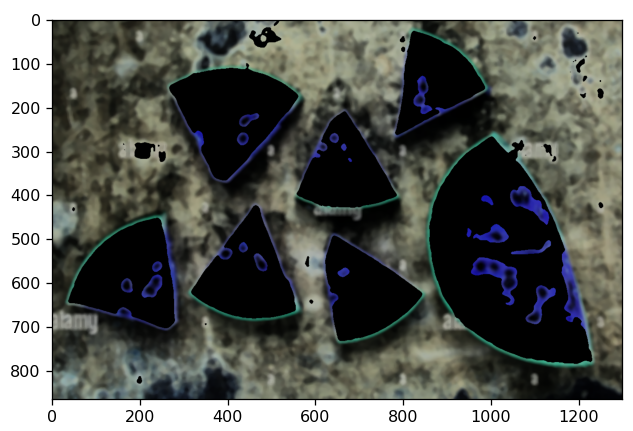


Рисунок 12 – Зображення після обробки

Бачимо, що після обробки шматки кавуна виділилися чорним кольором.

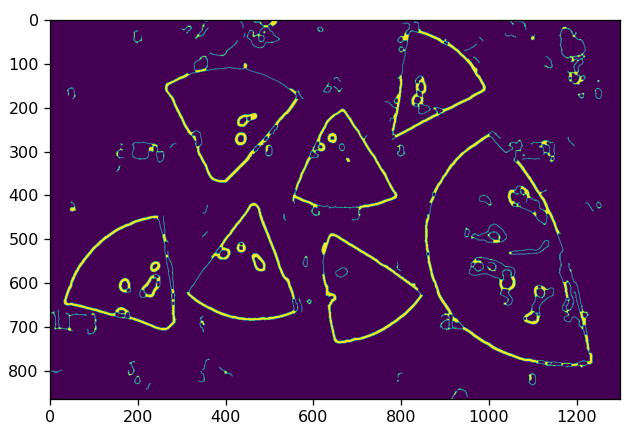


Рисунок 13 – Знайдені кути та морфологічне закриття на зображенні

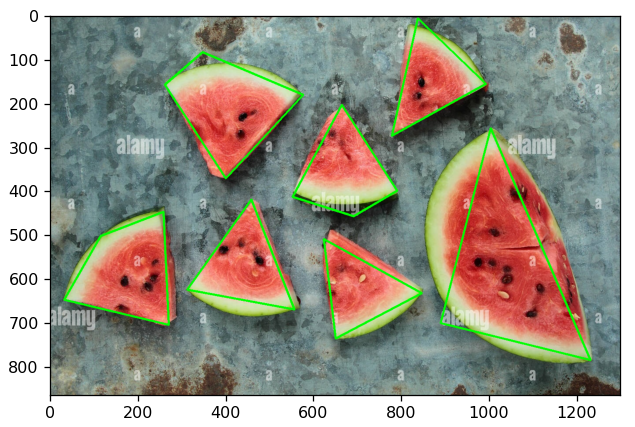


Рисунок 14 – Виділення контурів шматків кавуна



Рисунок 15 – Підрахунок кількості знайдених об’єктів

Бачимо, що контури для менших шматків знайдені непогано, для більшого шматка є деяка неточність. Проте сам підрахунок виконано правильно, на зображенні 7 шматків кавуна.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритми на рис.1-3 та спрямований на отримання результатів, поданих на рис.5-15.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: pandas, matplotlib, sklearn, numpy, OpenCV.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

task1.py

*"""  
Кластеризація даних за рік про значення індексу S&P 500 методом k-means  
"""  
  
import* matplotlib.pyplot *as* plt  
*import* pandas *as* pd  
*from* sklearn.cluster *import* KMeans  
  
*# Зчитуємо дані та обираємо стовпці 2 (індекс дня) та 1 (значення) як ознаки*data = pd.read\_csv('s&p500.csv', index\_col=0)  
data['Date'] = pd.to\_datetime(data['Date'])  
data = data.sort\_values(by='Date', ascending=*True*).reset\_index(drop=*True*)  
data['Days'] = (data['Date'] - data['Date'].min()).dt.days  
X = data.iloc[:, [2, 1]].values  
print(X)  
  
