МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ШТУЧНИЙ ІНТЕЛЕКТ ТА ГІБРИДНІ МЕРЕЖІ

Методичні вказівки до виконання лабораторних робіт
для здобувачів ОП другого магістерського рівня вищої освіти
спеціальності 122 «Комп'ютерні науки»

Київ 2024

УДК 004.8
М

Укладачі: С. Ю. Долгополов, асистент

Рецензент І. А. Ачкасов, доктор технічних наук, професор

Відповідальний за випуск Т. А. Гончаренко, канд. техн. наук, доцент, завідувач кафедри інформаційних технологій

Затверджено на засіданні кафедри інформаційних технологій, протокол № 7 від 09 лютого 2024 року.

В авторській редакції.

М Штучний інтелект та гібридні мережі: Методичні вказівки до виконання лабораторних робіт / уклад.: С. Ю. Долгополов. – Київ: КНУБА, 2024. – 261 с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт.

Призначено для здобувачів ОПП першого рівня вищої освіти (бакалавр) спеціальності 122 «Комп’ютерні науки».

ЗМІСТ

Вступ

Методичні вказівки до виконання лабораторних робіт розроблені відповідно до робочої програми дисципліни «Штучний інтелект та гібридні мережі». Сучасні технології штучного інтелекту та машинного навчання відкривають нові можливості для аналізу даних, розробки інтелектуальних систем і створення ефективних рішень для складних проблем. Завдання, що представлені в цих методичних вказівках, допоможуть здобувачам вищої освіти глибше зрозуміти принципи роботи алгоритмів машинного навчання, навчитися підбирати оптимальні моделі для різних типів даних і застосовувати їх для вирішення практичних завдань.

Перед проведенням лабораторних робіт здобувачі повинні ознайомитися з теоретичними аспектами задач штучного інтелекту, принципами підготовки та обробки даних, а також з основними методами оцінки ефективності моделей. Також необхідно звернути увагу на вивчення інструментів і середовищ розробки, таких як Google Colab, Kaggle, Jupyter Lab, та мов програмування Python і R, що будуть використані під час виконання робіт. Важливим етапом є також ознайомлення з правилами техніки безпеки при роботі з електронними пристроями та програмним забезпеченням.

Ці методичні вказівки спрямовані на розвиток аналітичних здібностей здобувачів, навичок роботи з великими обсягами даних та застосування теоретичних знань на практиці. Виконання лабораторних робіт дасть можливість здобувачам здобути практичний досвід у галузі штучного інтелекту, що є важливим кроком на шляху до формування професійних компетенцій в цій галузі.

Лабораторна робота № 1. Інтегрований аналіз даних в Data Science: практичне застосування та візуалізація

Мета: сформулювати практичні навички аналізу та візуалізації даних в контексті Data Science. Розвинути компетенції в обробці даних, їх статистичному аналізі, моделюванні, а також візуалізації результатів аналізу для ефективної інтерпретації.

Теоретичні відомості

В області Data Science інтегрований аналіз даних відіграє ключову роль, об'єднуючи статистичний аналіз, машинне навчання, візуалізацію та обробку даних для отримання цінної інформації з великих та різноманітних наборів даних. Аналіз даних охоплює широкий спектр методів та процедур, які дозволяють перетворити сиру інформацію у корисні знання та інсайти.

Обробка даних передбачає очищення, нормалізацію, трансформацію та агрегування даних. Очищення даних включає в себе видалення або корекцію неповних, некоректних або нерелевантних частин даних. Нормалізація даних – це процес приведення різноманітних форматів даних до єдиного стандарту, що спрощує аналіз.

Статистичний аналіз забезпечує математичне осмислення даних, дозволяючи визначити тенденції, залежності та закономірності. Він включає в себе розрахунок основних статистичних показників, таких як середнє значення, медіана, мода, стандартне відхилення, а також складніші методи, такі як кореляційний аналіз та регресійний аналіз.

Машинне навчання використовується для створення моделей, які можуть робити прогнози або класифікувати дані на основі історичних даних. Воно охоплює різні типи навчання, включаючи навчання з учителем, без учителя та підкріплювальне навчання.

Візуалізація даних – це ефективний спосіб комунікації інформації, який дозволяє швидко ідентифікувати нові тренди та залежності. Вона включає в себе створення різних типів графіків, таких як лінійні діаграми, гістограми, точкові діаграми, коробчасті діаграми, теплові карти та інші.

Комплексний підхід до аналізу даних у Data Science вимагає знань та навичок у всіх цих областях, дозволяючи проводити глибокий та всеохоплюючий аналіз даних.

Приклад виконання роботи

Завдання

1. Створіть DataFrame зі стовпцями «City», «Latitude» та «Longitude». Згенеруйте дані для 10 різних міст з випадковими координатами.
2. Додайте стовпець «Distance_to_Capital», що показує відстань кожного міста до столиці вашої країни.
3. Створіть стовпець «Region», який категоризує міста на «North», «South», «East», «West» на основі їхніх координат.
4. Використовуючи бібліотеку для візуалізації геоданих (наприклад, Geopandas), створіть хороплетну карту, яка відображає «City» залежно від «Region».
5. Додайте стовпець «Population» з випадковими значеннями. Проаналізуйте кореляцію між «Population» та «Distance_to_Capital».

Контрольне запитання

Вкажіть, яка кількість міст визначена як «East» у стовпці «Region»?
Відповідь: _____

Хід роботи

1. Імпорт бібліотек, що будуть використані в ході роботи

```
import pandas as pd
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
from geopy.distance import great_circle
```

2. Створення DataFrame зі стовпцями «City», «Latitude» та «Longitude»

```
# Завдання №1. Створення Dataframe
np.random
cities = [f'City_{i}' for i in range(1, 11)]
latitudes = np.random.uniform(-90, 90, size=10)
longitudes = np.random.uniform(-180, 180, size=10)

df = pd.DataFrame({'City': cities, 'Latitude': latitudes, 'Longitude':
longitudes})
df.head(10)
```

	City	Latitude	Longitude
0	City_1	77.582328	-26.348098
1	City_2	13.785159	82.280663
2	City_3	61.136138	92.280741
3	City_4	22.192580	-36.859938
4	City_5	-31.586851	153.072039
5	City_6	41.042096	-106.736284
6	City_7	4.092593	-177.119024
7	City_8	42.626064	153.486255
8	City_9	-60.226899	-73.975844
9	City_10	33.670493	-119.897426

Рисунок 1.1. Результат виконання завдання №1

3. Реалізація стовпця «Distance_to_Capital»

```
# Завдання №2. Обчислення відстаней до столиці

capital_coords = (0, 0) # столиця знаходиться в координатах (0;0)
def calculate_distance(lat, lon):
    return great_circle(capital_coords, (lat, lon)).kilometers

df['Distance_to_Capital'] = df.apply(lambda x:
calculate_distance(x['Latitude'], x['Longitude']), axis=1)
df.head(10)
```

	City	Latitude	Longitude	Distance_to_Capital
0	City_1	-1.121451	175.901816	19542.691359
1	City_2	21.592029	28.708279	3931.877434
2	City_3	3.550000	-43.149178	4811.003766
3	City_4	-61.777549	18.341359	7041.834219
4	City_5	-86.656284	88.320395	9996.665953
5	City_6	-77.396014	60.923842	9330.674715
6	City_7	-2.457880	-84.628959	9410.875265
7	City_8	19.139303	-156.119460	16651.619365
8	City_9	12.393259	-46.769689	5338.723290
9	City_10	-32.874766	46.698303	6096.791795

Рисунок 1.2. Результат виконання завдання №2

4. Реалізація стовпця «Region»

```
# Завдання №3. Категоризація міст
def categorize_region(lat, lon):
    if lat >= 0:
        return 'North' if lon >= 0 else 'West'
    else:
        return 'South' if lon >= 0 else 'East'

df['Region'] = df.apply(lambda x: categorize_region(x['Latitude'],
x['Longitude']), axis=1)
df.head(10)
```

	City	Latitude	Longitude	Distance_to_Capital	Region
0	City_1	-1.121451	175.901816	19542.691359	South
1	City_2	21.592029	28.708279	3931.877434	North
2	City_3	3.550000	-43.149178	4811.003766	West
3	City_4	-61.777549	18.341359	7041.834219	South
4	City_5	-86.656284	88.320395	9996.665953	South
5	City_6	-77.396014	60.923842	9330.674715	South
6	City_7	-2.457880	-84.628959	9410.875265	East
7	City_8	19.139303	-156.119460	16651.619365	West
8	City_9	12.393259	-46.769689	5338.723290	West
9	City_10	-32.874766	46.698303	6096.791795	South

Рисунок 1.3. Результат виконання завдання №3

5. Візуалізація геоданих за допомогою «Geopandas»

```
# Завдання №4. Хороплетена карта

gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.Longitude,
df.Latitude))

world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
fig, ax = plt.subplots(figsize=(20, 6))
world.plot(ax=ax, color='lightgrey', edgecolor='black')
gdf.plot(ax=ax, color='red', marker='o', markersize=100, label='Cities')
plt.title('Географічне Розташування Міст', fontsize=20, fontweight='bold')
plt.xlabel('Довгота', fontsize=15)
plt.ylabel('Широта', fontsize=15)
plt.legend(fontsize=12)
plt.show()
```

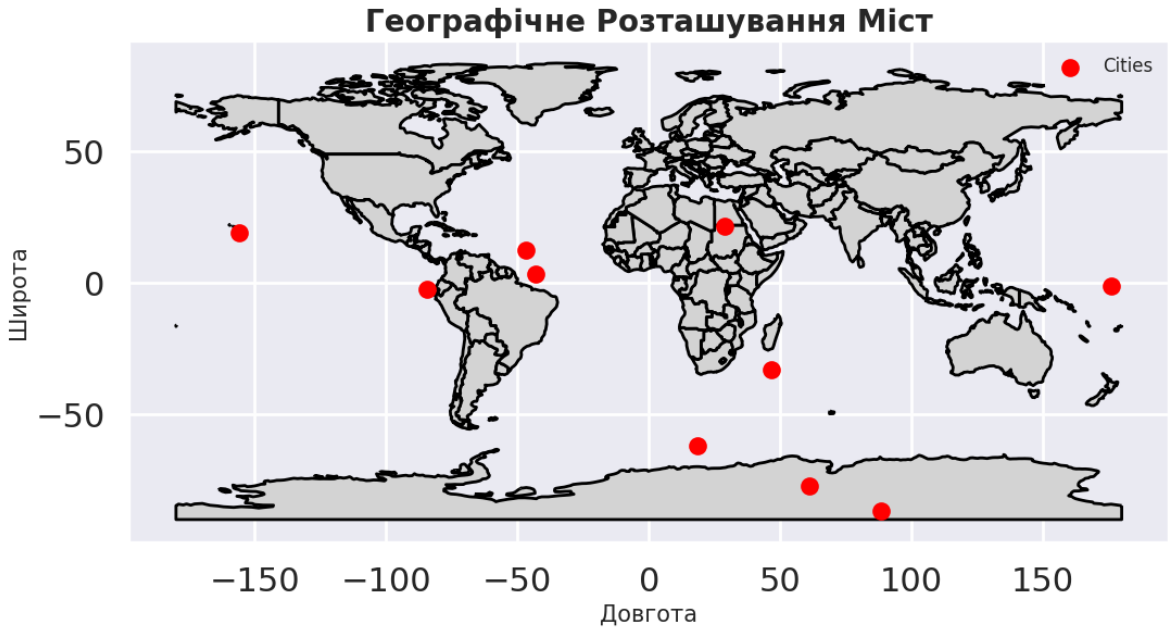


Рисунок 1.4. Результат виконання завдання №4

6. Аналіз кореляції між новим стовпцем «Population» та «Distance_to_Capital»

```
# Завдання №5. Кореляція між 'Population' та 'Distance_to_Capital'
df['Population'] = np.random.randint(1000, 100000, size=10)
df.head(10)
correlation = df[['Population', 'Distance_to_Capital']].corr()
print("Кореляція між населенням та відстанню до столиці:")
print(correlation)
```

	City	Latitude	Longitude	Distance_to_Capital	Region	Population
0	City_1	-1.121451	175.901816	19542.691359	South	92462
1	City_2	21.592029	28.708279	3931.877434	North	39553
2	City_3	3.550000	-43.149178	4811.003766	West	20686
3	City_4	-61.777549	18.341359	7041.834219	South	36056
4	City_5	-86.656284	88.320395	9996.665953	South	79896
5	City_6	-77.396014	60.923842	9330.674715	South	1690
6	City_7	-2.457880	-84.628959	9410.875265	East	7078
7	City_8	19.139303	-156.119460	16651.619365	West	26900
8	City_9	12.393259	-46.769689	5338.723290	West	58961
9	City_10	-32.874766	46.698303	6096.791795	South	60574

Рисунок 1.5. Результат виконання завдання №5

```
Кореляція між населенням та відстанню до столиці:
                    Population  Distance_to_Capital
Population             1.00000              0.30226
Distance_to_Capital    0.30226              1.00000
```

Рисунок 1.6. Результат виконання завдання №5

Контрольне запитання

Вкажіть, яка кількість міст визначена як «East» у стовпці «Region»?

Відповідь: **1**

Висновки: …

Завдання для виконання лабораторної роботи

Варіант 1.

1. Імпортуйте набір даних у DataFrame з файлу CSV. Файл повинен містити дані з трьома стовпцями: «Name» (string), «Age» (int), «City» (string). Кількість рядків повинна бути більшою за 30.
2. Видаліть стовпець «City» з імпортованого DataFrame.
3. Перейменуйте стовпець «Name» на «FirstName» та «Age» на «PersonAge».
4. Відфільтруйте дані, щоб в DataFrame залишились тільки рядки, де значення в «PersonAge» більше 30 років.
5. Виведіть 2, 6, 12, 24 рядки оновленого DataFrame.

Контрольне запитання

Вкажіть значення «PersonAge» у 12 рядку фінального DataFrame.
Відповідь: _____

Варіант 2.

1. Створіть DataFrame з двома стовпцями: «Temperature» (температура) та «Humidity» (вологість). Згенеруйте 100 випадкових значень для кожного стовпця, де «Temperature» в межах від -10 до 30 градусів, а «Humidity» – від 30% до 90%.
2. На основі значень у стовпці «Temperature» створіть новий стовпець «Temperature_Category», який класифікує температуру як «Low» (низька) для значень < 0, «Moderate» (помірна) для значень від 0 до 20, та «High» (висока) для значень > 20.
3. Створіть стовпець «Comfortable», де значення буде True, якщо «Temperature» знаходиться в межах від 18 до 24 градусів та «Humidity» від 40% до 60%, інакше False.
4. Додайте стовпець «Heat_Index», який розраховується за формулою:
$$Heat\_Index = \frac{Temperature \cdot Humidity}{100}$$
5. Виведіть описову статистику (середнє, мінімальне, максимальне значення) для стовпців «Temperature», «Humidity» та «Heat_Index».

Контрольне запитання

Вкажіть кількість значень «True» в стовпці «Comfortable».
Відповідь: _____

Варіант 3.

1. Створіть DataFrame з трьома стовпцями: «Product», «Sales», та «Region». «Product» містить назви трьох продуктів, «Sales» – кількість продажів (випадкові числа), «Region» – один з трьох регіонів продажу. Згенеруйте 50 випадкових записів.
2. Створіть гістограму для візуалізації розподілу продажів («Sales»).
3. Візуалізуйте продажі («Sales») для кожного продукту («Product») за допомогою коробчастої діаграми (boxplot).
4. Створіть точкову діаграму (scatter plot), щоб показати залежність між «Sales» та «Region».

5. Використовуйте стовпчикову діаграму для порівняння загальних продажів по кожному регіону («Region»).

Контрольне запитання

Вкажіть регіон з найбільшою кількістю загальних продажів.
Відповідь: _____

Варіант 4.

1. Створіть DataFrame з двома стовпцями: «Age» та «Income». Згенеруйте 100 випадкових значень для кожного стовпця, де «Age» випадково розподілене від 18 до 65 років, а «Income» – від 30.000 до 100.000.
2. Виведіть основні статистичні показники (середнє, медіана, стандартне відхилення) для «Age» та «Income».
3. Обчисліть та візуалізуйте кореляцію між «Age» та «Income».
4. Створіть новий стовпець «Log_Income», який є логарифмом значень «Income».
5. Виконайте нормалізацію стовпця «Income» за допомогою Min-Max scaling і додайте це як новий стовпець «Normalized_Income».

Контрольне запитання

Вкажіть показник «середнє» для «Income».
Відповідь: _____

Варіант 5.

1. Створіть DataFrame зі стовпцем «Date», який містить щоденні дати з початку поточного року до сьогодні. Додайте стовпець «Sales», що містить випадкові значення продажів.
2. Згрупуйте дані за місяцями та обчисліть загальні продажі в кожному місяці.
3. Якщо у «Sales» є пропущені значення, заповніть їх середнім значенням продажів.
4. Створіть стовпець «Previous_Day_Sales», який містить значення продажів попереднього дня. Для першого запису використовуйте значення 0.
5. Додайте стовпець «Cumulative_Sales», який показує кумулятивну (загальну) суму продажів з початку року.

Контрольне запитання

Вкажіть показник кумулятивної (загальної) суми продажів з початку року.
Відповідь: _____

Варіант 6.

1. Створіть два DataFrame: один зі стовпцями «ID», «Name», «Age» і другий зі стовпцями «ID», «Occupation». Згенеруйте 50 випадкових записів для кожного DataFrame і виконайте об'єднання (merge) даних за стовпцем «ID».
2. У результуючому DataFrame після об'єднання, створіть новий стовпець «Age_Group», який класифікує «Age» на категорії «Young», «Adult», «Senior».

3. Застосуйте функції для обробки текстових даних, щоб замінити символи «r» на «g» у стовпці «Name».
4. Використовуйте метод apply для створення нового стовпця «Length_Name», який містить кількість символів у кожному імені у стовпці «Name».
5. Створіть візуалізацію, що поєднує гістограму розподілу «Age» та кругову діаграму розподілу «Occupation».

Контрольне запитання

Вкажіть кількість рядків, що мають категорію «Senior» у стовпці «Age_Group».
Відповідь: _____

Варіант 7.

1. Створіть DataFrame зі стовпцями «Month», «Product_A_Sales», «Product_B_Sales». Згенеруйте 12 рядків, що представляють місяці року, та випадкові значення продажів для двох продуктів.
2. Створіть лінійний графік для порівняння місячних продажів «Product_A_Sales» та «Product_B_Sales».
3. Створіть гістограму для візуалізації розподілу продажів «Product_A_Sales».
4. Використовуйте скрипкову діаграму для візуалізації розподілу продажів обох продуктів.
5. Створіть теплову карту (heatmap), щоб візуалізувати кореляцію між «Product_A_Sales» та «Product_B_Sales» у різні місяці.

Контрольне запитання

Вкажіть показник кореляції «Product_A_Sales» та «Product_B_Sales» у лютому.
Відповідь: _____

Варіант 8.

1. Створіть DataFrame, який містить стовпець «Timestamp» з випадковими датами та часом протягом останнього року. Додайте стовпець «Sales», що містить випадкові значення продажів.
2. Додайте стовпець «Sales_Change», який показує відсоткову зміну продажів порівняно з попереднім днем.
3. Згрупуйте дані за місяцями та розрахуйте середні продажі в кожному місяці.
4. Створіть графік, який відображає середні місячні продажі.
5. Визначте 5 днів із найвищими продажами та виведіть відповідні дати та значення продажів.

Контрольне запитання

Вкажіть дату дня з найвищими продажами.
Відповідь: _____

Варіант 9.

1. Створіть DataFrame зі стовпцями «Player», «Points», «Assists», «Rebounds». Згенеруйте випадкові спортивні статистики для 30 гравців. Виведіть загальну кількість «Points», «Assists» і «Rebounds» для всіх гравців.
2. Виберіть та виведіть дані гравців, які набрали більше 20 очок («Points») та 5 асистів («Assists»).
3. Відсортуйте DataFrame за «Points» у спадаючому порядку, а потім за «Assists» у зростаючому.
4. Додайте стовпець «Team», який містить назви команд (наприклад, TeamA, TeamB, TeamC). Розподіліть гравців по цих трьох командах випадковим чином.
5. Визначте середню кількість «Points» та «Rebounds» для кожної команди.

Контрольне запитання

Вкажіть кількість гравців, що були розподілені до першої команди.
Відповідь: _____

Варіант 10.

1. Створіть DataFrame зі стовпцями «Name» та «Review». «Name» містить імена осіб, а «Review» – короткі текстові відгуки. Згенеруйте 50 випадкових записів.
2. Переведіть усі відгуки («Review») до нижнього регістру.
3. Використовуйте регулярні вирази для видалення всіх цифр із текстових відгуків.
4. Додайте стовпець «Word_Count», який показує кількість слів у кожному відгуку.
5. Створіть стовпець «Sentiment», який категоризує кожен відгук як «Positive», «Neutral», або «Negative» на основі випадково згенерованих значень.

Контрольне запитання

Вкажіть кількість відгуків, що були категоризовані як «Neutral».
Відповідь: _____

Варіант 11.

1. Створіть DataFrame зі стовпцями «Date» (дати за 2023 рік) та «Stock_Price». Згенеруйте випадкові значення для «Stock_Price».
2. Обчисліть та додайте стовпець з 7-денним рухомим середнім для «Stock_Price».
3. Виконайте сезонну декомпозицію для «Stock_Price» та візуалізуйте результат.
4. Обчисліть та візуалізуйте лагову кореляцію (кореляції між значеннями в часовому ряді) для «Stock_Price».
5. Використовуйте просту лінійну регресію для прогнозування «Stock_Price» на наступні 30 днів на основі існуючих даних.

Контрольне запитання

Вкажіть результат прогнозування «Stock_Price» на 30 день.
Відповідь: _____

Варіант 12.

1. Створіть DataFrame зі стовпцями «Date», «Company», «Revenue» та «Expenses». Згенеруйте дані для трьох компаній за останній рік з випадковими значеннями доходів та витрат.
2. Додайте стовпець «Profit», що розраховується як різниця між «Revenue» та «Expenses».
3. Групуйте дані за «Company» та обчисліть загальний прибуток для кожної компанії.
4. Визначте, в яку дату кожна компанія мала максимальний та мінімальний прибуток.
5. Створіть графіки, що показують місячні зміни «Revenue», «Expenses» та «Profit» для кожної компанії.

Контрольне запитання

Вкажіть значення мінімального прибутку компанії за весь період.

Відповідь: _____

Варіант 13.

1. Створіть DataFrame зі стовпцями «Person_ID», «Age», «Income», «Social_Media_Hours». Згенеруйте дані для 100 осіб з випадковими значеннями для кожного стовпця.
2. Категоризуйте осіб за віком: «Youth» (до 18 років), «Adult» (від 18 до 60 років) та «Senior» (понад 60 років).
3. Виведіть кореляційну матрицю для всіх числових стовпців.
4. Створіть гістограму, що показує розподіл кількості годин, проведених в соціальних мережах.
5. Обчисліть та візуалізуйте середній дохід у кожній віковій групі.

Контрольне запитання

Вкажіть середній дохід у віковій групі «Youth».

Відповідь: _____

Варіант 14.

1. Створіть DataFrame, який відображає щотижневі продажі продукту протягом року. Дані повинні включати стовпці «Week» та «Sales».
2. Проаналізуйте часовий ряд, виявіть основні тренди та сезонні коливання в даних.
3. Обчисліть 4-тижневу рухому середню продажів та додайте її як новий стовпець «Mean_Week» у DataFrame.
4. Використовуючи будь-яку методику прогнозування часових рядів, створіть прогноз продажів на наступні 3 місяці і візуалізуйте його разом із історичними даними.
5. Розрахуйте та проаналізуйте відхилення фактичних продажів від прогнозованих за останній місяць.

Контрольне запитання

Вкажіть тиждень, в якому було виконано найбільше продажів.

Відповідь: _____

Варіант 15.

1. Створіть DataFrame, який містить стовпці «Student_ID», «Subject», «Score». Згенеруйте дані для оцінок студентів з різних предметах.
2. Проведіть аналіз розподілу оцінок у кожному предметі. Визначте предмети з найвищими та найнижчими середніми оцінками.
3. Створіть візуалізації, що відображають розподіл оцінок у кожному предметі.
4. Розрахуйте середню оцінку кожного студента з усіх предметів.
5. Визначте студентів з найбільшими відхиленнями в оцінках між предметами.

Контрольне запитання

Вкажіть «Student_ID» студента, що має найбільше відхилення в оцінках.

Відповідь: _____

Варіант 16.

1. Створіть DataFrame зі стовпцями «Patient_ID», «Age», «Blood_Pressure», «Cholesterol_Level». Згенеруйте дані для 100 пацієнтів з випадковими значеннями для кожного стовпця.
2. Використовуючи дані про кров'яний тиск («Blood_Pressure») та рівень холестерину («Cholesterol_Level»), ідентифікуйте пацієнтів з потенційно високим ризиком серцевих захворювань. Створіть новий стовпець («Health_Hazards»).
3. Створіть графіки, які відображають розподіл «Blood_Pressure» та «Cholesterol_Level» серед пацієнтів.
4. Проаналізуйте середній «Blood_Pressure» та «Cholesterol_Level» у різних вікових групах.
5. Обчисліть кореляцію між «Age», «Blood_Pressure» та «Cholesterol_Level» та інтерпретуйте отримані результати.

Контрольне запитання

Вкажіть найбільший показник у стовпці «Health_Hazards».

Відповідь: _____

Варіант 17.

1. Створіть DataFrame зі стовпцями «Company_ID», «Quarter», «Revenue», «Expenses». Згенеруйте дані для кількох компаній за останні два роки.
2. Визначте компанії з найвищим та найнижчим квартальним приростом доходів.
3. Створіть лінійні діаграми, що відображають динаміку доходів і витрат для обраних компаній.
4. Розрахуйте маржу прибутку для кожної компанії та аналізуйте її зміни протягом часу.
5. Виконайте дослідження кореляції між «Revenue» та «Expenses» у компаніях.

Контрольне запитання

Вкажіть назву компанії з найвищим квартальним приростом доходів.

Відповідь: _____

Варіант 18.

1. Створіть DataFrame зі стовпцями «Incident_ID», «Date», «Type», «Location». Згенеруйте дані про різні безпекові інциденти, що сталися протягом останнього року.
2. Визначте, які типи інцидентів трапляються найчастіше, використовуючи групування та підрахунок.
3. Створіть графік, що показує кількість інцидентів у різних місцях.
4. Визначте, в які періоди року інциденти трапляються частіше.
5. Виконайте ручну заміну «Location» для 2, 6, 12 на NaN (Null).

Контрольне запитання

Вкажіть період року з найменшою кількістю інцидентів.
Відповідь: _____

Варіант 19.

1. Створіть DataFrame зі стовпцями «Observation_ID», «Date», «Time», «Location», «Description». Згенеруйте дані про заявлені спостереження невпізнаних літаючих об'єктів (НЛО) за останні декілька років.
2. Вивчіть розподіл спостережень НЛО за географічними регіонами та часом (в період з 00:00 до 06:00 та в період з 06:01 до 23:59).
3. Створіть графіки, які відображають розподіл спостережень за роками та місцезнаходженням.
4. Проаналізуйте описи спостережень, щоб виявити найчастіші слова або фрази, які використовуються свідками.
5. На основі описів та даних, оцініть потенційну надійність спостережень. Визначте, які з них можуть бути високо вірогідними та які – можливими помилковими повідомленнями.

Контрольне запитання

Вкажіть локацію де кількість заявлених спостережень є максимальною.
Відповідь: _____

Варіант 20.

1. Створіть DataFrame зі стовпцями «Car_ID», «Model», «Year», «Battery_Capacity», «Range». Згенеруйте дані про різні моделі автомобілів Tesla, включаючи інформацію про рік випуску, ємність батареї та пробіг.
2. Дослідіть, як ємність батареї впливає на загальний пробіг автомобіля.
3. Створіть лінійні діаграми або скатерплоти, що показують еволюцію ємності батареї та пробігу автомобілів Tesla за роками.
4. Порівняйте різні моделі автомобілів Tesla за ключовими характеристиками, включаючи ємність батареї та пробіг.
5. Розрахуйте та проаналізуйте показники енергоефективності для різних моделей Tesla, враховуючи ємність батареї та пробіг.

Контрольне запитання

Вкажіть модель автомобіля Tesla, яка має найбільший пробіг.

Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1NIK8xBm8ZbmzhuqUUgIcwbY3YtAHJjOZ#scrollTo=Je6rmFt5ApbY

Головне про Google Colab: **Google Colab** – це хмарний сервіс від Google, який дозволяє писати, запускати та ділитися кодом на Python за допомогою Jupyter Notebooks. Він надає безкоштовний доступ до обчислювальних ресурсів, що робить його чудовим вибором для запуску скриптів Python, особливо якщо ви не хочете встановлювати Python на своїй локальній машині.

Інструкція з використання Google Colab:

Старт роботи з Google Colab:

1. Перейдіть на https://colab.research.google.com/ і увійдіть за допомогою свого облікового запису Google.

Створіть новий блокнот або використовуйте вже готовий за посиланням https://colab.research.google.com/drive/1NIK8xBm8ZbmzhuqUUgIcwbY3YtAHJjOZ#scrollTo=Je6rmFt5ApbY

1. Натисніть «*File*» у верхньому лівому куті, а потім натисніть «*New notebook*». Відкриється нове вікно Jupyter Notebook у вашому браузері.
2. Натисніть «*Connect*» у верхньому правому куті та досліджуйте розроблений курс.

Перейменування блокноту

1. Натисніть на назву блокноту за замовчуванням «*Untitled0.ipynb*» у верхньому лівому кутку і дайте йому змістовну назву, наприклад, «*Лабораторна робота №1*».

Виконайте завдання лабораторної роботи

1. Використовуйте вже створені комірки блокноту для внесення коду на Python.
2. За необхідності створюйте нові комірки за допомогою комбінації клавіш *Ctrl + M + B* або просто натиснувши «*+ Code*» в проміжку між блоками.
3. Зберігайте позитивний настрій.

За додатковою інформацією щодо Google Colab звертайтеся до документації https://docs.github.com/en/pages

Лабораторна робота № 2. Аналіз та візуалізація великих наборів даних

Мета: ознайомити здобувачів з основами аналізу даних та візуалізації з використанням бібліотек Pandas та Seaborn в Python. Навчити здобувачів завантажувати дані з CSV-файлів, проводити попередню обробку даних, виконувати статистичний аналіз та будувати інформативні візуалізації. Розвинути навички командної роботи, взаємодії та послідовного виконання завдань.

Теоретичні відомості

Великі набори даних (Big Data) відіграють ключову роль у сучасному світі, охоплюючи різні сфери діяльності, від науки та бізнесу до державного управління та соціальних досліджень. Ефективний аналіз та візуалізація таких даних є важливими інструментами для отримання цінних інсайтів, виявлення прихованих закономірностей та підтримки прийняття обґрунтованих рішень.

Pandas – це потужна бібліотека Python, яка надає високоефективні, прості у використанні структури даних та інструменти аналізу даних. Вона спеціально розроблена для роботи з табличними даними, такими як дані з електронних таблиць, баз даних або CSV-файлів. Основними структурами даних в Pandas є Series (одномірний масив з мітками) та DataFrame (двовимірна таблиця з мітками рядків та стовпців). Pandas надає широкий спектр функцій для завантаження, очищення, перетворення, фільтрації, агрегації та аналізу даних.

Seaborn – це бібліотека Python для створення статистичних графіків, яка базується на бібліотеці Matplotlib. Seaborn надає високорівневий інтерфейс для створення інформативних та естетично привабливих візуалізацій даних. Вона тісно інтегрована з Pandas, що дозволяє легко будувати графіки на основі DataFrame. Seaborn автоматично оброблює багато деталей, пов'язаних з візуалізацією, таких як вибір кольорової палітри, налаштування осей та легенд, що дозволяє зосередитися на інтерпретації даних, а не на технічних деталях.

Приклад виконання роботи

Завдання

1. Завантажити набір даних «recent-grads.csv» у DataFrame Pandas. Цей набір даних містить інформацію про випускників коледжів США за 2010-2012 роки, включаючи демографічні дані, спеціальності, рівень зайнятості та доходи.
2. Провести попередню обробку даних: видалити стовпці, які не потрібні для аналізу, та обробити пропущені значення.
3. **Здобувач 1**. Розрахувати середній рівень безробіття серед випускників.
4. **Здобувач 1**. Побудувати гістограму розподілу медіанного заробітку випускників.
5. **Здобувач 1**. Знайти медіану середнього заробітку випускників, які мають частку жінок у спеціальності більше 0.5.
6. **Здобувач 2**. Використовуючи дані, передані **Здобувачем 1**, побудувати діаграму розсіювання, що показує залежність між часткою жінок у спеціальності та медіанним заробітком. Додати на графік горизонтальну лінію, що відповідає медіані середнього заробітку випускників, які мають частку жінок у спеціальності більше 0.5, розраховану **Здобувачем 1**.
7. **Здобувач 2**. Використовуючи дані, передані **Здобувачем 1**, побудувати ящик з вусами для медіанного заробітку випускників різних категорій спеціальностей.
8. **Здобувач 2**. Розрахувати **secret key** як суму індексів (номер рядка) шести спеціальностей з найвищим рівнем безробіття, використовуючи дані, передані **Здобувачем 1**.

Контрольне запитання

Як співвідноситься частка жінок у спеціальності з медіанним заробітком випускників? Чи є суттєва різниця у медіанному заробітку між різними категоріями спеціальностей?

Відповідь: _____

Хід роботи

0. Комунікація та планування (виконується обома учасниками).

Обговоріть завдання лабораторної роботи та переконайтеся, що обидва учасники розуміють цілі, завдання та послідовність виконання.

Сплануйте етапи виконання роботи, враховуючи дедлайни та час, необхідний для кожного етапу. Домовтесь про спосіб передачі даних між учасниками (наприклад, збереження проміжних результатів у CSV-файл, або безпосередньо в Google Colab).

1. Імпорт бібліотек та налаштування середовища (виконує **Здобувач 1**).

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_style("whitegrid")
```

На цьому етапі ми імпортуємо необхідні бібліотеки: Pandas для роботи з даними та Seaborn для візуалізації. Ми також налаштовуємо стиль графіків Seaborn, щоб зробити їх більш привабливими.

2. Завантаження даних (виконує **Здобувач 1**).

```
df = pd.read_csv("recent-grads.csv")
df.head(10)
```

	Rank	Major_code	Major	Total	Men	Women	Major_category	ShareWomen	Sample_size	Employed	...	Part_time	Full_t:
0	1	2419	PETROLEUM ENGINEERING	2339.0	2057.0	282.0	Engineering	0.120564	36	1976	...	270	
1	2	2416	MINING AND MINERAL ENGINEERING	756.0	679.0	77.0	Engineering	0.101852	7	640	...	170	
2	3	2415	METALLURGICAL ENGINEERING	856.0	725.0	131.0	Engineering	0.153037	3	648	...	133	
3	4	2417	NAVAL ARCHITECTURE AND MARINE ENGINEERING	1258.0	1123.0	135.0	Engineering	0.107313	16	758	...	150	
4	5	2405	CHEMICAL ENGINEERING	32260.0	21239.0	11021.0	Engineering	0.341631	289	25694	...	5180	
5	6	2418	NUCLEAR ENGINEERING	2573.0	2200.0	373.0	Engineering	0.144967	17	1857	...	264	
6	7	6202	ACTUARIAL SCIENCE	3777.0	2110.0	1667.0	Business	0.441356	51	2912	...	296	
7	8	5001	ASTRONOMY AND ASTROPHYSICS	1792.0	832.0	960.0	Physical Sciences	0.535714	10	1526	...	553	
8	9	2414	MECHANICAL ENGINEERING	91227.0	80320.0	10907.0	Engineering	0.119559	1029	76442	...	13101	
9	10	2408	ELECTRICAL ENGINEERING	81527.0	65511.0	16016.0	Engineering	0.196450	631	61928	...	12695	

10 rows × 21 columns

Рисунок 2.1. Результат виконання завдання №2

Тут ми використовуємо функцію `read_csv()` з бібліотеки Pandas, щоб завантажити дані з файлу «recent-grads.csv» у DataFrame, який ми назвали `df`. Цей файл повинен бути заздалегідь завантажений у робоче середовище Google Colab.

3. Попередня обробка даних (виконує **Здобувач 1**).

```
df = df.drop(['Rank', 'P25th', 'P75th'], axis=1)

df = df.dropna()
df.head(10)
```

	Major_code	Major	Total	Men	Women	Major_category	ShareWomen	Sample_size	Employed	Full_time	Part_time	F
0	2419	PETROLEUM ENGINEERING	2339.0	2057.0	282.0	Engineering	0.120564	36	1976	1849	270	
1	2416	MINING AND MINERAL ENGINEERING	756.0	679.0	77.0	Engineering	0.101852	7	640	556	170	
2	2415	METALLURGICAL ENGINEERING	856.0	725.0	131.0	Engineering	0.153037	3	648	558	133	
3	2417	NAVAL ARCHITECTURE AND MARINE ENGINEERING	1258.0	1123.0	135.0	Engineering	0.107313	16	758	1069	150	
4	2405	CHEMICAL ENGINEERING	32260.0	21239.0	11021.0	Engineering	0.341631	289	25694	23170	5180	
5	2418	NUCLEAR ENGINEERING	2573.0	2200.0	373.0	Engineering	0.144967	17	1857	2038	264	
6	6202	ACTUARIAL SCIENCE	3777.0	2110.0	1667.0	Business	0.441356	51	2912	2924	296	

Рисунок 2.2. Результат виконання завдання №3

На цьому етапі ми видаляємо стовпці «Rank», «P25th» та «P75th», які не будуть використовуватися в нашому аналізі, використовуючи метод `drop()`. Потім ми видаляємо рядки, які містять пропущені значення, за допомогою методу `dropna()`. Це важливо зробити, оскільки багато функцій аналізу та візуалізації не можуть коректно працювати з даними, що містять пропуски.

4. Розрахунок середнього рівня безробіття (виконує **Здобувач 1**).

```
df_variant35 = df[df['Major_category'].isin(["Law & Public Policy",
"Communications & Journalism"])]

employment_rate = df_variant35.groupby('Major')['Employed'].sum() /
(df_variant35.groupby('Major')['Employed'].sum() +
df_variant35.groupby('Major')['Unemployed'].sum())
print("Середній рівень працевлаштування для кожної спеціальності:")
print(employment_rate)

average_employment_rate_variant = employment_rate.mean()
print(f"Середній рівень працевлаштування для варіанту:
{average_employment_rate_variant}")

employment_rate_sorted = employment_rate.sort_values()
```

```
Середній рівень працевлаштування для кожної спеціальності:
Major
ADVERTISING AND PUBLIC RELATIONS        0.932039
COMMUNICATIONS                          0.924823
COURT REPORTING                         0.988310
CRIMINAL JUSTICE AND FIRE PROTECTION    0.917548
JOURNALISM                              0.930824
MASS MEDIA                              0.910163
PRE-LAW AND LEGAL STUDIES               0.928035
PUBLIC ADMINISTRATION                   0.840509
PUBLIC POLICY                           0.871574
dtype: float64
Середній рівень працевлаштування для варіанту: 0.915980575920436
```

Рисунок 2.3. Результат виконання завдання №4

Тут ми групуємо дані за спеціальностями «Major» за допомогою методу `groupby()`, а потім розраховуємо середній рівень безробіття «Unemployment_rate» для кожної спеціальності за допомогою методу `mean()`. Результат зберігається у змінній `average_unemployment_rate` та виводиться на екран.

5. Побудова гістограми розподілу медіанного заробітку (виконує **Здобувач 1**).

```
plt.figure(figsize=(12, 6))
sns.histplot(df_variant35['Total'], bins=20, kde=True)
plt.title('Розподіл кількості студентів на спеціальностях категорій Law & Public Policy та Communications & Journalism')
plt.xlabel('Кількість студентів')
plt.ylabel('Кількість спеціальностей')
plt.show()
```

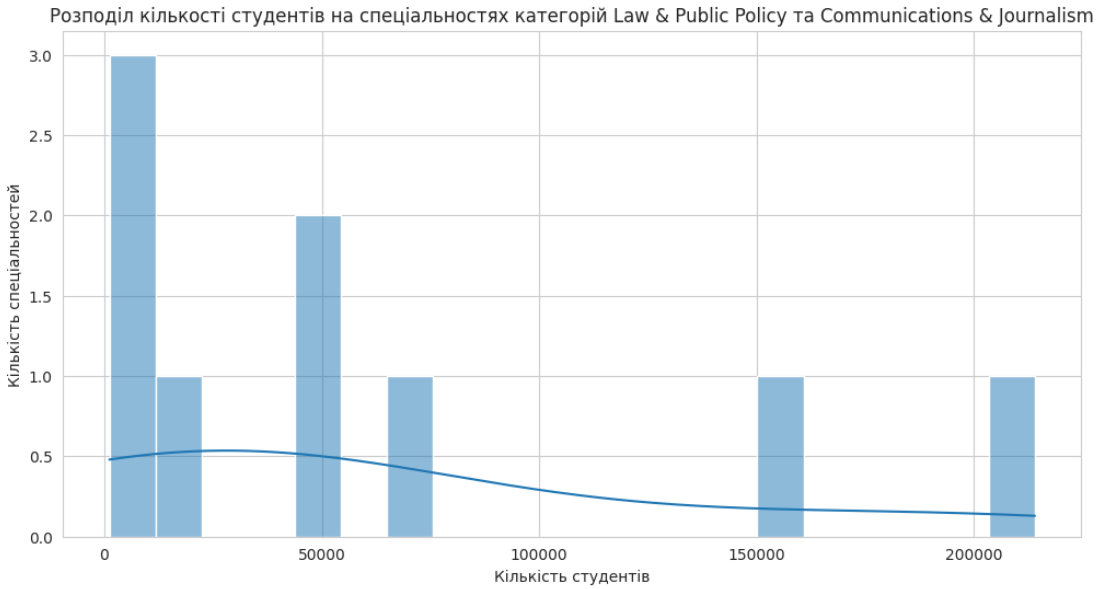


Рисунок 2.4. Результат виконання завдання №5

Ми використовуємо функцію `histplot()` з бібліотеки Seaborn для побудови гістограми розподілу медіанного заробітку «Median». Параметр `bins` задає кількість стовпчиків гістограми, а `kde=True` додає криву оцінки щільності розподілу. Ми також додаємо заголовок та підписи осей для кращого розуміння графіку.

6. Розрахунок медіани середнього заробітку для спеціальностей з часткою жінок > 0.5 (виконує **Здобувач 1**).

```
df_filtered = df_variant35[df_variant35.apply(lambda x: (x['Employed'] / (x['Employed'] + x['Unemployed'])) > average_employment_rate_variant, axis=1)]

median_men_high_employment_rate = df_filtered['Men'].median()

print(f"Медіана кількості чоловіків для спеціальностей з рівнем працевлаштування вище середнього: {median_men_high_employment_rate}")
```

```
Медіана кількості чоловіків для спеціальностей з рівнем працевлаштування вище середнього: 18299.0
```

Рисунок 2.5. Результат виконання завдання №6

7. Передача даних **Здобувачу 2** (виконує **Здобувач 1**).

```
employment_rate_sorted.to_csv("employment_rate_variant35.csv")
df_variant35.to_csv("variant35_data.csv", index=False)

print(f"Передайте Учаснику 2 наступне значення:
{median_men_high_employment_rate}")
```

```
Передайте Учаснику 2 наступне значення: 18299.0
```

Рисунок 2.6. Результат виконання завдання №7

8. Отримання даних від **Здобувача 1** (виконує **Здобувач 2**).

```
employment_rate_sorted = pd.read_csv("employment_rate_variant35.csv")
df_variant35 = pd.read_csv("variant35_data.csv")

median_men_high_employment_rate = 18299.0 # Замініть на значення, передане
Учасником 1!
```

9. Побудова діаграми розсіювання з горизонтальною лінією (виконує **Здобувач 2**).

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Men', y='Women', data=df_variant35)
plt.title('Залежність кількості жінок від кількості чоловіків на
спеціальностях (Варіант 35)')
plt.xlabel('Кількість чоловіків')
plt.ylabel('Кількість жінок')
plt.axhline(y=median_men_high_employment_rate, color='r', linestyle='--',
label=f'Медіана чоловіків: {median_men_high_employment_rate}')
plt.legend()
plt.show()
```

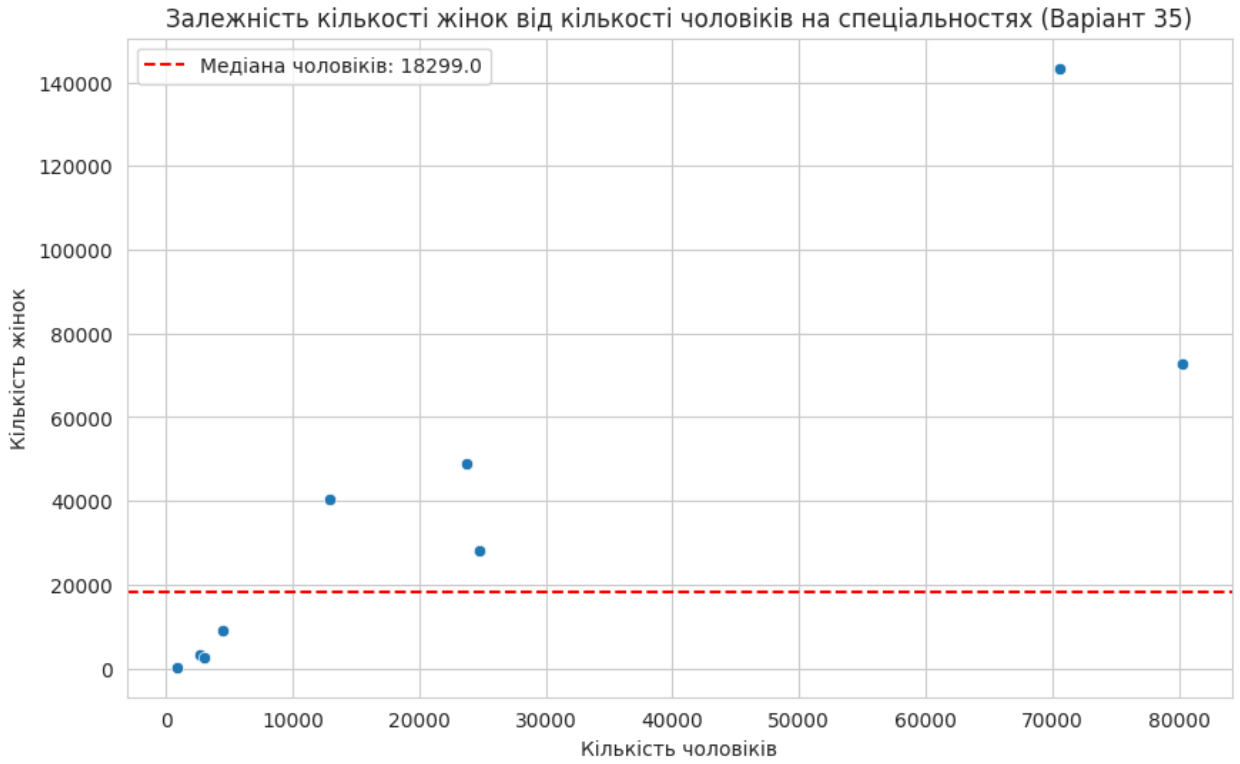


Рисунок 2.7. Результат виконання завдання №9

Ми використовуємо функцію `scatterplot()` з Seaborn для побудови діаграми розсіювання, яка показує залежність між часткою жінок у спеціальності «ShareWomen» та медіанним заробітком «Median». Кожна точка на графіку відповідає одній спеціальності.

10. Побудова ящика з вусами (виконує **Здобувач 2**).

```
plt.figure(figsize=(12, 8))
sns.boxplot(x='Major_category', y='Full_time', data=df_variant35)
plt.title('Розподіл кількості повного робочого дня для спеціальностей
(Варіант 35)')
plt.xlabel('Категорія спеціальності')
plt.ylabel('Кількість повного робочого дня')
plt.xticks(rotation=90)
plt.show()
```

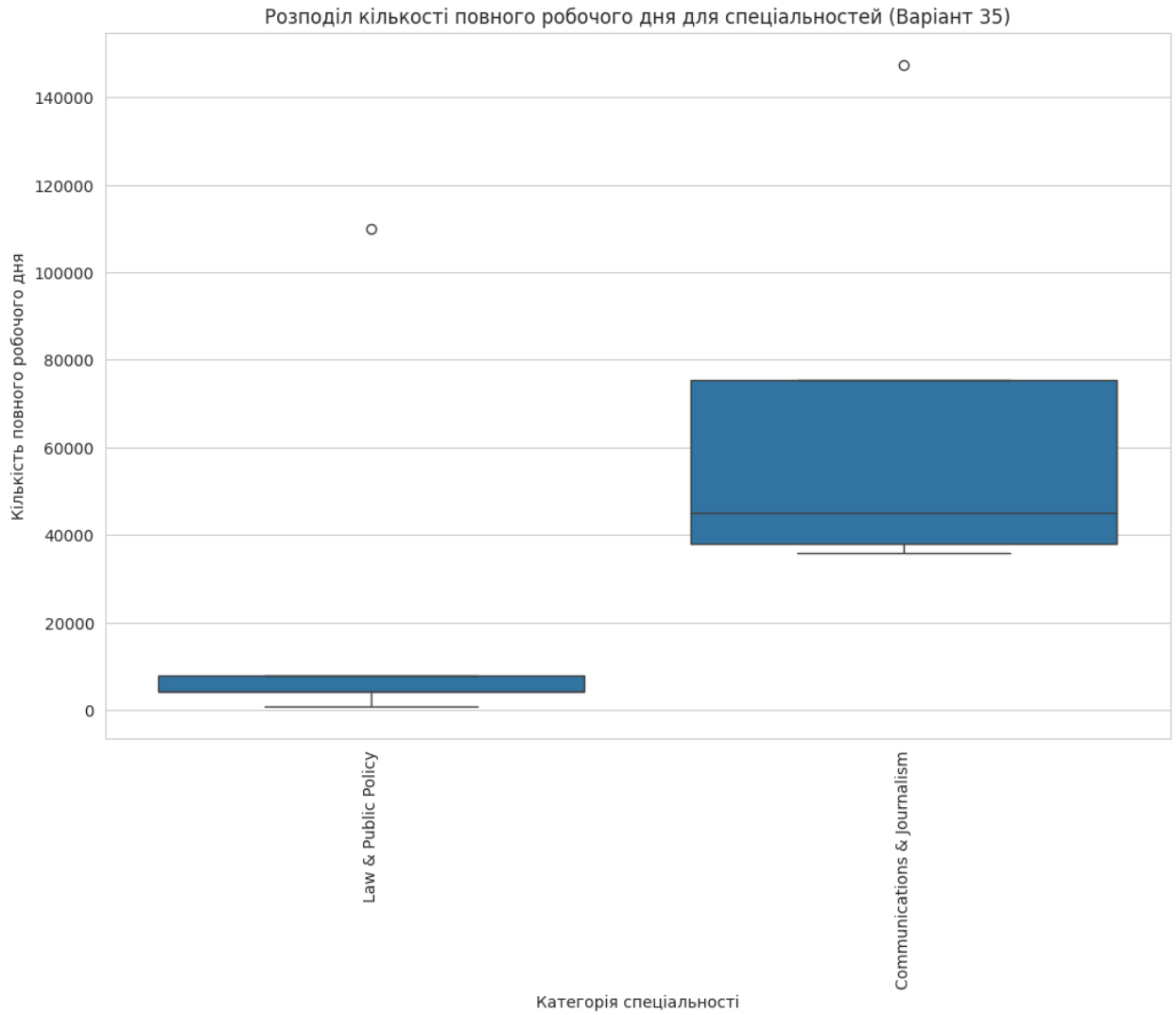


Рисунок 2.8. Результат виконання завдання №10

Ми використовуємо функцію `boxplot()` з Seaborn для побудови ящика з вусами, який показує розподіл медіанного заробітку для різних категорій спеціальностей «Major_category». Цей тип графіку дозволяє порівняти медіану, квартилі та викиди для різних груп даних. Ми також повертаємо підписи осі x на 90 градусів для кращої читабельності.

11. Розрахунок **secret key** (виконує **Здобувач 2**).

```
employment_rate_sorted = employment_rate.sort_values()
last_6_majors = employment_rate_sorted.head(6).index.tolist()
last_6_indices =
df_variant35.loc[df_variant35['Major'].isin(last_6_majors)].index.tolist()
unique_indices = []
[unique_indices.append(x) for x in last_6_indices if x not in
unique_indices]
checked_indices = [idx if idx != 0 else 1 for idx in unique_indices]


# Розраховуємо secret key як суму індексів
```

```
from math import prod

secret_key = prod(checked_indices)

print(f"Secret key: {secret_key}")
```

Secret key: 360360

12. Обговорення результатів та відповідь на контрольне запитання (виконують обидва здобувачі).

1. Обговоріть отримані результати та зробіть висновки на основі проведеного аналізу та візуалізацій.
2. Підготуйте спільну відповідь на контрольне запитання.
3. Перевірте secret key на https://keychecker.vercel.app/.

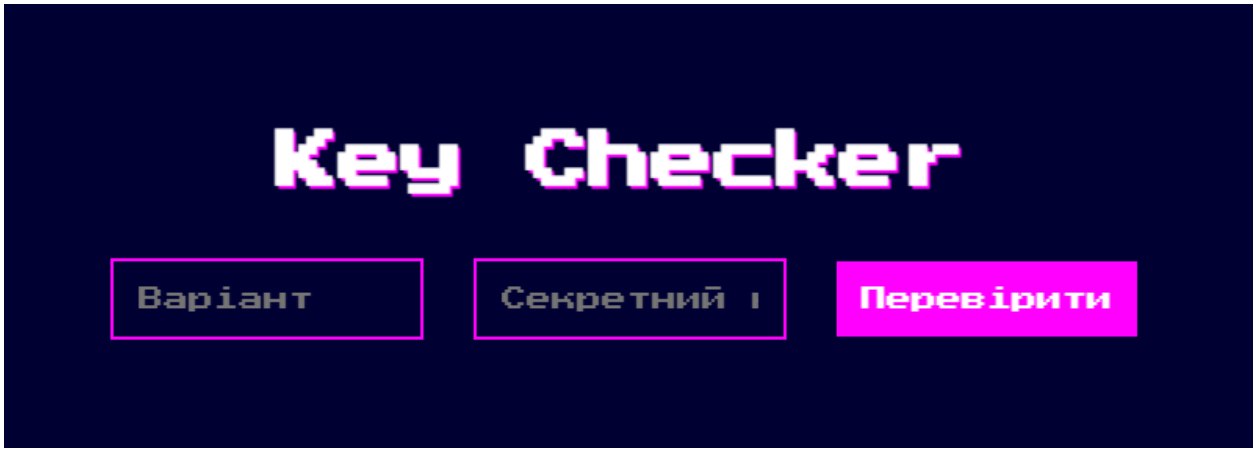


Рисунок 2.9. Результат виконання завдання №11

Контрольне запитання

Як співвідноситься частка жінок у спеціальності з медіанним заробітком випускників? Чи є суттєва різниця у медіанному заробітку між різними категоріями спеціальностей?
Відповідь: _____

Висновки: …

Завдання для виконання лабораторної роботи

1. Завантажити набір даних «recent-grads.csv» у DataFrame Pandas.
2. Провести попередню обробку даних: видалити стовпці, які не потрібні для аналізу, та обробити пропущені значення, якщо вони є.
3. **Здобувач 1**. Розрахувати середній рівень працевлаштування (Employed / (Employed + Unemployed)) серед випускників різних спеціальностей для кожного варіанту.
4. **Здобувач 1**. Побудувати гістограму розподілу кількості студентів на спеціальності (Total) для кожного варіанту.
5. **Здобувач 1**. Знайти медіану кількості чоловіків (Men) на спеціальностях, де рівень працевлаштування вище середнього для даного варіанту.
6. **Здобувач 2**. Використовуючи дані, передані **Здобувачем 1**, побудувати діаграму розсіювання, що показує залежність між кількістю чоловіків (Men) та кількістю жінок (Women) на спеціальності для кожного варіанту. Додати на графік горизонтальну лінію, що відповідає медіані кількості чоловіків на спеціальностях, де рівень працевлаштування вище середнього, розраховану **Здобувачем 1**.
7. **Здобувач 2**. Використовуючи дані, передані Учасником 1, побудувати ящик з вусами для кількості повного робочого дня (Full_time) випускників різних категорій спеціальностей для кожного варіанту.
8. **Здобувач 2**. Розрахувати **secret key** як суму індексів (номер рядка) шести спеціальностей з найнижчим рівнем працевлаштування, використовуючи дані, передані **Здобувачем 1**.
9. Спільне завдання для **Здобувача 1** та **Здобувача 2**. Побудувати стовпчикову діаграму з накопиченням (stacked bar chart), яка показує розподіл кількості працевлаштованих на повний робочий день (Full_time), неповний робочий день (Part_time) та безробітних (Unemployed) для кожної спеціальності у вашому варіанті.

Варіант 1.
Категорія спеціальності: «Engineering»

Варіант 2.
Категорія спеціальності: «Business»

Варіант 3.
Категорія спеціальності: «Physical Sciences»

Варіант 4.
Категорія спеціальності: «Health»

Варіант 5.
Категорія спеціальності: «Computers & Mathematics»

Варіант 6.
Категорія спеціальності: «Agriculture & Natural Resources»

Варіант 7.
Категорія спеціальності: «Computers & Mathematics», «Physical Sciences»

Варіант 8.
Категорія спеціальності: «Arts»

Варіант 9.
Категорія спеціальності: «Social Science»

Варіант 10.
Категорія спеціальності: «Psychology & Social Work»

Варіант 11.
Категорія спеціальності: «Communications & Journalism»

Варіант 12.
Категорія спеціальності: «Law & Public Policy»

Варіант 13.
Категорія спеціальності: «Education»

Варіант 14.
Категорія спеціальності: «Humanities & Liberal Arts»

Варіант 15.
Категорія спеціальності: «Biology & Life Science»

Варіант 16.
Категорія спеціальності: «Engineering», «Business»

Варіант 17.
Категорія спеціальності: «Physical Sciences», «Health»

Варіант 18.
Категорія спеціальності: «Computers & Mathematics», «Agriculture & Natural Resources»

Варіант 19.
Категорія спеціальності: «Social Science», «Agriculture & Natural Resources»

Варіант 20.
Категорія спеціальності: «Social Science», «Psychology & Social Work»
Варіант 21.
Категорія спеціальності: «Communications & Journalism», «Law & Public Policy»

Варіант 22.
Категорія спеціальності: «Education», «Humanities & Liberal Arts»

Варіант 23.
Категорія спеціальності: «Biology & Life Science», «Engineering»

Варіант 24.
Категорія спеціальності: «Business», «Physical Sciences»

Варіант 25.
Категорія спеціальності: «Health», «Computers & Mathematics»

Варіант 26.
Категорія спеціальності: «Education», «Psychology & Social Work»

Варіант 27.
Категорія спеціальності: «Arts», «Social Science»

Варіант 28.
Категорія спеціальності: «Psychology & Social Work», «Communications & Journalism»

Варіант 29.
Категорія спеціальності: «Law & Public Policy», «Education»

Варіант 30.
Категорія спеціальності: «Humanities & Liberal Arts», «Biology & Life Science»

Welcome to Google Colab

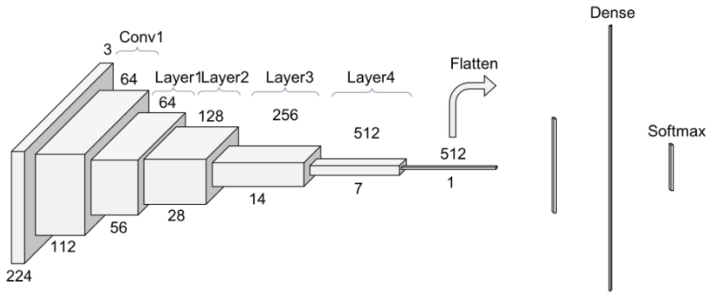
https://colab.research.google.com/drive/1tb5_4PqsEefLDr6r2j29GdWSPnV86nDg?usp=sharing

Лабораторна робота № 3. Використання Label Studio для анотації, класифікація та детекція об’єктів на зображеннях з використанням ResNet та Yolo

Мета: сформулювати практичні навички у сфері анотації зображень, використання переднавчених конволюційних нейронних мереж (CNN) для класифікації зображень, а також основ роботи з YOLOv5 й YOLOv8 для детекції об’єктів. Учасники лабораторної роботи навчаться організовувати та підготовлювати датасети, застосовувати інструменти для анотування зображень, проводити тести та порівнювати різноманітні архітектури нейронних мереж. Також вони отримають досвід у розробці власних моделей для ефективного розв’язання специфічних задач класифікації та детекції об’єктів.

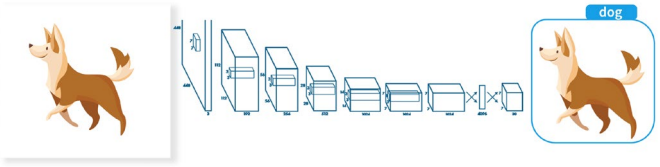
Теоретичні відомості

В останні роки область комп’ютерного зору зазнала значних змін завдяки розвитку глибинного навчання та конволюційних нейронних мереж (CNN). CNN стали основою багатьох сучасних систем обробки зображень, здатних виконувати широкий спектр завдань, від класифікації зображень до детекції об’єктів.

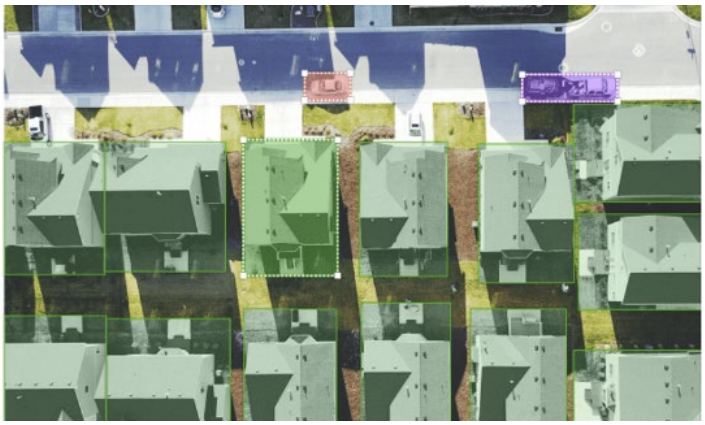


Класифікація зображень – це процес визначення, до якого класу належить зображення з заданого набору категорій. Переднавчені моделі, такі як ResNet (з різними варіантами глибини: 18, 34, 50, 101, 152), дозволяють використовувати переваги глибокого навчання без необхідності збирати величезні набори даних та витрачати значні ресурси на навчання моделей з нуля.

Детекція об’єктів зосереджена на визначенні місцезнаходження об’єктів на зображенні та їх класифікації. YOLO (You Only Look Once) є однією з передових архітектур для детекції об’єктів, що дозволяє виконувати детекцію в реальному часі з високою точністю. YOLOv5, використана у цій лабораторній роботі, є потужною ітерацією цієї архітектури, яка вирізняється покращеною продуктивністю, ефективністю та гнучкістю. Вона забезпечує значні поліпшення у швидкості та точності детекції порівняно з попередніми версіями, роблячи її ідеальним



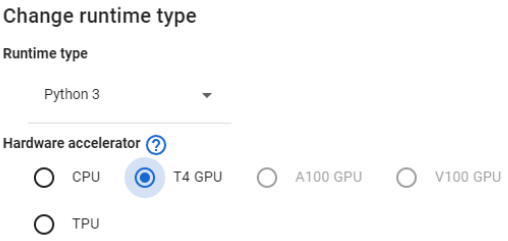
вибором для широкого спектру застосунків від простого виявлення об’єктів до складних систем візуального спостереження. YOLOv8 продовжує вдосконалювати можливості детекції, враховуючи найновіші досягнення у галузі глибинного навчання та комп’ютерного зору.



Анотація зображень є критично важливим етапом підготовки датасетів для тренування моделей глибинного навчання. Вона включає маркування зображень з метою навчання моделі розпізнавати і класифікувати об’єкти. Використання інструментів, таких як Label Studio, спрощує процес анотації, дозволяючи точно вказувати місцезнаходження та класи об’єктів на зображеннях.

Рекомендації до лабораторної роботи

При виконанні лабораторної роботи в Google Colab використовуйте «T4 GPU». Для цього оберіть «Runtime» → «Change runtime type» → «T4 GPU».



Приклад виконання роботи

Завдання

1. Використовуючи запропонований код, зберіть датасет зображень за темою, вказаною у вашому варіанті. Завантажте не менше 30 зображень для кожного класу об’єктів.
2. Використовуючи бібліотеку FastAI, протестуйте різні моделі ResNet (вказані у вашому варіанті) для класифікації зображень. Визначте, яка з моделей демонструє кращу точність, при параметрах навчання вашого датасету.
3. Використайте Label Studio для анотації зібраних зображень. Кожне зображення повинно містити принаймні 1 анотований об’єкт, що вказаний у вашому варіанті.
4. Налаштуйте та навчіть модель YoloV5 на вашому анотованому датасеті для детекції об'єктів. Використовуйте параметри навчання (кількість епох, розмір пакету), вказані у вашому варіанті.
5. Проаналізуйте результати класифікації та детекції. Визначте, які класи об’єктів модель визначає найточніше, та у яких виникають складнощі при розпізнавані.
6. Використайте претреновану модель YoloV8 для детекції довільних об’єктів на фото або відео.

Частина 1					**Частина 2**		
Варіант	**Тема***	**Перша Модель ResNet*****	**Друга Модель ResNet*****	**Кіль-кість епох (fine_tune)****	**Анотації в Label Studio***	**Кількість епох (YoloV5)****	**Розмірність пакету (batch_size)****
Т	Забори	18	34	5	Забори, ворота	5	16

Контрольне запитання №1

Вкажіть показник accuracy для останньої епохи навчання першої ResNet.
Відповідь: _____

Контрольне запитання №2

Вкажіть показник accuracy для останньої епохи навчання другої ResNet.
Відповідь: _____

Контрольне запитання №3

Вкажіть кількість проанотованих зображень.
Відповідь: _____

Контрольне запитання №4

Як впливає кількість епох на процес навчання моделі, і які потенційні ризики пов’язані з великою кількістю епох?

Відповідь: _____

_____

_____

Хід роботи

1. Встановлення необхідних бібліотек та модулів

```
import os
from google.colab import files
import socket,warnings
from fastbook import *
from fastdownload import download_url
from fastai.vision.all import *
from fastai.vision.widgets import *
from matplotlib import pyplot as plt
from time import sleep
from google.colab import files
import torch

!pip install fastbook
try:
    socket.setdefaulttimeout(1)
    socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect(('1.1.1.1',
53))
except socket.error as ex: raise Exception("STOP: No internet. Click '>|' in
top right and set 'Internet' switch to on")
iskaggle = os.environ.get('KAGGLE_KERNEL_RUN_TYPE', '')

if iskaggle:
    !pip install -U duckduckgo_search
    !pip install fastai
    !pip install fastbook

!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!pip install -qr requirements.txt

!pip install pillow==10.1.0
```

2. Пошук зображень за тематикою забори (fence) та ворота (gate).

```
# Завдання №1. Збір датасету зображень
def search_images(term, max_images=30):
    print(f"Searching for '{term}'")
    return L(search_images_ddg(term, max_images=max_images))

dest = 'fence.jpg'
download_url(urls[0], dest, show_progress=False)

im = Image.open(dest)
im.to_thumb(256,256)
```

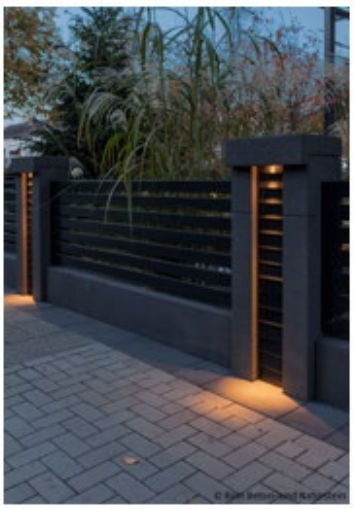


Рисунок 3.1. Екземпляр класу «fence»

```
download_url(search_images('gate photos', max_images=1)[0], 'gate.jpg',
show_progress=False)
Image.open('gate.jpg').to_thumb(256,256)
```

Рисунок 3.2. Екземпляр класу «gate»

1. Пошук та завантаження зображень при різному освітлені

```
searches = 'gate','fence'
path = Path('fence_or_not')

for o in searches:
    dest = (path/o)
    dest.mkdir(exist_ok=True, parents=True)
    download_images(dest, urls=search_images(f'{o} photo'))
    sleep(10)  # Pause between searches to avoid over-loading server
    download_images(dest, urls=search_images(f'{o} sun photo'))
    sleep(10)
    download_images(dest, urls=search_images(f'{o} shade photo'))
    sleep(10)
    resize_images(path/o, max_size=400, dest=path/o
```

2. Аналіз переліку завантажених зображень

```
print(path)
searches = 'gate','fence'

for imageType in searches:
    print(imageType)
    content = os.listdir(path/imageType)
    for element in content:
        print(element)
```

```
27f53785-ff92-4af2-aba9-cd73542d748c.jpg
62ae10aa-06a5-48c0-8ae9-4a9ba752bcdc.jpg
e3f2348b-cde1-40a5-a31d-2807a92dd170.jpg
e3b433ca-3a28-48df-aeae-8db594ae3187.jpg
6b5ea308-5851-4c3f-94e9-256e2372ef73.jpg
dc0f7491-a76f-4d78-b633-114c5d831f00.jpg
abb321d2-a971-44ba-a277-fc65879802bc.jpg
1fc3d332-4cdc-4a22-b52c-37f5c45260b7.jpg
306865db-6672-4d1d-a0ac-192a284688e8.jpg
927e0429-1d5b-4907-9435-2cf82dad61b4.jpg
08de0e4f-3b65-425a-a181-016845cf55a3.jpg
f1c670b0-067a-40c3-a2ef-f6ca05d48632.jpg
1f820bd1-02ed-4ecd-932d-771edd13d387.jpg
e916e477-906d-44f7-935f-8e45f9f74679.jpg
97463ee7-4eeb-4bd0-8ec1-20b53358bce5.jpg
a31352ba-dd4f-47dd-8b59-da6ba1a110f9.jpg
0a55b0a1-ba4e-4cdf-b74e-052143e57873.jpg
8cf00466-8aee-41cf-b42e-843115daa8aa.jpg
d89f84ac-6839-4be2-81f3-60790e4330b2.jpg
cd37b7ec-a60d-45ea-815a-dba490e84cb3.jpg
c2344974-5499-45f9-aed6-2026ae124d31.jpg
ca122e1a-8ab2-45f1-afaa-482e9d3ea4fc.jpg
68d7b202-1ed0-48a4-9825-a6bd3af54021.jpg
d2800340-6d66-4ff7-9128-8c6894cc5b75.jpg
0319ac99-1ff3-40bb-a592-de0d70e2fbd1.jpg
```

Рисунок 3.3. Частина завантажених зображень за темою

3. Аналіз завантажених зображень, пошук нерелевантних екземплярів

```
def list_images(directory):
    return [f for f in os.listdir(directory) if
os.path.isfile(os.path.join(directory, f))]

failed = verify_images(get_image_files(path))
failed.map(Path.unlink)
len(failed)

dls = ImageDataLoaders.from_folder(
    path, valid_pct=0.2, seed=42,
    item_tfms=Resize(224),
    batch_tfms=aug_transforms(mult=2)
)

dls.show_batch(max_n=6)
```

Рисунок 3.4. Аналіз екземплярів класів у датасеті

3. Тестування моделей ResNet18 та ResNet34.

```
# Завдання №2. Тестування різних моделей ResNet
learn = vision_learner(dls, resnet18, metrics=[error_rate, accuracy])
learn.fine_tune(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.397259	0.939227	0.363636	0.636364	00:38
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.925423	0.816139	0.393939	0.606061	00:40
1	0.775136	0.708150	0.333333	0.666667	00:41
2	0.661138	0.669650	0.242424	0.757576	00:40
3	0.627082	0.666702	0.272727	0.727273	00:39
4	0.550512	0.659468	0.272727	0.727273	00:40

Рисунок 3.5. Результат навчання моделі ResNet18

```
learn2 = vision_learner(dls, resnet34, metrics=[error_rate, accuracy])
learn2.fine_tune(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.390454	1.368886	0.393939	0.606061	00:52
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.882794	0.907909	0.424242	0.575758	01:14
1	0.807758	0.650826	0.303030	0.696970	01:14
2	0.772568	0.593983	0.333333	0.666667	01:17
3	0.682128	0.587102	0.272727	0.727273	01:17
4	0.621130	0.597385	0.272727	0.727273	01:14

Рисунок 3.6. Результат навчання моделі ResNet34

Таким чином, в однаковому діапазоні epoch при порівнянні моделей ResNet18 та ResNet34 вдалося досягти ідентичного показника точності (accuracy), проте моделі ResNet34 затребувалося на ~50% більше часу.

4. Анотування зображень в Label Studio

```
# Завантажимо Dataset
folder_path = '/content/fence_or_not'
!zip -r /content/fence_or_not.zip "$folder_path"
files.download('/content/fence_or_not.zip')
```

Перейдемо до сторінки з Label Studio та створимо новий інтерфейс клавішою «Create».

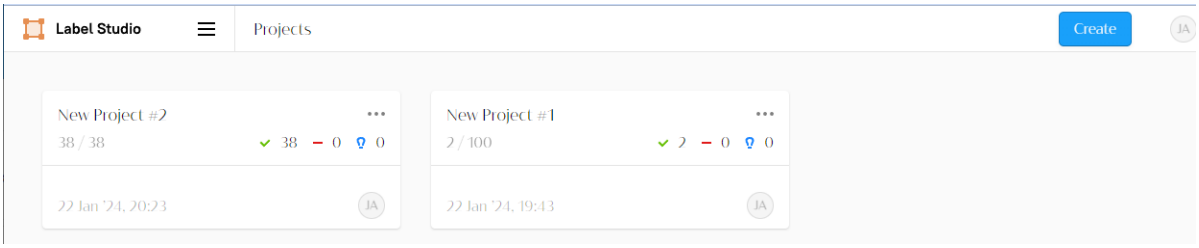


Рисунок 3.7. Загальний інтерфейс стартової сторінки Label Studio

Надамо назву проєкту «FENCE_OR_GATE».

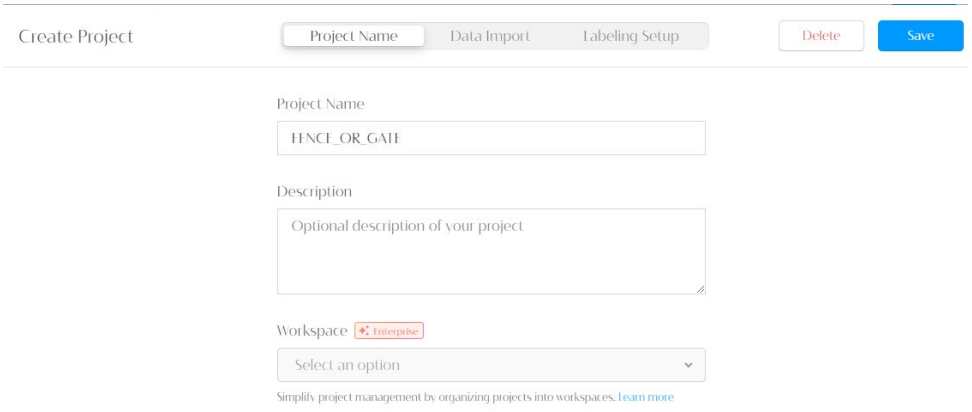


Рисунок 3.8. Встановлення назви проєкту

Перейдемо до завантаження зображень.

Примітка. Завчасно перегляньте завантажені зображення та видаліть такі, що не відповідають вашій темі, мають некоректний формат або пошкоджені, щоб уникнути помилки

⚠ *«is not supported»*

Також завантажуйте зображення партіями по 50 зображень, щоб не отримати помилку

⚠ *«The number of files exceeded settings. DATA_UPLOAD_MAX_NUMBER_FILES».*

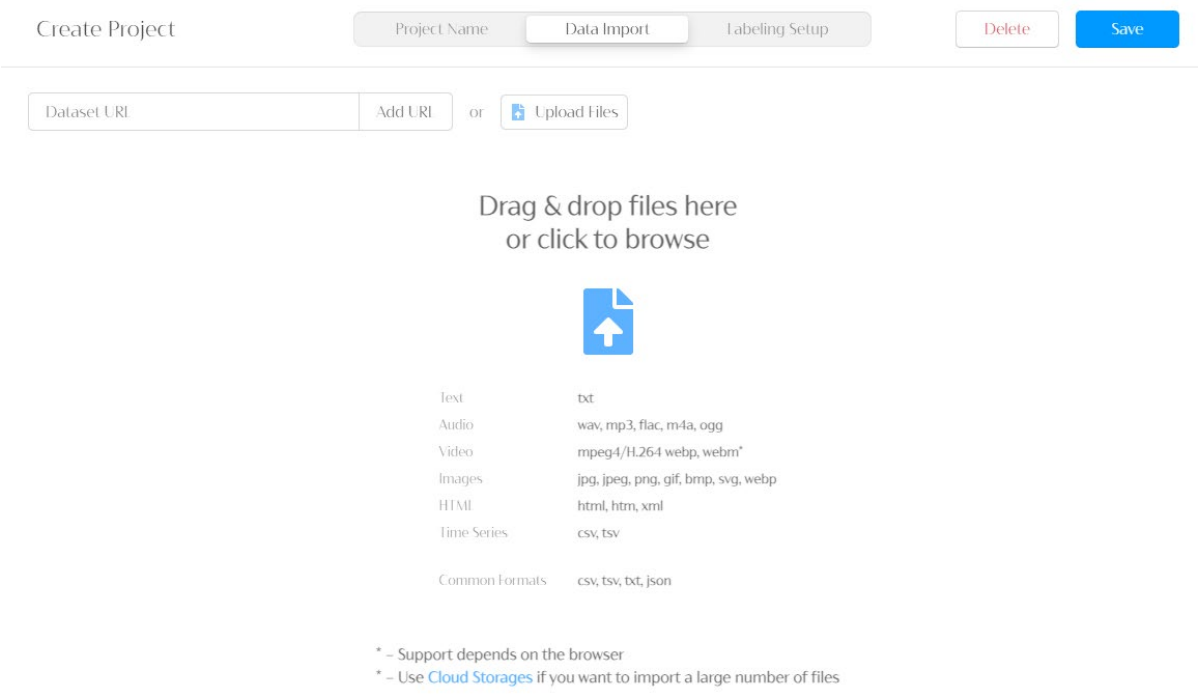


Рисунок 3.9. Вікно завантаження зображень

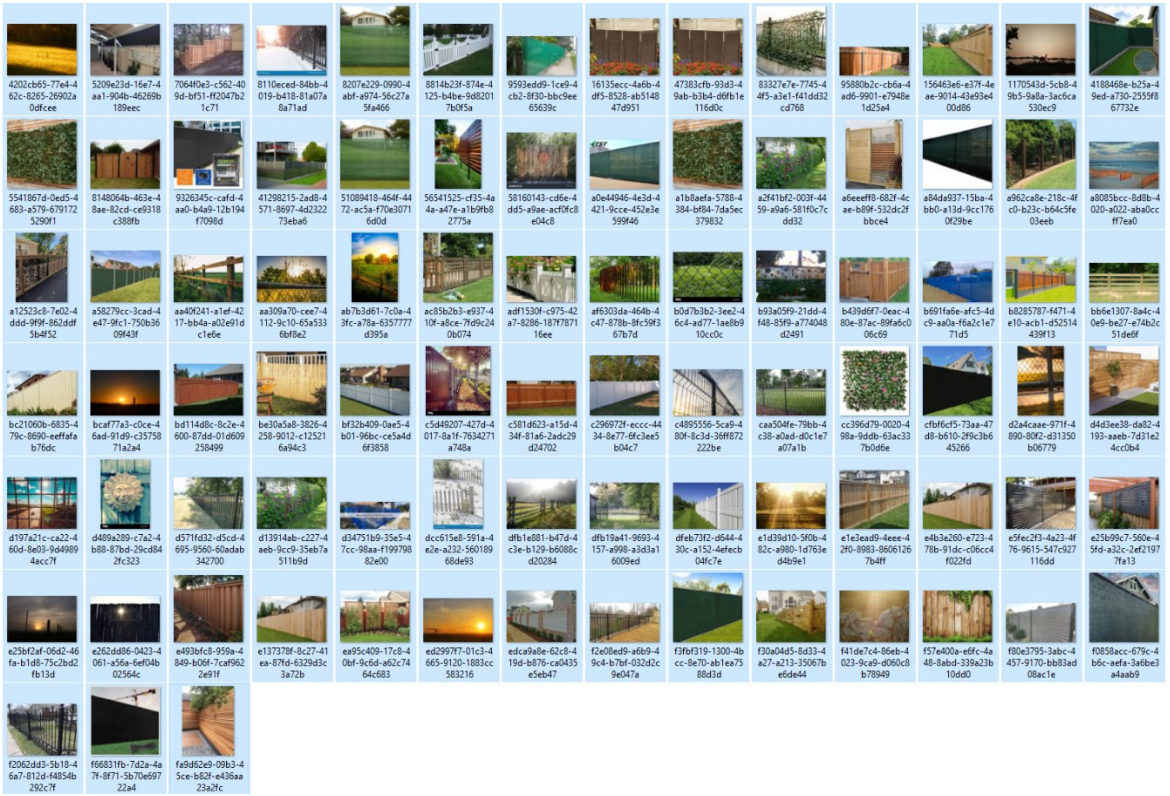


Рисунок 3.10. Вибір зображень заборів «fence»

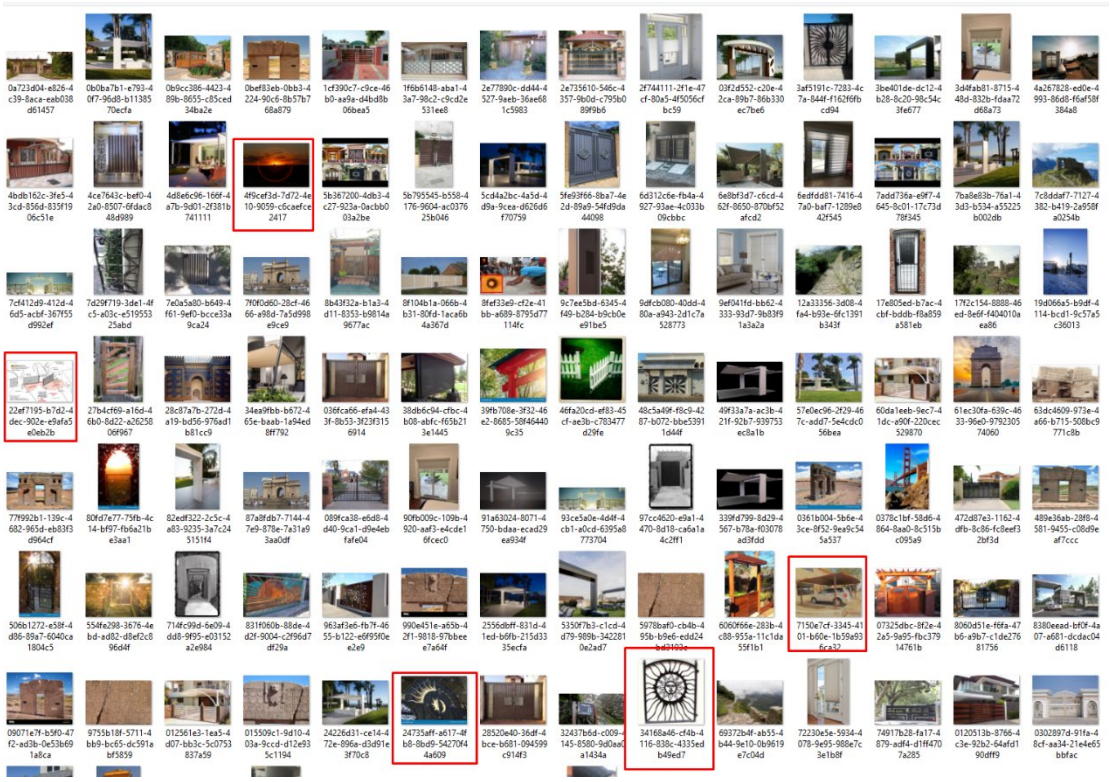


Рисунок 3.11. Вибір зображень воріт «gate», з видаленням невідповідних

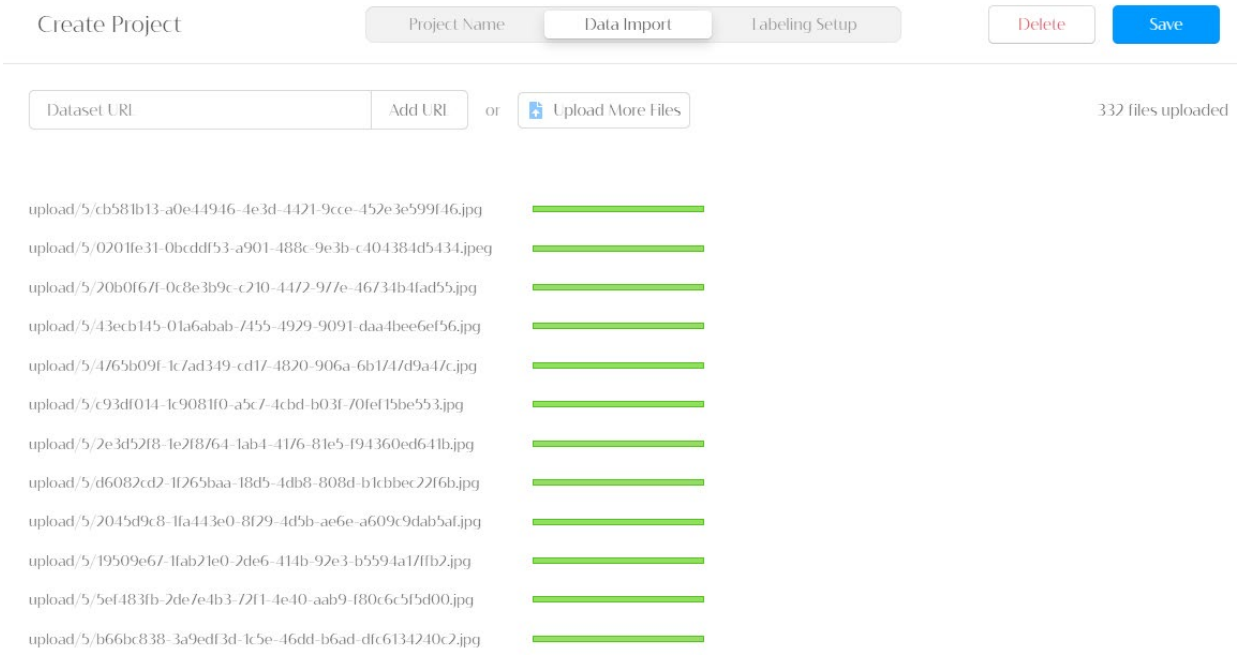


Рисунок 3.12. Завантажені файли зображень у Label Studio

Оберемо задачу «Object Detecting with Bounding Boxes» тобто розпізнавання об’єктів з використанням обмежувальної рамки.

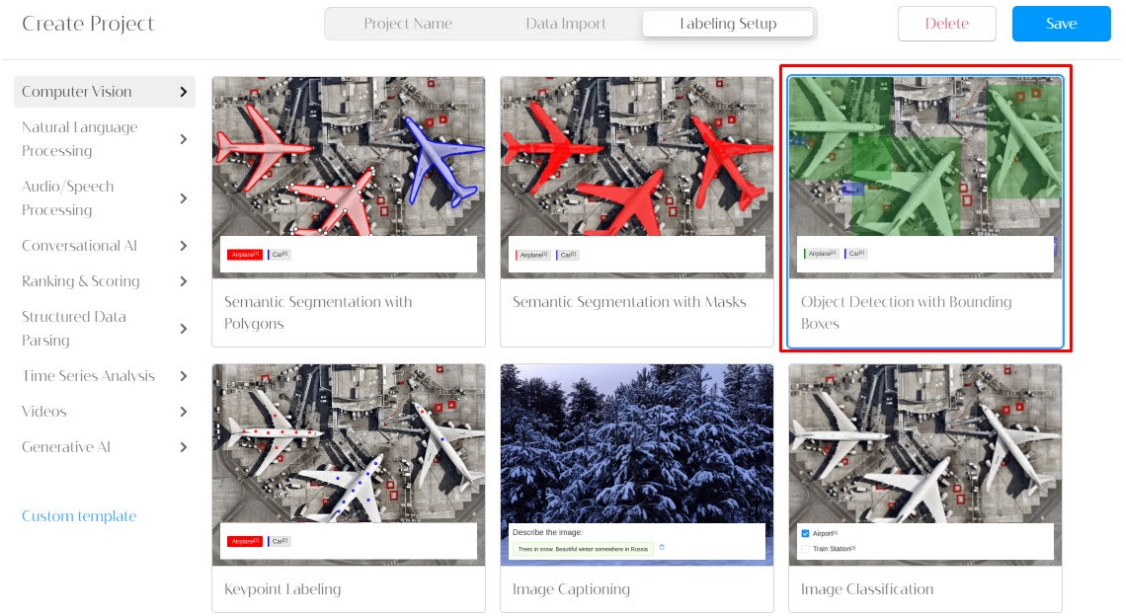


Рисунок 3.13. Завантажені файли зображень у Label Studio

Внесемо назви класів, що будуть розпізнаватися до відповідного поля.

Рисунок 3.14. Внесення класів, які будуть використані для анотування зображень

Рисунок. 3.15. Внесені мітки

Натиснемо на довільне зображень з переліченого списку та виконаємо анотування об’єкта за допомогою 1 для fence та 2 для gate.

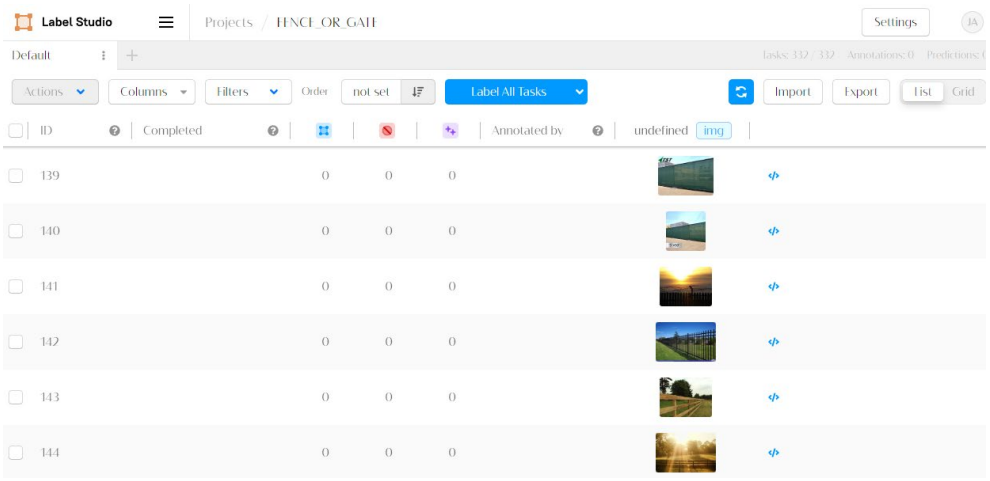


Рисунок 3.16. Загальний вигляд неанотованих зображень у Label Studio

Рисунок 3.17. Анотування забору та воріт одночасно

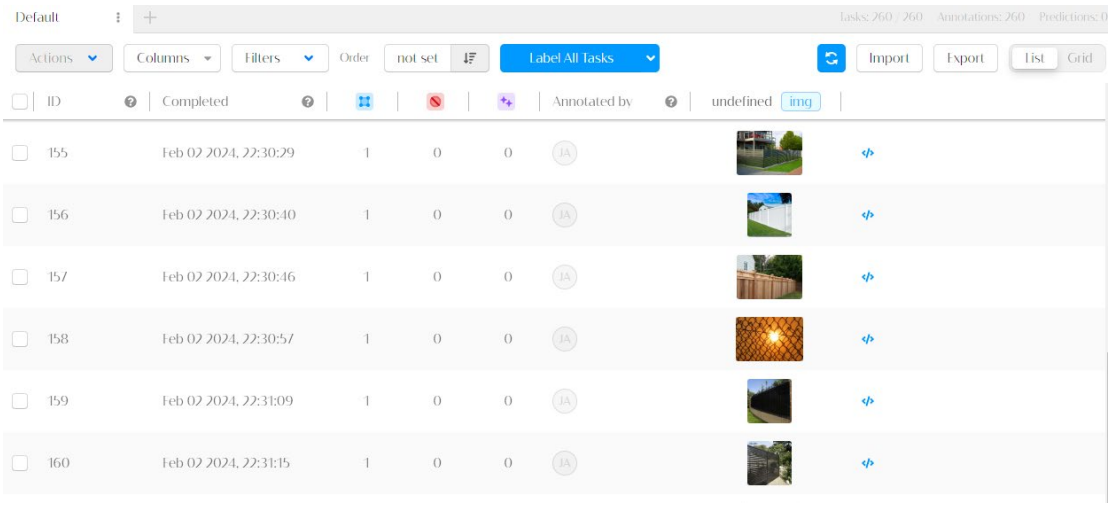


Рисунок 3.18. Проанотовані зображення

Експортуємо зображення та створені мітки для моделей YOLO за допомогою «Export» → «YOLO».

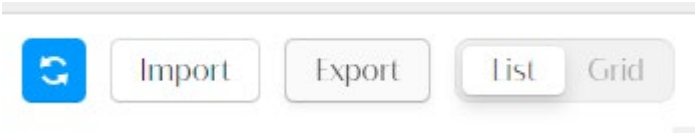


Рисунок 3.19. Експорт анотованих зображень

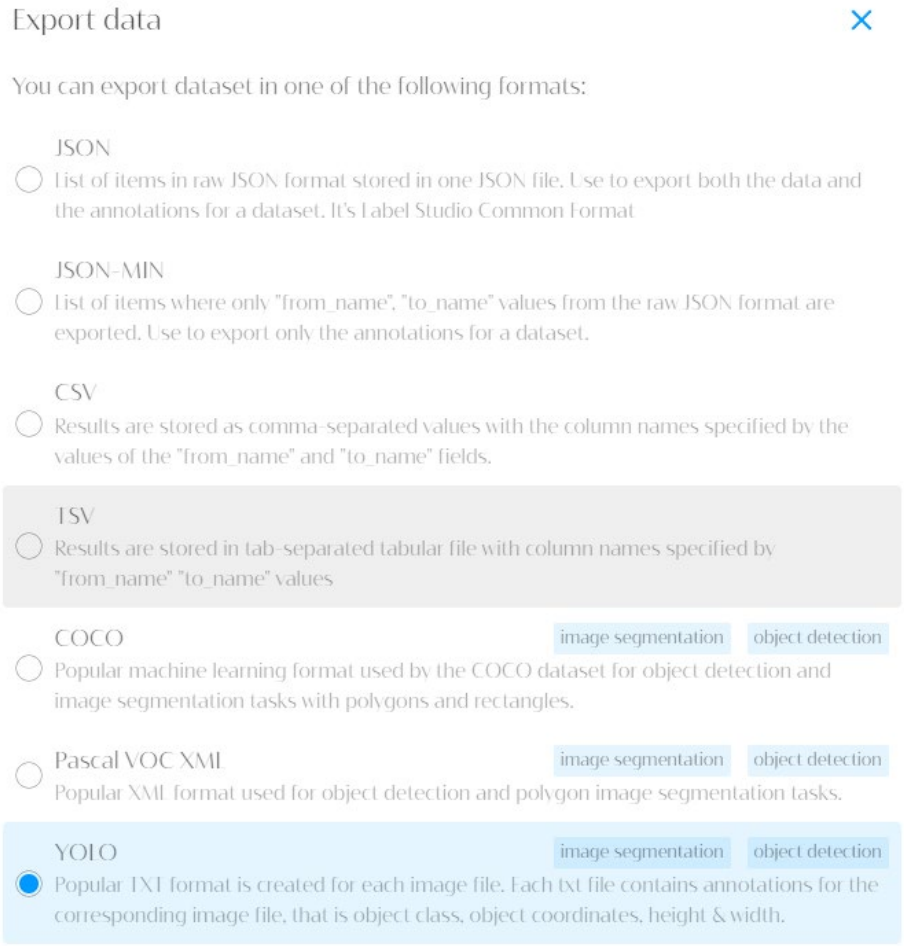


Рисунок 3.20. Експорт анотованих зображень для YOLO

5. Налаштування та тренування моделі YoloV5.