*# Визначимо оптимальну кількість кластерів з допомогою методу Elbow*inertia = []  
*for* i *in* range(1, 11):  
 kmeans = KMeans(n\_clusters=i, n\_init=10, random\_state=42)  
 kmeans.fit(X)  
 inertia.append(kmeans.inertia\_)  
*# Оптимальна кількість кластерів - 4*plt.plot(range(1, 11), inertia)  
plt.title('Метод Elbow')  
plt.xlabel('Кількість кластерів')  
plt.ylabel('Інерція')  
plt.show()  
  
*# Виконуємо кластеризацію*kmeans = KMeans(n\_clusters=4, n\_init=10, random\_state=42)  
y\_kmeans = kmeans.fit\_predict(X)  
  
*# Виконуємо візуалізацію  
for* i *in* range(4):  
 plt.scatter(X[y\_kmeans == i, 0], X[y\_kmeans == i, 1], s=100, label=f'Кластер {i}')  
plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, label='Центри кластерів')  
plt.title('Кластеризація даних про індекс S&P 500')  
plt.xlabel('Індекс дня')  
plt.ylabel('Значення індексу S&P 500')  
plt.legend()  
plt.show()

task2.py

*"""  
Кластеризація за кольором водойм на зображенні з використанням методу k-means  
https://www.bing.com/maps?cp=51.524207%7E23.858193&lvl=12.3&style=a  
"""  
  
import* cv2  
*import* numpy *as* np  
*from* matplotlib *import* pyplot *as* plt  
  
  
*def* show\_image(image):  
 *""" Показ зображення """* plt.imshow(image)  
 plt.show()  
  
  
*def* image\_read(file\_image):  
 *""" Зчитування зображення з файлу """* image = cv2.imread(file\_image)  
 image\_in\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 *return* image\_in\_rgb  
  
  
*def* image\_processing(image):  
 *""" Обробка зображення """  
 # Застосовуємо Гаусівське розмиття* blurred\_image = cv2.GaussianBlur(image, (5, 5), 5)  
 *# Виділяємо темні області на зображенні* hsv\_image = cv2.cvtColor(blurred\_image, cv2.COLOR\_RGB2HSV)  
 \_, \_, v\_channel = cv2.split(hsv\_image)  
 \_, binary\_image = cv2.threshold(v\_channel, 40, 255, cv2.THRESH\_BINARY\_INV)  
 *# Перетворюємо маски в трьох канальне зображення* binary\_mask\_3channel = cv2.merge([binary\_image, binary\_image, binary\_image])  
 *# Застосовуємо маску до основного зображення* result = cv2.bitwise\_and(blurred\_image, binary\_mask\_3channel)  
 *# Застосовуємо згладжування* result = cv2.medianBlur(result, 9)  
 *return* result  
  
  
*def* save\_result(file\_name, image\_in\_rgb):  
 *""" Збереження результату у файл """* image\_in\_bgr = cv2.cvtColor(image\_in\_rgb, cv2.COLOR\_RGB2BGR)  
 cv2.imwrite(file\_name, image\_in\_bgr)  
  
  
*def* clustering(image):  
 *""" Кластеризація з допомогою k-means """  
 # Перетворюємо зображення у двовимірний масив* two\_dimension = image.reshape((-1, 3))  
 two\_dimension = np.float32(two\_dimension)  
 *# Критерії зупинки для алгоритму k-means* criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)  
 *# Кількість кластерів та спроб* k = 2  
 attempts = 10  
 *# Застосовуємо алгоритм k-means* ret, label, center = cv2.kmeans(two\_dimension, k, *None*, criteria, attempts, cv2.KMEANS\_PP\_CENTERS)  
 center = np.uint8(center)  
 *# Отримуємо результати та відновлюємо форму зображення* res = center[label.flatten()]  
 edged\_img = res.reshape(image.shape)  
 *return* edged\_img  
  
  
image\_entrance = image\_read('lakes.jpg')  
show\_image(image\_entrance)  
image\_exit = image\_processing(image\_entrance)  
show\_image(image\_exit)  
img\_clustered = clustering(image\_exit)  
show\_image(img\_clustered)  
save\_result('lakes\_result.jpg', img\_clustered)

task3.py

*"""  
Підрахунок кількості об'єктів на зображенні  
"""  
  
import* cv2  
*from* matplotlib *import* pyplot *as* plt  
  
  
*def* show\_image(image):  
 *""" Показ зображення """* plt.imshow(image)  
 plt.show()  
  