Перш за все організуємо експортовані дані за наступною схемою, де буде створена загальна папка «dataset», яка включатиме папку train, що налічуватиме 200 анотованих зображень та папку val – 60 анотованих зображень:

```
├── dataset
│   ├── classes.txt
│   │
│   ├── train
│   │   ├── images
│   │   └── labels
│   └── val
│       ├── images
│       └── labels
└── yolov5
    ├── data.yaml
    └── yolo5s.pt
```

```
# Завдання №4. Налаштування та тренування моделі YoloV5.

# Завантаження zip архіву з датасетом
uploaded = files.upload()
!unzip /content/proj5.zip -d /content/dataset/

images_path = '/content/dataset/train'  # шлях для загального пулу зображень
та анотацій (тренувальний набір)
val_images_path = '/content/dataset/validation'  # шлях до зображень для
перевірки

# Створеня тек за їх відсутності
os.makedirs(images_path, exist_ok=True)
os.makedirs(val_images_path, exist_ok=True)

# Підготовка файлу конфігурації для навчання
# Це файл YAML, в якому вказані шляхи до даних, параметрів моделі та
параметрів навчання

yaml_content = f"""
path: ../datasets  # шлях до набору даних
train: {images_path}  # шлях до тренувальних даних
val: {val_images_path}  # шлях до валідаційних даних

# Кількість класів
nc: 2

# Назви класів
names: ['fence', 'gate']
"""

with open("data.yaml", "w") as file:
    file.write(yaml_content)

# Навчання моделі, де --img - розмір зображення,
# --batch - кількість зображень, що використовуються за одну епоху,
# epoch -- кількість повних проходів навчальних даних
# -- data - шлях до конфігураційного файлу yaml
# --weights ваги, що в даному випадку визначаються моделю yolov5s.pt
```

```
# -- cache кеш для зображень та анотацій

# Примітка! Дані параметри вказані як демонстраційні.
# Задля досягнення якісного та швидкого навчання бажано використовувати GPU власного комп'ютера або іншої платформи
# зі збільшенням кількості epoch в діапазоні значень 30-50.

!python train.py --img 416 --batch 16 --epochs 5 --data data.yaml --weights yolov5s.pt --cache
```

```
30 epochs completed in 0.459 hours.
Optimizer stripped from runs\train\exp18\weights\last.pt, 14.3MB
Optimizer stripped from runs\train\exp18\weights\best.pt, 14.3MB

Validating runs\train\exp18\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 2/2 [00:04<0
                   all         60        123       0.63      0.582      0.604      0.333
                 fence         60         78      0.502      0.609      0.561      0.289
                  gate         60         45      0.758      0.556      0.647      0.377
Results saved to runs\train\exp18
```

Рисунок 3.21. Результат тренування моделі Yolov5

6. Проаналізуйте результати класифікації та детекції

Таким чином, загальний показник Precision для всіх класів становить 0.63, в той час як для fence – 0.502, для gate – 0.758. Показник Recall закономірно демонструє для всіх класів – 0.582, fence – 0.609, gate – 0.556. Можемо зробити висновок, що модель найкращим чином натренувалася на анотованих зображеннях воріт (gate), незважаючи на те, що екземплярів заборів (fence) було значно більше, що демонструє, що вхідні ознаки воріт є більш яскравими та доступними для навчання моделлю Yolov5, в той час як показник виявлення заборів наближається до показника відгадування, що може свідчити про недостатньо якісний або неповний набір даних. Але, при детальному аналізі на валідаційному наборі було також відмічено, що модель здебільшого коректно виділяє класи об’єктів, або взагалі не виділяє нічого.

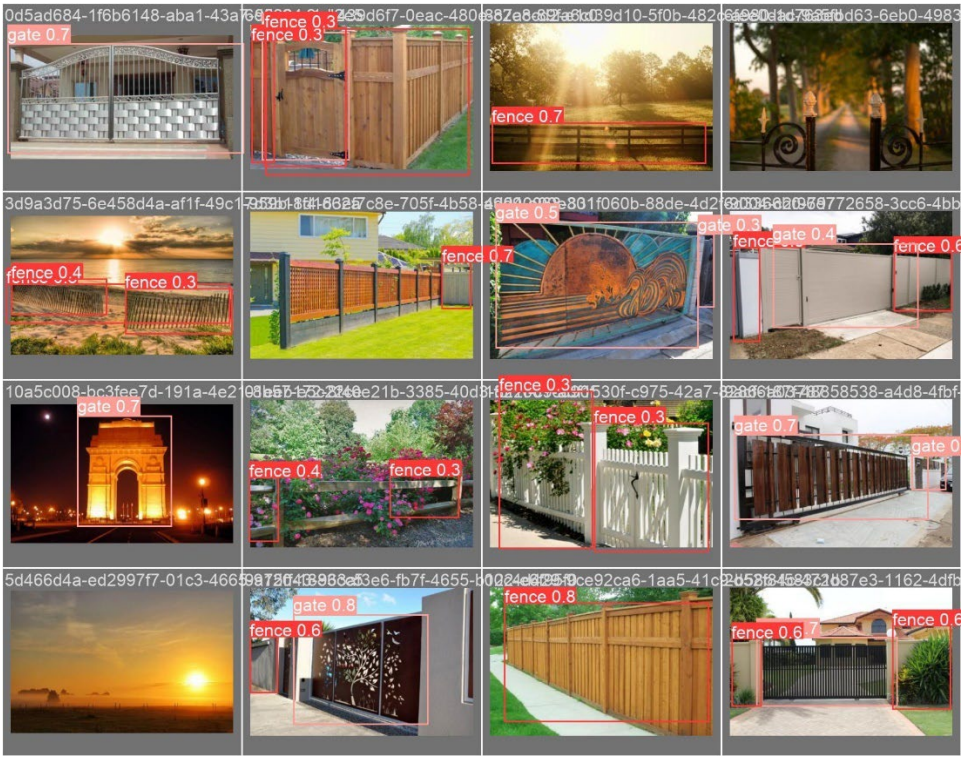


Рисунок 3.22. Результат детекції об’єктів моделі Yolov5

7. Використайте моделі YoloV8 для детекції довільних об’єктів на фото або відео.

```
# Завдання №6. Використання моделі Yolov8
# 1) pip install ultralytics
# 2) yolo predict model=yolov8n.pt
source='https://ultralytics.com/images/bus.jpg' # замініть посилання на
зображення.
# 3) yolo predict model=yolov8n.pt source='dance.mp4' # замініть шлях до
свого локального файлу.
```

```
C:\Users\Marano\PycharmProjects\pythonProject7\yolov5>yolo predict model=yolov8n.pt source='https://ultralytics.com/imag
es/bus.jpg'
Downloading https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8n.pt to 'yolov8n.pt'...
100%|                                                                                | 6.23M/6.23M [00:02<00:00, 2.55MB/s]
Ultralytics YOLOv8.1.9 🚀 Python-3.10.6 torch-2.2.0+cpu CPU (Intel Core(TM) i9-10940X 3.30GHz)
YOLOv8n summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFLOPs

Downloading https://ultralytics.com/images/bus.jpg to 'bus.jpg'...
100%|                                                                                | 476k/476k [00:00<00:00, 2.51MB/s]
image 1/1 C:\Users\Marano\PycharmProjects\pythonProject7\yolov5\bus.jpg: 640x480 4 persons, 1 bus, 1 stop sign, 123.7ms
Speed: 12.0ms preprocess, 123.7ms inference, 8.0ms postprocess per image at shape (1, 3, 640, 480)
Results saved to runs\detect\predict
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

Рисунок. 3.23. Результат виконання завдання №6

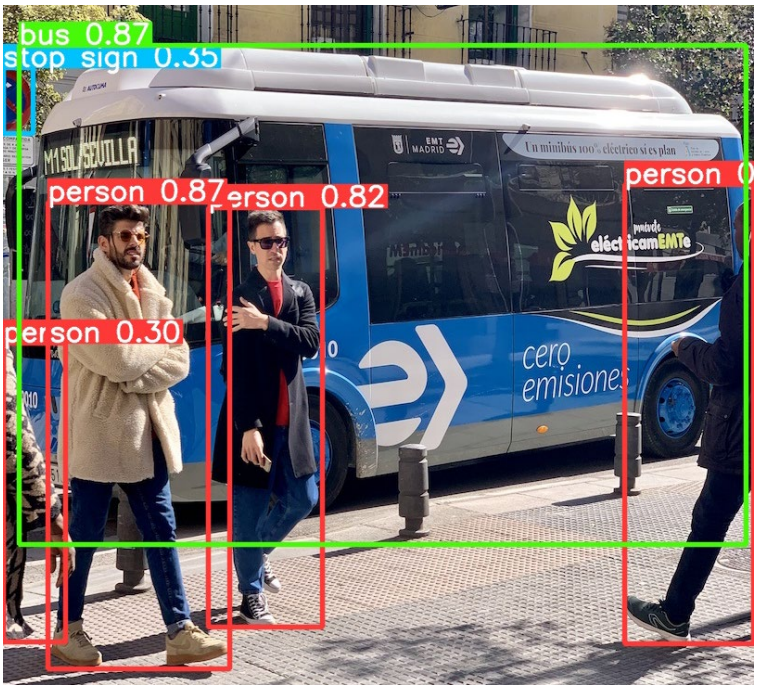


Рисунок. 3.24. Результат виконання завдання №6

```
video 1/1 (43/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 59.8ms
video 1/1 (44/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 1 person, 1 bottle, 55.9ms
video 1/1 (45/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 1 person, 1 bottle, 61.8ms
video 1/1 (46/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 1 person, 1 bottle, 49.9ms
video 1/1 (47/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 55.9ms
video 1/1 (48/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 56.8ms
video 1/1 (49/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 56.9ms
video 1/1 (50/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 53.9ms
video 1/1 (51/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 50.9ms
```

Рисунок. 3.25. Результат виконання завдання №6

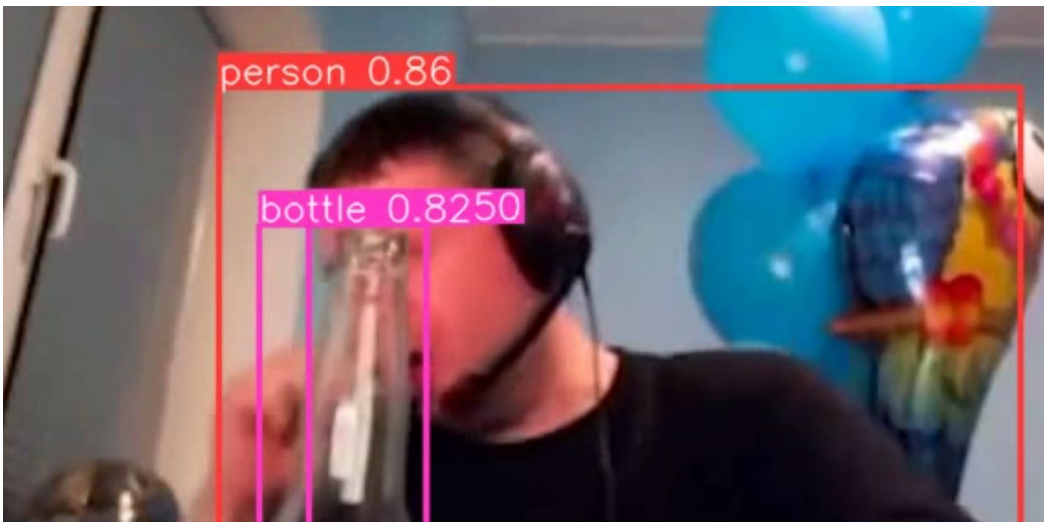


Рисунок. 3.26. Результат виконання завдання №6

Контрольне запитання №1

Вкажіть показник accuracy для останньої епохи навчання першої ResNet.

Відповідь: **0.727273**

Контрольне запитання №2

Вкажіть показник accuracy для останньої епохи навчання другої ResNet

Відповідь: **0.727273**

Контрольне запитання №3

Вкажіть кількість проанотованих зображень

Відповідь: **260**

Контрольне запитання №4

Як впливає кількість епох на процес навчання моделі, і які потенційні ризики пов’язані з великою кількістю епох?

Відповідь: …

Висновки: …

Завдання для виконання лабораторної роботи

1. Використовуючи запропонований код, зберіть датасет зображень за темою, вказаною у вашому варіанті. Завантажте не менше 30 зображень для кожного класу об’єктів.
2. Використовуючи бібліотеку FastAI, протестуйте різні моделі ResNet (вказані у вашому варіанті) для класифікації зображень. Визначте, яка з моделей демонструє кращу точність, при параметрах навчання вашого датасету.
3. Використайте Label Studio для анотації зібраних зображень. Кожне зображення повинно містити принаймні 1 анотований об’єкт, що вказаний у вашому варіанті.
4. Налаштуйте та навчіть модель YoloV5 на вашому анотованому датасеті для детекції об'єктів. Використовуйте параметри навчання (кількість епох, розмір пакету), вказані у вашому варіанті.
5. Проаналізуйте результати класифікації та детекції. Визначте, які класи об’єктів модель визначає найточніше, та у яких виникають складнощі при розпізнавані.
6. Використайте претреновану модель YoloV8 для детекції довільних об’єктів на фото або відео.

Частина 1					**Частина 2**		
Варіант	**Тема***	**Перша Модель ResNet*****	**Друга Модель ResNet*****	**Кіль-кість епох (fine_tune)****	**Анотації в Label Studio***	**Кількість епох (YoloV5)****	**Розмірність пакету (batch_size)****
1	Автомобілі	18	34	3	Автомобілі, дорожні знаки	5	16
2	Дерева	18	50	5	Дерева, квіти	7	32
3	Тварини	18	34	4	Тварини, птахи	9	8
4	Фрукти	18	50	6	Фрукти, овочі	5	16
5	Міські пейзажі	18	34	3	Будівлі, автомобілі	7	32
6	Пляжі	18	50	4	Море, пісок	9	16
7	Комп’ютерне обладнання	18	34	5	Комп’ютери, монітори	5	8
8	Спорт	18	50	6	Спортсмени, спортивне обладнання	7	32
9	Рослини	18	34	3	Квіти, дерева	9	16

10	Кава	18	50	5	Кавові чашки, кавові зерна	5	8
11	Годинники	18	34	4	Годинники, браслети	7	32
12	Взуття	18	50	6	Взуття, шкарпетки	9	16
13	Техніка	18	34	3	Смартфони, навушники	5	8
14	Велосипеди	18	50	5	Велосипеди, шоломи	7	32
15	Парки	18	34	3	Дерева, лавки	9	16
16	Живопис	18	50	4	Картини, скульптури	5	8
17	Страви	18	34	6	Страви, напої	7	32
18	Космос	18	50	3	Зірки, планети	9	16
19	Іграшки	18	34	5	Іграшки, ігрові консолі	5	8
20	Книги	18	50	4	Книги, бібліотеки	7	32
21	Гори	18	34	5	Гори, річки	9	16
22	Техніка	18	50	3	Клавіатури, приставки	5	8
23	Офіс	18	34	6	Офісне обладнання, робочі місця	7	32
24	Зоопарк	18	50	4	Тварини в зоопарку, клітки	9	16
25	Сади	18	34	5	Сади, парники	5	8
26	Мода	18	50	3	Модельєри, подіум	7	32
27	Скульптури	18	34	6	Скульптури, музеї	9	16
28	Музичні інструменти	18	50	4	Гітари, піаніно	5	8
29	Кулінарія	18	34	5	Страви, кухонне приладдя	7	32
30	Подорожі	18	50	3	Ландшафти, транспорт	9	16

* Студент може обрати свою тему самостійно, якщо вона не буде збігатися з темами інших студентів.
** Студент може змінювати параметри на більші, якщо бажає та має можливість задля досягнення кращих результатів навчання моделі.
*** Студент може додатково виконати випробування з моделями ResNet101, ResNet152.

Контрольне запитання №1

Вкажіть показник accuracy для останньої епохи навчання першої ResNet.
Відповідь: _____

Контрольне запитання №2

Вкажіть показник accuracy для останньої епохи навчання другої ResNet.

Відповідь: _____

Контрольне запитання №3

Вкажіть кількість проанотованих зображень.

Відповідь: _____

Контрольне запитання №4

Як впливає кількість епох на процес навчання моделі, і які потенційні ризики пов’язані з великою кількістю епох?

Відповідь: _____

_____

_____

Welcome to Google Colab

https://colab.research.google.com/drive/1Yykj2DxeQy3HlbZIpAIheRSSmXFnwuWB?usp=sharing

Головне про Label Studio: **Label Studio** – це інструмент для маркування даних з відкритим вихідним кодом та дозволяє інтегрувати Label Studio з моделями машинного навчання, щоб надавати прогнози для міток або виконувати безперервне активне навчання.

Інсталяція Label Studio на Windows з pip

Примітка. У вас попередньо повинен бути встановлений Python версії 3.8 або вище.

Щоб встановити Label Studio з pip і віртуальним середовищем, виконайте наступні дії:

```
python3 -m venv env
source env/bin/activate
python -m pip install label-studio
```

Щоб встановити Label Studio з pip, виконайте наступні дії:

```
pip install label-studio
```

Після встановлення Label Studio запустіть сервер наступною командою:

```
label-studio
```

Веб-браузер за замовчуванням автоматично відкриє сторінку http://localhost:8080 з Label Studio.

За додатковою інформацією щодо Label Studio звертайтеся до документації https://labelstud.io/guide/install

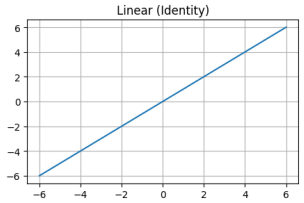
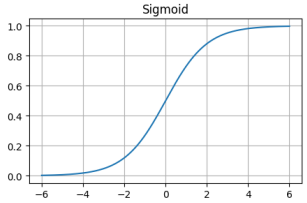
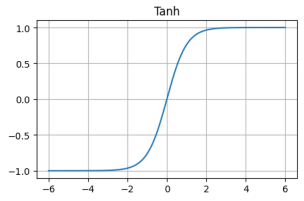
Лабораторна робота № 4. Оптимізація нейронних мереж: дослідження функцій активації та стратегій оптимізації

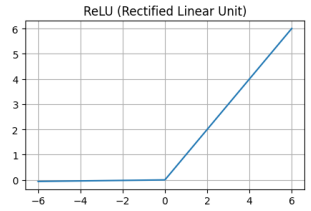
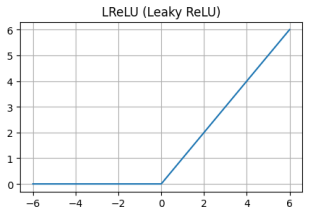
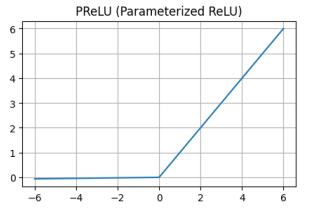
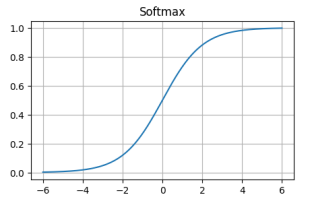
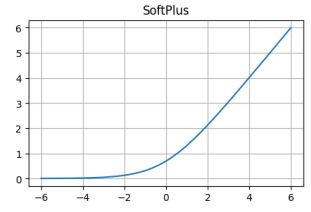
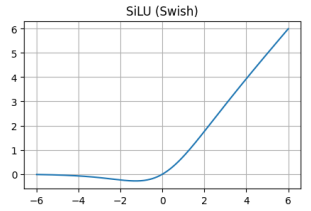
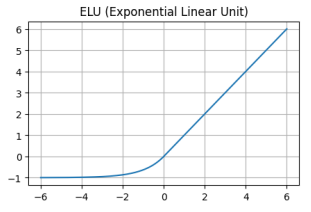
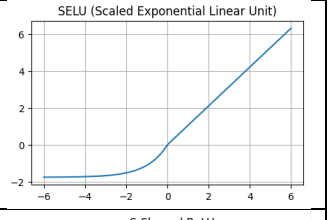
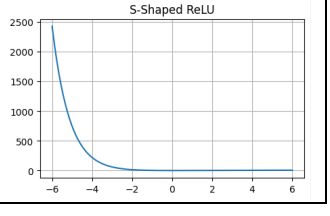
Мета: сформулювати практичні навички у виборі та застосуванні різноманітних функцій активації та оптимізаторів для ефективного навчання нейронних мереж. Лабораторна робота включає в себе аналітику та порівняння впливів різних комбінацій функцій активації та оптимізаторів на процес навчання та загальну продуктивність моделі на різних датасетах.

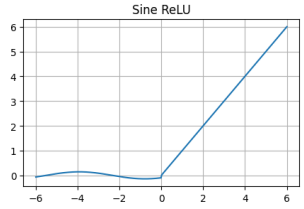
Теоретичні відомості

Оптимізація нейронних мереж є ключовим елементом у глибокому навчанні, що дозволяє моделям ефективно навчатися, адаптуючись до складних даних та вирішуючи різноманітні задачі. Центральними концепціями в оптимізації є функції активації та алгоритми оптимізації.

Функції активації відіграють критичну роль у нейронних мережах, додаючи нелінійність до моделі, що дозволяє їй виявляти складніші патерни в даних. Вони визначають, як нейрон буде активований, залежно від суми вхідних сигналів. Популярні функції активації включають ReLU, Sigmoid, Tanh, Leaky ReLU, та інші, кожна з яких має свої переваги та сценарії застосування.

№	Функція активації	Опис	Використання	
1	Linear (або Identity)	Повертає вхід без змін. $f(x) = x$	Застосовується в випадках, коли потрібно зберегти розподіл вхідних даних, наприклад, у вихідному шарі для задач регресії.	
2	Sigmoid	Приводить вихідні значення до діапазону між 0 та 1. $\frac{1}{1+e^{-x}}$	Часто використовується в вихідному шарі для бінарної класифікації.	
3	Tanh (гіперболічний тангенс)	Схожа на сигмоїд, але виводить значення між -1 та 1. $f(x) = \tanh(x)$	Добре підходить для прихованих шарів завдяки своїй центрованості близько нуля.	

4	ReLU (Rectified Linear Unit)	Виводить вхід, якщо він позитивний; інакше повертає нуль. $f(x) = \max(0, x)$	Найпопулярніша функція для прихованих шарів у глибоких нейронних мережах завдяки простоті та ефективності.	
5	LReLU (Leaky ReLU)	Схожа на ReLU, але пропускає маленьке значення для негативних вхідних даних. $f(x) = \max(0.01x, x)$	Допомагає уникнути проблеми «мертвих нейронів», яка може виникати при використанні ReLU.	
6	PReLU (Parameterized ReLU)	Загальний випадок Leaky ReLU, де коефіцієнт негативних значень є параметром, який може навчатися.	Дозволяє мережі самостійно визначати оптимальний коефіцієнт для негативних значень.	
7	Softmax	Перетворює вектор чисел на вектор ймовірностей, що сумуються в 1.	Застосовується у вихідному шарі для багатокласової класифікації.	
8	SoftPlus	Гладка версія ReLU. $f(x) = \ln(1 + e^x)$	Може використовуватися як альтернатива ReLU, надаючи гладку апроксимацію.	
9	SiLU (Swish)	Самоштовхуюча функція активації. $f(x) = x \cdot \sigma(x)$	Демонструє хороші результати в деяких задачах, може використовуватися як альтернатива ReLU.	
10	ELU (Exponential Linear Unit)	Подібна до ReLU, але з експоненціальним нахилом для негативних вхідних даних.	Допомагає зменшити зсув активацій, може покращити швидкість навчання.	
11	SELU (Scaled Exponential Linear Unit)	Самонормалізуюча версія ELU.	Забезпечує самонормалізацію вхідних даних, ефективна для глибоких мереж.	
12	S-Shaped ReLU	Варіація ReLU з S-подібною формою для покращення нелінійності.	Експериментальна функція, яка може покращити навчання в певних задачах.	

13	Sine ReLU	Використовує синусоїду для активації негативних значень замість лінійного або експоненційного нахилу.	Експериментальна функція для введення періодичності в активації, може бути корисною в специфічних задачах.	

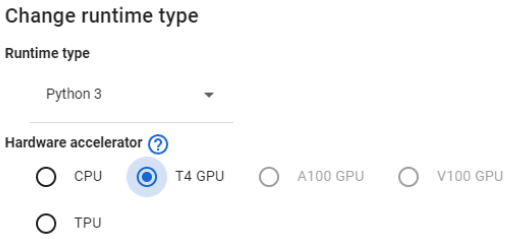
Оптимізатори керують процесом оновлення ваг моделі в процесі навчання, з метою мінімізації функції втрат. Вибір оптимізатора може значно вплинути на швидкість навчання та здатність моделі досягати глобального мінімуму. Оптимізатори, такі як SGD, Adam, RMSprop, та інші, різняться за своїми стратегіями оновлення ваг та адаптивністю до змін у градієнтах.

№	Оптимізатор	Опис	Використання
1	SGD	Класичний метод оптимізації, який оновлює ваги моделі на основі градієнта втрати по відношенню до кожного ваги.	Використовується для навчання різноманітних типів нейромереж завдяки своїй простоті та ефективності на великих датасетах.
2	Adam	Алгоритм оптимізації, який адаптивно коригує швидкість навчання для кожного параметра.	Дуже популярний в глибокому навчанні завдяки здатності швидко збігатися та ефективно працювати на різноманітних задачах.
3	AdamW	Варіант оптимізатора Adam, який включає в себе вагову регуляризацію в процес оновлення параметрів, що дозволяє краще контролювати перенавчання.	Використовується в задачах, де важливо зменшити вплив перенавчання та покращити загальну узагальнювану здатність моделі.
4	Adagrad	Алгоритм, що адаптує швидкість навчання для кожного параметра на основі частоти його оновлення, зменшуючи швидкість для часто оновлюваних параметрів.	Корисний при роботі з рідкісними даними, але може бути менш ефективний для тривалого навчання через необоротне зменшення швидкості навчання.
5	RMSprop	Алгоритм оптимізації, який налаштовує швидкість навчання, використовуючи квадратичний середній корінь останніх градієнтів.	Ефективний на невеликих датасетах і в задачах, де потрібна адаптивність швидкості навчання.
6	Adadelta	Вдосконалення Adagrad, яке намагається уникнути його агресивного, монотонного зменшення швидкості навчання.	Використовується для задач, де необхідно більш стабільне та адаптивне оновлення швидкості навчання.
7	Nadam	Комбінація Adam та Nesterov accelerated gradient (NAG), яка використовує передбачувальний градієнт для покращення збіжності.	Ефективний для широкого спектра задач глибокого навчання, особливо коли потрібна швидка збіжність.

Важливим аспектом ефективної оптимізації є розуміння, як вибір функції активації та оптимізатора впливає на здатність моделі до навчання, її збіжність, та уникнення проблем, таких як перенавчання або «мертві нейрони». Експериментування з різними комбінаціями може допомогти визначити оптимальні параметри для конкретної задачі.

Рекомендації до лабораторної роботи

При виконанні лабораторної роботи в Google Colab використовуйте «T4 GPU». Для цього оберіть «Runtime» → «Change runtime type» → «T4 GPU».



Приклад виконання роботи

Завдання

1. Створіть модель для класифікації зображень з датасету відповідно до вашого варіанту.
2. Застосуйте три функції активації відповідно до вашого варіанту у прихованих шарах вашої моделі та порівняйте їх вплив на точність класифікації та швидкість збіжності.
3. Використайте кожен з трьох оптимізаторів відповідно до вашого варіанту для тренування вашої моделі та проаналізуйте їх ефективність.
4. Зібрані результати тренування та тестування представте у вигляді порівняльної таблиці. Проаналізуйте, як комбінація функції активації та оптимізатора впливає на загальну продуктивність моделі на датасеті. Зробіть висновки про оптимальні стратегії навчання для цього конкретного датасету.

Варіант	Датасет	Функція активації №1	Функція активації №2	Функція активації №3	Опти-мізатор №1	Опти-мізатор №2	Опти-мізатор №3
T	MNIST	SiLU	SoftPlus	PReLU	Nadam	AdamW	SGD

Контрольне запитання №1

Вкажіть показник Test Accuracy для комбінації другої функції активації та другого оптимізатора.
Відповідь: _____

Контрольне запитання №2

Вкажіть показник Test Loss для комбінації третьої функції активації та першого оптимізатора.
Відповідь: _____

Контрольне запитання №3

Вкажіть найбільш ефективну комбінацію функції активації та оптимізатора за показником точності.
Відповідь: _____

Контрольне запитання №4

Як правильно налаштувати трансформації для зображень з 3 кольоровими каналами (RGB) в Pytorch, використовуючи transforms.Compose?
Відповідь: _____

Хід роботи

1. Встановлення необхідних бібліотек та модулів, завантаження та підготовка набору даних MNIST

```
# Завдання №1. Створення моделі для класифікації зображень з MNIST
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import torch.nn.functional as F
import torch.nn as nn
import torch.optim as optim

transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5,), (0.5,))]) # 2

train_data = datasets.MNIST(root='./data', train=True, download=True,
transform=transform)
test_data = datasets.MNIST(root='./data', train=False, download=True,
transform=transform)

train_loader = DataLoader(train_data, batch_size=64, shuffle=True)
test_loader = DataLoader(test_data, batch_size=64, shuffle=False)
```

Опишемо архітектуру нейронної мережі «Net», що містить 3 повністю з'єднанні шари для класифікації зображень розміром 28x28 пікселів у 10 класів.

```
class Net(nn.Module):
    def __init__(self, activation_func=F.relu):
        super(Net, self).__init__()
        self.activation_func = activation_func
        self.fc1 = nn.Linear(28*28, 512) # 1, 2
        self.fc2 = nn.Linear(512, 128)
        self.fc3 = nn.Linear(128, 10) # 3

    def forward(self, x):
        x = x.view(-1, 28*28) # 1, 2
        x = self.fc1(x)
        if self.activation_func is not None:
            x = self.activation_func(x)
        x = self.fc2(x)
        if self.activation_func is not None:
            x = self.activation_func(x, dim=1) if self.activation_func ==
F.softmax else self.activation_func(x)
        x = self.fc3(x)
        return x

class SShapedReLU(nn.Module):
    def __init__(self):
        super(SShapedReLU, self).__init__()

    def forward(self, x):
        return x * (torch.sigmoid(x) - torch.sigmoid(-x))
```

```
class SineReLU(nn.Module):
    def __init__(self, beta=0.1):
        super(SineReLU, self).__init__()
        self.beta = beta

    def forward(self, x):
        return torch.where(x > 0, x, self.beta * (torch.sin(x) -
torch.cos(x)))
```

Опишемо процес тренування та тестування нейронної мережі, обчислюючи втрати та точність на навчальному та тестовому наборах.

```
def train(model, train_loader, optimizer, epoch):
    model.train()
    total_loss = 0
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = model(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

        if batch_idx % 100 == 0:
            print(f'Train Epoch: {epoch} [{batch_idx *
len(data)}/{len(train_loader.dataset)} ({100. * batch_idx /
len(train_loader):.0f}%)]\tLoss: {loss.item():.6f}')

    average_loss = total_loss / len(train_loader)
    return average_loss

def test(model, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            output = model(data)
            test_loss += F.cross_entropy(output, target,
reduction='sum').item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
    test_loss /= len(test_loader.dataset)
    test_accuracy = correct / len(test_loader.dataset)

    print(f'\nTest set: Average loss: {test_loss:.4f}, Accuracy:
{correct}/{len(test_loader.dataset)} ({100. * correct /
len(test_loader.dataset):.0f}%)\n')
    return test_loss, test_accuracy
```

Опишемо доступний список функцій активації та список оптимізаторів.

```
from ipywidgets import interact, Dropdown, widgets

# Список функцій активації
activation_functions = {
    'Linear': None,
    'ReLU': F.relu,
    'Sigmoid': torch.sigmoid,
    'Tanh': torch.tanh,
    'Leaky ReLU': nn.LeakyReLU(),
    'Parameterized ReLU': nn.PReLU(),
    'Softmax': F.softmax,
    'SoftPlus': F.softplus,
    'SiLU (Swish)': F.silu,
    'ELU': F.elu,
    'SELU': F.selu,
    'S-Shaped ReLU': SShapedReLU(),
    'Sine ReLU': SineReLU(beta=0.1)
}

# Список оптимізаторів
optimizers_dict = {
    'SGD': optim.SGD,
    'Adam': optim.Adam,
    'AdamW': optim.AdamW,
    'RMSprop': optim.RMSprop,
    'Adagrad': optim.Adagrad,
    'Adadelta': optim.Adadelta,
    'Nadam': optim.NAdam
}
```

2. Ініціалізуємо та виконаємо експеримент з навчання нейронної мережі, використовуючи функції активації SiLU, SoftPlus, PReLU та оптимізатори Nadam, AdamW, SGD.

```
# Завдання №2. Застосуємо функції активації SiLU, Soft, Plus, PReLU
# Завдання №3. Використаємо оптимізатори Nadam, AdamW, SGD
results = []

def run_experiment(activation='ReLU', optimizer='SGD'):
    optimizer_label = optimizer
    if activation == 'Softmax':
        # Логіка для Softmax
        activation_func = lambda x: F.softmax(x, dim=1)
    else:
        activation_func = activation_functions[activation] if activation !=
'Linear' else lambda x: x
    model = Net(activation_func=activation_func)
    optimizer = optimizers_dict[optimizer](model.parameters(), lr=0.01)

    epoch_losses = []
```

```
    for epoch in range(1, 2):
        train_loss = train(model, train_loader, optimizer, epoch)
        epoch_losses.append(train_loss)

    test_loss, test_accuracy = test(model, test_loader)

    results.append({
        'Activation': activation,
        'Optimizer': optimizer_label,
        'Train Loss': sum(epoch_losses) / len(epoch_losses),
        'Test Loss': test_loss,
        'Test Accuracy': test_accuracy
    })

activation_dropdown =
widgets.Dropdown(options=list(activation_functions.keys()),
description='Activation:')
optimizer_dropdown = widgets.Dropdown(options=list(optimizers_dict.keys()),
description='Optimizer:')

start_button = widgets.Button(description="Start Experiment")

output = widgets.Output()



def on_start_button_clicked(b):
    with output:
        output.clear_output()
        run_experiment(activation=activation_dropdown.value,
optimizer=optimizer_dropdown.value)

start_button.on_click(on_start_button_clicked)

display(activation_dropdown, optimizer_dropdown, start_button, output)
```

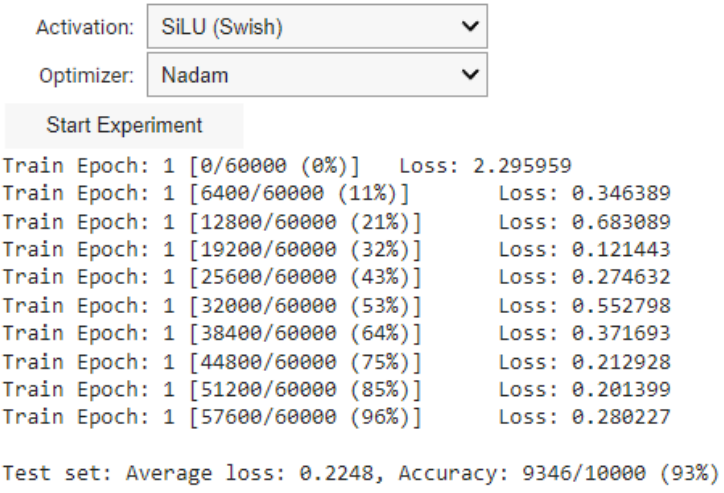


Рисунок 4.1. Результат класифікації з використанням функції активації SiLU та оптимізатора Nadam

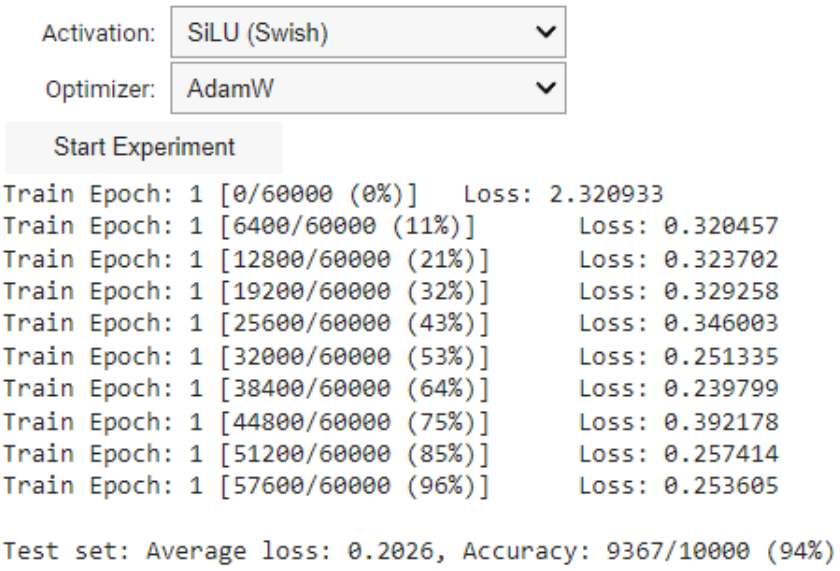


Рисунок 4.2. Результат класифікації з використанням функції активації SiLU та оптимізатора AdamW

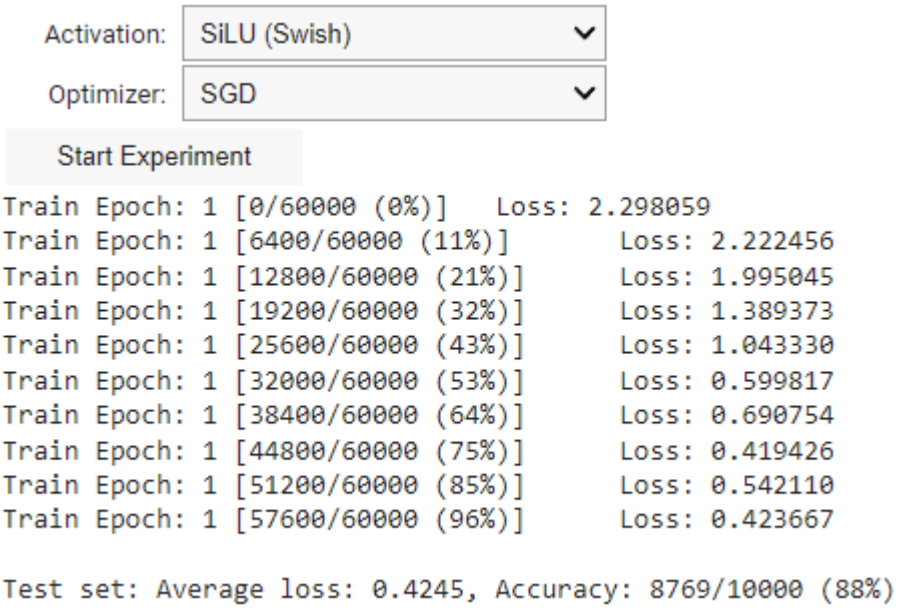


Рисунок 4.3. Результат класифікації з використанням функції активації SiLU та оптимізатора SGD

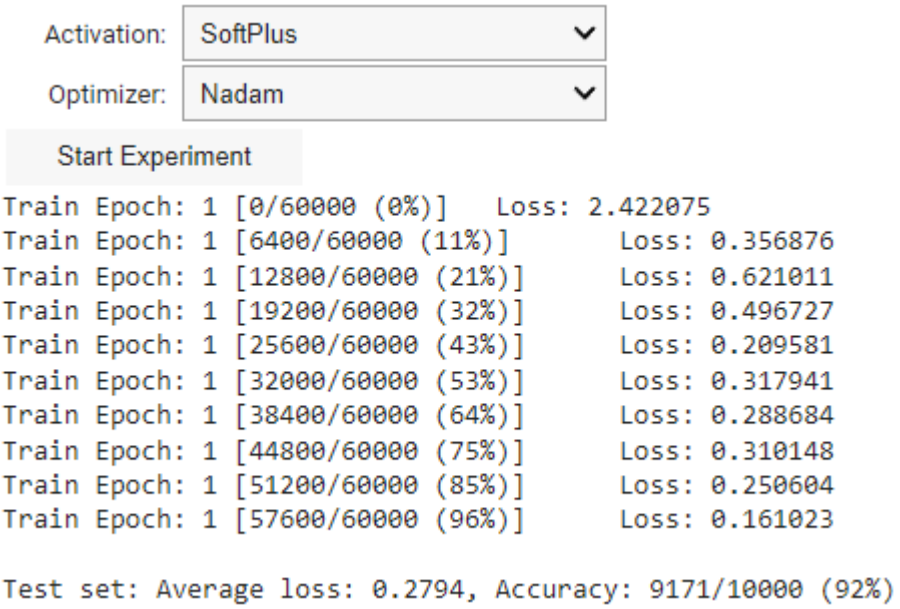


Рисунок 4.4. Результат класифікації з використанням функції активації SoftPlus та оптимізатора Nadam

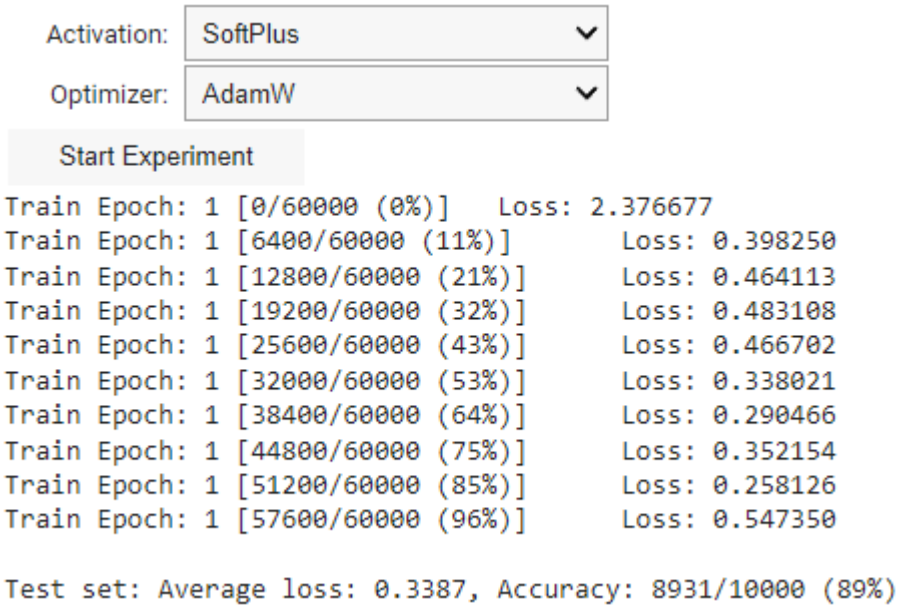


Рисунок 4.5. Результат класифікації з використанням функції активації SoftPlus та оптимізатора AdamW

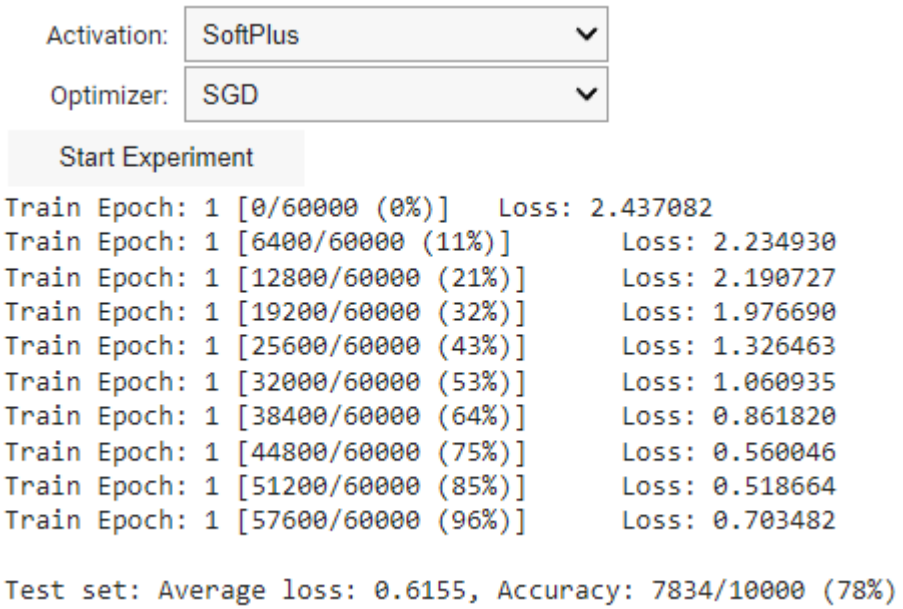


Рисунок 4.6. Результат класифікації з використанням функції активації SoftPlus та оптимізатора SGD

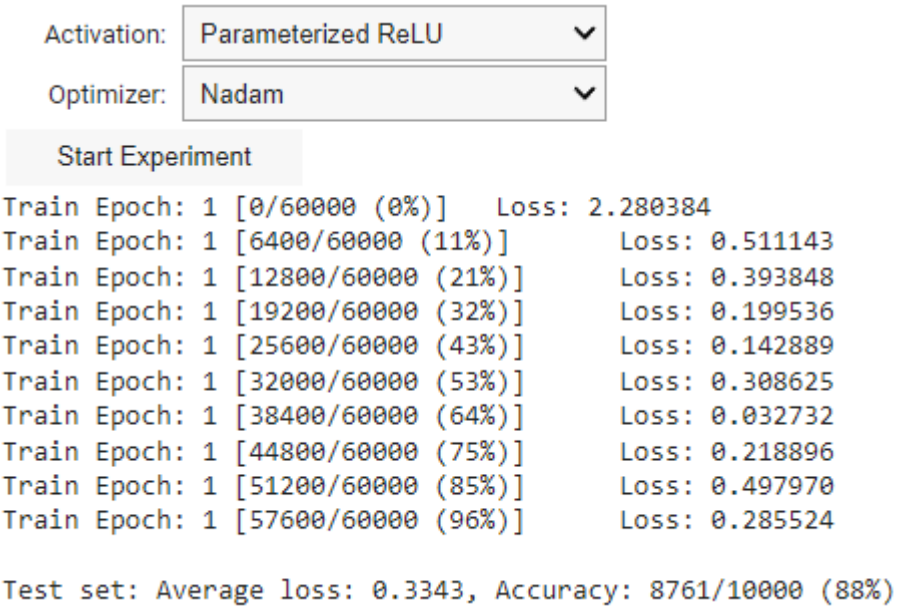


Рисунок 4.7. Результат класифікації з використанням функції активації PReLU та оптимізатора Nadam

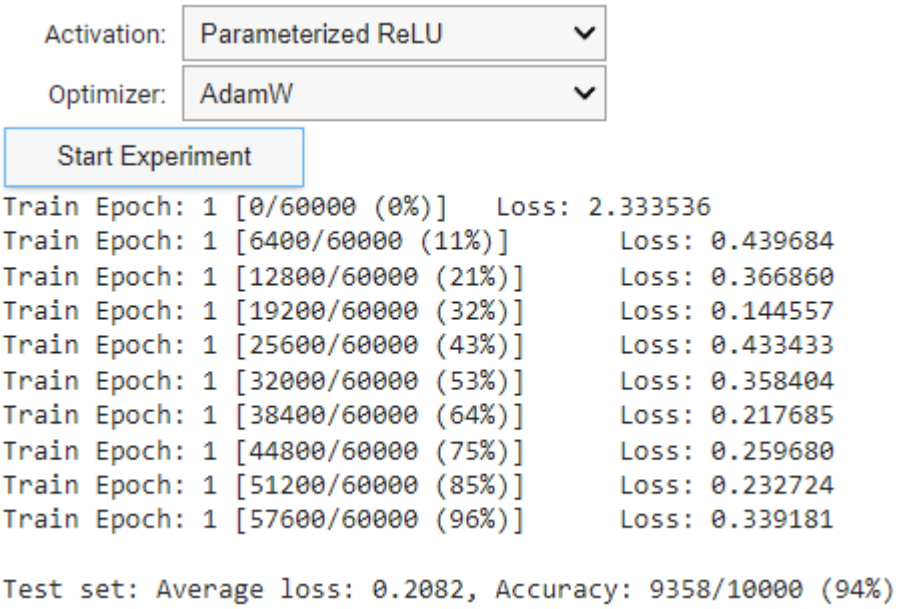


Рисунок 4.8. Результат класифікації з використанням функції активації PReLU та оптимізатора AdamW

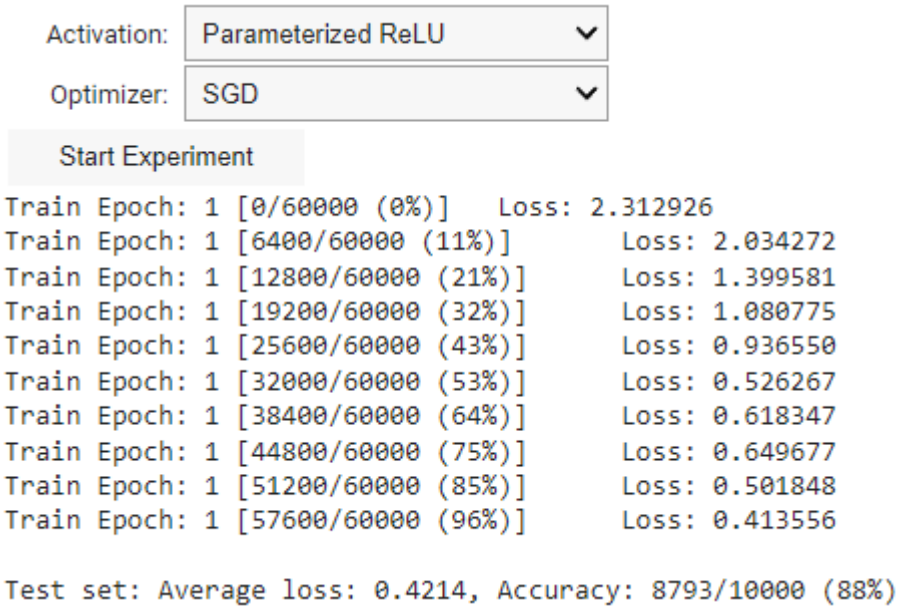


Рисунок 4.9. Результат класифікації з використанням функції активації PReLU та оптимізатора SGD

3. Створимо порівняльну таблицю та проаналізуємо різні комбінації функцій активації та оптимізаторів.

```
# Завдання №4. Створимо порівняльну таблицю результатів
import pandas as pd
results_df = pd.DataFrame(results)

styled_df = results_df.style.set_table_styles(
    [{'selector': 'th', 'props': [('background-color', 'lightblue'),
('color', 'black')]},
     {'selector': 'td', 'props': [('text-align', 'center')]}]
    ).set_properties(**{'background-color': 'lavender', 'color': 'black',
'border-color': 'darkgray'}
                     ).hide_index().format({'Train Loss': '{:.4f}', 'Test
Loss': '{:.4f}', 'Test Accuracy': '{:.2f}%'})

display(styled_df)
```

Activation	Optimizer	Train Loss	Test Loss	Test Accuracy
SiLU (Swish)	Nadam	0.2410	0.1805	95.12%
SiLU (Swish)	AdamW	0.2534	0.1872	95.39%
SiLU (Swish)	SGD	0.4829	0.2699	92.20%
SoftPlus	Nadam	0.2555	0.2135	94.37%
SoftPlus	AdamW	0.3794	0.2879	91.75%
SoftPlus	SGD	0.5491	0.3865	88.67%
Parameterized ReLU	Nadam	3.4909	0.3114	91.98%
Parameterized ReLU	AdamW	0.2871	0.2104	94.59%
Parameterized ReLU	SGD	0.4239	0.2033	93.82%

Рисунок 4.10. Таблиця результатів тестування з використанням різних функцій активації та оптимізації

Побудуємо графік для візуалізації отриманих результатів

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.barplot(x='Optimizer', y='Test Accuracy', hue='Activation',
data=results_df)
plt.title('Test Accuracy for Different Activation Functions and Optimizers')
plt.show()
```

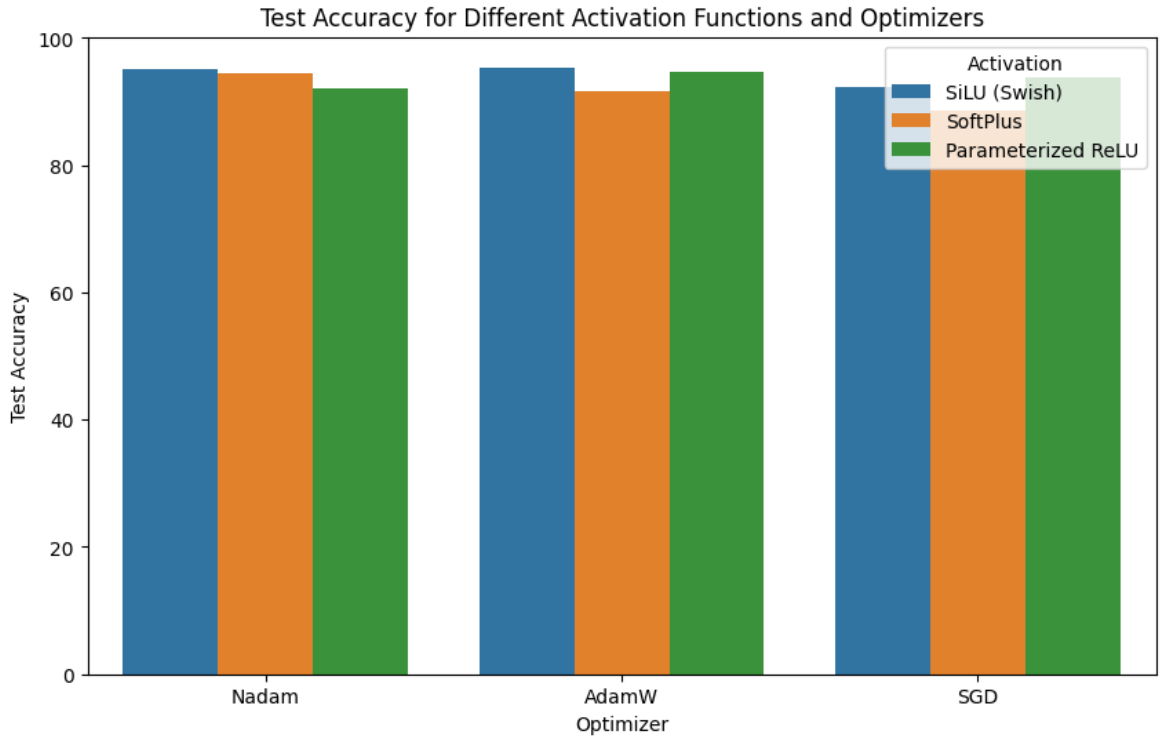


Рисунок 4.11. Графік точності на тестовому наборі за умови використання різних функцій активації та оптимізаторів

Таким чином, аналізуючи отримані дані, можна зазначити, що оптимізатори Nadam і AdamW забезпечують кращі результати для всіх трьох функцій активації порівняно з SGD, що відображається в менших втратах та вищій точності, що свідчить про їх більшу ефективність у оптимізації ваг моделі для даного набору даних.

Комбінація SiLU з AdamW демонструє найкращі результати, а саме Test Loss (0.1872), Test Accuracy (95.39%), що робить її найефективнішою для даної задачі.

Комбінація PReLU з AdamW також продемонструвала високу ефективність, маючи схожі показники Test Loss (0.2104), Test Accuracy (94.59%).

Комбінація SoftPlus з SGD продемонструвала найвищий показник Test Loss (0.3865) та найнижчий результат Test Accuracy (88.67%), що вказує на недостатню адаптацію цієї комбінації до умов даної задачі.

Контрольне запитання №1

Вкажіть показник Test Accuracy для комбінації другої функції активації та другого оптимізатора.
Відповідь: **91.75%**

Контрольне запитання №2

Вкажіть показник Test Loss для комбінації третьої функції активації та першого оптимізатора.
Відповідь: **0.3114**

Контрольне запитання №3

Вкажіть найбільш ефективну комбінацію функції активації та оптимізатора за показником точності.
Відповідь: **SiLU (Swish) та AdamW**

Контрольне запитання №4

Як правильно налаштувати трансформації для зображень з 3 кольоровими каналами (RGB) в Pytorch, використовуючи transforms.Compose?
Відповідь: **…**

Висновки: …

Завдання для виконання лабораторної роботи

1. Створіть модель для класифікації зображень з датасету відповідно до вашого варіанту.
2. Застосуйте три функції активації відповідно до вашого варіанту у прихованих шарах вашої моделі та порівняйте їх вплив на точність класифікації та швидкість збіжності.
3. Використайте кожен з трьох оптимізаторів відповідно до вашого варіанту для тренування вашої моделі та проаналізуйте їх ефективність.
4. Зібрані результати тренування та тестування представте у вигляді порівняльної таблиці. Проаналізуйте, як комбінація функції активації та оптимізатора впливає на загальну продуктивність моделі на датасеті. Зробіть висновки про оптимальні стратегії навчання для цього конкретного датасету.

Варіант	Датасет	Функція активації №1	Функція активації №2	Функція активації №3	Опти-мізатор №1	Опти-мізатор №2	Опти-мізатор №3
1	MNIST	ReLU	Sigmoid	Tanh	SGD	Adam	RMSprop
2	Fashion MNIST	Leaky ReLU	ELU	ReLU	Adam	Adagrad	Adadelta
3	MNIST	ELU	Softmax	Sigmoid	RMSprop	Adam	SGD
4	MNIST	Tanh	Softmax	PReLU	Adadelta	Adagrad	AdamW
5	Fashion MNIST	Sigmoid	ReLU	ELU	Adagrad	SGD	RMSprop
6	CIFAR-10	Softmax	Tanh	Sigmoid	AdamW	Adadelta	Adagrad
7	MNIST	ELU	Softplus	ReLU	RMSprop	Adadelta	SGD
8	Fashion MNIST	ReLU	Sigmoid	Tanh	Adagrad	Nadam	RMSprop
9	CIFAR-10	Tanh	SELU	Leaky ReLU	SGD	RMSprop	Adadelta
10	MNIST	Softmax	ELU	Sigmoid	Adadelta	Adagrad	AdamW
11	Fashion MNIST	Sine ReLU	ELU	Sigmoid	Adam	SGD	Adagrad
12	CIFAR-10	Softmax	Tanh	Leaky ReLU	RMSprop	Adadelta	Adam
13	MNIST	ELU	Sigmoid	Softmax	Adadelta	Adagrad	SGD
14	Fashion MNIST	Tanh	S-Shaped ReLU	ELU	SGD	RMSprop	Adam
15	CIFAR-10	Softmax	Sigmoid	Linear	Adagrad	AdamW	RMSprop
16	MNIST	ELU	Softmax	Tanh	Adadelta	SGD	Adagrad
17	Fashion MNIST	Sigmoid	SiLU	ReLU	RMSprop	Adagrad	Nadam
18	CIFAR-10	Leaky ReLU	Softmax	Linear	AdamW	Adadelta	SGD
19	MNIST	Tanh	Sigmoid	ELU	Adagrad	RMSprop	Adadelta
20	Fashion	ReLU	Tanh	Softmax	SGD	Nadam	RMSprop

	MNIST						
21	CIFAR-10	Softmax	PReLU	Leaky ReLU	Adadelta	RMSprop	Adagrad
22	MNIST	ELU	Tanh	Sigmoid	RMSprop	SGD	Adam
23	Fashion MNIST	Leaky ReLU	RReLU	Softmax	AdamW	Adagrad	SGD
24	CIFAR-10	Sigmoid	Softmax	PReLU	Adadelta	RMSprop	Adagrad
25	MNIST	Linear	Leaky ReLU	Softmax	SGD	Adagrad	RMSprop
26	Fashion MNIST	Softmax	Sigmoid	ReLU	Adam	SGD	Adadelta
27	CIFAR-10	ReLU	Softplus	Tanh	RMSprop	Adagrad	Nadam
28	MNIST	ELU	Softplus	Sigmoid	Adadelta	AdamW	SGD
29	Fashion MNIST	Sigmoid	Softmax	ELU	Adagrad	RMSprop	Adam
30	CIFAR-10	Tanh	ELU	Leaky ReLU	SGD	Adadelta	RMSprop

Контрольне запитання №1

Вкажіть показник Test Accuracy для комбінації другої функції активації та другого оптимізатора.

Відповідь: _____

Контрольне запитання №2

Вкажіть показник Test Loss для комбінації третьої функції активації та першого оптимізатора.

Відповідь: _____

Контрольне запитання №3

Вкажіть найбільш ефективну комбінацію функції активації та оптимізатора за показником точності.

Відповідь: _____

Контрольне запитання №4

Як правильно налаштувати трансформації для зображень з 3 кольоровими каналами (RGB) в Pytorch, використовуючи transforms.Compose?

Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1KbxoeoObNgr-e00QkB6UrRLNxZYTzLrY?usp=sharing

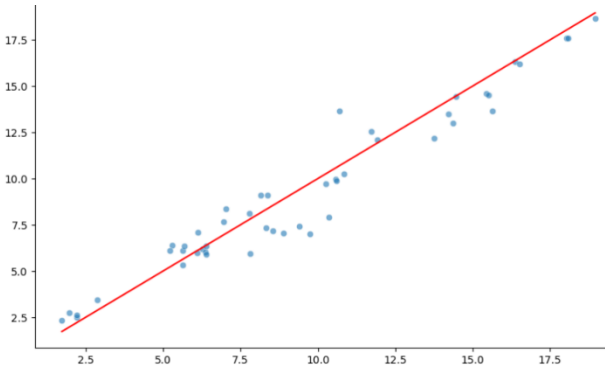
Головне про PyTorch: **PyTorch** – це відкрита бібліотека машинного навчання, яка широко використовується для наукових досліджень та розробки в галузі штучного інтелекту. PyTorch надає потужні інструменти для створення та тренування нейронних мереж, дозволяючи легко працювати з глибоким навчанням. Особливість PyTorch полягає в його динамічних обчислювальних графах, що робить експериментування та ітерації більш гнучкими та інтуїтивно зрозумілими. Бібліотека підтримує роботу з GPU для прискорення обчислень, має велику кількість попередньо навчених моделей через torchvision та інтегрується з багатьма іншими інструментами для наукових обчислень та візуалізації даних. PyTorch Datasets – це компонент PyTorch, який забезпечує зручний доступ до стандартних наборів даних, що дозволяє легко завантажувати та препроцесингувати дані для тренування моделей, забезпечуючи широкий спектр наборів даних для різних задач машинного навчання та комп'ютерного зору.

За додатковою інформацією щодо PyTorch звертайтеся до документації https://pytorch.org/vision/stable/datasets.html

Лабораторна робота № 5. Дослідження методів регресії: Linear Regression, Polynomial Regression, Ridge Regression та Lasso Regression

Мета: сформулювати практичні навички в застосуванні різних типів регресійних моделей для аналізу та передбачення даних. Здобувачі навчаться підбирати оптимальний тип регресійної моделі в залежності від специфіки даних та завдання, оцінювати ефективність моделей за допомогою стандартних метрик, а також розуміти та застосовувати методи регуляризації задля покращення загальної якості моделі.

Теоретичні відомості



Лінійна регресія є методом навчання з учителем, який використовується, коли цільова або залежна змінна є **неперервним дійсним числом**. Вона встановлює зв'язок між залежною змінною y та однією або декількома незалежними змінними x за допомогою «лінії найкращої відповідності». Цей метод працює на принципі методу найменших квадратів (Ordinary Least Squares, OLS) або середньоквадратичної помилки (Mean Square Error, MSE). У статистиці OLS є методом оцінки невідомих параметрів лінійної регресійної функції, його метою є мінімізація суми квадратів різниць між спостережуваними залежними змінними у наданому наборі даних та тими, що передбачені лінійною регресійною функцією.

Основною метою регресійного аналізу є визначення залежності цільової (залежної) змінної від однієї чи декількох предикторних (незалежних) змінних та прогнозування цільової змінної на основі відомих значень незалежних змінних.

Linear Regression (Лінійна регресія) виражає залежність через лінійну функцію: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \epsilon$, де y – цільова змінна, $x_1, \ldots, x_n$ – предиктори, $\beta_0, \ldots, \beta_n$ – коефіцієнти моделі, а ϵ – випадкова похибка.

Polynomial Regression (Поліноміальна регресія) розширює лінійну модель, дозволяючи предикторам впливати на цільову змінну нелінійно: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon$. Це дозволяє моделювати більш складні залежності.

Ridge та **Lasso Regression** (Регресія Ріджа та Лассо) використовуються для регуляризації моделі, що допомагає запобігати перенавчанню, шляхом додавання штрафу до коефіцієнтів моделі.

Ridge Regression мінімізує суму квадратів коефіцієнтів (L2-регуляризація) $L_{Ridge}(\theta) = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{m}\theta_j x_{ij})^2 + \alpha\sum_{j=1}^{m}\theta_j^2$, де $L_{Ridge}(\theta)$ – функція втрат для Ridge Regression, y_i – фактичне значення цільової змінної для *i*-го спостереження, x_{ij} значення *j*-ї ознаки для *i*-го спостереження, θ_j – коефіцієнт моделі для *j*-ознак, n – кількість спостережень, m – кількість ознак, α – параметр регуляризації.

Lasso Regression сприяє створенню розріджених моделей, додаванням штрафу у вигляді абсолютних значень коефіцієнтів (L1-регуляризація) $L_{Lasso}(\theta) = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{m}\theta_j x_{ij})^2 + \alpha\sum_{j=1}^{m}|\theta_j|$, де $L_{Lasso}(\theta)$ – функція втрат для Lasso Regression. Lasso Regression може призводити до того, що деякі коефіцієнти стануть рівними нулю.

Регуляризація допомагає уникнути перенавчання моделі, шляхом запровадження штрафу за великі значення коефіцієнтів. **Ridge Regression** зменшує розмір коефіцієнтів, але зазвичай не зводить їх до нуля. Натомість, **Lasso Regression** може звести деякі коефіцієнти до нуля, що дозволяє виконувати відбір ознак в ході тренування моделі.

Приклад виконання роботи

Завдання

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте цільову змінну та ознаки, які будуть використовуватися для регресії.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується. Створіть графіки для розуміння розподілу ознак та їх взаємозв'язку з цільовою змінною.
4. Виконайте тренування моделей: 1) лінійної регресії; 2) поліноміальної регресії; 3) Ridge регресії; 4) Lasso регресії. Виконайте оцінку та візуалізуйте результати за наступними метриками: MSE, R^2 для кожної моделі. Побудуйте графіки фактичних та передбачених значень для кожної моделі.
5. Виконайте прогнозування за допомогою різних моделей для довільних вхідних даних відповідно до вашої цільової змінної.
6. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
7. За бажанням проекспериментуйте з гіперпараметрами Degree та Alpha для моделей задля покращення результатів.

Контрольне запитання №1

Вкажіть результуюче значення MSE для Polynomial Regression.
Відповідь: _____

Контрольне запитання №2

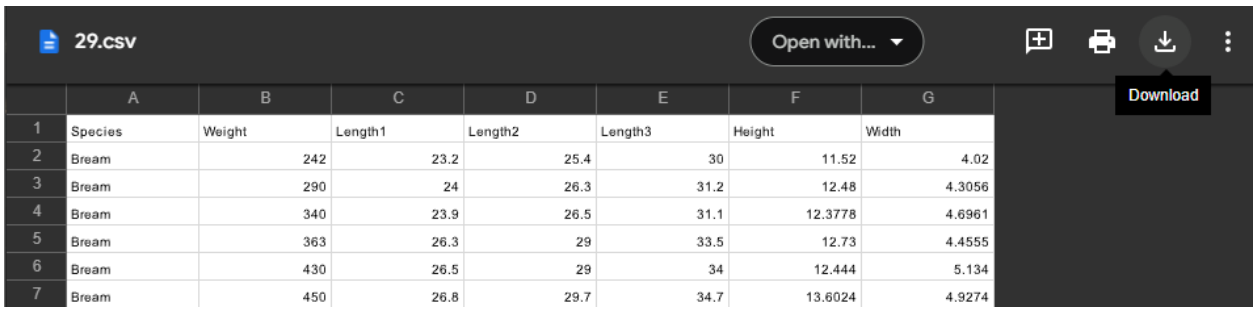
Вкажіть результуюче значення R^2 для Lasso Regression.
Відповідь: _____

Контрольне запитання №3

Вкажіть найкраще значення показника R^2 серед усіх моделей.
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набір даних за варіантом.



	A	B	C	D	E	F	G
1	Species	Weight	Length1	Length2	Length3	Height	Width
2	Bream	242	23.2	25.4	30	11.52	4.02
3	Bream	290	24	26.3	31.2	12.48	4.3056
4	Bream	340	23.9	26.5	31.1	12.3778	4.6961
5	Bream	363	26.3	29	33.5	12.73	4.4555
6	Bream	430	26.5	29	34	12.444	5.134
7	Bream	450	26.8	29.7	34.7	13.6024	4.9274

Рисунок 5.1. Датасет на Google Drive

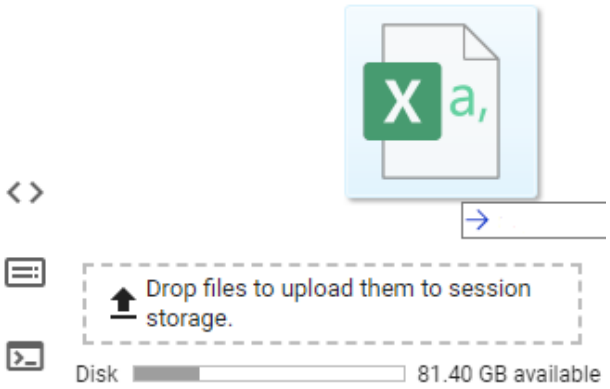


Рисунок 5.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.pipeline import make_pipeline
```

3. Ініціалізація набору даних в Google Colab.

```
# Завдання №1. # Завантаження датасету
data = pd.read_csv("/content/29.csv")
data.head()
```

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

Рисунок 5.3. Перші 5 рядків набору даних

4. Визначимо цільову змінну, як Height, ознаками будуть: Weight, Length1, Length2, Length3, Width.

5. Виконаємо очищення даних

```
# Перевірка на відсутні значення
print(data.isnull().sum())
```

```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

Рисунок 5.4. Пошук нульових значень у датасеті

Порожні значення відсутні, тож видалення або заповнення виконувати не треба.

```
# Видалення або заповнення відсутніх значень
# data.dropna(inplace=True) # Видалення

# data.fillna(data.mean(), inplace=True) # Заповнення середніми значеннями
```

Виконаємо візуалізацію однієї з ознак набору даних, а саме «Length2»

```
# Візуалізація викидів
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Length2'])
plt.title('Boxplot before removing outliers')
plt.show()
```

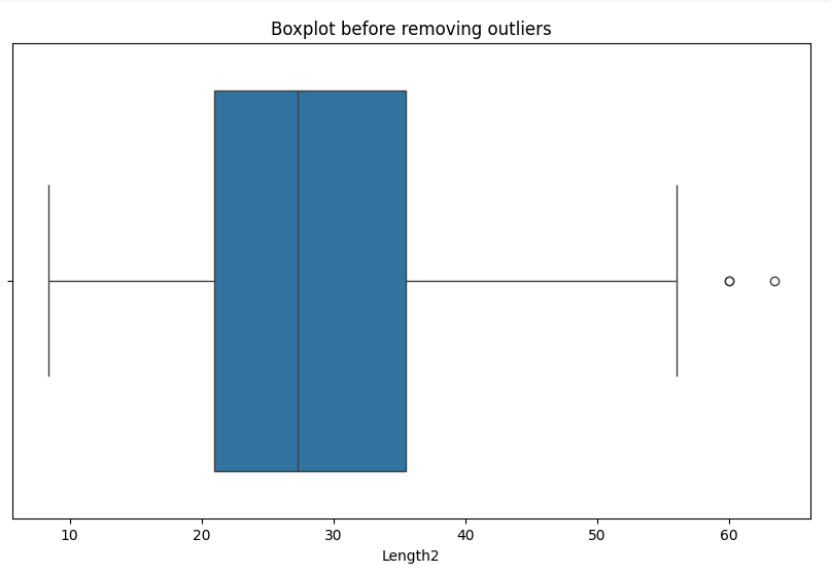


Рисунок 5.5. Діаграма викидів для Length2 перед очищенням

Виконаємо очищення викидів для «Length2».

```
# Видалення викидів
feature_processing = 'Length2'
q_low = data[feature_processing].quantile(0.01)
q_hi  = data[feature_processing].quantile(0.99)
data = data[(data[feature_processing] < q_hi) & (data[feature_processing] >
q_low)]
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Length2'])
plt.title('Boxplot after removing outliers')
plt.show()
```

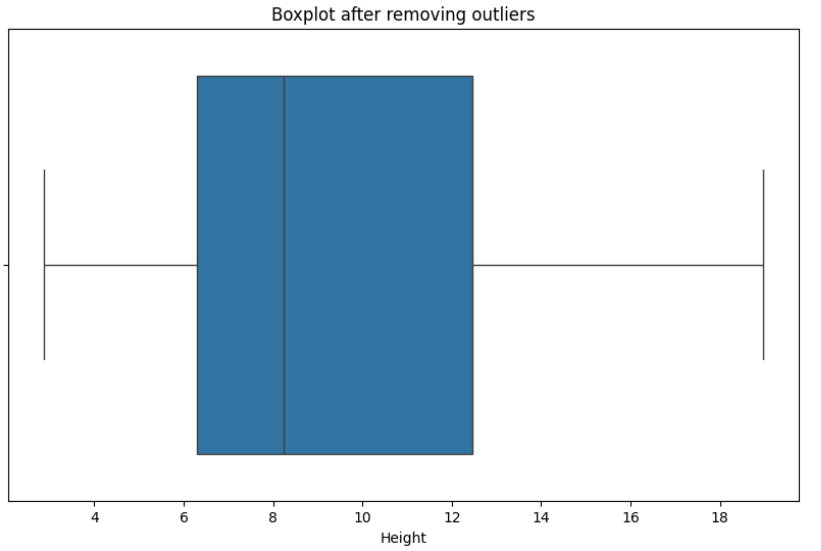


Рисунок 5.6. Діаграма викидів для Length2 після очищення

Подібні операції виконаємо для інших ознак.

6. Розподілимо дані на тренувальні та тестові набори з урахуванням ознак та цільової змінної

```
# Завдання №2. Застосуємо функції активації SiLU, Soft, Plus, PReLU
# Розділення даних на тренувальні та тестові набори
features = ['Weight', 'Length1', 'Length2', 'Length3', 'Width'] # ознаки
X = data[features]
y = data['Height'] # цільові змінні

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

7. Візуалізуємо взаємозв’язки між ознаками та цільовою змінною

```
# Візуалізація взаємозв'язку між ознаками та цільовою змінною
sns.pairplot(data, x_vars=features, y_vars='Height', kind='reg')
plt.show()
```

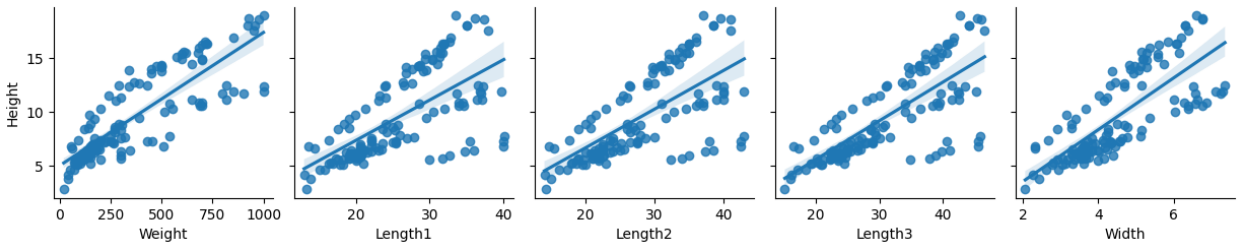


Рисунок 5.7. Візуалізація взаємозв’язку між ознаками та цільовою змінною

```
# Візуалізація кореляції між ознаками
sns.heatmap(data[features + ['Height']].corr(), annot=True, fmt=".2f")
plt.show()
```

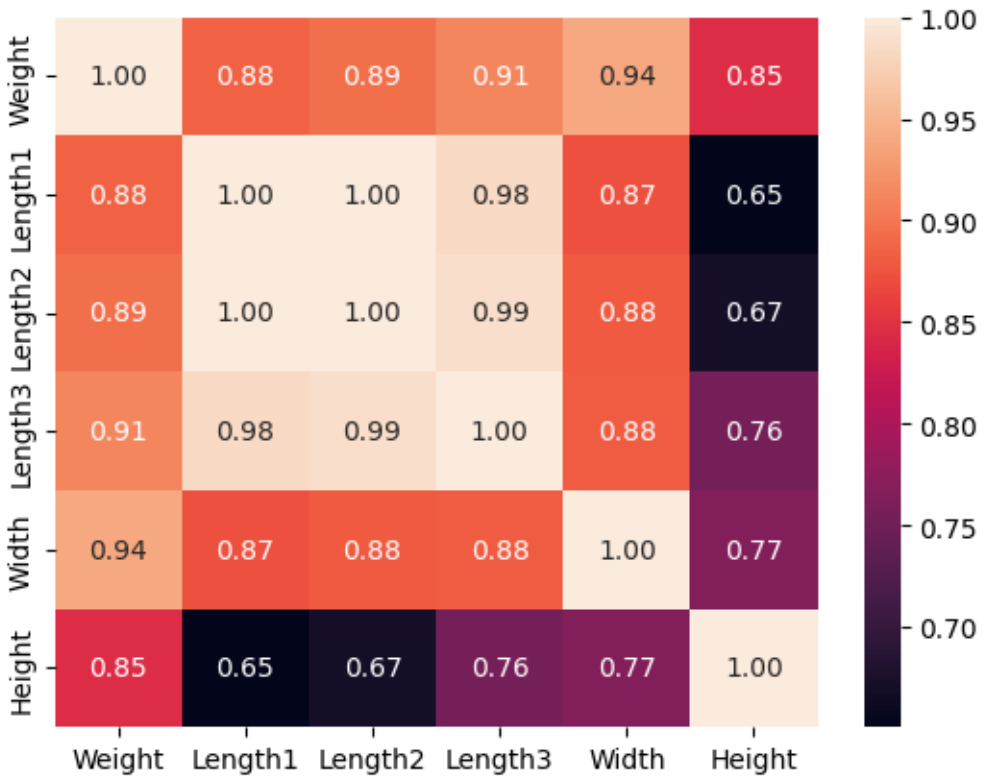


Рисунок 5.8. Візуалізація кореляції між ознаками

8. Виконаємо тренування моделей Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression та їх оцінку за показниками MSE та R^2.

```
# Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Polynomial Regression
polynomial_model = make_pipeline(PolynomialFeatures(degree=2),
LinearRegression())
polynomial_model.fit(X_train, y_train)

# Ridge Regression
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)

# Lasso Regression
lasso_model = Lasso(alpha=0.1)
lasso_model.fit(X_train, y_train)
```

9. Виконаємо оцінювання моделей за показниками MSE та R^2

```
# Оцінювання моделей
models = [linear_model, polynomial_model, ridge_model, lasso_model]
model_names = ['Linear Regression', 'Polynomial Regression', 'Ridge
Regression', 'Lasso Regression']

for model, name in zip(models, model_names):
```

```
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"{name}: MSE = {mse:.2f}, R^2 = {r2:.2f}")
```

```
Linear Regression: MSE = 1.15, R^2 = 0.93
Polynomial Regression: MSE = 0.92, R^2 = 0.94
Ridge Regression: MSE = 1.11, R^2 = 0.93
Lasso Regression: MSE = 1.19, R^2 = 0.92
```

Рисунок 5.9. Оцінка моделей регресії за показниками MSE та R^2

10. Виконаємо візуалізацію графіків моделей регресії.

```
import ipywidgets as widgets
from IPython.display import display

def plot_model_results(model_name):
    if model_name == 'Linear Regression':
        model = linear_model
    elif model_name == 'Polynomial Regression':
        model = polynomial_model
    elif model_name == 'Ridge Regression':
        model = ridge_model
    elif model_name == 'Lasso Regression':
        model = lasso_model
    else:
        print("Модель не вибрана")
        return

    y_pred = model.predict(X_test)

    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
    sns.lineplot(x=y_test, y=y_test, color='red')
    plt.xlabel('Actual Heights')
    plt.ylabel('Predicted Heights')
    plt.title(f'Actual vs Predicted Heights: {model_name}')
    plt.show()
```

```
# Створення випадаючого списку для вибору моделі
model_dropdown = widgets.Dropdown(
    options=['Linear Regression', 'Polynomial Regression', 'Ridge
Regression', 'Lasso Regression'],
    value='Linear Regression',
    description='Model:',
    disabled=False,
)

# Створення віджету для інтерактивного відображення графіків
interactive_plot = widgets.interactive(plot_model_results,
model_name=model_dropdown)
display(interactive_plot)
```

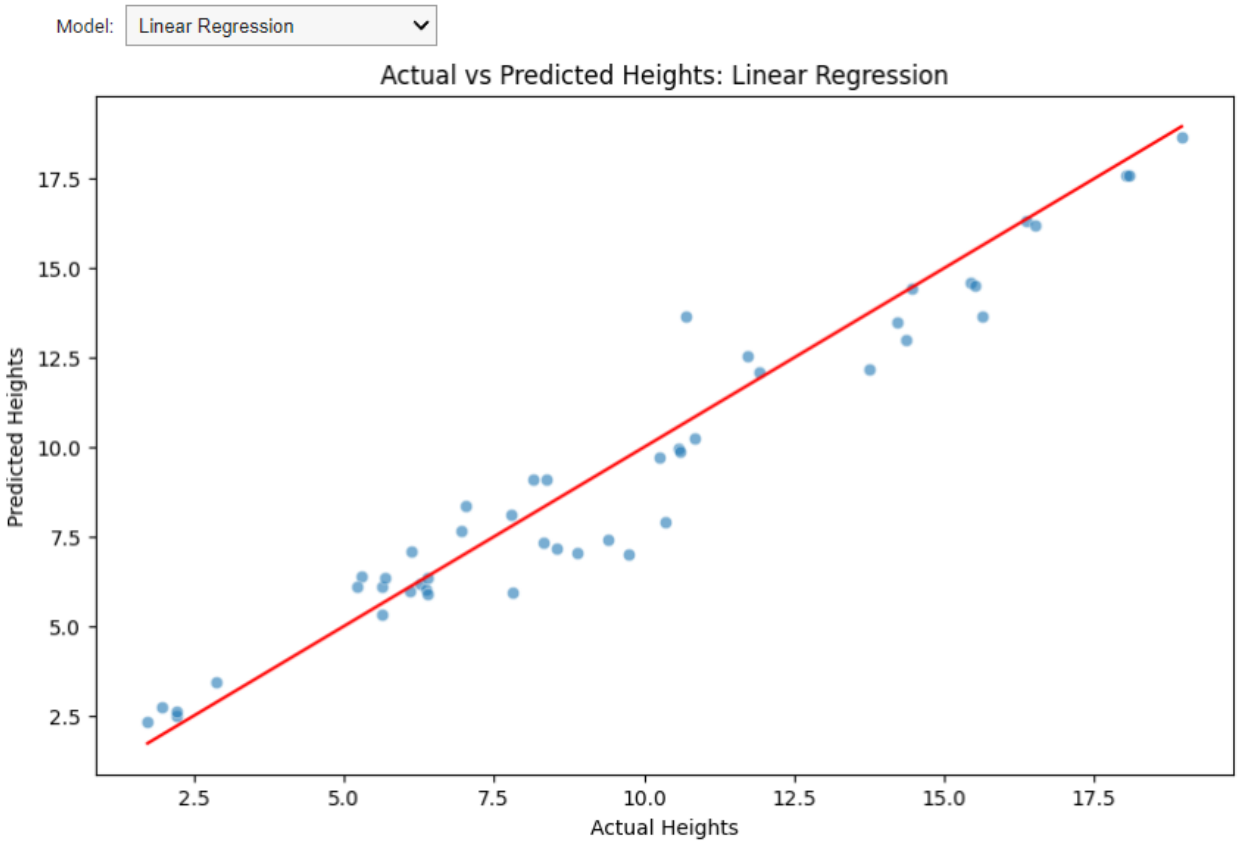


Рисунок 5.10. Графік реальних та прогнозованих значень Height за допомогою моделі Linear Regression

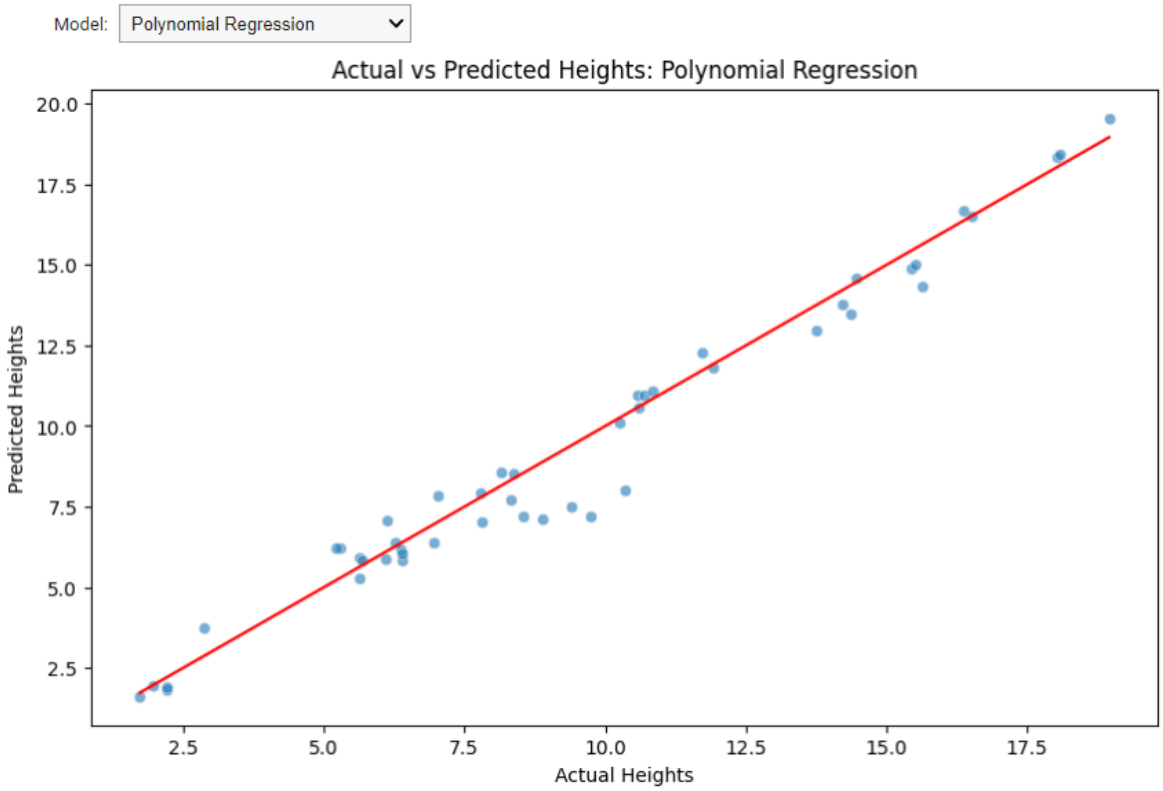


Рисунок 5.11. Графік реальних та прогнозованих значень Height за допомогою моделі Polynomial Regression

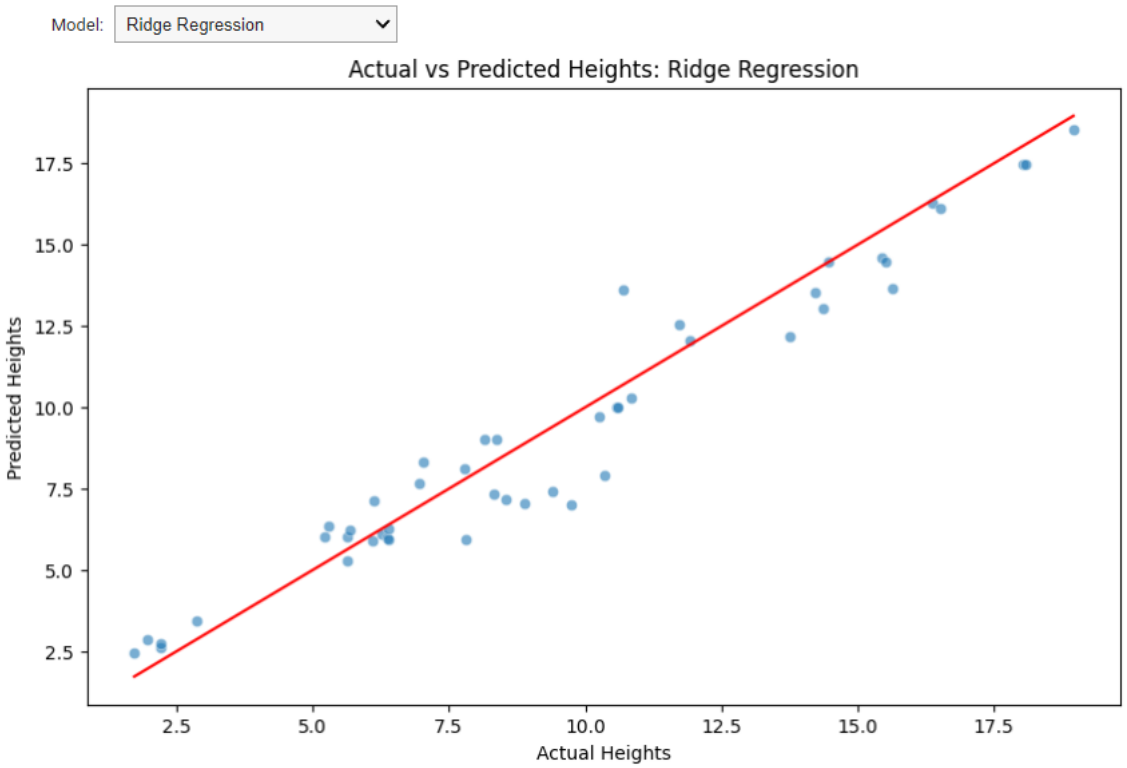


Рисунок 5.12. Графік реальних та прогнозованих значень Height за допомогою моделі Ridge Regression

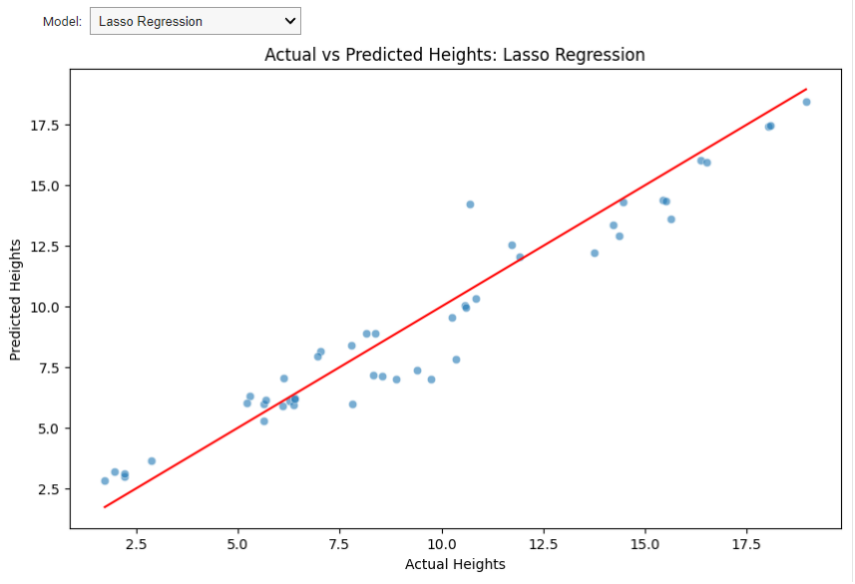


Рисунок 5.13. Графік реальних та прогнозованих значень Height за допомогою моделі Lasso Regression

11. Виконаємо прогнозування на довільних даних

```
new_X = [70, 16, 17, 18, 3]  # Приклад нового значення ...

predictions = {title: model.predict([new_X]) for model, title in zip(models, model_names)}

for title, pred in predictions.items():
    print(f"{title} predicted heights: {pred[0]}")
```

```
Linear Regression predicted heights: 4.2153492851751775
Polynomial Regression predicted heights: 3.8886067868371175
Ridge Regression predicted heights: 4.332190347354458
Lasso Regression predicted heights: 4.5056590783654045
```

Рисунок 5.14. Результати прогнозування різних моделей регресії

12. Виконаємо тестування гіперпараметрів Degree та Alpha

```
# Додавання віджетів для налаштування гіперпараметрів
degree_selector = widgets.IntSlider(min=1, max=5, value=2,
description='Degree:')
alpha_selector = widgets.FloatSlider(min=0.01, max=1, value=0.1, step=0.01,
description='Alpha:')
model_selector = widgets.Dropdown(options=model_names, description='Model:')
```

```
# Функція для тренування моделей з різними налаштуваннями
def train_with_parameters(degree, alpha, model_name):
    if model_name == 'Polynomial Regression':
        model = make_pipeline(PolynomialFeatures(degree=degree),
LinearRegression())
    elif model_name == 'Ridge Regression':
        model = Ridge(alpha=alpha)
    elif model_name == 'Lasso Regression':
        model = Lasso(alpha=alpha)
    else: # Linear Regression
        model = LinearRegression()

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"{model_name}: MSE = {mse:.2f}, R^2 = {r2:.2f}")