  
*def* image\_read(file\_image):  
 *""" Зчитування зображення з файлу """* image = cv2.imread(file\_image)  
 image\_in\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 *return* image\_in\_rgb  
  
  
*def* image\_processing(image):  
 *""" Обробка зображення """  
 # Збільшення контрасту* hsv\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)  
 h, s, v = cv2.split(hsv\_image)  
 v\_eq = cv2.equalizeHist(v)  
 hsv\_eq\_image = cv2.merge([h, s, v\_eq])  
 contrasted\_image = cv2.cvtColor(hsv\_eq\_image, cv2.COLOR\_HSV2RGB)  
 *# Застосування Гаусівського розмиття* blurred = cv2.GaussianBlur(contrasted\_image, (9, 9), 10)  
 *# Виділення світлих областей* hsv\_image = cv2.cvtColor(blurred, cv2.COLOR\_RGB2HSV)  
 \_, \_, v\_channel = cv2.split(hsv\_image)  
 \_, binary\_image = cv2.threshold(v\_channel, 180, 255, cv2.THRESH\_BINARY\_INV)  
 *# Перетворення маски в трьох канальне зображення* binary\_mask\_3channel = cv2.merge([binary\_image, binary\_image, binary\_image])  
 *# Застосування маски до основного зображення* image\_with\_mask = cv2.bitwise\_and(blurred, binary\_mask\_3channel)  
 *# Застосування згладжування* image\_with\_mask = cv2.medianBlur(image\_with\_mask, 7)  
 show\_image(image\_with\_mask)  
  
 *# Визначення кутів на зображенні та морфологічне закриття* edged = cv2.Canny(image\_with\_mask, 50, 150)  
 kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (7, 7))  
 closed = cv2.morphologyEx(edged, cv2.MORPH\_CLOSE, kernel)  
 *return* closed  
  
  
*def* image\_contours(image):  
 *""" Знаходження контурів """* contours, \_ = cv2.findContours(image.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 *return* contours  
  
  
*def* save\_result(file\_name, image\_in\_rgb):  
 *""" Збереження результату у файл """* image\_in\_bgr = cv2.cvtColor(image\_in\_rgb, cv2.COLOR\_RGB2BGR)  
 cv2.imwrite(file\_name, image\_in\_bgr)  
  
  
*def* image\_recognition(image, contours):  
 *""" Розпізнавання об'єктів """* total\_melons = 0  
 *for* contour *in* contours:  
 peri = cv2.arcLength(contour, *True*)  
 approx = cv2.approxPolyDP(contour, 0.05 \* peri, *True*)  
 *if* cv2.contourArea(contour) > 5000:  
 *if* 3 <= len(approx) <= 5:  
 cv2.drawContours(image, [approx], -1, (0, 255, 0), 4)  
 total\_melons += 1  
 *return* image, total\_melons  
  
  
image\_entrance = image\_read("watermelons.jpg")  
show\_image(image\_entrance)  
image\_exit = image\_processing(image\_entrance)  
show\_image(image\_exit)  
image\_contours = image\_contours(image\_exit)  
result, total\_figures = image\_recognition(image\_entrance, image\_contours)  
print(f"Знайдено {total\_figures} об'єктів")  
show\_image(result)  
save\_result("watermelons\_recognition.jpg", result)

## **IV. Висновки.**

У ході виконання лабораторної роботи було досліджено особливості аналізу даних з використанням методів машинного навчання. Було розроблено програмний скрипт, що виконує кластеризацію даних про індекс S&P 500, обробляє та кластеризує за кольором зображення Шацьких озер, обробляє та підраховує кількість шматків кавуна на зображенні.

Виконав: студент Трикош І. В.