    # Візуалізація результатів
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
    sns.lineplot(x=y_test, y=y_test, color='red')
    plt.xlabel('Actual Heights')
    plt.ylabel('Predicted Heights')
    plt.title(f'{model_name} Results')
    plt.show()

widgets.interactive(train_with_parameters, degree=degree_selector,
alpha=alpha_selector, model_name=model_selector)
```

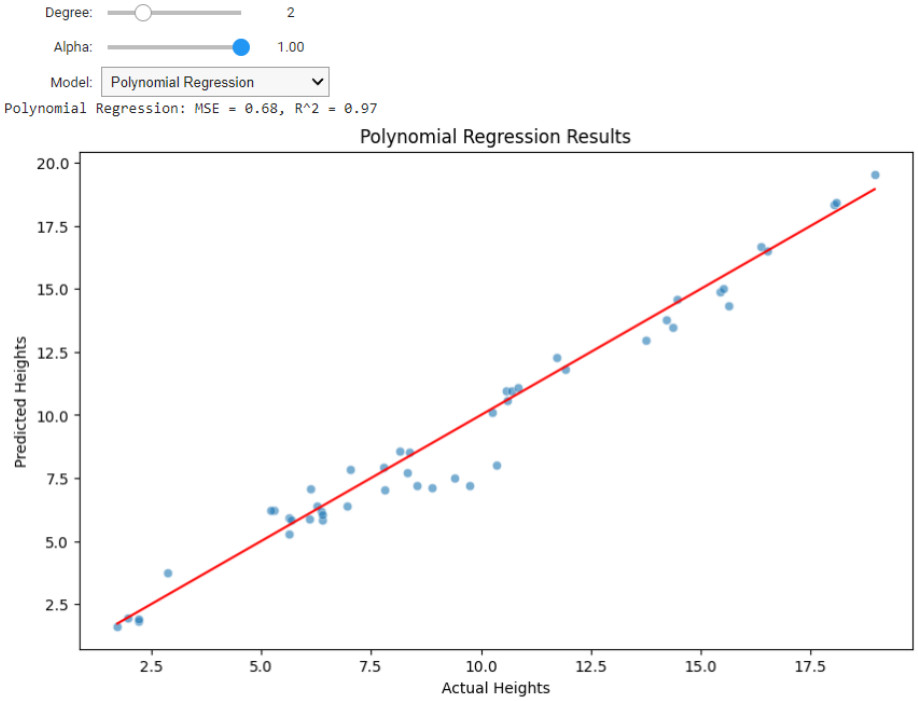


Рисунок 5.15. Можливості для кастомізації гіперпараметрів

Таким чином, аналізуючи отримані дані, можна зазначити, що використання моделі Polynomial Regression продемонструвало найкращі результати MSE (0.92) та R^2 (0.94). Моделі Linear Regression, Ridge Regression та Lasso Regression надали близькі один до одного результати за показниками MSE та R^2, що також є високими та дозволяють виконувати задачу регресії для даного набору даних.

Налаштування гіперпараметрів Degree та Alpha дозволило досягти результатів MSE (0.68), R2 (0.97) для моделі Polynomial Regression, тобто результати регресії були покращені.

Контрольне запитання №1

Вкажіть результуюче значення MSE для Polynomial Regression.
Відповідь: **0.92**

Контрольне запитання №2

Вкажіть результуюче значення R^2 для Lasso Regression.
Відповідь: **0.92**

Контрольне запитання №3

Вкажіть найкраще значення показника R^2 серед усіх моделей.
Відповідь: **0.94**

Висновки: …

Завдання для виконання лабораторної роботи

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.

Посилання на датасети					
1	6	11	16	21	26
2	7	12	17	22	27
3	8	13	18	23	28
4	9	14	19	24	29
5	10	15	20	25	30

2. Проаналізуйте датасет та визначте цільову змінну та ознаки, які будуть використовуватися для регресії.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується. Створіть графіки для розуміння розподілу ознак та їх взаємозв'язку з цільовою змінною.
4. Виконайте тренування моделей: 1) лінійної регресії; 2) поліноміальної регресії; 3) Ridge регресії; 4) Lasso регресії. Виконайте оцінку та візуалізуйте результати за наступними метриками: MSE, R^2 для кожної моделі. Побудуйте графіки фактичних та передбачених значень для кожної моделі.
5. Виконайте прогнозування за допомогою різних моделей для довільних вхідних даних відповідно до вашої цільової змінної.
6. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
7. За бажанням проекспериментуйте з гіперпараметрами Degree та Alpha для моделей задля покращення результатів.

Контрольне запитання №1

Вкажіть результуюче значення MSE для Polynomial Regression.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення R^2 для Lasso Regression.
Відповідь: _____

Контрольне запитання №3

Вкажіть найкраще значення показника R^2 серед усіх моделей.
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1w2OST7DJyDYcgeiFeT4ZIPmdr9zS2Wh-?usp=sharing

Лабораторна робота № 6. Дослідження методів класифікації: K-NN, Naive Bayes, SVM, Decision Trees, Logistic Regression

Мета: сформулювати практичні навички у застосуванні різних алгоритмів машинного навчання для вирішення задач класифікації. Здобувачі навчаться підбирати найбільш ефективний алгоритм для конкретного набору даних, а також здійснювати попередню обробку даних, налаштовувати гіперпараметри моделей та аналізувати результати класифікації.

Теоретичні відомості

Класифікація є однією з основних задач машинного навчання, яка полягає у визначенні категорії або класу для даного об'єкта з використанням набору ознак.

K-NN (K-Nearest Neighbors) – метод, який класифікує об'єкти на основі голосування найближчих сусідів. Він не має експліцитної тренувальної фази або вона дуже проста, що робить його простим у використанні, однак метод вимагає значного обчислювального ресурсу при класифікації нових об'єктів.

Naive Bayes – простий ймовірнісний класифікатор, який застосовує теорему Баєса з «наївним» припущенням про незалежність між ознаками. Попри свою простоту, Naive Bayes може бути дуже ефективним у деяких задачах, особливо при великій кількості ознак.

SVM (Support Vector Machine) – метод, який знаходить гіперплощину в багатовимірному просторі, що найкраще розділяє об'єкти різних класів. SVM ефективний у випадках, коли кількість ознак перевищує кількість спостережень.

Decision Trees – модель, яка використовує деревоподібну структуру для прийняття рішень. Кожен вузол дерева представляє ознаку, кожен лист – клас. Дерева рішень легко інтерпретувати, але схильні до перенавчання.

Logistic Regression – це статистичний метод для аналізу набору даних, в якому одна або кілька незалежних змінних визначають результат. Використовується для задач бінарної класифікації.

Приклад виконання роботи

Завдання

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте ознаки, які будуть використовуватися для класифікації. Як цільову зміну класифікації використовуйте клас заданий у таблиці.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується.
4. Виконайте тренування моделей: 1) K-NN; 2) Naïve Bayes; 3) SVM; 4) Decision Trees; 5) Logistic Regression. Виконайте оцінку та візуалізуйте результати за наступними метриками: Accuracy, Confusion matrix для кожної моделі.
5. Виконайте класифікацію за допомогою різних моделей для довільних вхідних даних відповідно до вашої цільової змінної.
6. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
7. За бажанням проекспериментуйте з гіперпараметрами для моделей задля покращення результатів.

Контрольне запитання №1

Вкажіть результуюче значення Accuracy для K-NN.
Відповідь: _____

Контрольне запитання №2

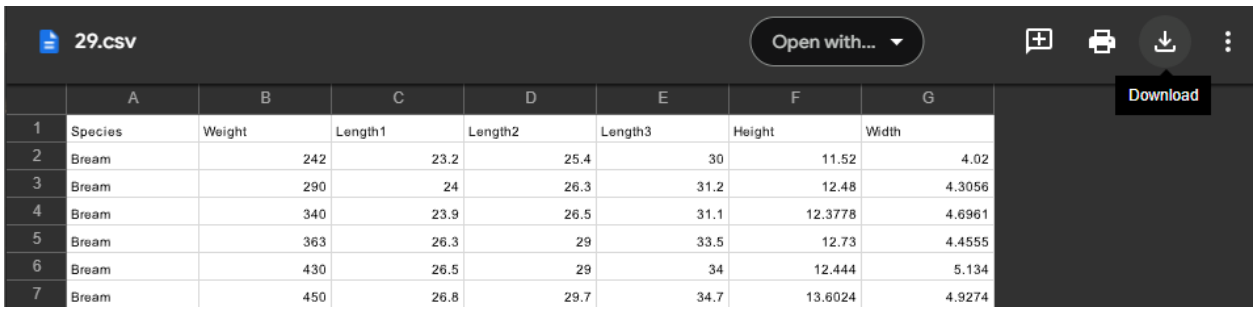
Вкажіть результуюче значення Accuracy для SVM.
Відповідь: _____

Контрольне запитання №3

Вкажіть результуюче значення Accuracy для Logistic Regression.
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набір даних за варіантом.



	A	B	C	D	E	F	G
1	Species	Weight	Length1	Length2	Length3	Height	Width
2	Bream	242	23.2	25.4	30	11.52	4.02
3	Bream	290	24	26.3	31.2	12.48	4.3056
4	Bream	340	23.9	26.5	31.1	12.3778	4.6961
5	Bream	363	26.3	29	33.5	12.73	4.4555
6	Bream	430	26.5	29	34	12.444	5.134
7	Bream	450	26.8	29.7	34.7	13.6024	4.9274

Рисунок 6.1. Датасет на Google Drive

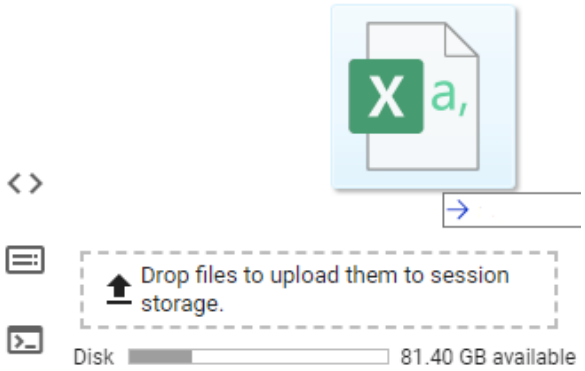


Рисунок 6.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve,
auc, precision_recall_curve
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.datasets import load_iris
import ipywidgets as widgets
from IPython.display import display, clear_output
```

3. Ініціалізація набору даних в Google Colab.

```
# Завдання №1. # Завантаження датасету
data = pd.read_csv("/content/29.csv")
data.head()
```

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

Рисунок 6.3. Перші 5 рядків набору даних

4. Визначимо цільову змінну, як Species, ознаками будуть: Weight, Length1, Length2, Length3, Height, Width.

5. Виконаємо очищення даних

```
# Перевірка на відсутні значення
print(data.isnull().sum())
```

```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

Рисунок 6.4. Пошук нульових значень у датасеті

Порожні значення відсутні, тож видалення або заповнення виконувати не треба.

```
# Видалення або заповнення відсутніх значень
# data.dropna(inplace=True) # Видалення
# data.fillna(data.mean(), inplace=True) # Заповнення середніми значеннями
```

Виконаємо візуалізацію однієї з ознак набору даних, а саме «Length2»

```
# Візуалізація викидів
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Length2'])
plt.title('Boxplot before removing outliers')
```

```
plt.show()
```

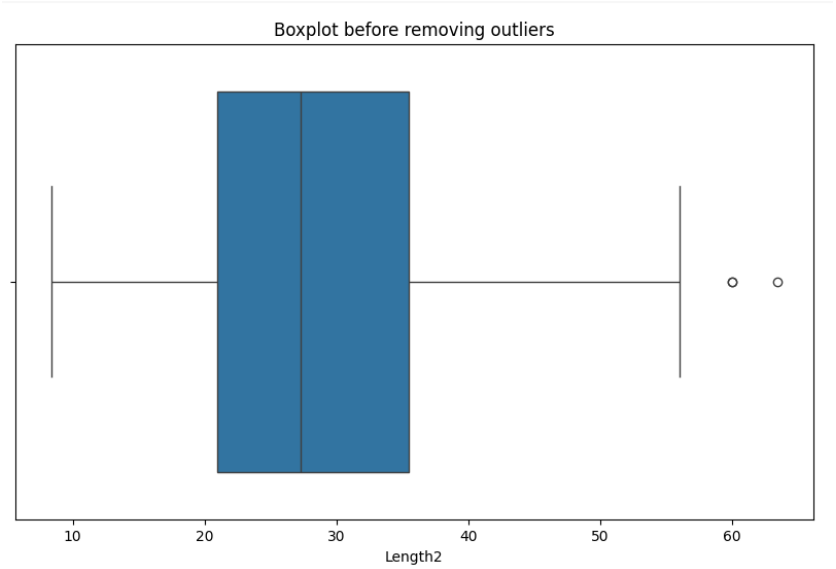


Рисунок 6.5. Діаграма викидів для Length2 перед очищенням

Виконаємо очищення викидів для «Length2».

```
# Видалення викидів
feature_processing = 'Length2'
q_low = data[feature_processing].quantile(0.01)
q_hi  = data[feature_processing].quantile(0.99)
data = data[(data[feature_processing] < q_hi) & (data[feature_processing] >
q_low)]
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Length2'])
plt.title('Boxplot after removing outliers')
plt.show()
```

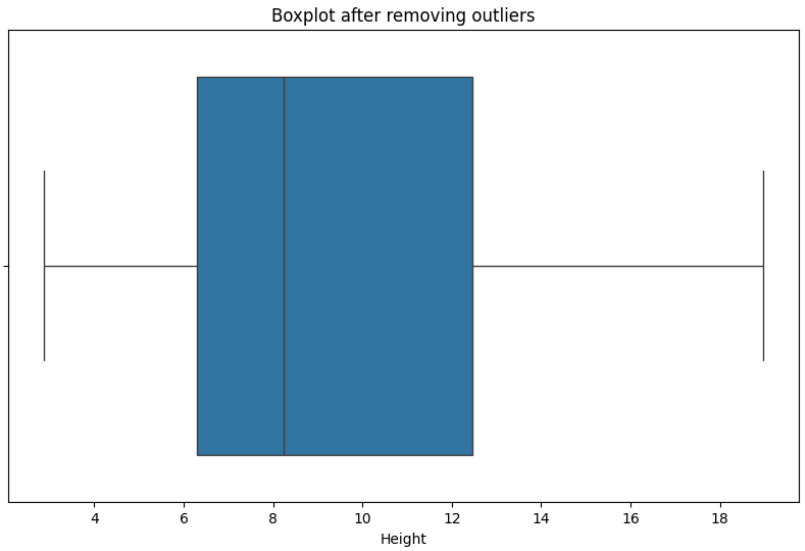


Рисунок 6.6. Діаграма викидів для Length2 після очищення

Подібні операції виконаємо для інших ознак.

6. Розподілимо дані на тренувальні та тестові набори з урахуванням ознак та цільової змінної. В тестовому наборі даних видалимо стовпець «Species», що класифікується, та виконаємо розподіл тестових та тренувальних наборів даних.

```
X = data.drop('Species', axis=1)
y = data['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Виконаємо нормалізацію даних.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

7. Виконаємо тренування моделей K-NN, Naïve Bayes, SVM, Decision Trees, Logistic Regression та їх оцінку за показниками Accuracy та Confusion Matrix.

```
model_widget = widgets.Dropdown(
    options=['K-NN', 'Naive Bayes', 'SVM', 'Decision Trees', 'Logistic
Regression'],
    description='Model:',
)

hyperparameter_widget = widgets.IntSlider(
    value=3,
    min=1,
    max=10,
    step=1,
    description='Hyperparameter:',
)

@widgets.interact(model_choice=model_widget,
hyperparameter=hyperparameter_widget)
def train_model(model_choice, hyperparameter):
    print(f"Починаємо обробку з моделлю {model_choice} та гіперпараметром
{hyperparameter}...")
    if model_choice == 'K-NN':
        model = KNeighborsClassifier(n_neighbors=hyperparameter)
    elif model_choice == 'SVM':
        model = SVC(kernel='linear', C=hyperparameter)
    elif model_choice == 'Decision Trees':
        model = DecisionTreeClassifier(max_depth=hyperparameter)
    elif model_choice == 'Logistic Regression':
        model = LogisticRegression(C=hyperparameter)
    else:
        model = GaussianNB()
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d")
plt.show()
```

8. Виконаємо оцінювання моделей за показниками Accuracy та Confusion matrix.

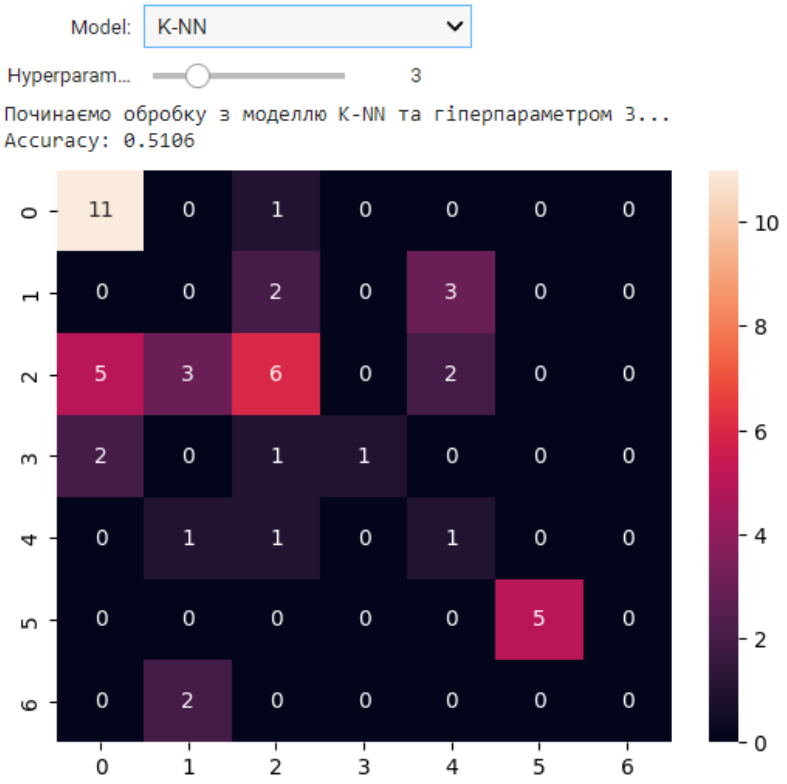


Рисунок 6.7. Результати моделі K-NN

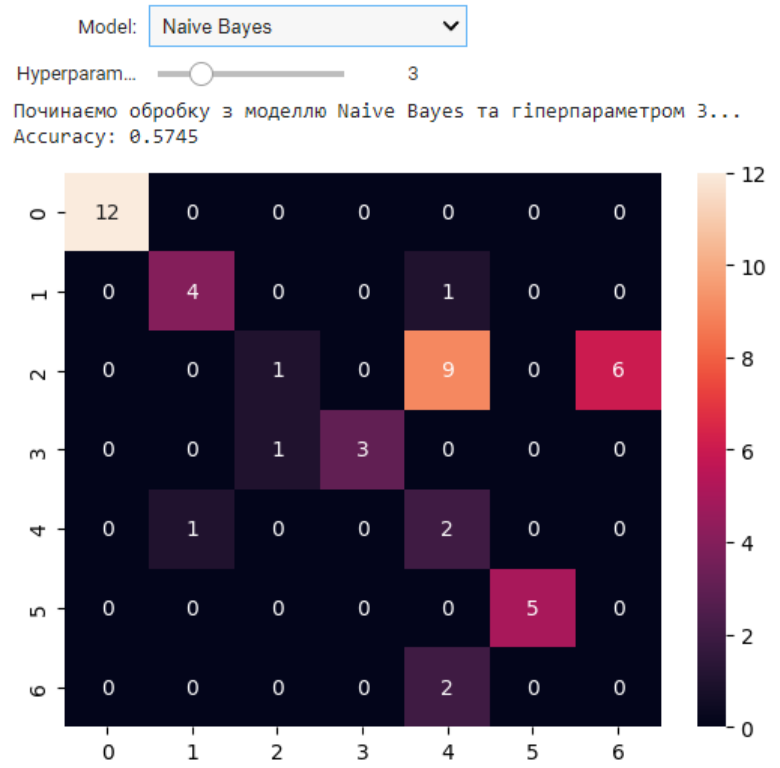


Рисунок 6.8. Результати моделі Naïve Bayes

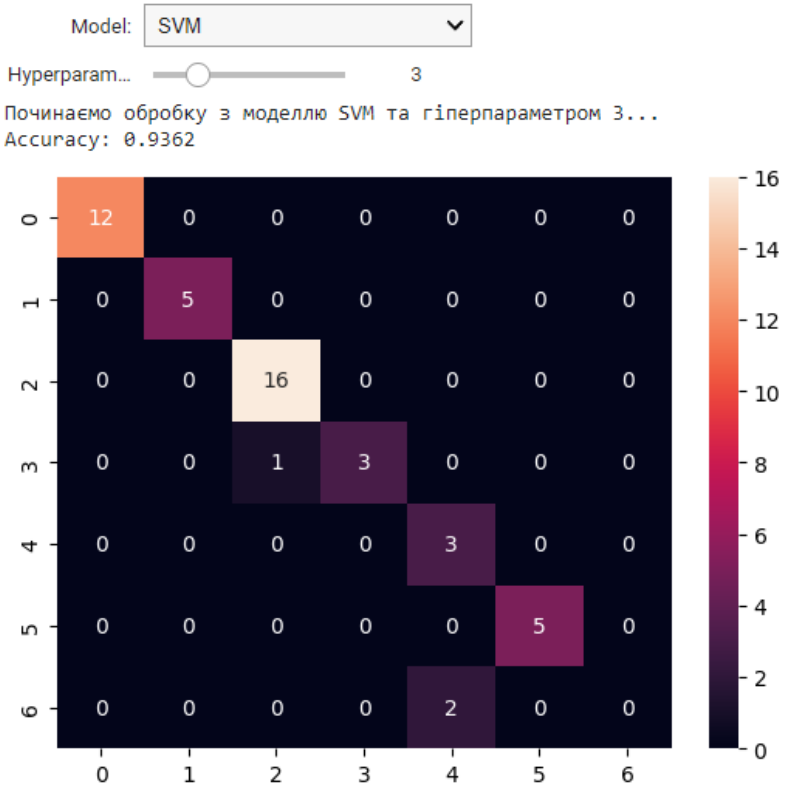


Рисунок 6.9. Результати моделі SVM

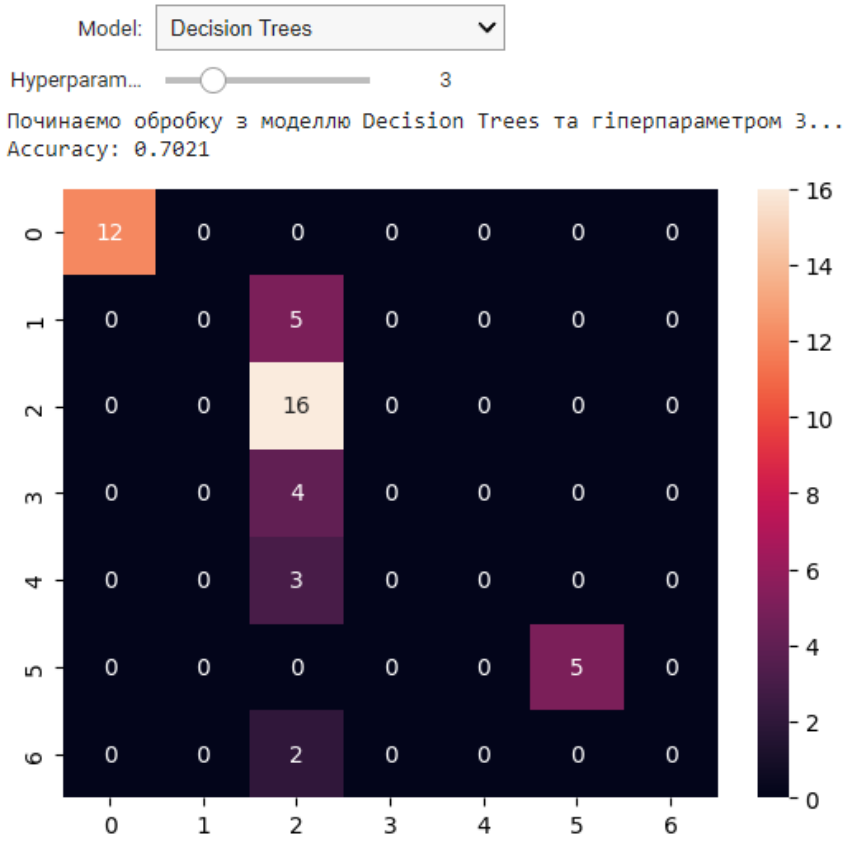


Рисунок 6.10. Результати моделі Decision Trees

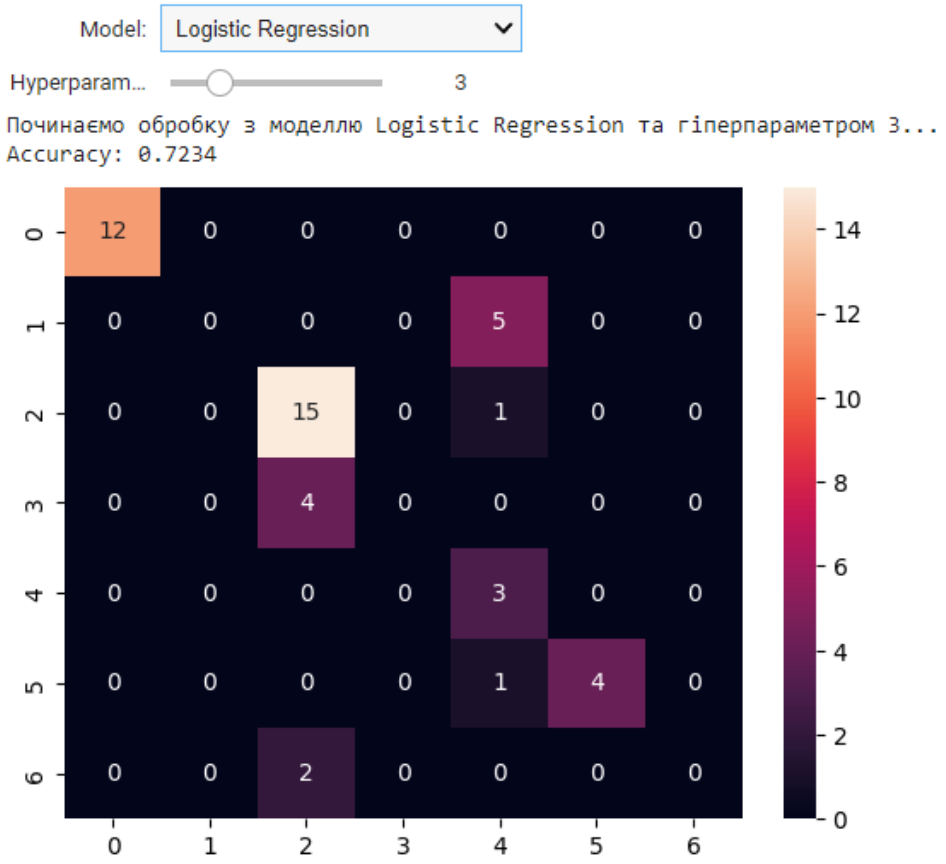


Рисунок 6.11. Результати моделі Logistic Regression

9. Виконаємо прогнозування на довільних даних

```
input_widgets = [widgets.FloatText(description=f'Feature {i+1}:') for i in
range(X_train.shape[1])]

classify_button = widgets.Button(description="Classify")

output = widgets.Output()

def classify(b):
    with output:
        output.clear_output()
        data_to_classify = [w.value for w in input_widgets]

        model_choice = model_widget.value
        hyperparameter = hyperparameter_widget.value
        model = prepare_model(model_choice, hyperparameter)


        prediction = model.predict([data_to_classify])
        print(f"Predicted class: {prediction[0]}")

classify_button.on_click(classify)
def prepare_model(model_choice, hyperparameter):
    if model_choice == 'K-NN':
        model = KNeighborsClassifier(n_neighbors=hyperparameter)
    elif model_choice == 'SVM':
        model = SVC(kernel='linear', C=hyperparameter)
    elif model_choice == 'Decision Trees':
        model = DecisionTreeClassifier(max_depth=hyperparameter)
    elif model_choice == 'Logistic Regression':
        model = LogisticRegression(C=hyperparameter)
    else:
        model = GaussianNB()
    model.fit(X_train, y_train)
    return model

display(*input_widgets, model_widget, hyperparameter_widget,
classify_button, output)
```

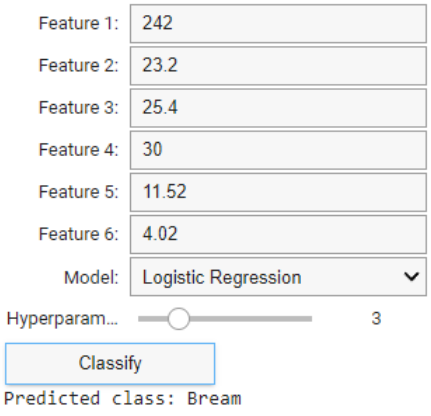


Рисунок 6.12. Результати прогнозування для моделі Logistic Regression

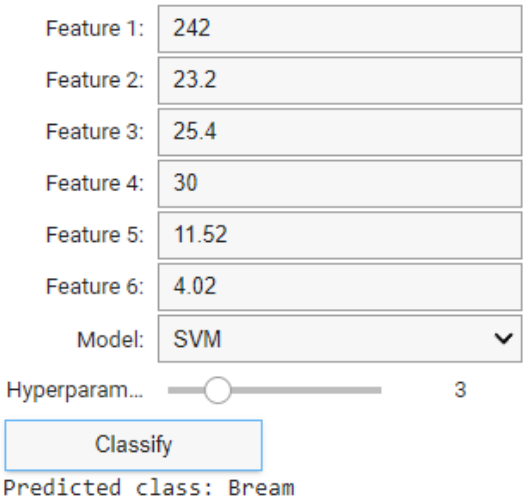


Рисунок 6.13. Результати прогнозування для моделі SVM

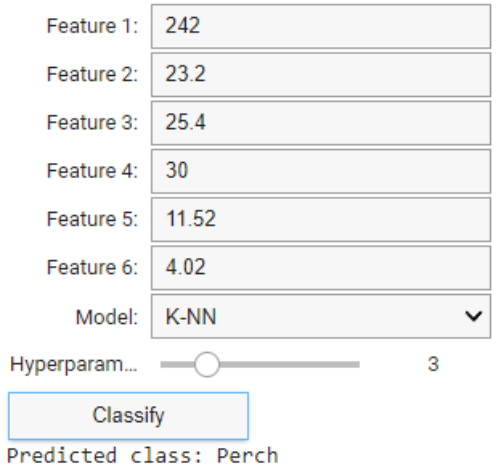


Рисунок 6.14. Результати прогнозування для моделі K-NN

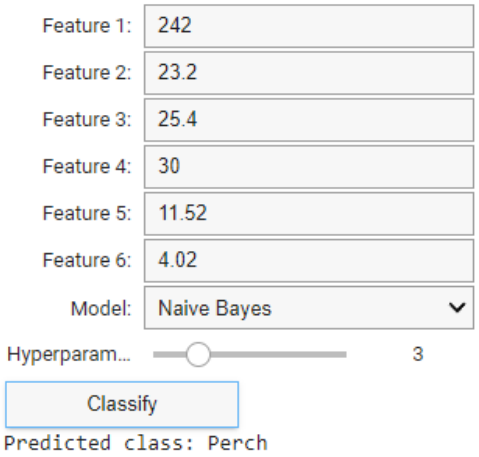


Рисунок 6.15. Результати прогнозування для моделі Naïve Bayes

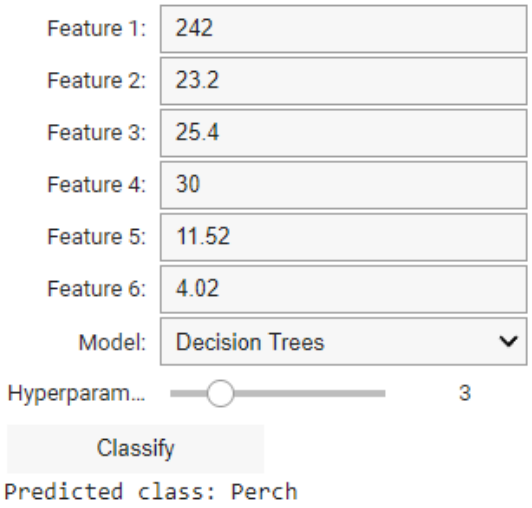


Рисунок 6.16. Результати прогнозування для моделі Decision Trees

10. Виконаємо тестування гіперпараметрів

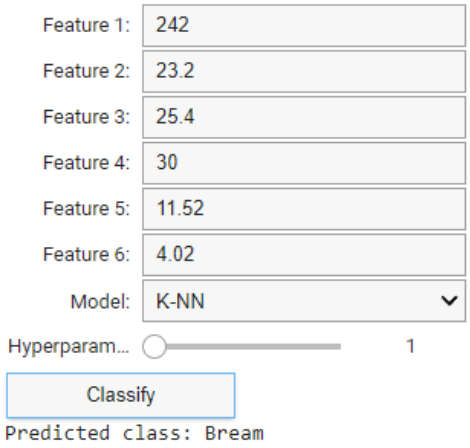


Рисунок 6.17. Результати прогнозування для моделі K-NN зі значенням гіперпараметрів «1»

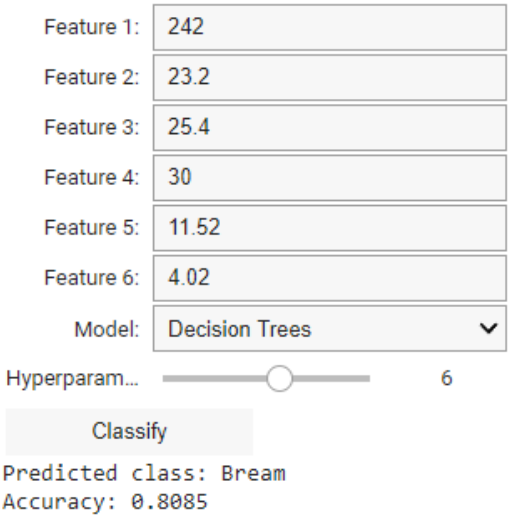


Рисунок 6.18. Результати прогнозування для моделі Decision Trees зі значенням гіперпараметрів «6»

Таким чином, аналізуючи отримані дані, можна зазначити, що використання моделі SVM продемонструвало найкращі результати за показником Accuracy (0.9362). Моделі Decision Trees (0.7021) та Logistic Regression (0.7234) надали близькі один до одного результати за показником Accuracy, що є достатнім та дозволяє виконувати задачу класифікації для даного набору даних. Проте зі стандартним налаштуванням гіперпараметрів модель Decision Trees надала некоректний результат класифікації на довільному прикладі. При збільшенні значення гіперпараметрів, модель почала надавати більшу точність (0.8085) та правильно класифікувала приклад. Що стосується моделей K-NN (0.5106) та Naïve Bayes (0.5745), то їх показники точності є незадовільними та не дозволяють якісно виконувати класифікацію на заданому наборі даних, потребують доопрацювання.

Контрольне запитання №1

Вкажіть результуюче значення Accuracy для K-NN.
Відповідь: **0.5106**

Контрольне запитання №2

Вкажіть результуюче значення Accuracy для SVM.
Відповідь: **0.9362**

Контрольне запитання №3

Вкажіть результуюче значення Accuracy для Logistic Regression.
Відповідь: **0.7234**

Висновки: …

Завдання для виконання лабораторної роботи

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте ознаки, які будуть використовуватися для класифікації. Як цільову зміну класифікації використовуйте клас заданий у таблиці.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується.
4. Виконайте тренування моделей: 1) K-NN; 2) Naïve Bayes; 3) SVM; 4) Decision Trees; 5) Logistic Regression. Виконайте оцінку та візуалізуйте результати за наступними метриками: Accuracy, Confusion matrix для кожної моделі.
5. Виконайте класифікацію за допомогою різних моделей для довільних вхідних даних відповідно до вашої цільової змінної.
6. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
7. За бажанням проекспериментуйте з гіперпараметрами для моделей задля покращення результатів.

	Посилання на датасети				
№	**1**	**5**	**9**	**13**	**17**
Classification	Brand	label	Microcalcification	category	wifi
№	**2**	**6**	**10**	**14**	**18**
Classification	class	Type	diabetes	R	is_safe
№	**3**	**7**	**11**	**15**	**19**
Classification	POP	Species	Car	class	quality
№	**4**	**8**	**12**	**16**	**20**
Classification	ocean_proximity	CarName	poutcome	custcat	Species

Контрольне запитання №1

Вкажіть результуюче значення Accuracy для K-NN.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Accuracy для SVM.
Відповідь: _____

Контрольне запитання №3

Вкажіть результуюче значення Accuracy для Logistic Regression.
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1eAe0Z89Tb3HNyNbenUcn7RSUIqr00i5I?usp=sharing

Лабораторна робота № 7. Дослідження методів кластеризації: K-Means, DBSCAN, Mean-Shift, Agglomerative Clustering, Fuzzy C-Means

Мета: сформулювати практичні навички у застосуванні та аналізі різних методів кластеризації, таких як K-Means, DBSCAN, Mean-Shift, Agglomerative Clustering, Fuzzy C-Means, з метою глибшого розуміння їх особливостей, переваг та обмежень. Розвинути здатність студентів до самостійного вибору найбільш коректного методу кластеризації для конкретного аналітичного завдання, а також навички роботи з реальними даними, включаючи попередню обробку, нормалізацію даних і візуалізацію результатів.

Теоретичні відомості

Методи кластеризації – це фундаментальні інструменти в машинному навчанні та аналізі даних, які дозволяють групувати схожі об'єкти в кластери. Вони відіграють ключову роль у зниженні складності даних, виявленні шаблонів та аномалій, а також у попередній обробці даних для подальшого аналізу.

K-**Means** – один з найпопулярніших методів кластеризації, який розділяє датасет на K кластери, мінімізуючи суму квадратів відстаней між точками та центрами кластерів. Цей метод чутливий до вибору початкових центрів і кількості кластерів.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) – метод кластеризації, заснований на густині, ідеально підходить для виявлення кластерів довільної форми та викидів. Він використовує поняття досяжності густини, щоб визначити кластери.

Mean-Shift – метод кластеризації, який використовує рух до вищих густин даних, визначаючи кластери без заздалегідь заданої кількості. Це робить його ефективним для виявлення кластерів нестандартних форм.

Agglomerative Clustering – метод ієрархічної кластеризації, який починає з того, що кожен об'єкт представляє собою окремий кластер, а потім послідовно об’єднує кластери на основі міри схожості, поки не буде досягнута задана кількість кластерів або інший критерій зупинки.

Fuzzy C-Means – це розширення K-Means, яке дозволяє об'єктам належати до кількох кластерів одночасно з різним ступенем приналежності, що забезпечує більшу гнучкість у виявленні кластерів, що перекриваються.

Приклад виконання роботи

Завдання

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте ознаки, які будуть використовуватися для кластеризації. Як цільову зміну кластеризації за замовчуванням визначено останній стовпець у наборах даних.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується.
4. Виконайте тренування моделей: 1) K-Means; 2) DBSCAN; 3) Mean-Shift; 4) Agglomerative Clustering; 5) Gaussian Mixture. Виконайте оцінку та візуалізуйте результати за наступними метриками: Silhouette для кожної моделі.
5. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
6. За бажанням проекспериментуйте з гіперпараметрами для моделей задля покращення результатів.

Контрольне запитання №1

Вкажіть результуюче значення Silhouette для K-Means.
Відповідь: _____

Контрольне запитання №2

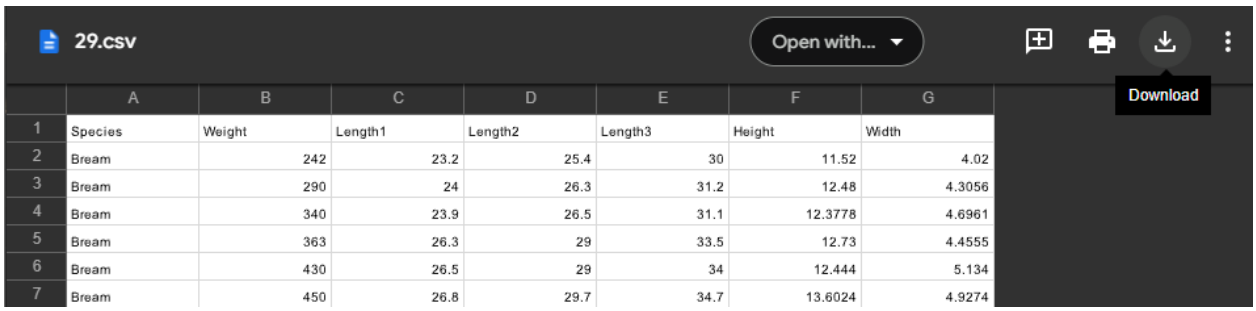
Вкажіть результуюче значення Silhouette для Agglomerative Clustering.
Відповідь: _____

Контрольне запитання №3

Вкажіть результуюче значення Silhouette для Gaussian Mixture.
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набір даних за варіантом.



	A	B	C	D	E	F	G
1	Species	Weight	Length1	Length2	Length3	Height	Width
2	Bream	242	23.2	25.4	30	11.52	4.02
3	Bream	290	24	26.3	31.2	12.48	4.3056
4	Bream	340	23.9	26.5	31.1	12.3778	4.6961
5	Bream	363	26.3	29	33.5	12.73	4.4555
6	Bream	430	26.5	29	34	12.444	5.134
7	Bream	450	26.8	29.7	34.7	13.6024	4.9274

Рисунок 7.1. Датасет на Google Drive

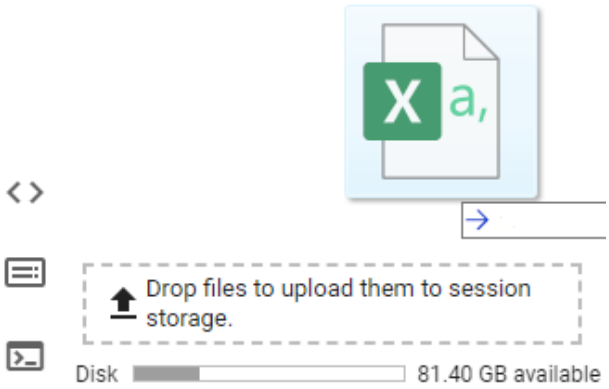


Рисунок 7.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans, DBSCAN, MeanShift,
AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
import ipywidgets as widgets
from IPython.display import display
```

3. Ініціалізація набору даних в Google Colab.

```
# Завдання №1. # Завантаження датасету
data = pd.read_csv("/content/5.csv")
features = data.columns[:-1].tolist()  # Отримання списку ознак
data.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Рисунок 7.3. Перші 5 рядків набору даних

4. Визначимо цільову змінну, як Chance of Admit, ознаками будуть: GRE Score та TOEFL Score. Обрані ознаки простираються на широкий діапазон значень та можуть бути досить інформативними під час кластеризації.

5. Виконаємо очищення даних

```
# Перевірка на відсутні значення
print(data.isnull().sum())
```

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

Рисунок 7.4. Пошук нульових значень у датасеті

Порожні значення відсутні, тож видалення або заповнення виконувати не треба.

```
# Видалення або заповнення відсутніх значень
# data.dropna(inplace=True) # Видалення

# data.fillna(data.mean(), inplace=True) # Заповнення середніми значеннями
```

Виконаємо візуалізацію однієї з ознак набору даних, а саме «Length2»

```
# Візуалізація викидів
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['GRE Score'])
plt.title('Boxplot before removing outliers')
plt.show()
```

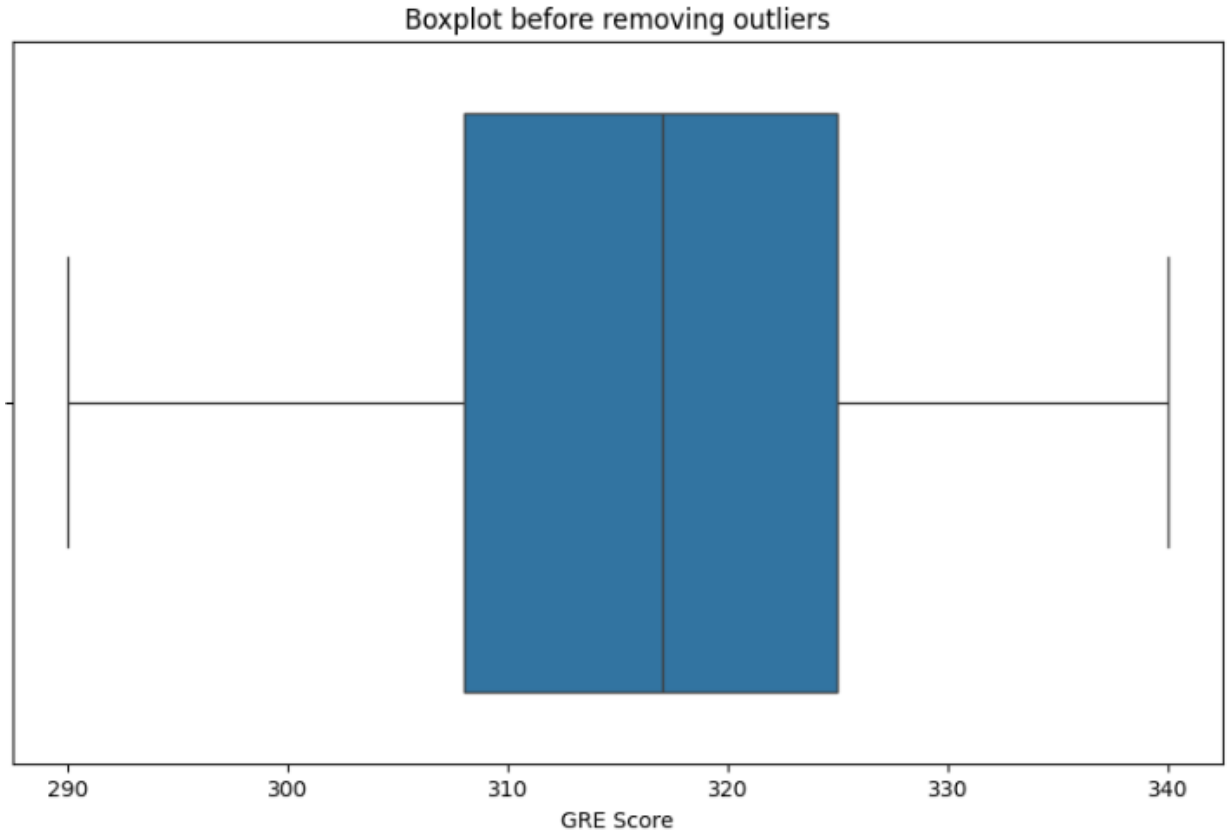


Рисунок 7.5. Діаграма викидів для GRE Score перед очищенням

Викидів не зафіксовано, тож додаткове очищення виконувати немає сенсу.

6. Побудуємо логіку взаємодії з віджетом задля обробки методів кластеризації

```
# Віджети для вибору ознак
selected_features = []

def add_feature(b):
    if feature_dropdown.value and feature_dropdown.value not in
selected_features:
        selected_features.append(feature_dropdown.value)
        update_feature_list()



def remove_feature(b):
    if feature_list_widget.value:
        selected_features.remove(feature_list_widget.value)
        update_feature_list()

def update_feature_list():
    feature_list_widget.options = selected_features

feature_dropdown = widgets.Dropdown(options=features,
description='Features:')
add_button = widgets.Button(description="Add Feature")
remove_button = widgets.Button(description="Remove Feature")
```

```
feature_list_widget = widgets.Select(options=selected_features,
description="Selected Features:")

add_button.on_click(add_feature)
remove_button.on_click(remove_feature)

# Віджет для вибору методу кластеризації
clustering_widget = widgets.Dropdown(
    options=['K-Means', 'DBSCAN', 'Mean-Shift', 'Agglomerative Clustering',
'Gaussian Mixture'],
    description='Method:',
)

# Віджет для вибору гіперпараметру EPS для DBSCAN
eps_widget = widgets.FloatSlider(description='EPS:', min=0.1, max=10.0,
value=0.5, step=0.1)

# Віджет для вибору гіперпараметру Min Samples для DBSCAN
min_samples_widget = widgets.IntSlider(description='Min Samples:', min=1,
max=20, value=5)

n_clusters_widget = widgets.IntSlider(
    value=3,
    min=2,
    max=10,
    step=1,
    description='N Clusters:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='d'
)

build_button = widgets.Button(description="Build Clusters")

output = widgets.Output()
method_widget = widgets.Dropdown(
    options=['K-Means', 'DBSCAN', 'Mean-Shift', 'Agglomerative Clustering',
'Gaussian Mixture'],
    description='Method:',
    disabled=False,
)
```

7. Виконаємо кластеризацію та оцінку моделей кластеризації.

```
def build_clusters(b):
    with output:
        output.clear_output()
        if not selected_features:
            print("Please add features to cluster.")
```

```
            return
        # Попередня обробка та нормалізація даних
        scaler = StandardScaler()
        scaled_data = scaler.fit_transform(data[selected_features])

        n_clusters = n_clusters_widget.value
        method = method_widget.value

        if method == 'K-Means':
            model = KMeans(n_clusters=n_clusters)
        elif method == 'DBSCAN':
            model = DBSCAN(eps=eps_widget.value,
min_samples=min_samples_widget.value)
        elif method == 'Mean-Shift':
            model = MeanShift()
        elif method == 'Agglomerative Clustering':
            model = AgglomerativeClustering(n_clusters=n_clusters)
        elif method == 'Gaussian Mixture':
            model = GaussianMixture(n_components=n_clusters)
        model.fit(scaled_data)

        labels = model.labels_ if hasattr(model, 'labels_') else
model.predict(scaled_data)

        # Візуалізація
        plt.figure(figsize=(10, 6))
        sns.scatterplot(x=scaled_data[:, 0], y=scaled_data[:, 1],
hue=labels, palette='viridis')
        plt.title(f'Clustering Result with {method}')
        plt.show()

        silhouette = silhouette_score(scaled_data, labels)
        print(f'Silhouette Score: {silhouette:.2f}')

build_button.on_click(build_clusters)

# Відображення віджетів
feature_selector_widgets = widgets.VBox([feature_dropdown, add_button,
remove_button, feature_list_widget])
settings_widgets = widgets.VBox([method_widget, eps_widget,
min_samples_widget, n_clusters_widget, build_button])
display(widgets.HBox([feature_selector_widgets, settings_widgets]), output)
```

Рисунок 7.6. Результати моделі K-Means

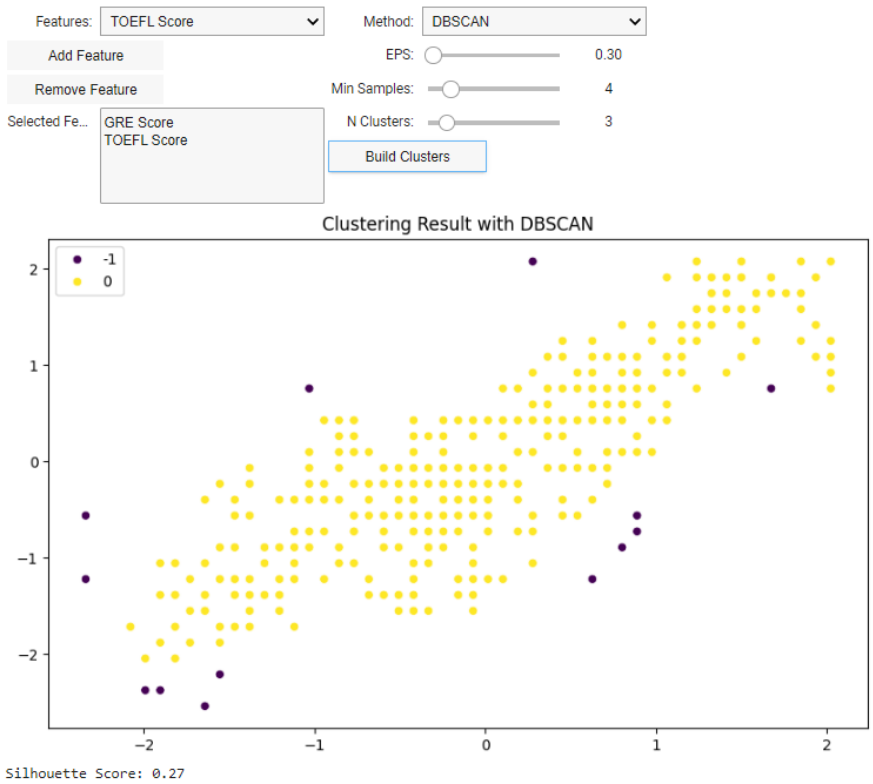


Рисунок 7.7. Результати моделі DBSCAN

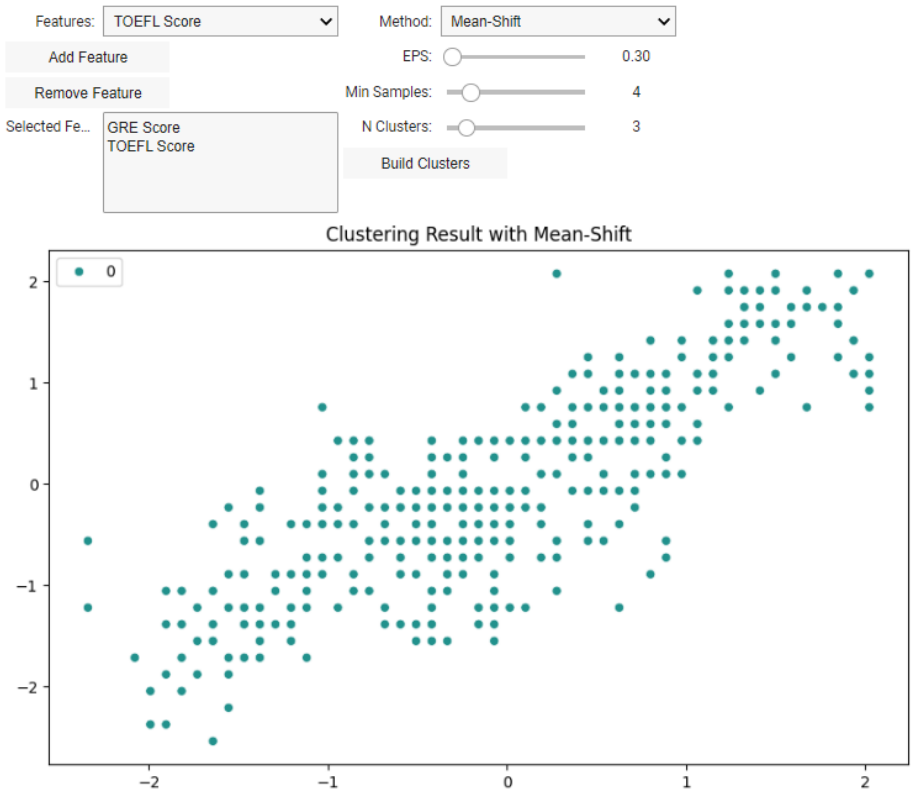


Рисунок 7.8. Результати моделі Mean-Shift

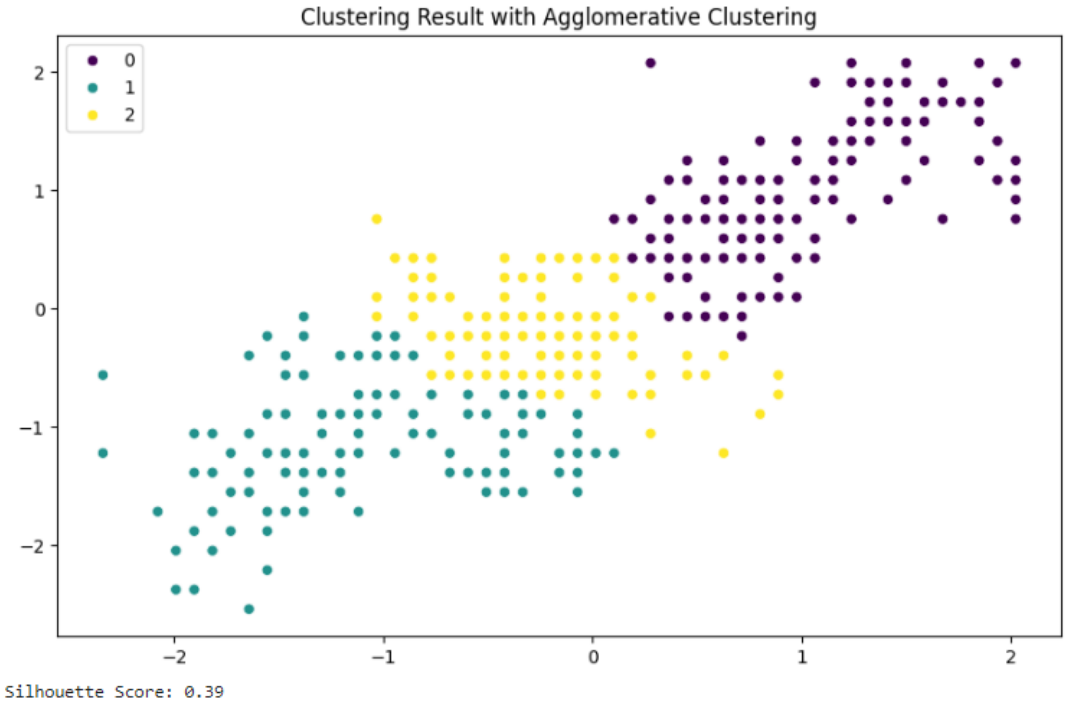


Рисунок 7.9. Результати моделі Agglomerative Clustering

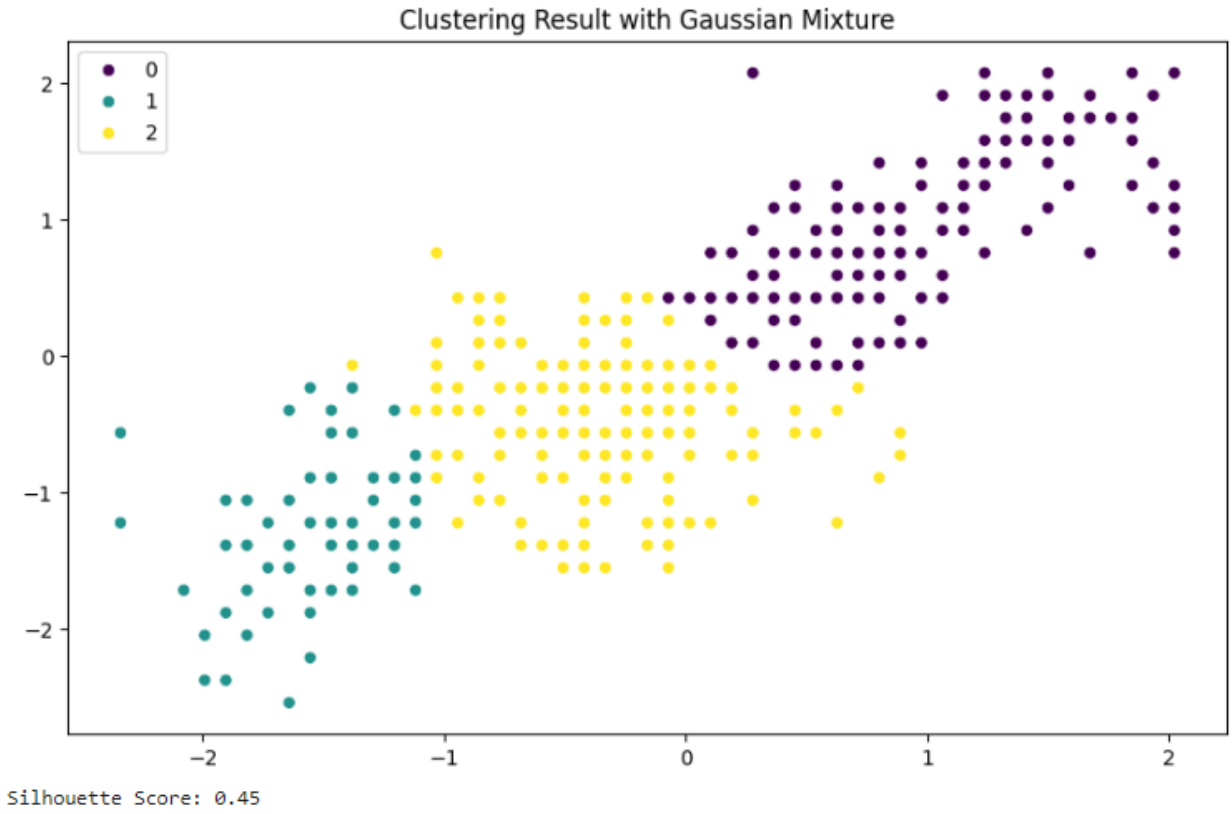


Рисунок 7.10. Результати моделі Gaussian Mixture

8. Виконаємо тестування гіперпараметрів

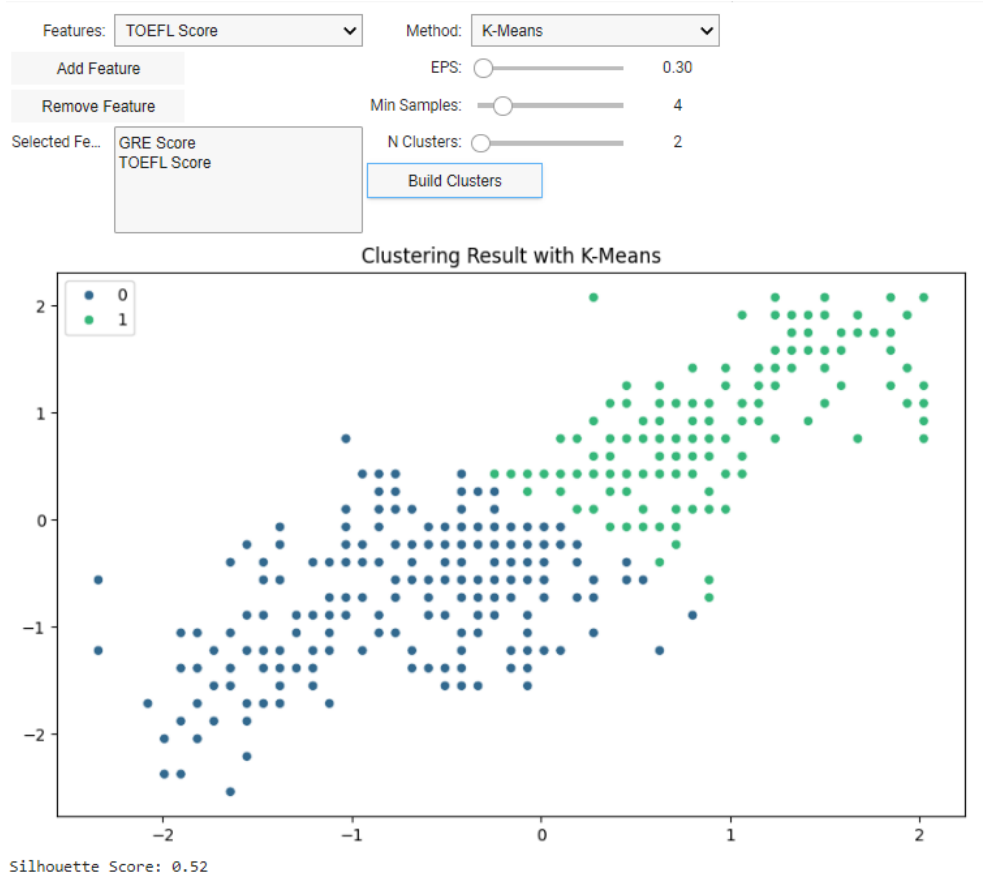


Рисунок 7.11. Зменшення кількості кластерів до «2» у моделі K-Means

Рисунок 7.12. Збільшення гіперпараметру Min Samples до «13» у моделі DBSCAN

Таким чином, аналізуючи отримані дані, можна зазначити, що використання моделі Gaussian Mixture продемонструвало найкращі результати за показником Silhouette (0.45) за умови поділу на 3 кластери. Моделі Mean-Shift не вистачило даних або ознак, тому вона не надала коректних результатів. Модель K-Means (0.44) та Agglomerative Clustering (0.39) також продемонстрували непогані результати Silhouette. Що стосується моделі DBSCAN (0.07) – її показник Silhouette є незадовільними та не дозволяє якісно виконувати кластеризацію на заданому наборі даних, потребується доопрацювання. За рахунок маніпуляції з гіперпараметрами EPS та Min Samples вдалося підвищити показник Silhouette до 0.21.

Контрольне запитання №1

Вкажіть результуюче значення Silhouette для K-Means.
Відповідь: **0.44**

Контрольне запитання №2

Вкажіть результуюче значення Silhouette для Agglomerative Clustering.
Відповідь: **0.39**

Контрольне запитання №3

Вкажіть результуюче значення Silhouette для Gaussian Mixture.

Відповідь: **0.45**

Висновки: …

Завдання для виконання лабораторної роботи

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте ознаки, які будуть використовуватися для кластеризації. Як цільову зміну кластеризації за замовчуванням визначено останній стовпець у наборах даних.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується.
4. Виконайте тренування моделей: 1) K-Means; 2) DBSCAN; 3) Mean-Shift; 4) Agglomerative Clustering; 5) Gaussian Mixture. Виконайте оцінку та візуалізуйте результати за наступними метриками: Silhouette для кожної моделі.
5. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
6. За бажанням проекспериментуйте з гіперпараметрами для моделей задля покращення результатів.

	Посилання на датасети				
№	**1**	**5**	**9**	**13**	**17**
Clusterization	Customer_Segment	Chance of Admit	Shape	category	profit
№	**2**	**6**	**10**	**14**	**18**
Clusterization	Profit	Sales	diabetes	R	Population
№	**3**	**7**	**11**	**15**	**19**
Clusterization	epsilon	Species	CO2	fiber_ID	class
№	**4**	**8**	**12**	**16**	**20**
Clusterization	median_house_value	Profit	quality	Median_Family_Income	Total

Контрольне запитання №1

Вкажіть результуюче значення Silhouette для K-Means.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Silhouette для Agglomerative Clustering.
Відповідь: _____

Контрольне запитання №3

Вкажіть результуюче значення Silhouette для Gaussian Mixture.
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/131_zrC7dhzn9LpNjBzhRZ_e51pgKpcdi?usp=sharing

Лабораторна робота № 8. Дослідження алгоритмів асоціативних правил: Apriori, FP-Growth

Мета: сформулювати практичні навички у використанні алгоритмів асоціативних правил, таких як Apriori та FP-Growth, для аналізу великих даних і виявлення сильних асоціативних зв'язків між різними елементами у базі даних транзакцій. Ця лабораторна робота має на меті розвинути здатність здобувачів аналізувати реальні або синтетичні набори даних, виявляти значущі зв'язки між елементами та використовувати отримані знання для прийняття обґрунтованих рішень у сфері маркетингу, рекомендаційних систем, управління запасами та інших областях.

Теоретичні відомості

Асоціативні правила є одним з основних інструментів аналізу даних у машинному навчанні та інтелектуальному аналізі даних, які дозволяють виявляти важливі зв'язки, асоціації та кореляції між різними елементами в наборах даних. Правила можуть бути використані для виявлення закономірностей покупок у роздрібній торгівлі, для аналізу поведінки користувачів на веб-сайтах, у медицині для виявлення комбінацій симптомів та лікувань тощо.

Apriori – класичний алгоритм для пошуку асоціативних правил, що працює шляхом виявлення наборів елементів, які з'являються разом у транзакціях з частотою вище встановленого порога (порога мінімальної підтримки). **Apriori** використовує ітеративний підхід, де на кожному кроці генеруються кандидати з більшою кількістю елементів на основі наборів, що пройшли попередній крок, та перевіряються на мінімальну підтримку.

FP-Growth (Frequent Pattern Growth) – це ефективніший алгоритм для пошуку асоціативних правил, який не потребує «кандидатської генерації» (процесу створення можливих наборів елементів (кандидатів), які можуть задовольняти встановлені критеріїв), як **Apriori**. Замість цього, **FP-Growth** використовує структуру даних, відому як FP-дерево, для зберігання частотної інформації про елементи. Алгоритм працює в два кроки: спочатку будується FP-дерево, а потім, використовуючи це дерево, виявляються часті набори елементів.

Обидва алгоритми мають свої переваги та недоліки. **Apriori** простий у розумінні та реалізації, але може бути неефективним на великих наборах даних через велику кількість сканувань бази даних і генерації кандидатів. **FP-Growth**, водночас, є значно швидшим за рахунок зменшення кількості сканувань бази даних та відсутності необхідності генерації кандидатських наборів, проте він вимагає більше пам'яті для зберігання FP-дерева.

Важливими поняттями при роботі з асоціативними правилами є підтримка (**support**), впевненість (**confidence**) і підйом (**lift**). Підтримка вказує, наскільки часто набір елементів з’являється в транзакціях; впевненість показує, наскільки часто правила застосовні, а підйом оцінює, наскільки асоціація між елементами сильніша, ніж випадково.

Приклад виконання роботи

Завдання

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Виконайте згрупування даних по транзакціям та перетворіть транзакції для подальшого аналізу.
3. Використайте алгоритми пошуку асоціативних правил: 1) Apriori; 2) FP-Growth. Виконайте оцінку та візуалізуйте результати у вигляді графу.
4. Сформуйте рекомендації на основні виявлених асоціативних правил.
5. Проаналізуйте отримані результати.

Контрольне запитання №1

Вкажіть максимальне значення support для Apriori.
Відповідь: _____

Контрольне запитання №2

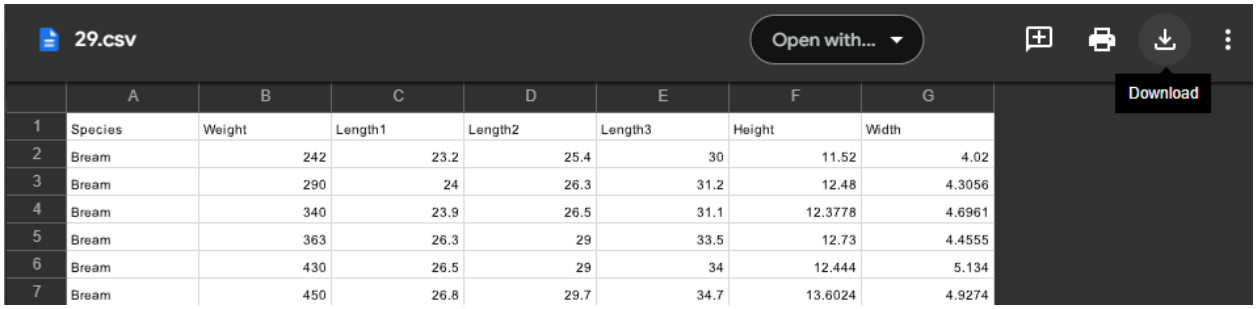
Вкажіть максимальне значення confidence для FP-Growth.
Відповідь: _____

Контрольне запитання №3

Вкажіть кількість виявлених правил при support = 0.10; confidence = 0.3.
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набір даних за варіантом.



	A	B	C	D	E	F	G
1	Species	Weight	Length1	Length2	Length3	Height	Width
2	Bream	242	23.2	25.4	30	11.52	4.02
3	Bream	290	24	26.3	31.2	12.48	4.3056
4	Bream	340	23.9	26.5	31.1	12.3778	4.6961
5	Bream	363	26.3	29	33.5	12.73	4.4555
6	Bream	430	26.5	29	34	12.444	5.134
7	Bream	450	26.8	29.7	34.7	13.6024	4.9274

Рисунок 8.1. Датасет на Google Drive

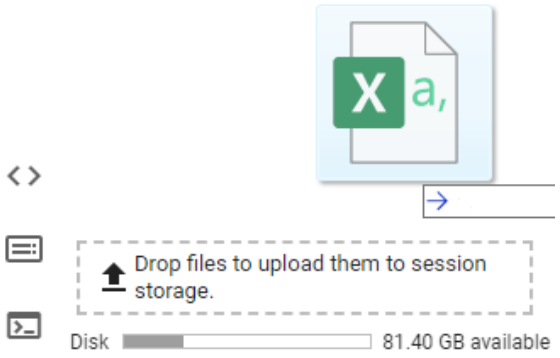


Рисунок 8.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів

```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
from mlxtend.preprocessing import TransactionEncoder
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
import ipywidgets as widgets
from IPython.display import display
```

3. Ініціалізація набору даних в Google Colab.

```
# Завдання №1. # Завантаження датасету
data = pd.read_csv("/content/33.csv", names=['items'], header=None)
data.head()
```

	items
0	MILK,BREAD,BISCUIT
1	BREAD,MILK,BISCUIT,CORNFLAKES
2	BREAD,TEA,BOURNVITA
3	JAM,MAGGI,BREAD,MILK
4	MAGGI,TEA,BISCUIT

Рисунок 8.3. Перші 5 рядків набору даних

4. Виконаємо згрупування даних по транзакціям та перетворення транзакцій

```
# Групування даних по транзакціям
transactions = data['items'].str.split(',')
# Перетворення транзакцій
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_transformed = pd.DataFrame(te_ary, columns=te.columns_)
transactions.head()
```

```
0                [MILK, BREAD, BISCUIT]
1    [BREAD, MILK, BISCUIT, CORNFLAKES]
2               [BREAD, TEA, BOURNVITA]
3            [JAM, MAGGI, BREAD, MILK]
4                 [MAGGI, TEA, BISCUIT]
Name: items, dtype: object
```

Рисунок 8.4. Згруповані та перетворені транзакції

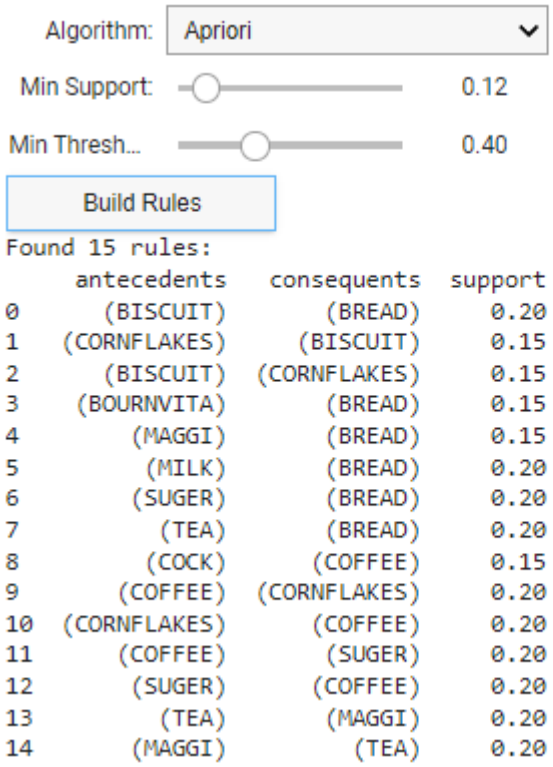
5. Підготуємо логіку для використання алгоритмів пошуку асоціативних правил Apriori та FP-Growth.

```
# Віджет для вибору алгоритму асоціативних правил
algorithm_widget = widgets.Dropdown(
    options=['Apriori', 'FP-Growth'],
    description='Algorithm:',
)

# Віджет для встановлення мінімальної підтримки
# % появи об'єкту у транзакціях
min_support_widget = widgets.FloatSlider(
    value=0.05,
    min=0.01,
    max=1,
    step=0.01,
    description='Min Support:',
)

# Віджет для вибору продукту
product_widget = widgets.Dropdown(options=te.columns_, description='Select
Product:')
recommend_button = widgets.Button(description="Recommend")
recommendation_output = widgets.Output()

min_threshold_widget = widgets.FloatSlider(
    value=0.2,
    min=0.1,
    max=1,
    step=0.1,
    description='Min Threshold:',
)

build_button = widgets.Button(description="Build Rules")

output = widgets.Output()

def visualize_rules(rules):
    G = nx.DiGraph()
    labels_map = {}
    label_counter = 1

    for index, row in rules.iterrows():
        antecedents = tuple(row['antecedents'])
```

```
        consequents = tuple(row['consequents'])
        confidence = row['confidence']

        for antecedent in antecedents:
            if antecedent not in labels_map:
                labels_map[antecedent] = str(label_counter)
                label_counter += 1
            for consequent in consequents:
                if consequent not in labels_map:
                    labels_map[consequent] = str(label_counter)
                    label_counter += 1
                G.add_edge(labels_map[antecedent], labels_map[consequent],
confidence=f"{confidence:.2f}")

    pos = nx.spring_layout(G)
    edges = G.edges(data=True)
    nx.draw_networkx_nodes(G, pos, node_size=700)
    nx.draw_networkx_edges(G, pos, edgelist=edges, edge_color='black')
    nx.draw_networkx_labels(G, pos)
    edge_labels = dict(((u, v), d['confidence']) for u, v, d in edges)
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
    plt.title('Association Rules')
    plt.axis('off')
    plt.show()

    print("Label mapping:")
    for item, label in labels_map.items():
        print(f"{label}: {item}")

def build_rules(button):
    global rules
    with output:
        output.clear_output()
        min_support = min_support_widget.value
        min_threshold = min_threshold_widget.value
        algorithm = algorithm_widget.value

        if algorithm == 'Apriori':
            frequent_itemsets = apriori(df_transformed,
min_support=min_support, use_colnames=True)
        elif algorithm == 'FP-Growth':
            frequent_itemsets = fpgrowth(df_transformed,
min_support=min_support, use_colnames=True)

        rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=min_threshold)

        if not rules.empty:
            print(f"Found {len(rules)} rules:")
            print(rules[['antecedents', 'consequents', 'support',
'confidence', 'lift']])
```

```
        else:
            print("No rules found with the given parameters.")

build_button.on_click(build_rules)

display(widgets.VBox([algorithm_widget, min_support_widget,
min_threshold_widget, build_button, output]))
```

Використаємо алгоритм пошуку асоціативних правил Apriori з параметрами Min Support = 0.12, Min threshold = 0.40.



Found 15 rules:

	antecedents	consequents	support	confidence	lift
0	(BISCUIT)	(BREAD)	0.20	0.571429	0.879121
1	(CORNFLAKES)	(BISCUIT)	0.15	0.500000	1.428571
2	(BISCUIT)	(CORNFLAKES)	0.15	0.428571	1.428571
3	(BOURNVITA)	(BREAD)	0.15	0.750000	1.153846
4	(MAGGI)	(BREAD)	0.15	0.600000	0.923077
5	(MILK)	(BREAD)	0.20	0.800000	1.230769
6	(SUGER)	(BREAD)	0.20	0.666667	1.025641
7	(TEA)	(BREAD)	0.20	0.571429	0.879121
8	(COCK)	(COFFEE)	0.15	1.000000	2.500000
9	(COFFEE)	(CORNFLAKES)	0.20	0.500000	1.666667
10	(CORNFLAKES)	(COFFEE)	0.20	0.666667	1.666667
11	(COFFEE)	(SUGER)	0.20	0.500000	1.666667
12	(SUGER)	(COFFEE)	0.20	0.666667	1.666667
13	(TEA)	(MAGGI)	0.20	0.571429	2.285714
14	(MAGGI)	(TEA)	0.20	0.800000	2.285714

Рисунок 8.5. Результат пошуку асоціативних правил за алгоритмом Apriori

Використаємо алгоритм пошуку асоціативних правил FP-Growth з параметрами Min Support = 0.15, Min threshold = 0.60.

Algorithm: FP-Growth

Min Support: 0.15

Min Thresh... 0.60

Build Rules

Found 8 rules:

	antecedents	consequents	support	confidence	lift
0	(MILK)	(BREAD)	0.20	0.800000	1.230769
1	(CORNFLAKES)	(COFFEE)	0.20	0.666667	1.666667
2	(BOURNVITA)	(BREAD)	0.15	0.750000	1.153846
3	(MAGGI)	(BREAD)	0.15	0.600000	0.923077
4	(MAGGI)	(TEA)	0.20	0.800000	2.285714
5	(COCK)	(COFFEE)	0.15	1.000000	2.500000
6	(SUGER)	(COFFEE)	0.20	0.666667	1.666667
7	(SUGER)	(BREAD)	0.20	0.666667	1.025641

Рисунок 8.6. Результат пошуку асоціативних правил за алгоритмом FP-Growth

6. Виконаємо побудову графів для Apriori та FP-Growth

```
visualize_rules(rules)
```


```
Label mapping:
1: BISCUIT
2: BREAD
3: CORNFLAKES
4: BOURNVITA
5: MAGGI
6: MILK
7: SUGER
8: TEA
9: COCK
10: COFFEE
```

Рисунок 8.7. Граф асоціативних правил для Apriori

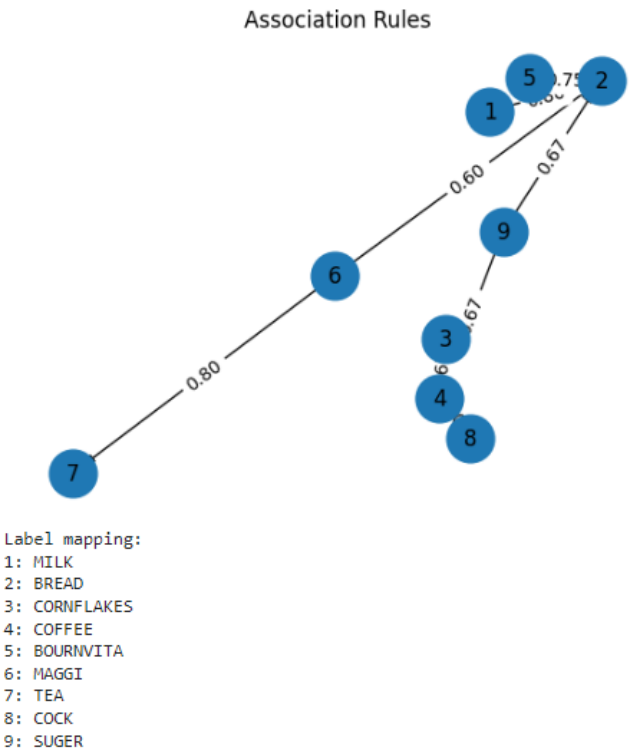


Рисунок 8.8. Граф асоціативних правил для FP-Growth

7. Виконаємо процес надання рекомендацій відповідно до моделі Apriori та FP-Growth

```
# Функція для отримання рекомендацій
def get_recommendations(button):
    with recommendation_output:
        recommendation_output.clear_output()
        selected_product = product_widget.value
        recommendations_set = set()

        recommended = rules[rules['antecedents'].apply(lambda x:
selected_product in x)]['consequents']

        if not recommended.empty:
            print(f"Recommendations for {selected_product}:")
            for index, item in recommended.iteritems():
                recommendations_set.update(item)

            for item in recommendations_set:
                print(f"- {item}")
        else:
            print(f"No recommendations found for {selected_product}")

recommend_button.on_click(get_recommendations)

display(widgets.VBox([product_widget, recommend_button,
recommendation_output]))
```

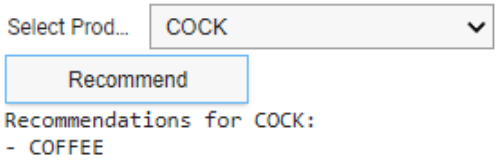


Рисунок 8.9. Рекомендації моделі Apriori для «COCK»

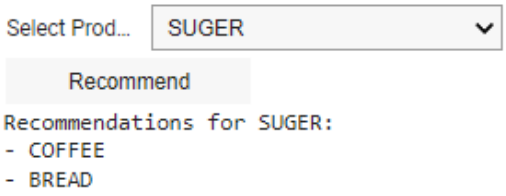


Рисунок 8.10. Рекомендації моделі FP-Growth для «COCK»

Таким чином, аналізуючи отримані дані, можна зазначити, що використання моделі SVM продемонструвало найкращі результати за показником Accuracy (0.9362). Моделі Decision Trees (0.7021) та Logistic Regression (0.7234) надали близькі один до одного результати за показником Accuracy, що є достатнім та дозволяє виконувати задачу класифікації для даного набору даних. Проте зі стандартним налаштуванням гіперпараметрів модель Decision Trees надала некоректний результат класифікації на довільному прикладі. При збільшенні значення гіперпараметрів, модель почала надавати більшу точність (0.8085) та правильно класифікувала приклад. Що стосується моделей K-NN (0.5106) та Naïve Bayes (0.5745), то їх показники точності є незадовільними та не дозволяють якісно виконувати класифікацію на заданому наборі даних, потребують доопрацювання.

Контрольне запитання №1

Вкажіть максимальне значення support для Apriori.
Відповідь: **0.20**

Контрольне запитання №2

Вкажіть максимальне значення confidence для FP-Growth.
Відповідь: **1.00**

Контрольне запитання №3

Вкажіть кількість виявлених правил при support = 0.10; confidence = 0.3.
Відповідь: **90**

Висновки: …

Завдання для виконання лабораторної роботи

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.

Посилання на датасети					
1	6	11	16	21	26
2	7	12	17	22	27
3	8	13	18	23	28
4	9	14	19	24	29
5	10	15	20	25	30

2. Виконайте згрупування даних за транзакціями та перетворіть транзакції для подальшого аналізу.
3. Використайте алгоритми пошуку асоціативних правил: 1) Apriori; 2) FP-Growth. Виконайте оцінку та візуалізуйте результати у вигляді графу.
4. Сформуйте рекомендації на основні виявлених асоціативних правил.
5. Проаналізуйте отримані результати.

Контрольне запитання №1

Вкажіть максимальне значення support для Apriori.
Відповідь: _____

Контрольне запитання №2

Вкажіть максимальне значення confidence для FP-Growth.
Відповідь: _____

Контрольне запитання №3

Вкажіть кількість виявлених правил при support = 0.10; confidence = 0.3.
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1fK7wWaaifxKbJrOUftnDuYB88BfnqdIg?usp=sharing

Лабораторна робота № 9. Дослідження методів узагальнення: LDA, SVD (LSA), PCA, t-SNE та ансамблевих методів: Random Forest, XGBoost, LightGBM, CatBoost, AdaBoost

Мета: сформулювати практичні навички у використанні та порівнянні методів узагальнення даних та ансамблевих методів машинного навчання, включаючи LDA, SVD (LSA), PCA, t-SNE, Random Forest, XGBoost, LightGBM, CatBoost та AdaBoost. Розвинути здатність аналізувати великі набори даних, визначати найбільш ефективні методи для конкретних завдань класифікації або редукції розмірності, а також опанувати навички налаштування гіперпараметрів для покращення результатів моделі.

Теоретичні відомості

Методи узагальнення даних, також відомі як методи редукції розмірності, використовуються для зменшення кількості вхідних змінних у наборі даних, шукаючи новий набір ознак, який зберігає найважливішу інформацію з оригінального набору. Ці методи дозволяють спростити моделі, зменшити обчислювальні витрати та поліпшити інтерпретацію даних, зберігаючи при цьому якомога більше корисної інформації.

Приклади методів узагальнення:

1. **PCA (Principal Component Analysis)** – метод лінійної редукції розмірності, який використовується для зменшення кількості змінних у датасеті, зберігаючи при цьому якомога більше інформації. PCA шукає новий, менш розмірний простір для представлення даних.
2. **t-SNE (t-Distributed Stochastic Neighbor Embedding)** – нелінійний метод зниження розмірності, особливо ефективний для візуалізації великих наборів високорозмірних даних.
3. **LDA (Linear Discriminant Analysis)** – метод, що використовується для знаходження лінійних комбінацій ознак, які найкраще розділяють два або більше класів об’єктів або подій.
4. **SVD (Singular Value Decomposition)** та **LSA (Latent Semantic Analysis)** – **SVD** є математичним методом, який дозволяє розкладати матрицю на три інші матриці, а **LSA** використовує **SVD** для аналізу відносин між набором документів та термінами, що містяться в них, для виявлення семантичних структур.

Ансамблеві методи в машинному навчанні – це підходи, що комбінують прогнози з кількох моделей навчання для покращення загальної точності, надійності та ефективності моделі. Ансамблеві методи базуються на принципі, що група «слабких» моделей, які працюють разом, може перевершити одну «сильну» модель.

Приклади ансамблевих методів:

- **Random Forest** – ансамблевий метод, що будує багато дерев рішень під час тренування та виводить клас, що є модою класів (класифікація) або середнє прогнозоване деревами (регресія).
- **XGBoost (Extreme Gradient Boosting)** – оптимізований алгоритм градієнтного бустингу, що використовується для збільшення швидкості та ефективності обчислень.
- **LightGBM** – швидкий, ефективний алгоритм градієнтного бустингу, який використовує алгоритми розбиття на основі гістограм для обробки великих обсягів даних.
- **CatBoost** – алгоритм, який автоматично обробляє категорійні змінні та забезпечує високу точність прогнозування.
- **AdaBoost (Adaptive Boosting)** – алгоритм, що послідовно виправляє помилки попередніх класифікаторів та надає їм ваги для формування сильного класифікатора.

Приклад виконання роботи

Завдання

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте ознаки, які будуть використовуватися у методах узагальнення та ансамблевих методах. Як цільову зміну класифікації використовуйте клас заданий у таблиці.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується.
4. Використайте алгоритми узагальнення: 1) PCA; 2) t-SNE; 3) LDA; 4) SVD (LSA) та ансамблеві методи: 1) Random Forest; 2) XGBoost; 3) LightGBM; 4) CatBoost; 5) AdaBoost. Виконайте оцінку та візуалізуйте результати для кожної моделі.
5. Виконайте класифікацію за допомогою різних ансамблевих моделей для довільних вхідних даних відповідно до вашої цільової змінної.
6. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
7. За бажанням проекспериментуйте з гіперпараметрами для моделей задля покращення результатів.

Контрольне запитання №1

Вкажіть результуюче значення Accuracy для Random Forest.
Відповідь: _____

Контрольне запитання №2

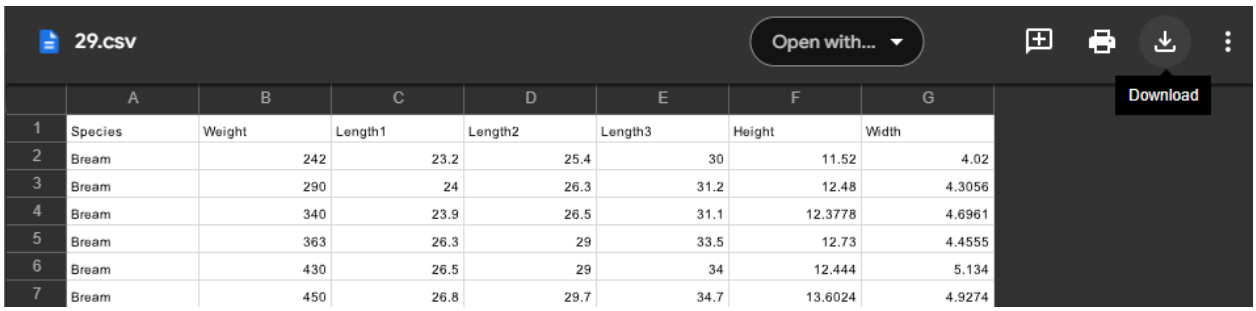
Вкажіть результуюче значення Accuracy для CatBoost.
Відповідь: _____

Контрольне запитання №3

Вкажіть результуюче значення Accuracy для LightGBM.
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набір даних за варіантом.



	A	B	C	D	E	F	G
1	Species	Weight	Length1	Length2	Length3	Height	Width
2	Bream	242	23.2	25.4	30	11.52	4.02
3	Bream	290	24	26.3	31.2	12.48	4.3056
4	Bream	340	23.9	26.5	31.1	12.3778	4.6961
5	Bream	363	26.3	29	33.5	12.73	4.4555
6	Bream	430	26.5	29	34	12.444	5.134
7	Bream	450	26.8	29.7	34.7	13.6024	4.9274

Рисунок 9.1. Датасет на Google Drive

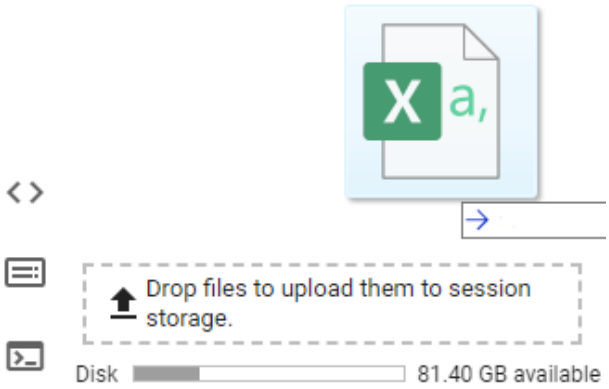


Рисунок 9.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.manifold import TSNE
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import ipywidgets as widgets
from IPython.display import display, clear_output
from sklearn.preprocessing import LabelEncoder
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
```

3. Ініціалізація набору даних в Google Colab.

```
# Завдання №1. # Завантаження датасету
data = pd.read_csv("/content/29.csv")
data.head()
```

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

Рисунок 9.3. Перші 5 рядків набору даних

4. Визначимо цільову змінну, як Species, ознаками будуть: Weight, Length1, Length2, Length3, Height, Width.

5. Виконаємо очищення даних

```
# Перевірка на відсутні значення
print(data.isnull().sum())
```

```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

Рисунок 9.4. Пошук нульових значень у датасеті

Порожні значення відсутні, тож видалення або заповнення виконувати не треба.

```
# Видалення або заповнення відсутніх значень
# data.dropna(inplace=True) # Видалення

# data.fillna(data.mean(), inplace=True) # Заповнення середніми значеннями
```

Виконаємо візуалізацію однієї з ознак набору даних, а саме «Length2»

```
# Візуалізація викидів
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Length2'])
plt.title('Boxplot before removing outliers')
plt.show()
```

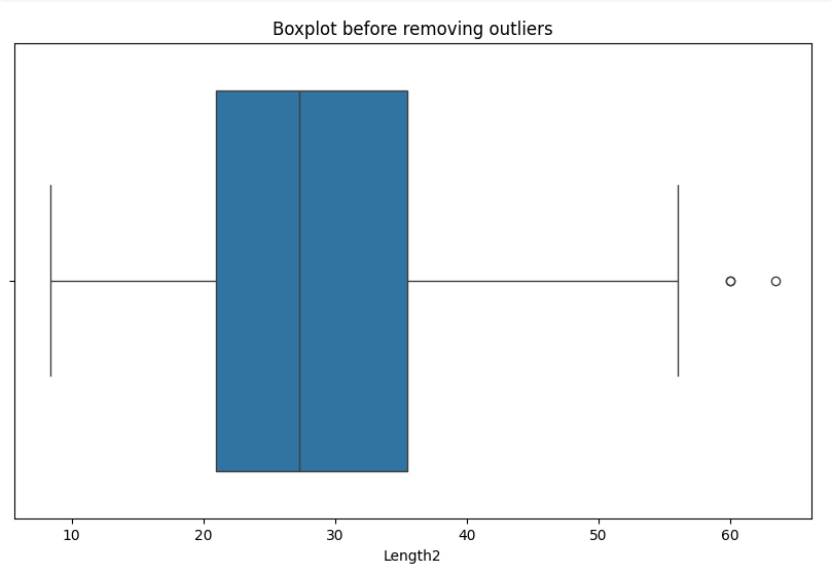


Рисунок 9.5. Діаграма викидів для Length2 перед очищенням

Виконаємо очищення викидів для «Length2».

```
# Видалення викидів
feature_processing = 'Length2'
q_low = data[feature_processing].quantile(0.01)
q_hi  = data[feature_processing].quantile(0.99)
data = data[(data[feature_processing] < q_hi) & (data[feature_processing] >
q_low)]
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Length2'])
plt.title('Boxplot after removing outliers')
plt.show()
```

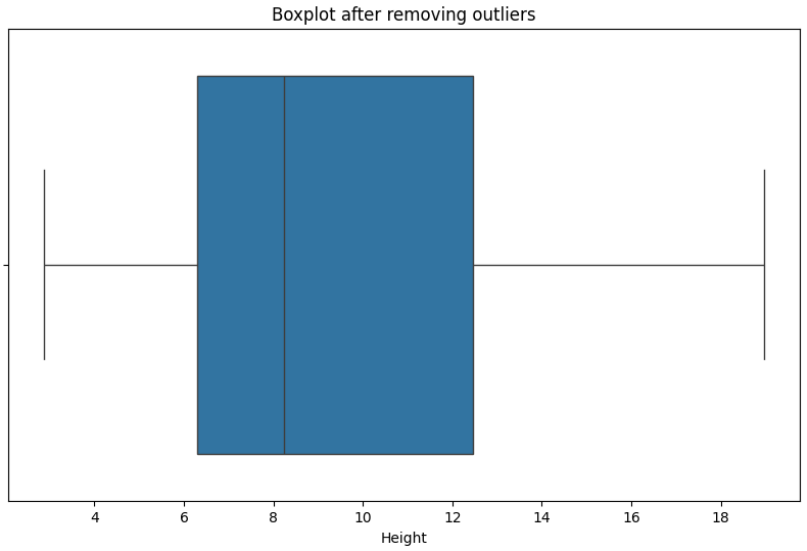


Рисунок 9.6. Діаграма викидів для Length2 після очищення

Подібні операції виконаємо для інших ознак.

6. Розподілимо дані на тренувальні та тестові набори з урахуванням ознак та цільової змінної. В тестовому наборі даних видалимо стовпець «Species», що класифікується та виконаємо розподіл тестових та тренувальних наборів даних. Водночас виконаємо перетворення міток класів у числовий формат за допомогою LabelEncoding.

```
X = data.drop('Species', axis=1)
y = data['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

Виконаємо нормалізацію даних.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

7. Підготуємо та використаємо алгоритми узагальнення PCA, t-SNE, LDA та SVD (LSA), а також ансамблеві методи Random Forest, XGBoost, LightGBM, CatBoost та Adaboost.

```
model_widget = widgets.Dropdown(
    options=['PCA', 't-SNE', 'LDA', 'SVD (LSA)', 'Random Forest', 'XGBoost',
'LightGBM', 'CatBoost', 'AdaBoost'],
    description='Model:',
)

# Віджети для налаштування гіперпараметрів
hyperparameter_widgets = {
    'n_components': widgets.IntSlider(min=2, max=5, value=2,
description='n_components:'),
    'max_depth': widgets.IntSlider(min=1, max=10, value=3, description='Max
Depth:'),
    'learning_rate': widgets.FloatSlider(min=0.01, max=0.3, value=0.1,
step=0.01, description='Learning Rate:'),
    'n_estimators': widgets.IntSlider(min=10, max=200, value=100,
description='N Estimators:'),
    'silent': widgets.Checkbox(value=True, description='Silent:'),
}

hyperparameters_ui = widgets.VBox([])

def update_hyperparameters_ui(*args):
    model_name = model_widget.value
    children = [model_widget]
```

```
    if model_name in ['Random Forest', 'XGBoost', 'LightGBM', 'CatBoost',
'AdaBoost']:
        children.append(widgets.IntSlider(min=10, max=200, value=100,
description='n_estimators:'))
        children.append(widgets.IntSlider(min=1, max=10, value=3,
description='max_depth:'))
    if model_name in ['XGBoost', 'LightGBM', 'CatBoost']:
        children.append(widgets.FloatSlider(min=0.01, max=0.3, value=0.1,
step=0.01, description='learning_rate:'))
    if model_name == 'CatBoost':
        children.append(widgets.Checkbox(value=False,
description='Silent:'))
    hyperparameters_ui.children = children

model_widget.observe(update_hyperparameters_ui, 'value')

output = widgets.Output()

def train_and_visualize(button):
    with output:
        output.clear_output(wait=True)
        model_name = model_widget.value
        n_components = hyperparameter_widgets['n_components'].value
        max_depth = hyperparameter_widgets['max_depth'].value
        learning_rate = hyperparameter_widgets['learning_rate'].value
        n_estimators = hyperparameter_widgets['n_estimators'].value
        silent = hyperparameter_widgets['silent'].value

        global y_train, y_test
        if model_name in ['PCA', 't-SNE', 'LDA', 'SVD (LSA)']:
            if model_name == 'PCA':
                model = PCA(n_components=n_components)
            elif model_name == 't-SNE':
                model = TSNE(n_components=n_components)
            elif model_name == 'LDA':
                model = LDA(n_components=n_components)
            elif model_name == 'SVD (LSA)':
                model = TruncatedSVD(n_components=n_components)

            X_transformed = model.fit_transform(X_train_scaled, y_train)
            plt.figure(figsize=(10, 6))
            plt.scatter(X_transformed[:, 0], X_transformed[:, 1],
c=y_train_encoded)
            plt.title(f'{model_name} visualization')
            plt.colorbar()
            plt.show()
        else:
            if model_name == 'Random Forest':
                model = RandomForestClassifier(n_estimators=n_estimators,
max_depth=max_depth)
            elif model_name == 'XGBoost':
                model = XGBClassifier(n_estimators=n_estimators,
max_depth=max_depth, learning_rate=learning_rate)
```

```
            elif model_name == 'LightGBM':
                model = LGBMClassifier(verbose= -1,
n_estimators=n_estimators, max_depth=max_depth, learning_rate=learning_rate)
            elif model_name == 'CatBoost':
                model = CatBoostClassifier(n_estimators=n_estimators,
max_depth=max_depth, learning_rate=learning_rate, silent=silent)
            elif model_name == 'AdaBoost':
                model = AdaBoostClassifier(n_estimators=n_estimators)

            model.fit(X_train_scaled, y_train_encoded)
            y_pred_encoded = model.predict(X_test_scaled)
            accuracy = accuracy_score(y_test_encoded, y_pred_encoded)
            cm = confusion_matrix(y_test_encoded, y_pred_encoded)
            print(f"Model: {model_name}, Accuracy: {accuracy:.4f}")
            sns.heatmap(cm, annot=True, fmt="d")
            plt.title(f'Confusion Matrix: {model_name}')
            plt.show()

train_button = widgets.Button(description="Train and Visualize")
train_button.on_click(train_and_visualize)

hyperparameter_ui = widgets.VBox([v for k, v in
hyperparameter_widgets.items()])
ui = widgets.VBox([model_widget, hyperparameter_ui, train_button, output])

display(ui)
```

8. Виконаємо тестування алгоритмів узагальнення та ансамблевих методів

Використаємо алгоритм узагальнення PCA з показником «n_components» = 2.

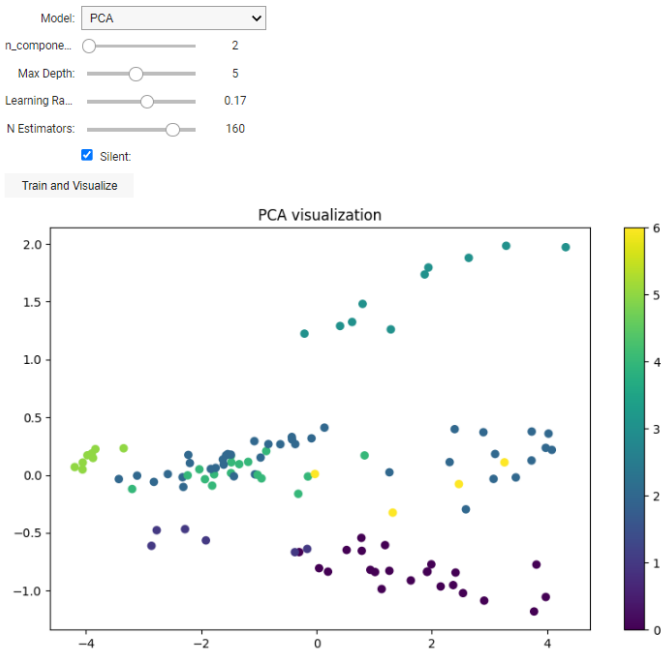


Рисунок 9.7. Результат використання моделі PCA

Використаємо алгоритм узагальнення t-SNE з показником «n_components» = 2.

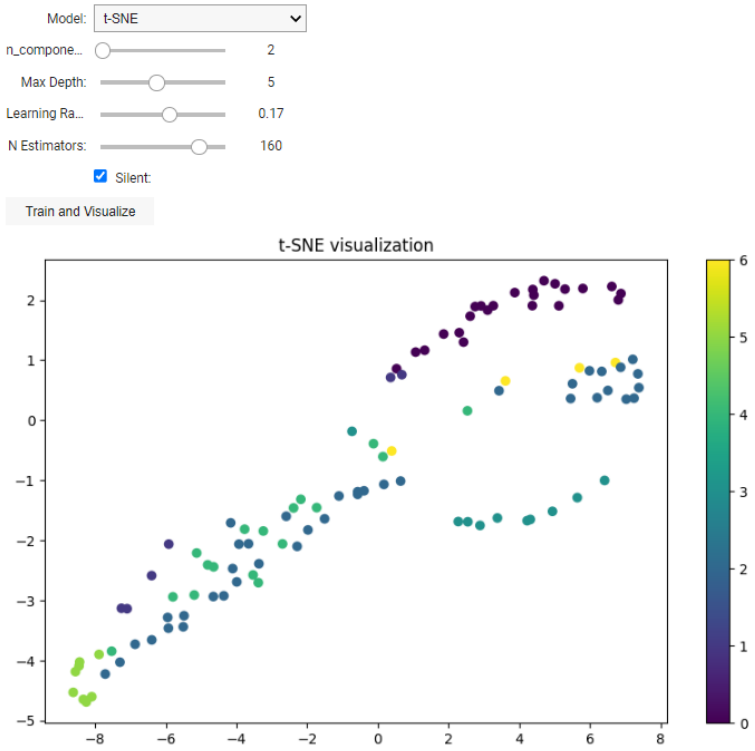


Рисунок 9.8. Результат використання моделі t-SNE

Використаємо алгоритм узагальнення LDA з показником «n_components» = 2.

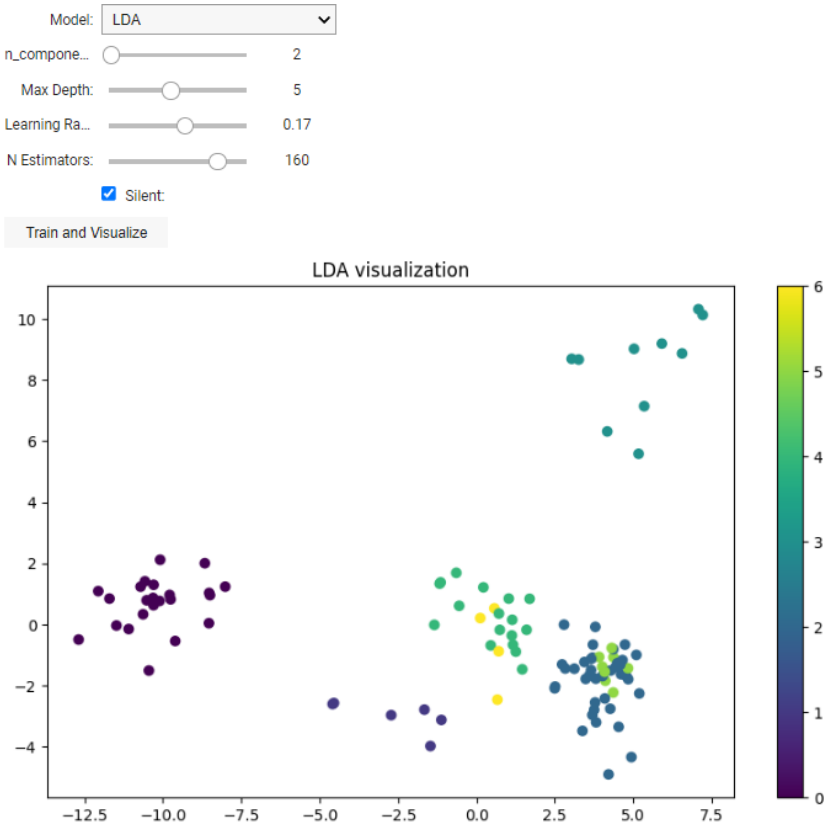


Рисунок 9.9. Результат використання моделі LDA

Використаємо алгоритм узагальнення SVD (LSA) з показником «n_components» = 2.

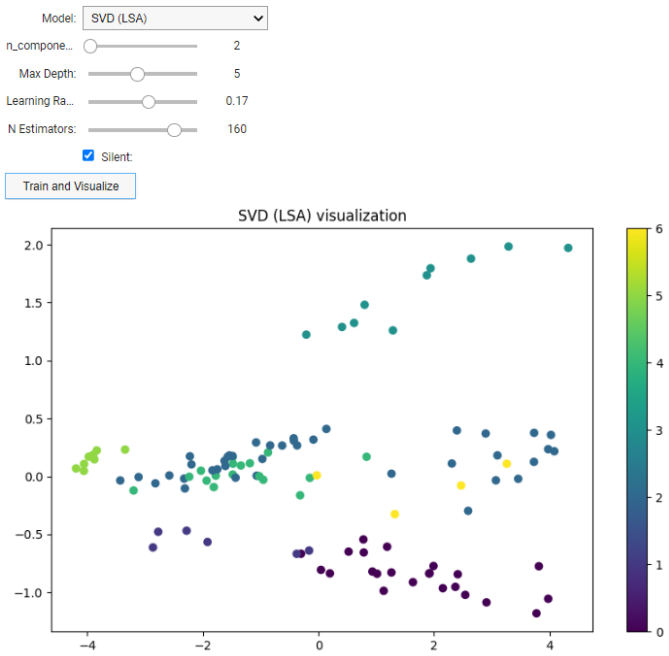


Рисунок 9.10. Результат використання моделі SVD (LSA)

Використаємо ансамблевий метод Random Forest з показниками «n_estimators» = 160, «max_depth» = 5.

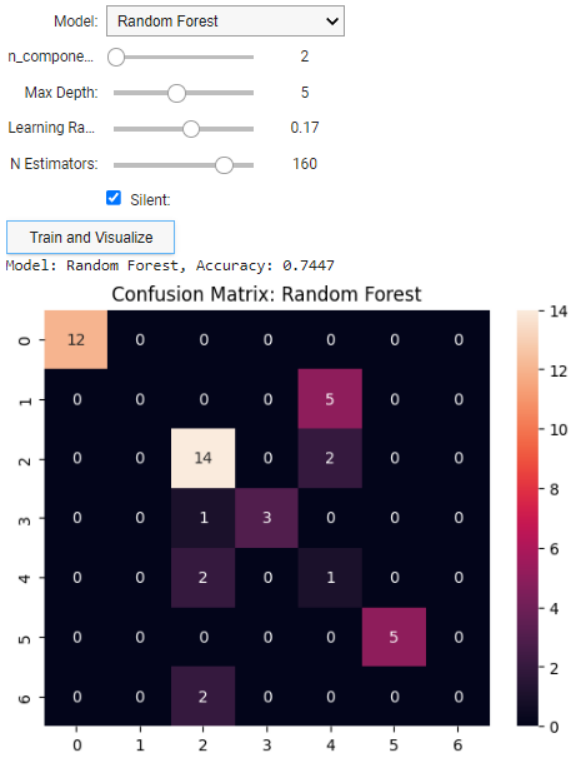


Рисунок 9.11. Результат використання ансамблевої моделі Random Forest

Використаємо ансамблевий метод XGBoost з показниками «n_estimators» = 160, «max_depth» = 5, «learning_rate» = 0.17.

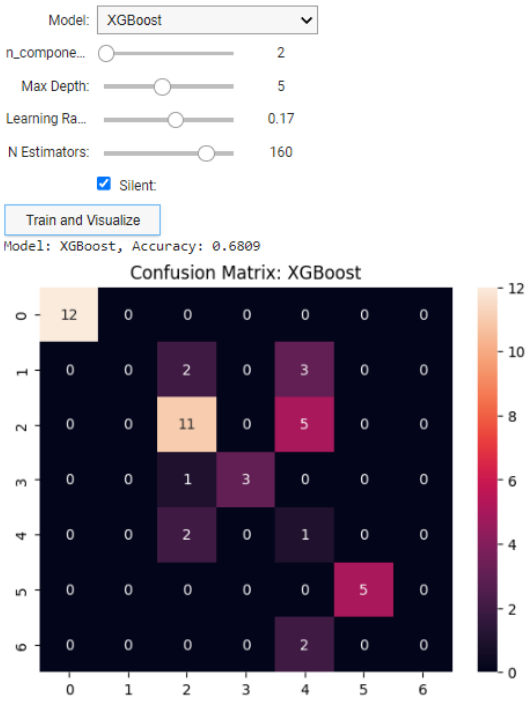


Рисунок 9.12. Результат використання ансамблевої моделі GXBoost

Використаємо ансамблевий метод LightGBM з показниками «n_estimators» = 160, «max_depth» = 5, «learning_rate» = 0.17.

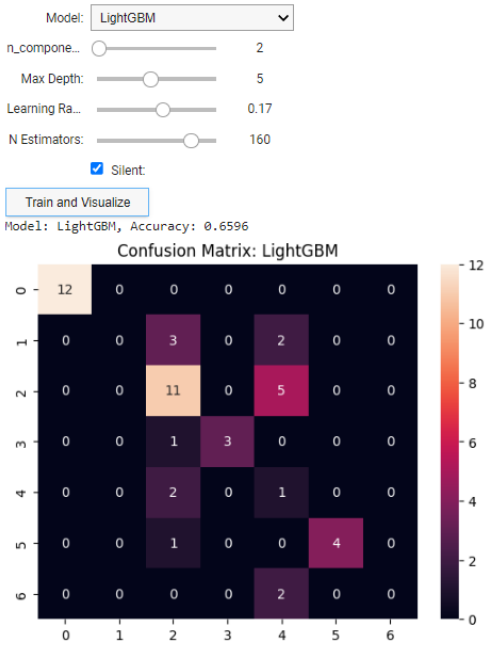


Рисунок 9.13. Результат використання ансамблевої моделі LightGBM

Використаємо ансамблевий метод CatBoost з показниками «n_estimators» = 160, «max_depth» = 5, «learning_rate» = 0.17, «silent» = True.

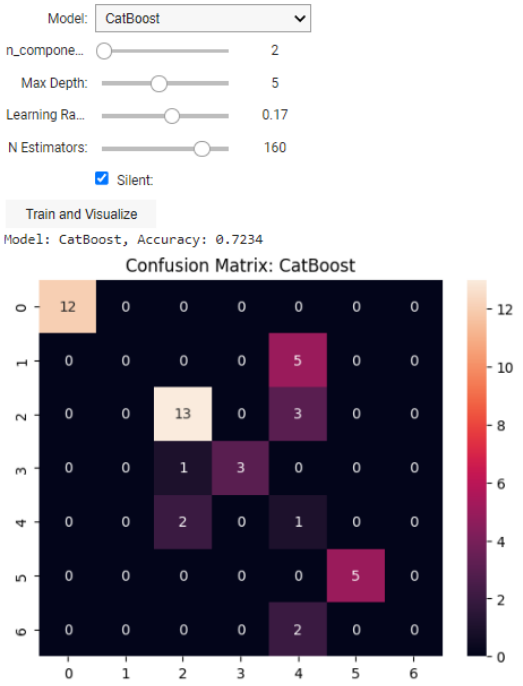


Рисунок 9.14. Результат використання ансамблевої моделі CatBoost

Використаємо ансамблевий метод AdaBoost з показниками «n_estimators» = 160.

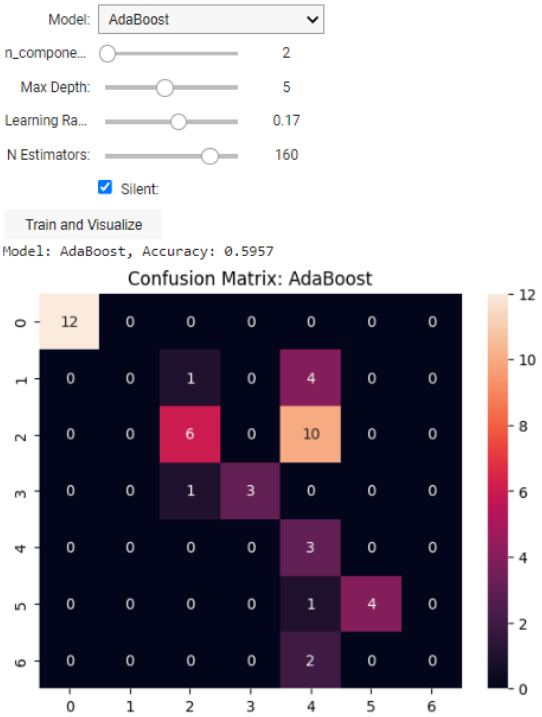


Рисунок 9.15. Результат використання ансамблевої моделі AdaBoost

9. Виконаємо прогнозування на довільних даних

```
model_widget = widgets.Dropdown(
    options=['Random Forest', 'XGBoost', 'LightGBM', 'CatBoost',
'AdaBoost'],
    description='Model:',
)

n_estimators_widget = widgets.IntSlider(min=10, max=200, value=100,
description='n_estimators:')
max_depth_widget = widgets.IntSlider(min=1, max=10, value=3,
description='max_depth:')
learning_rate_widget = widgets.FloatSlider(min=0.01, max=0.3, value=0.1,
step=0.01, description='learning_rate:')


hyperparameter_widgets = widgets.VBox([n_estimators_widget,
max_depth_widget, learning_rate_widget])

input_widgets = [widgets.FloatText(description=f'Feature {i+1}:') for i in
range(X_train.shape[1])]

classify_button = widgets.Button(description="Classify")
output = widgets.Output()

def classify(b):
    with output:
        clear_output()
        data_to_classify = [w.value for w in input_widgets]
        model = prepare_model(
            model_choice=model_widget.value,
            n_estimators=n_estimators_widget.value,
            max_depth=max_depth_widget.value,
            learning_rate=learning_rate_widget.value,
        )
        prediction = model.predict([data_to_classify])
        print(f"Predicted class: {prediction[0]}")
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        print(f"Accuracy: {accuracy:.4f}")

classify_button.on_click(classify)

def prepare_model(model_choice, n_estimators, max_depth, learning_rate):
    if model_choice == 'Random Forest':
        model = RandomForestClassifier(n_estimators=n_estimators,
max_depth=max_depth)
    elif model_choice == 'XGBoost':
        model = XGBClassifier(n_estimators=n_estimators,
max_depth=max_depth, learning_rate=learning_rate)
    elif model_choice == 'LightGBM':
        model = LGBMClassifier(verbose= -1, n_estimators=n_estimators,
max_depth=max_depth, learning_rate=learning_rate)
```

```
    elif model_choice == 'CatBoost':
        model = CatBoostClassifier(n_estimators=n_estimators,
depth=max_depth, learning_rate=learning_rate, silent=True)
    elif model_choice == 'AdaBoost':
        model = AdaBoostClassifier(n_estimators=n_estimators,
learning_rate=learning_rate)
    model.fit(X_train, y_train)
    return model

display(widgets.VBox([model_widget, hyperparameter_widgets] + input_widgets
+ [classify_button, output]))
```

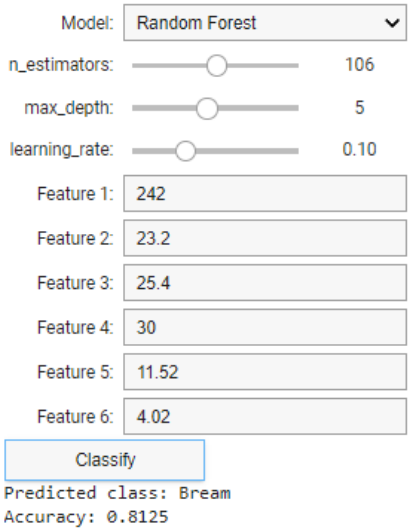


Рисунок 9.16. Результати прогнозування для моделі Random Forest

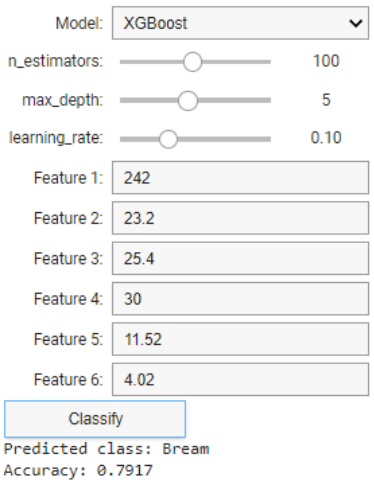


Рисунок 9.17. Результати прогнозування для моделі XGBoost

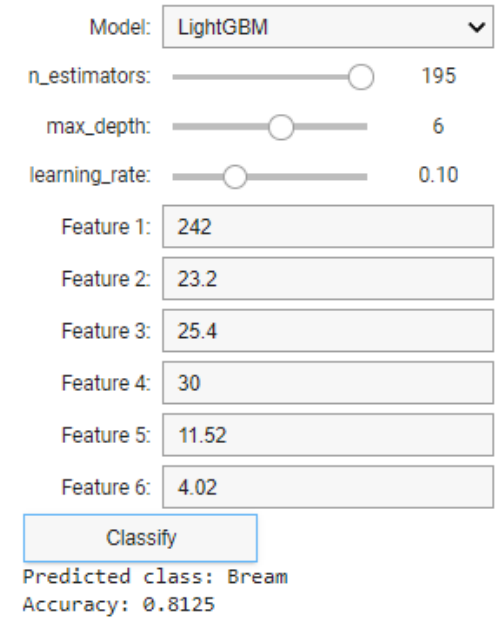


Рисунок 9.18. Результати прогнозування для моделі LightGBM

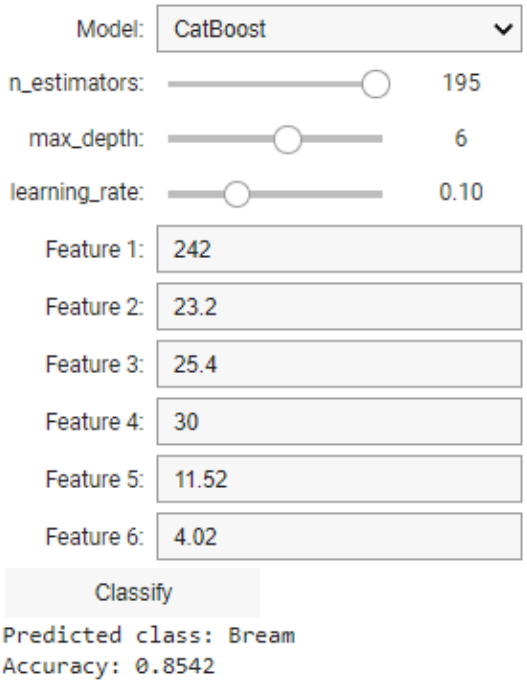


Рисунок 9.19. Результати прогнозування для моделі CatBoost

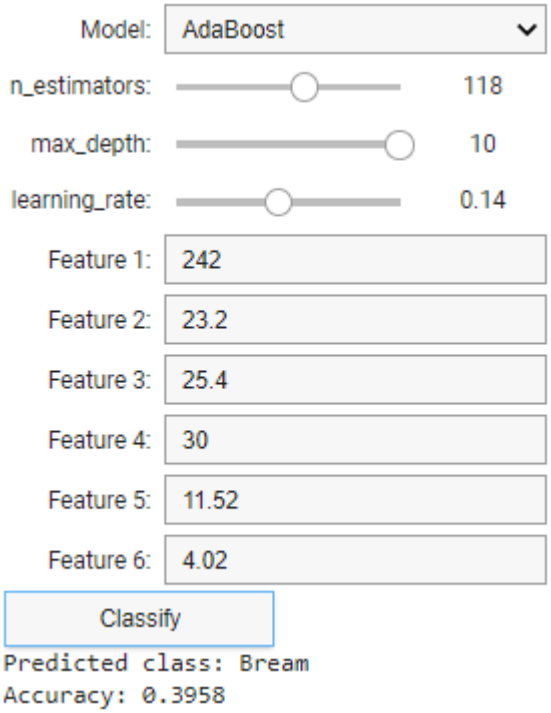


Рисунок 9.20. Результати прогнозування для моделі AdaBoost

Таким чином, …

Контрольне запитання №1

Вкажіть результуюче значення Accuracy для Random Forest.
Відповідь: **0.7447**

Контрольне запитання №2

Вкажіть результуюче значення Accuracy для CatBoost.
Відповідь: **0.7234**

Контрольне запитання №3

Вкажіть результуюче значення Accuracy для LightGBM.
Відповідь: **0.6596**

Висновки: …

Завдання для виконання лабораторної роботи

1. Оберіть та завантажте датасет відповідно до свого варіанту. Завантажте ваш набір даних як тимчасовий файл у середовище Google Colab та ініціалізуйте його.
2. Проаналізуйте датасет та визначте ознаки, які будуть використовуватися у методах узагальнення та ансамблевих методах. Як цільову зміну класифікації використовуйте клас заданий у таблиці.
3. Виконайте очищення даних: видаліть або заповніть пропущені значення, опрацюйте викиди якщо це потребується.
4. Використайте алгоритми узагальнення: 1) PCA; 2) t-SNE; 3) LDA; 4) SVD (LSA) та ансамблеві методи: 1) Random Forest; 2) XGBoost; 3) LightGBM; 4) CatBoost; 5) AdaBoost. Виконайте оцінку та візуалізуйте результати для кожної моделі.
5. Виконайте класифікацію за допомогою різних ансамблевих моделей для довільних вхідних даних відповідно до вашої цільової змінної.
6. Проаналізуйте отримані результати, обґрунтуйте вибір найкращої моделі для даного набору даних.
7. За бажанням проекспериментуйте з гіперпараметрами для моделей задля покращення результатів.

	Посилання на датасети				
№	**1**	**5**	**9**	**13**	**17**
Classification	Brand	label	Microcalcification	category	wifi
№	**2**	**6**	**10**	**14**	**18**
Classification	class	Type	diabetes	R	is_safe
№	**3**	**7**	**11**	**15**	**19**
Classification	POP	Species	Car	class	quality
№	**4**	**8**	**12**	**16**	**20**
Classification	ocean_proximity	CarName	poutcome	custcat	Species

Контрольне запитання №1

Вкажіть результуюче значення Accuracy для Random Forest.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Accuracy для CatBoost.
Відповідь: _____

Контрольне запитання №3

Вкажіть результуюче значення Accuracy для LightGBM.

Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1spYcnLtT7DdbErQLG7PsTZHftc02Dpq7?usp=sharing

Лабораторна робота № 10. Моделювання нейронної мережі. Розв’язання класичної задачі «Титанік»

Мета: сформулювати практичні навички розробки та тренування нейронних мереж за допомогою сучасних бібліотек глибокого навчання. Робота спрямована на розвиток вмінь аналізувати та обробляти реальні набори даних, виконувати інженерію ознак, підготовку даних до тренування моделі, а також оптимізацію гіперпараметрів для досягнення найкращої можливої точності прогнозування. Ця робота також має на меті розвинути розуміння процесів, які лежать в основі глибокого навчання, та показати, як ці технології можуть бути застосовані для розв’язання практичних завдань.

Теоретичні відомості

Глибоке навчання є підгалуззю **машинного навчання**, яке використовує нейронні мережі з численними прихованими шарами для моделювання складних абстракцій. Воно знайшло широке застосування у багатьох областях, таких як комп'ютерний зір, обробка природної мови, рекомендаційні системи та інші.

Нейронні мережі імітують роботу людського мозку і складаються з великої кількості вузлів, званих нейронами, які з’єднані між собою зв'язками. Кожен зв’язок має вагу, що регулює вплив одного нейрона на інший. Навчання нейронної мережі полягає у визначенні оптимальних значень цих ваг на основі навчальних даних.

Задача «Титанік» є класичним прикладом бінарної класифікації, де необхідно на основі набору характеристик пасажирів (вік, стать, клас каюти та інші) передбачити, чи вижив пасажир під час катастрофи. Ця задача вимагає від дослідника виконання кількох ключових етапів: попередньої обробки даних, включаючи заповнення пропущених значень та кодування категоріальних ознак; інженерії ознак для створення нових змінних, які можуть покращити модель; тренування моделі, використовуючи нейронні мережі; а також оцінку точності моделі на тестовому наборі даних.

Для ефективного тренування моделі важливо вибрати відповідні гіперпараметри, такі як швидкість навчання, кількість епох, розмір партії та архітектура мережі (кількість шарів і нейронів в кожному шарі). Оптимізація гіперпараметрів може значно покращити продуктивність моделі.

Приклад виконання роботи

Завдання

Титанік – британський пасажирський лайнер компанії White Star Line, який затонув у північній частині Атлантичного океану рано вранці 15 квітня 1912 року після зіткнення з айсбергом під час свого першого рейсу з Саутгемптона до Нью-Йорка з приблизно 2224 пасажирами і членами екіпажу, які перебували на борту, понад 1500 загинули, що робить цю катастрофу однією з найсмертоносніших комерційних морських катастроф мирного часу в сучасній історії. Хоча на борту було близько 2224 пасажирів і членів екіпажу, ми маємо дані про 1300 пасажирів. З цих 1300, близько 900 даних використовуються з навчальною метою, а решта 400 – з тестовою. У цьому завдані вам надано близько 400 тестових даних з відсутнім стовпчиком виживших. Вам треба змоделювати модель нейронної мережі, щоб передбачити, чи вижили пасажири у тестових даних, чи ні. Як навчальні, так і тестові дані не є чистими (містять багато пропущених значень). Побудуйте модель з найкращою точністю.

Характеристика даних:

- Survival (виживання): 0 – ні, 1 – так;
- Pclass (номер класу): 1-й – вищий, 2-й – середній, 3-й – нижчий;
- sibsp (кількість братів і сестер / подружжя на борту): Sibling – брат, сестра, зведений брат, зведена сестра, Spouse – чоловік, дружина (коханки та наречені не враховуються);
- parch (кількість батьків / дітей на борту): Parent – мати, батько, Child – донька, син, падчерка, пасинок. Деякі діти подорожували лише з нянею, тому для них parch = 0;
- Ticket (квиток): номер квитка;
- Fare (вартість): пасажирський тариф;
- Cabin (каюта): номер каюти;
- Port of Embarkation (порт посадки): C – Cherbourg, Q – Queenstown, S – Southampton;
- Name (ім’я), Sex (стать), Age (вік).

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.

Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Precision.

Відповідь: _____

Контрольне запитання №3

Вкажіть яке значення Survived приймає PassengerId 900.

Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набори даних.

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen H	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mrs. John I	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, Miss. Lain	female	26	0	0	STON/O2. 3101282	7.925		S
5	4	1	1	Futrelle, Mrs. Jacque	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William H	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy, Mr. Timot	male	54	0	0	17463	51.8625	E46	S

Рисунок 10.1. Датасет на Google Drive

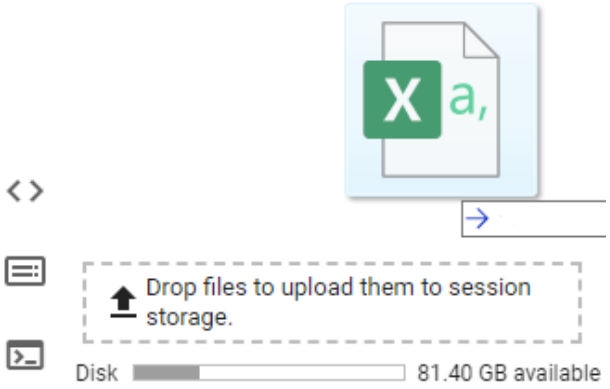


Рисунок 10.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів.

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf
```

```
import keras
from keras.layers import Dense, Dropout, Input
from keras.models import Sequential
```

3. Завантаження наборів даних.

```
train_data = pd.read_csv('/content/train.csv')
test_data = pd.read_csv('/content/test.csv')

train_data.head()
test_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Рисунок 10.3. Перші 5 рядків навчального набору даних

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Рисунок 10.4. Перші 5 рядків тестового набору даних

Ми бачимо, що загальна кількість стовпців у тестовому наборі даних на один менше, ніж у навчальному. Відсутній стовпець у тестових даних – це Survived (Вижившi) стовпець, який ми повинні передбачити.

```
print("Загальна кількість рядків у навчальному наборі даних ",
train_data.shape[0])
print("Загальна кількість стовпців у навчальному наборі даних ",
train_data.shape[1])
print("Загальна кількість рядків у тестовому наборі даних ",
test_data.shape[0])
print("Загальна кількість стовпців у тестовому наборі даних ",
test_data.shape[1])
```

```
Загальна кількість рядків у навчальному наборі даних  891
Загальна кількість стовпців у навчальному наборі даних  12
Загальна кількість рядків у тестовому наборі даних  418
Загальна кількість стовпців у тестовому наборі даних  11
```

Рисунок 10.5. Загальна кількість рядків та стовпців у тестовому та навчальному наборах даних

4. Візуалізація та аналіз даних

Виконаємо візуалізацію кількості нульових значень в обох наборах даних

```
plt.figure(figsize = (13,5))
plt.bar(train_data.columns, train_data.isna().sum())
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у навчальному наборі даних")
plt.show()
```

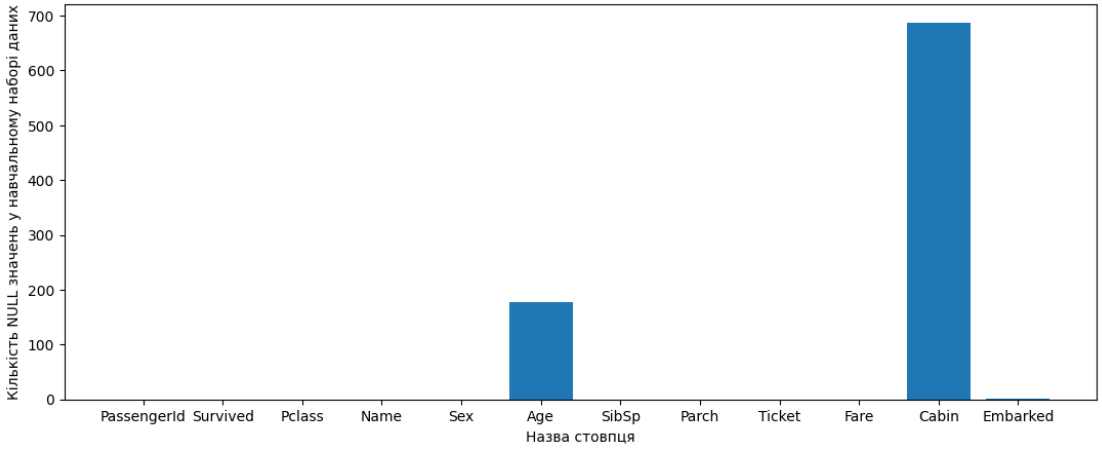


Рисунок 10.6. Графік «Кількість NULL значень у навчальному наборі даних»

```
plt.figure(figsize = (13,5))
plt.bar(test_data.columns, test_data.isnull().sum().values, color = 'red')
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у тестовому наборі даних")
plt.show()
```

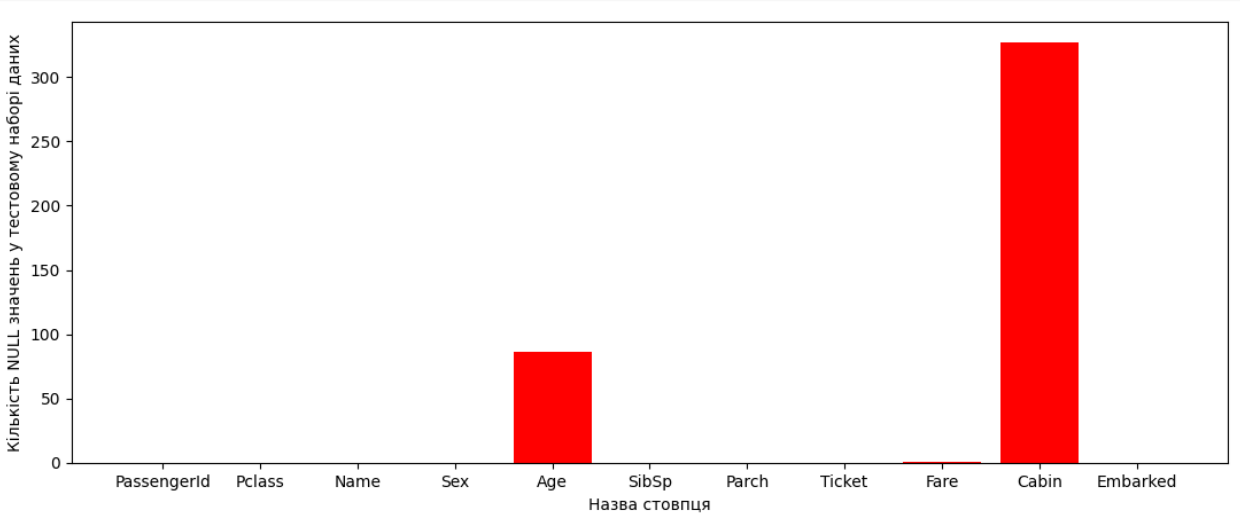


Рисунок 10.7. Графік «Кількість NULL значень у тестовому наборі даних»

Виконаємо візуалізацію кількості пасажирів, що вижили.

```
# Візуалізація кількості пасажирів, що вижили
sns.countplot(x='Survived', data = train_data)
plt.show()
```

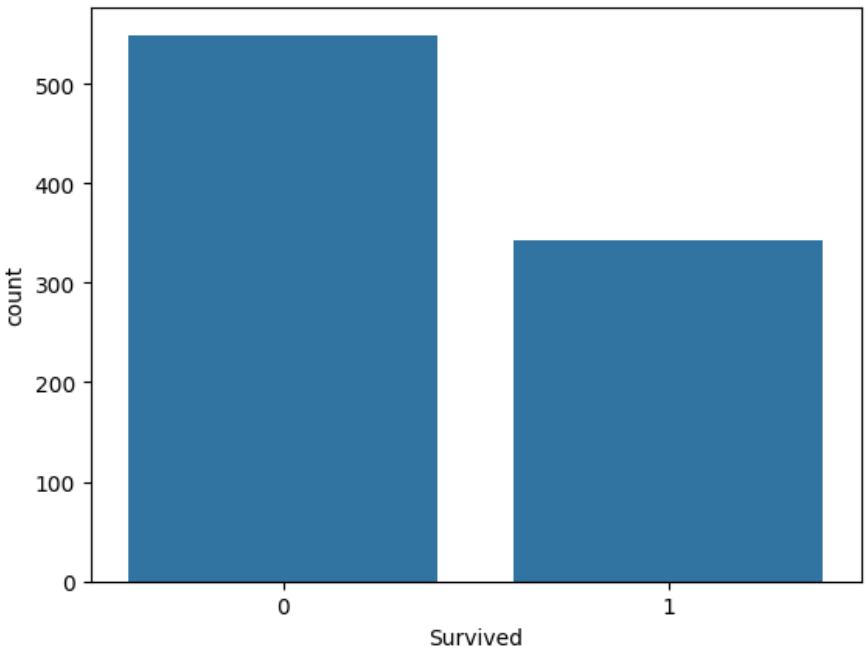


Рисунок 10.8. Графік «Кількість пасажирів, що вижили»

Виконаємо візуалізацію кількості пасажирів з посадкою у різних портах.

```
# Візуалізація кількості пасажирів з різних портів у навчальному наборі
даних
sns.countplot(x='Embarked', data = train_data)
plt.show()
```

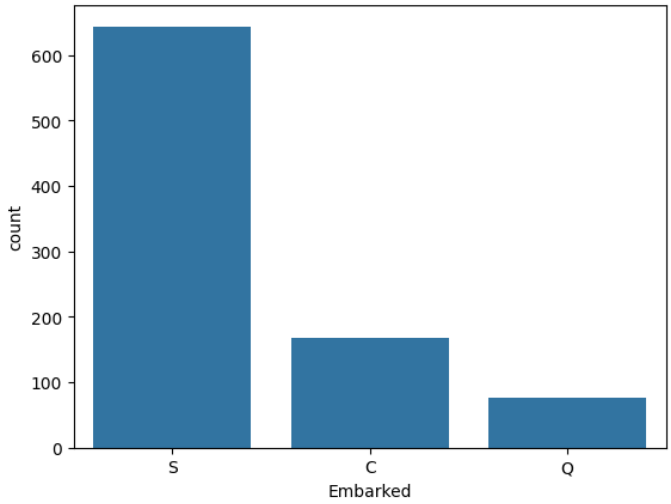


Рисунок 10.9. Графік «Кількість пасажирів з різних портів»

Виконаємо візуалізацію впливу гендеру на рівень виживання.

```
# Візуалізація впливу гендеру на рівень виживання
sns.countplot(x='Survived', hue = 'Sex', data = train_data)
plt.plot()
```

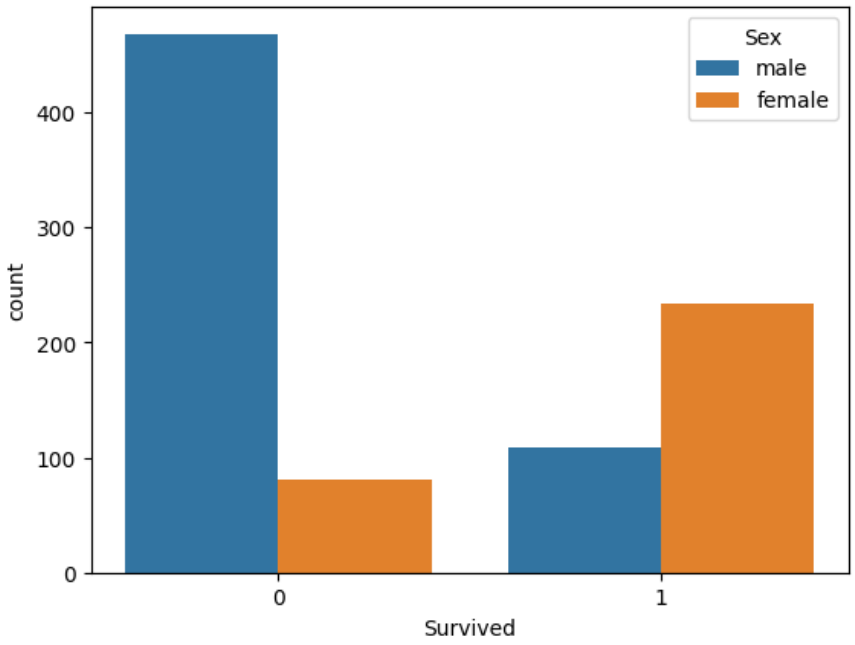


Рисунок 10.10. Графік «Вплив гендеру на виживання»

Виконаємо візуалізацію впливу рівня класу (pclass) на рівень виживання.

```
# Візуалізація впливу рівня класу (pclass) на рівень виживання
sns.countplot(x="Survived", hue = 'Pclass', data = train_data)
plt.show()
```

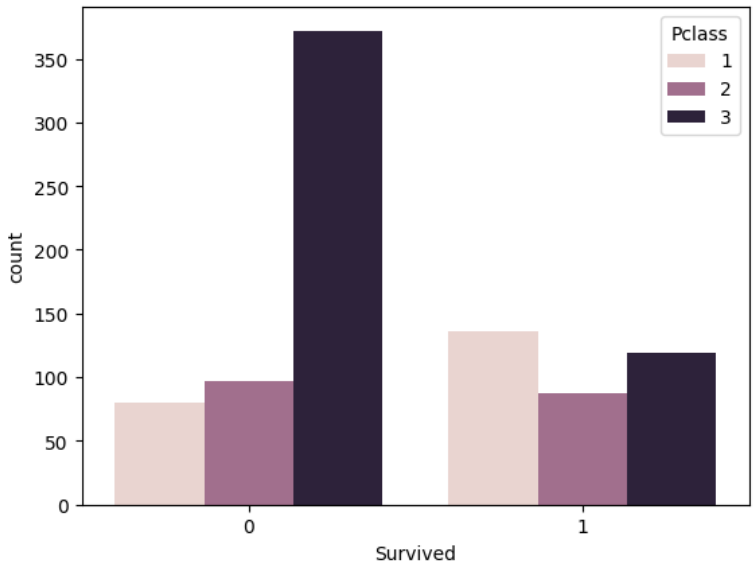


Рисунок 10.11. Графік «Вплив рівня класу (pclass) на рівень виживання»

Виконаємо візуалізацію впливу порту посадки на рівень виживання

```
# Візуалізація впливу порту посадки на рівень виживання
sns.countplot(x='Survived', hue = 'Embarked', data = train_data)
plt.show()
```

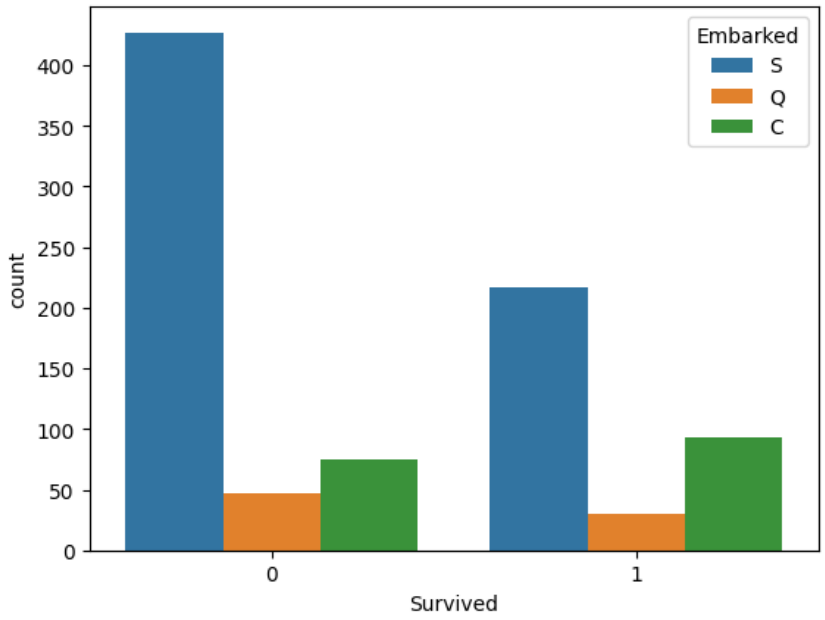


Рисунок 10.12. Графік «Вплив порту посадки на рівень виживання»

```
sns.boxplot(x='Fare', data = train_data)
plt.show()
# Результати свідчать про те, що було дуже мало людей, які заплатили більше
100$ за квиток
```

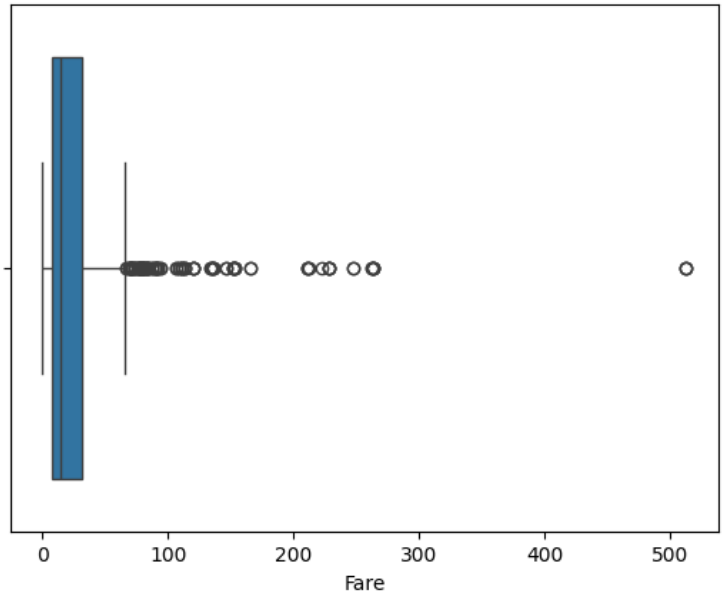


Рисунок 10.13. Графік викидів «Fare»

```
sns.boxplot(x='Age', data = train_data)
plt.show()
# Результати свідчать про те, що в навчальних даних було дуже мало людей,
старших за 65 років
```

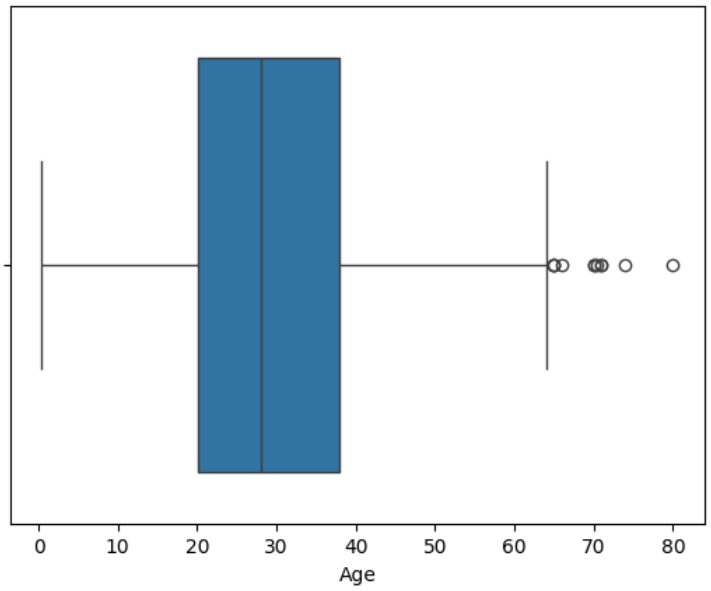


Рисунок 10.14. Графік викидів «Age»

```
interval = 10
value_for_bin = np.ceil((train_data.Age.max() - train_data.Age.min()) /
interval).astype(int)

plt.hist(train_data.Age, bins = value_for_bin)
plt.xlabel("Age")
plt.ylabel("Number")
plt.show()
# Результати свідчать про те, що більшість пасажирів були у віці від 20 до
40 років
```

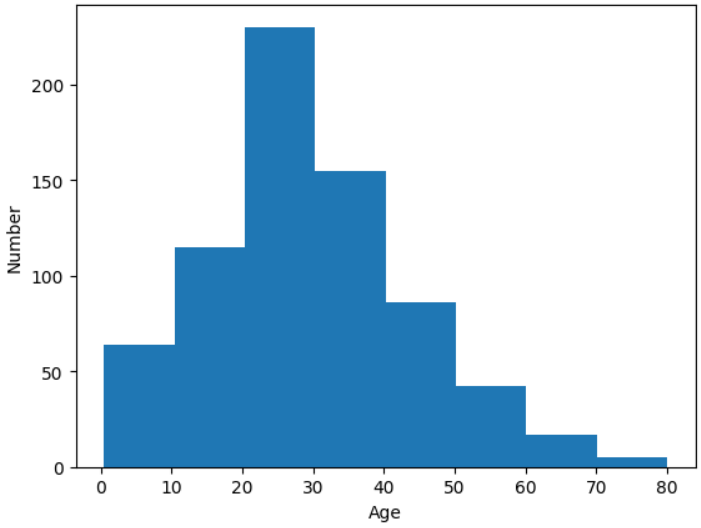


Рисунок 10.15. Графік «Віковий діапазон пасажирів»

```
plt.figure(figsize = (10,4))
plt.hist(train_data.Fare, bins = 10, color = 'lime')
plt.xlabel("Fare")
plt.ylabel("Number")
plt.show()
# Результати свідчать, що близько 700 людей заплатили від 0 до 50$
```

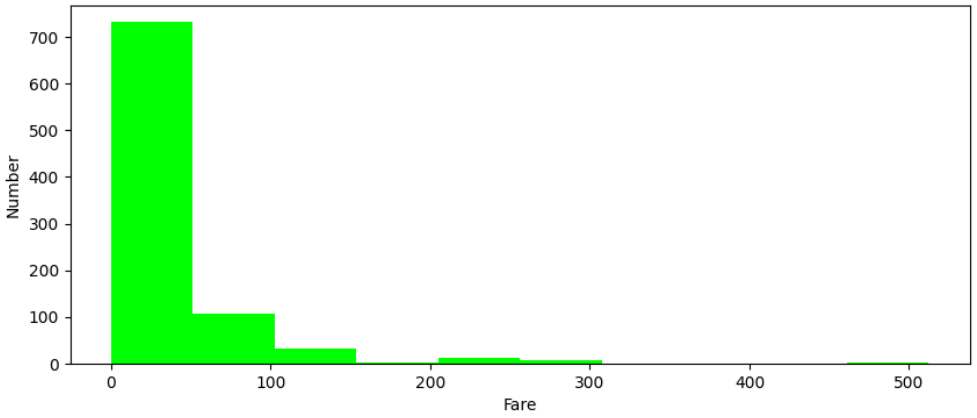


Рисунок 10.16. Графік «Витрати на квиток»

```
grid = sns.FacetGrid(train_data, col='Survived', row='Pclass', height=2.2,
aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
plt.show()
```

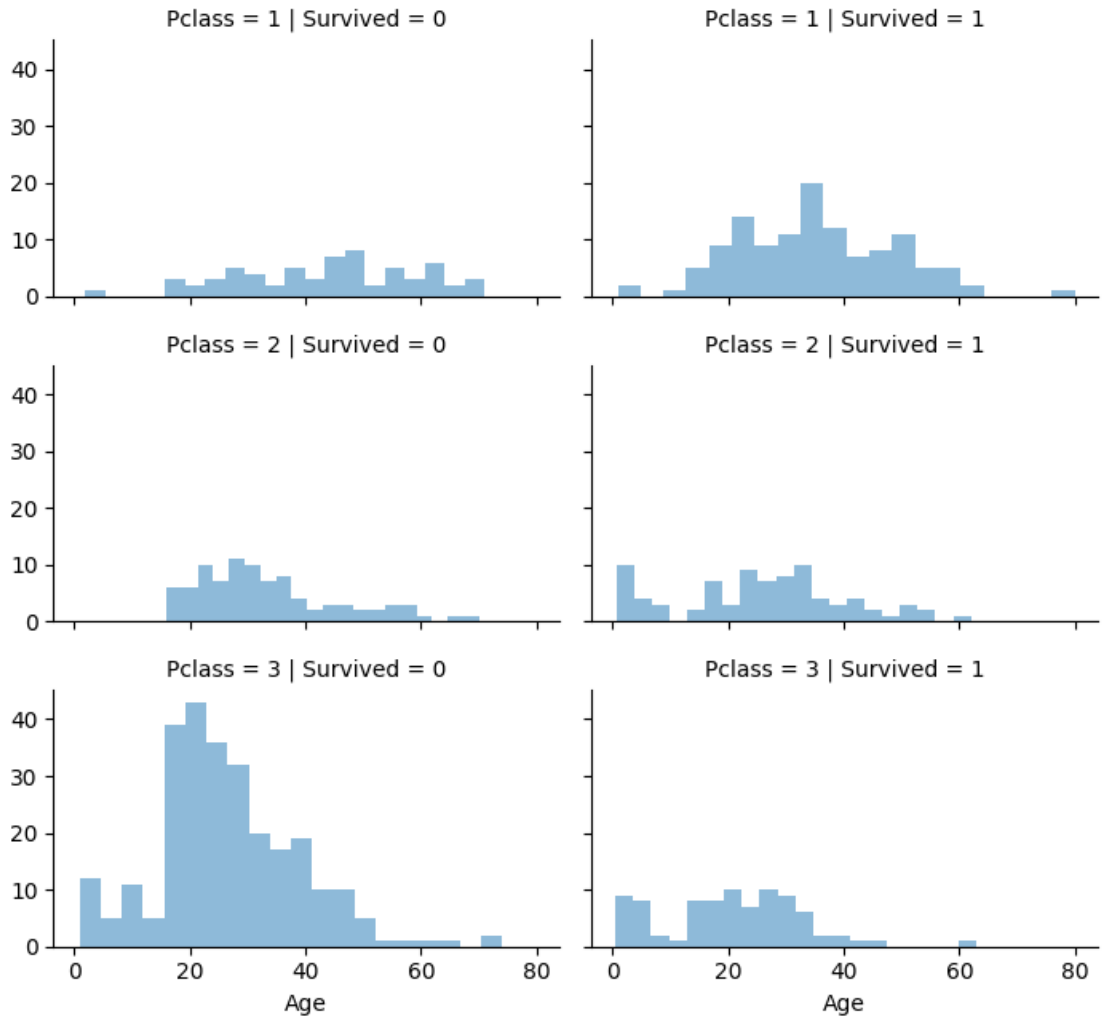


Рисунок 10.17. Графіки співвідношення класу та виживання

```
grid = sns.FacetGrid(train_data, row='Embarked', col='Survived', height=2.2,
aspect=1.6)
grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
grid.add_legend()
plt.show()
```

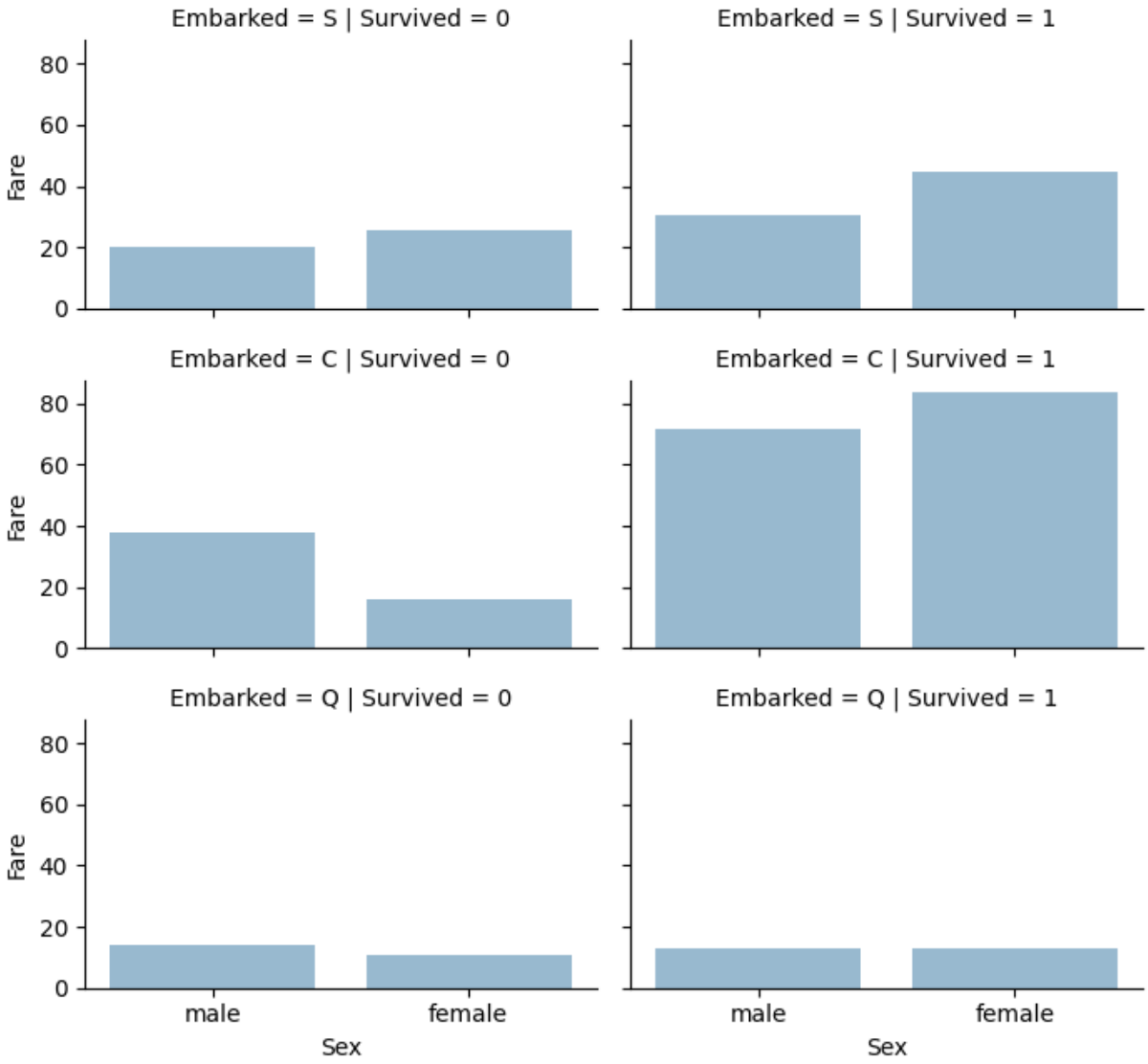


Рисунок 10.18. Графіки співвідношення місця посадки та виживання

```
corr_train = train_data.corr()
sns.heatmap(corr_train)
plt.show()
# Результати свідчать, що стовпці SibSp та Parch видалено, тому ми можемо
об'єднати їх для зменшення розмірності набору даних.
```

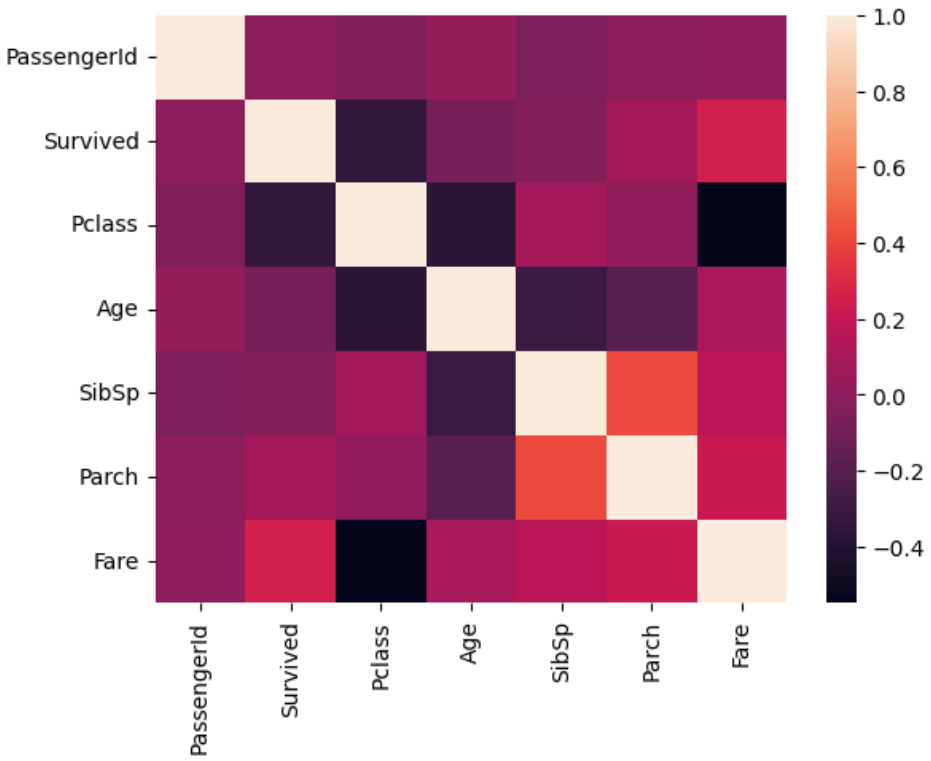


Рисунок 10.19. Теплова карта кореляційного зв’язку

5. Виконаємо простий аналіз даних.

Результати свідчать, що жінки мали близько 74% на виживання, в той час як чоловіки мали 81% на смерть.

```
((train_data.groupby(['Sex','Survived']).Survived.count() * 100) /
train_data.groupby('Sex').Survived.count())
```

```
Sex     Survived
female  0           25.796178
        1           74.203822
male    0           81.109185
        1           18.890815
Name: Survived, dtype: float64
```

Рисунок 10.20. Гендерне розподілення виживання

Результати свідчать, що люди, які належали до третього класу (бідніший), швидше за все, помруть, тоді як люди, які належать до першого класу (багатші), швидше за все, виживуть.

```
(train_data.groupby(['Pclass','Survived']).Survived.count() * 100) /
train_data.groupby('Pclass').Survived.count()
```

```
Pclass  Survived
1       0           37.037037
        1           62.962963
2       0           52.717391
        1           47.282609
3       0           75.763747
        1           24.236253
Name: Survived, dtype: float64
```

Рисунок 10.21. Класове розподілення виживання

Результати свідчать, про те, що люди, які вирушили з порту Southampton, швидше за все, загинуть.

```
(train_data.groupby(['Embarked','Survived']).Survived.count() * 100) /
train_data.groupby('Embarked').Survived.count()
```

```
Embarked  Survived
C         0           44.642857
          1           55.357143
Q         0           61.038961
          1           38.961039
S         0           66.304348
          1           33.695652
Name: Survived, dtype: float64
```

Рисунок 10.22. Географічне розподілення виживання

Результати свідчать, що середній вік людей, які вижили, становив близько 28 років.

```
train_data.groupby(by=['Survived']).mean()["Age"]
```

```
Survived
0    30.626179
1    28.343690
Name: Age, dtype: float64
```

Рисунок 10.23. Середній вік особ, що вижили та померли

6. Виконаємо роботу над NULL значенням.

Перед тим, як заповнити відсутні значення, видалимо стовпець Cabin з обох наборів даних.

```
train_data.drop('Cabin', axis = 1, inplace = True)
test_data.drop('Cabin', axis = 1, inplace = True)

combined_data = [train_data, test_data]
for data in combined_data:
    print(data.isnull().sum())
    print('*' * 20)
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64
********************
PassengerId     0
Pclass          0
Name            0
Sex             0
Age            86
SibSp           0
Parch           0
Ticket          0
Fare            1
Embarked        0
dtype: int64
********************
```

Рисунок 10.24. Визначення кількості NULL значень за стовпцями

Заповнимо стовпчики Age та Fare середніми значеннями, а стовпчик Embarked – значеннями, які зустрічаються найчастіше.

```
for data in combined_data:
    data.Age.fillna(data.Age.mean(), inplace = True)
    data.Fare.fillna(data.Fare.mean(), inplace = True)

# З візуалізацій, нам відомо, що порт Southampton є найпопулярнішим місцем посадки, тому, заповнимо пропущені значення "S"

train_data.Embarked.fillna('S', inplace = True)
```

7. Конвертуємо категоріальні ознаки

Почнемо з перетворення ознаки Sex у категоричні значення female = 1, male = 0

```
def change_gender(x):
    if x == 'male':
        return 0
    elif x == 'female':
        return 1
train_data.Sex = train_data.Sex.apply(change_gender)
test_data.Sex = test_data.Sex.apply(change_gender)

# або використаємо
# train_data.Sex = train_data.Sex.map({'female':1, 'male':0})
```

За допомогою функції map замінимо значення стовпчику Embarked S = 1, C = 2, Q = 0.

```
change = {'S':1,'C':2,'Q':0}
train_data.Embarked = train_data.Embarked.map(change)
test_data.Embarked = test_data.Embarked.map(change)
```

8. Виконаємо вилучення особливостей

Під час візуалізації кореляційної теплової карти ми виявили, що стовпці Sibsp та Parch тісно пов'язані між собою, тому створимо новий стовпець під назвою Alone, використовуючи ці два стовпці -------> 1 = Alone , 0 = not Alone.

```
train_data['Alone'] = train_data.SibSp + train_data.Parch
test_data['Alone'] = test_data.SibSp + test_data.Parch

train_data.Alone = train_data.Alone.apply(lambda x: 1 if x == 0 else 0)
test_data.Alone = test_data.Alone.apply(lambda x: 1 if x == 0 else 0)

# Тепер вилучимо стовпці SibSp та Parch для навчального та тестового наборів даних
train_data.drop(['SibSp','Parch'], axis = 1, inplace = True)
test_data.drop(['SibSp','Parch'], axis = 1, inplace = True )
```

Створимо новий елемент Title назви з існуючого елемента Name.

```
train_data.Name.str.extract(' ([A-Za-z]+)\.', expand=False).unique().size
# всього 17 унікальних назв

# Створимо елемент Title, що буде містити назву пасажира, вилучимо стовпець Name
for data in combined_data:
    data['Title'] = data.Name.str.extract('([A-Za-z]+)\.', expand = False)
    data.drop('Name', axis = 1, inplace = True)

train_data.Title.value_counts()
```

```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Mlle          2
Major         2
Col           2
Countess      1
Capt          1
Ms            1
Sir           1
Lady          1
Mme           1
Don           1
Jonkheer      1
Name: Title, dtype: int64
```

Рисунок 10.25. Визначення популярності значень Title

```
test_data.Title.unique()
```

```
array(['Mr', 'Mrs', 'Miss', 'Master', 'Ms', 'Col', 'Rev', 'Dr', 'Dona'],
      dtype=object)
```

Рисунок 10.26. Масив значень Title

Замінимо назви, що зустрічаються найменше на Rare.

```
least_occuring = [ 'Don', 'Rev', 'Dr', 'Mme', 'Ms',
       'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'Countess','Dona',
       'Jonkheer']
for data in combined_data:
    data.Title = data.Title.replace(least_occuring, 'Rare')

# Виконаємо відображення заголовків, замінивши їх на порядкові номери
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
for data in combined_data:
    data['Title'] = data['Title'].map(title_mapping)
```

Видалимо стовпці PassengerId та Ticket

```
columns_to_drop = ['PassengerId','Ticket']
train_data.drop(columns_to_drop, axis = 1, inplace = True)
test_data.drop(columns_to_drop[1], axis = 1, inplace = True)
```

Об’єднаємо стовпці Age та Fare

```
for dataset in combined_data:
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
```

```
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age'] = 4

for data in combined_data:
    data.loc[data['Fare'] < 30, 'Fare'] = 1
    data.loc[(data['Fare'] >= 30) & (data['Fare'] < 50),'Fare'] = 2
    data.loc[(data['Fare'] >= 50) & (data['Fare'] < 100),'Fare'] = 3
    data.loc[(data['Fare'] >= 100),'Fare'] = 4

corr_train = train_data.corr()
sns.heatmap(corr_train)
plt.show()
```

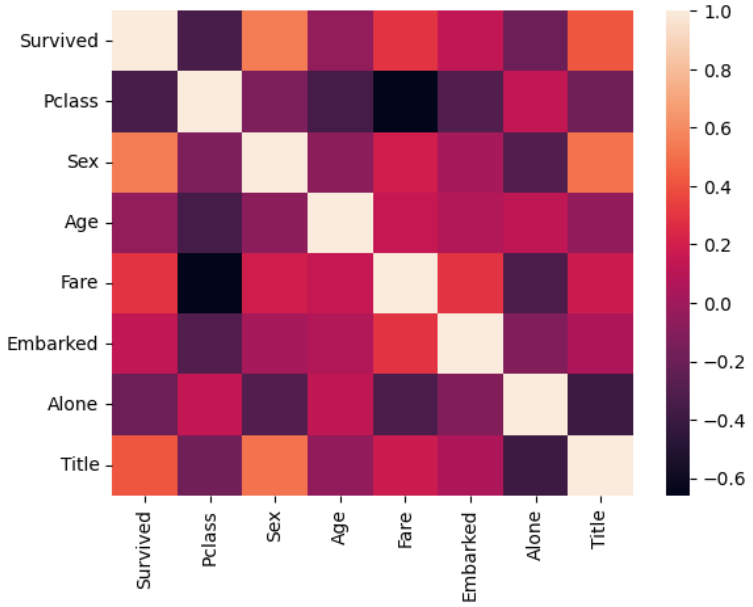


Рисунок 10.27. Результуюча теплова мапа

9. Виконаємо підготовку даних для навчання та тестування

```
X_train = train_data.drop("Survived", axis=1)
Y_train = train_data["Survived"]
X_test = test_data.drop("PassengerId", axis = 1)
print("shape of X_train",X_train.shape)
print("Shape of Y_train",Y_train.shape)
print("Shape of x_test",X_test.shape)
```

10. Розробимо та охарактеризуємо модель нейронної мережі

Примітка. Під час розробки ви можете використовувати різну кількість нейронів для кожного шару, різне значення для відсіву. Ви також можете змінювати значення гіперпараметрів для отримання кращого результату.

```
# Нейронна мережа
model = Sequential([
    Dense(32, input_dim=7, activation='relu'),
    Dense(64, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    tf.keras.layers.BatchNormalization(),
    Dense(128, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    Dropout(0.1),
    Dense(64, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    Dropout(0.1),
    Dense(32, activation='relu'),
    Dropout(0.15),
    Dense(16, activation='relu'),
    Dense(8, activation='relu', kernel_initializer='he_normal',
use_bias=False),
    Dense(1, activation='sigmoid')
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                256

 dense_1 (Dense)             (None, 64)                2048

 batch_normalization (Batch  (None, 64)                256
 Normalization)

 dense_2 (Dense)             (None, 128)               8192

 dropout (Dropout)           (None, 128)               0

 dense_3 (Dense)             (None, 64)                8192

 dropout_1 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 32)                2080

 dropout_2 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 16)                528

 dense_6 (Dense)             (None, 8)                 128

 dense_7 (Dense)             (None, 1)                 9

=================================================================
Total params: 21689 (84.72 KB)
Trainable params: 21561 (84.22 KB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
```

Рисунок 10.28. Характеристика моделі нейронної мережі

Скомпілюємо та натренуємо модель.

```
model.compile(loss=tf.keras.losses.binary_crossentropy,
              optimizer=tf.keras.optimizers.Adam(),
              metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, verbose=2, epochs=50)
```

```
Epoch 43/50
28/28 - 0s - loss: 0.3730 - accuracy: 0.8462 - 151ms/epoch - 5ms/step
Epoch 44/50
28/28 - 0s - loss: 0.3737 - accuracy: 0.8361 - 201ms/epoch - 7ms/step
Epoch 45/50
28/28 - 0s - loss: 0.3668 - accuracy: 0.8339 - 172ms/epoch - 6ms/step
Epoch 46/50
28/28 - 0s - loss: 0.3648 - accuracy: 0.8406 - 195ms/epoch - 7ms/step
Epoch 47/50
28/28 - 0s - loss: 0.3678 - accuracy: 0.8429 - 248ms/epoch - 9ms/step
Epoch 48/50
28/28 - 0s - loss: 0.3677 - accuracy: 0.8361 - 129ms/epoch - 5ms/step
Epoch 49/50
```

Рисунок 10.29. Процес навчання нейронної мережі

11. Прогнозування на основі тестових даних

```
predict = model.predict(X_test)
predict = (predict > 0.5).astype(int).ravel()
print(predict)
```

```
14/14 [==============================] - 1s 4ms/step
[0 1 0 0 1 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1 0 0 0 0 0 1 1 1 0 0
 0 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1
 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0
 1 1 1 1 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
 0 0 1 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1
 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1
 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 1 0 0
 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0
 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0
 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0
 0 1 1 1 1 0 0 1 0 0 1]
```

Рисунок 10.30. Прогнозування «вижив/не вижив»

```
submit = pd.DataFrame({"PassengerId":test_data.PassengerId,
'Survived':predict})
submit.to_csv("final_submission.csv",index = False)

from sklearn import metrics
Y_pred_rand = (model.predict(X_train) > 0.5).astype(int)
```

```
print('Precision : ', np.round(metrics.precision_score(Y_train,
Y_pred_rand)*100,2))
print('Accuracy : ', np.round(metrics.accuracy_score(Y_train,
Y_pred_rand)*100,2))
print('Recall : ', np.round(metrics.recall_score(Y_train,
Y_pred_rand)*100,2))
print('F1 score : ', np.round(metrics.f1_score(Y_train, Y_pred_rand)*100,2))
print('AUC : ', np.round(metrics.roc_auc_score(Y_train, Y_pred_rand)*100,2))
```

```
28/28 [==============================] - 0s 7ms/step
Precision :  80.42
Accuracy :  84.62
Recall :  79.24
F1 score :  79.82
AUC :  83.61
```

Рисунок 10.31. Результуючі показники моделі

```
# Побудова Confusion Matrix на тепловій карті
matrix = metrics.confusion_matrix(Y_train, Y_pred_rand)
sns.heatmap(matrix, annot = True,fmt = 'g')
plt.show()
```

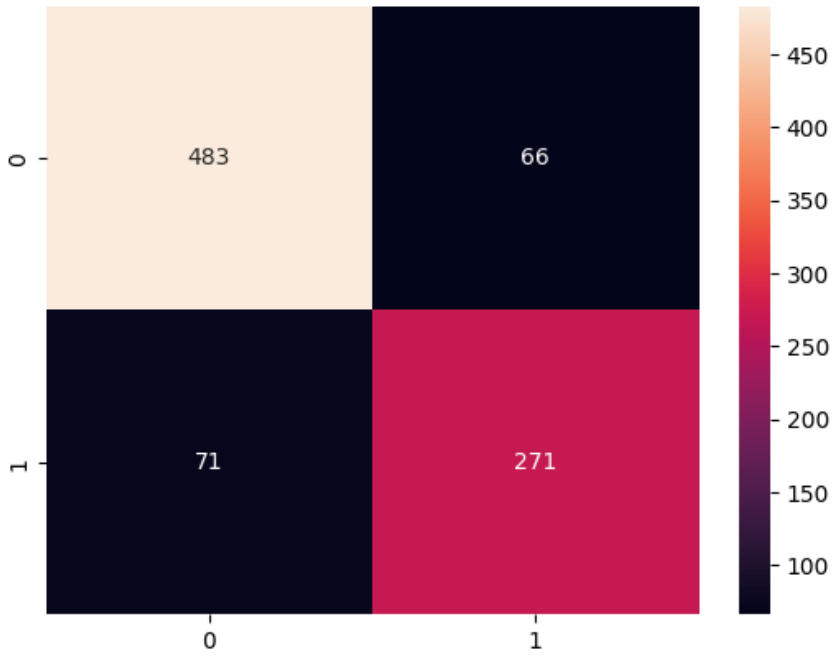


Рисунок 10.32. Confusion Matrix моделі

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.

Відповідь: **84.62**

Контрольне запитання №2

Вкажіть результуюче значення Precision.

Відповідь: **80.42**

Контрольне запитання №3

Вкажіть яке значення Survived приймає PassengerId 900.

Відповідь: …

Висновки: …

Завдання для виконання лабораторної роботи

Титанік – британський пасажирський лайнер компанії White Star Line, який затонув у північній частині Атлантичного океану рано вранці 15 квітня 1912 року після зіткнення з айсбергом під час свого першого рейсу з Саутгемптона до Нью-Йорка з приблизно 2224 пасажирами і членами екіпажу, які перебували на борту, понад 1500 загинули, що робить цю катастрофу однією з найсмертоносніших комерційних морських катастроф мирного часу в сучасній історії. Хоча на борту було близько 2224 пасажирів і членів екіпажу, ми маємо дані про 1300 пасажирів. З цих 1300, близько 900 даних використовуються з навчальною метою, а решта 400 – з тестовою. У цьому завдані вам надано близько 400 тестових даних з відсутнім стовпчиком виживших. Вам треба змоделювати модель нейронної мережі, щоб передбачити, чи вижили пасажири у тестових даних, чи ні. Як навчальні, так і тестові дані не є чистими (містять багато пропущених значень). Побудуйте модель з найкращою точністю.

Характеристика даних:

- Survival (виживання): 0 – ні, 1 – так;
- Pclass (номер класу): 1-й – вищий, 2-й – середній, 3-й – нижчий;
- sibsp (кількість братів і сестер / подружжя на борту): Sibling – брат, сестра, зведений брат, зведена сестра, Spouse – чоловік, дружина (коханки та наречені не враховуються);
- parch (кількість батьків / дітей на борту): Parent – мати, батько, Child – донька, син, падчерка, пасинок. Деякі діти подорожували лише з нянею, тому для них parch = 0;
- Ticket (квиток): номер квитка;
- Fare (вартість): пасажирський тариф;
- Cabin (каюта): номер каюти;
- Port of Embarkation (порт посадки): C – Cherbourg, Q – Queenstown, S – Southampton;
- Name (ім’я), Sex (стать), Age (вік).

Примітка. Дана лабораторна робота не має конкретних варіантів, але передбачає, що виконання буде базуватися на принципах креативності та зацікавленості здобувачів. Додаткову інформацію та приклади реалізації можна знайти на сайті Kaggle.

Посилання на навчальний та тестовий набори даних

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: _____

Контрольне запитання №2

Вкажіть результуюче значення Precision.
Відповідь: _____

Контрольне запитання №3

Вкажіть яке значення Survived приймає PassengerId 900.
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1hcaHTcz4n5gMoFiTrfEjHF2sMuBCpN1f?usp=sharing

Головне про Kaggle: **Kaggle** – це найбільша у світі платформа для змагань з даними, де спільнота з понад мільйоном користувачів, включаючи дослідників, інженерів з даними, та статистиків, співпрацюють для вирішення складних наукових завдань. Заснована у 2010 році, Kaggle надає компаніям, організаціям та дослідникам можливість опублікувати свої дані та поставити перед спільнотою виклики у вигляді змагань з призовими фондами. Учасники можуть взяти участь у змаганнях для побудови моделей прогнозування, аналізу даних та розробки алгоритмів, використовуючи машинне навчання та глибоке навчання.

За додатковою інформацією щодо Kaggle та задачі «Titanic» звертайтеся до https://www.kaggle.com/competitions/titanic

Лабораторна робота № 11. Моделювання нейронної мережі. Класифікація текстів з використанням RNN

Мета: сформулювати практичні навички у розробці, тренуванні та оцінці нейронних мереж з використанням рекурентних нейронних мереж (RNN) для задач класифікації текстів. Розвинути уміння працювати з текстовими даними, виконувати їх попередню обробку та векторизацію, а також оцінювати ефективність моделей на практичних прикладах.

Теоретичні відомості

Рекурентні нейронні мережі (RNN) є класом нейронних мереж, що ефективно обробляють послідовні дані або часові ряди. Вони здатні зберігати інформацію про попередні елементи послідовності в своїй внутрішній пам’яті, що робить їх ідеальними для задач обробки природної мови, включаючи класифікацію текстів, генерацію тексту, переклад мови та інші.

Основні компоненти RNN включають вузли (нейрони), які пов’язані між собою зворотними зв’язками, дозволяючи інформації циркулювати всередині мережі. Ця особливість дозволяє RNN «пам’ятати» попередні входи та використовувати цю інформацію для вирахування виходу.

Проте, стандартні RNN стикаються з проблемою зникнення або вибуху градієнтів при тренуванні на довгих послідовностях. Цю проблему частково вирішують варіанти RNN, такі як **LSTM (Long Short-Term Memory)** та **GRU (Gated Recurrent Unit)**, які впроваджують механізми забування та оновлення інформації, щоб балансувати збереження та відкидання інформації через час.

Попередня обробка тексту є критичним кроком при роботі з задачами класифікації текстів. Це включає видалення шуму (наприклад, пунктуації та стоп-слів), токенізацію тексту, векторизацію слів або використання предтренованих векторних представлень слів (наприклад, Word2Vec або GloVe), та нормалізацію даних перед подачею їх у модель.

Для оцінки ефективності моделей класифікації текстів використовуються такі метрики, як точність (accuracy), точність класифікації для кожного класу (precision), повнота (recall), F1-оцінка, яка є гармонійним середнім між точністю та повнотою, а також матриця помилок для візуалізації помилок класифікації.

Приклад виконання роботи

Завдання

1. **Підготовка даних.**
   - Завантажте набір даних для класифікації текстів, який містить текстові зразки та їх відповідні мітки класів (наприклад, позитивний, негативний, нейтральний);
   - Виконайте попередню обробку текстових даних: очищення тексту, видалення зайвих символів, токенізація, конвертація тексту в послідовності та їх подальше вирівнювання (padding).
2. **Розробка моделі RNN.**
   - Створіть модель RNN з використанням шарів Embedding, LSTM або GRU (за вибором) для обробки послідовностей;
   - Додайте необхідні шари для класифікації, такі як Dense, Dropout для запобігання перенавчанню;
   - Компілюйте модель, вибравши відповідну функцію втрат та оптимізатор.
3. **Тренування та оцінка моделі.**
   - Тренуйте модель на тренувальному наборі даних, використовуючи валідаційний набір для моніторингу процесу навчання;
   - Використайте колбеки, такі як EarlyStopping та ReduceLROnPlateau, для оптимізації процесу тренування;
   - Оцініть ефективність моделі на тестовому наборі даних, аналізуючи метрики, такі як точність (accuracy).
4. **Аналіз результатів.**
   - Візуалізуйте історію тренувань за допомогою графіків для оцінки змін точності та втрат в процесі тренування;
   - Проведіть декілька тестів з власними текстами для перевірки здатності моделі правильно класифікувати текстові дані.
5. **Експериментування.**
   - Модифікуйте архітектуру моделі, кількість епох, розмір пакету (batch size), оптимізатор та інші параметри тренування для покращення результатів;
   - Спробуйте використати різні стратегії попередньої обробки даних та інженерії ознак для аналізу їх впливу на ефективність моделі.

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.

Відповідь: _____

Контрольне запитання №2

Яка основна проблема стандартних RNN при тренуванні на довгих послідовностях?

Відповідь: _____

Контрольне запитання №3

Які дві модифікації RNN розроблені для вирішення цієї проблеми?

Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набори даних.

textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km◆)	Density (P/Km◆)
cb774db0d1	I'd have responded,	I'd have responded,	neutral	morning	0-20	Afghanistan	38928346	652860	60
549e992a42	Sooo SAD I will miss	Sooo SAD	negative	noon	21-30	Albania	2877797	27400	105
088c60f138	my boss is bullying m	bullying me	negative	night	31-45	Algeria	43851044	2381740	18
9642c003ef	what interview! leave	leave me alone	negative	morning	46-60	Andorra	77265	470	164
358bd9e861	Sons of ****, why co	Sons of ****,	negative	noon	60-70	Angola	32866272	1246700	26
28b57f3990	http://www.dothebou	http://www.dothebou	neutral	night	70-100	Antigua and Barbuda	97929	440	223

Рисунок 11.1. Датасет на Google Drive

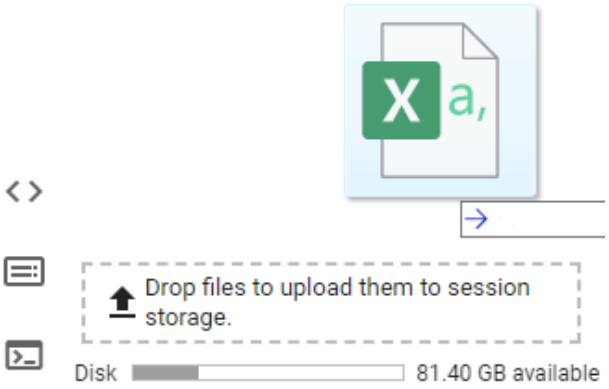


Рисунок 11.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів.

```
import warnings
warnings.filterwarnings('ignore')
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Dropout, Bidirectional
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from keras.utils import to_categorical
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

3. Завантаження наборів даних.

```
train_ds = pd.read_csv('/content/train.csv',encoding='latin1');
validation_ds = pd.read_csv('/content/test.csv',encoding='latin1');

train_ds.head()
validation_ds.head()
```

	textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km²)	Density (P/Km²)
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	morning	0-20	Afghanistan	38928346	652860.0	60
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	noon	21-30	Albania	2877797	27400.0	105
2	088c60f138	my boss is bullying me...	bullying me	negative	night	31-45	Algeria	43851044	2381740.0	18
3	9642c003ef	what interview! leave me alone	leave me alone	negative	morning	46-60	Andorra	77265	470.0	164
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	noon	60-70	Angola	32866272	1246700.0	26

Рисунок 11.3. Перші 5 рядків навчального набору даних

	textID	text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km²)	Density (P/Km²)
0	f87dea47db	Last session of the day http://twitpic.com/67ezh	neutral	morning	0-20	Afghanistan	38928346.0	652860.0	60.0
1	96d74cb729	Shanghai is also really exciting (precisely -...	positive	noon	21-30	Albania	2877797.0	27400.0	105.0
2	eee518ae67	Recession hit Veronique Branquinho, she has to...	negative	night	31-45	Algeria	43851044.0	2381740.0	18.0
3	01082688c6	happy bday!	positive	morning	46-60	Andorra	77265.0	470.0	164.0
4	33987a8ee5	http://twitpic.com/4w75p - I like it!!	positive	noon	60-70	Angola	32866272.0	1246700.0	26.0

Рисунок 11.4. Перші 5 рядків тестового набору даних

Ми бачимо, що загальна кількість стовпців у тестовому наборі даних на один менше, ніж у навчальному. Відсутній стовпець у тестових даних – це selected_text (виділений текст) стовпець, який ми повинні передбачити.

```
print("Загальна кількість рядків у навчальному наборі даних ",
train_ds.shape[0])
print("Загальна кількість стовпців у навчальному наборі даних ",
train_ds.shape[1])
print("Загальна кількість рядків у тестовому наборі даних ",
validation_ds.shape[0])
print("Загальна кількість стовпців у тестовому наборі даних ",
validation_ds.shape[1])
```

```
Загальна кількість рядків у навчальному наборі даних  27481
Загальна кількість стовпців у навчальному наборі даних  10
Загальна кількість рядків у тестовому наборі даних  4815
Загальна кількість стовпців у тестовому наборі даних  9
```

Рисунок 11.5. Загальна кількість рядків та стовпців у тестовому та навчальному наборах даних

4. Візуалізація та аналіз даних

Виконаємо візуалізацію кількості нульових значень в обох наборах даних

```
plt.figure(figsize = (13,5))
plt.bar(train_ds.columns, train_ds.isna().sum())
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у навчальному наборі даних")
plt.show()
```

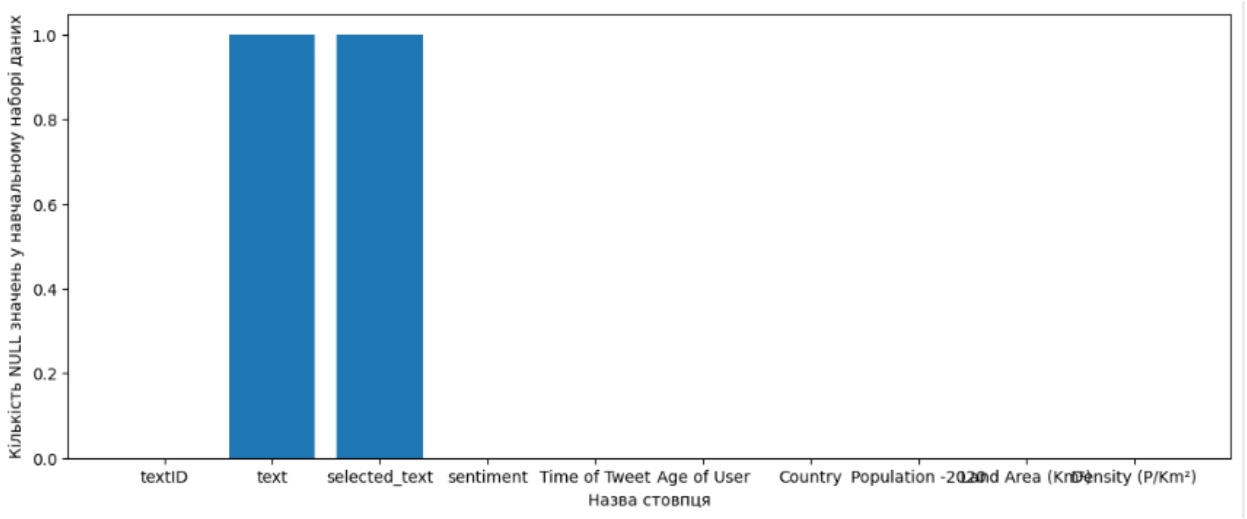


Рисунок 11.6. Графік «Кількість NULL значень у навчальному наборі даних»

```
plt.figure(figsize = (13,5))
plt.bar(validation_ds.columns, validation_ds.isnull().sum().values, color =
'red')
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у тестовому наборі даних")
plt.show()
```

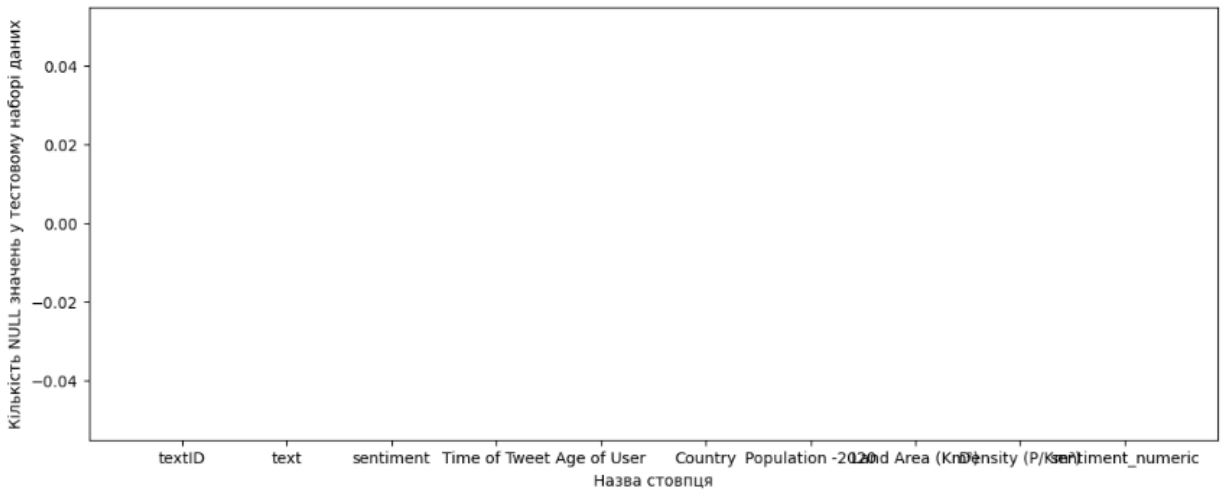


Рисунок 11.7. Графік «Кількість NULL значень у тестовому наборі даних»

Визначимо функцію для перетворення текстових міток настроїв («позитивний», «негативний», «нейтральний») на числові значення (0, 1, 2). Застосуємо цю функцію до навчального та валідаційного наборів даних для створення нових стовпчиків з числовими значеннями настроїв. Рядки з невідображеними настроями (позначені -1) відфільтровуються.

```
def sentiment_to_numeric(sentiment):
    mapping = {'positive': 0, 'negative': 1, 'neutral': 2}
    return mapping.get(sentiment, -1)

train_ds['sentiment_numeric'] =
train_ds['sentiment'].apply(sentiment_to_numeric)
validation_ds['sentiment_numeric'] =
validation_ds['sentiment'].apply(sentiment_to_numeric)

train_ds = train_ds[train_ds['sentiment_numeric'] != -1]
validation_ds = validation_ds[validation_ds['sentiment_numeric'] != -1]
```

Визначимо масив ознак (текст) і цільовий масив (настрої).

```
x_train, y_train = np.array(train_ds['text'].tolist()),
np.array(train_ds['sentiment'].tolist())
x_test, y_test = np.array(validation_ds['text'].tolist()),
np.array(validation_ds['sentiment'].tolist())
```

Перетворимо числові мітки настроїв у формат однократного кодування.

```
y_train = np.array(train_ds['sentiment_numeric'].tolist())
y_test = np.array(validation_ds['sentiment_numeric'].tolist())

y_train = to_categorical(y_train, 3)
y_test = to_categorical(y_test, 3)
```

Переглянемо розподіл категорій настроїв.

```
print("Unique values in y_train:", np.unique(y_train))
print("Unique values in y_test:", np.unique(y_test))
```

```
Unique values in y_train: [0. 1.]
Unique values in y_test: [0. 1.]
```

Рисунок 11.8. Унікальні значення категорій настроїв у навчальному та тестовому наборах

Підготуємо текстові дані для введення в модель шляхом токенізації та розбиття на частини.

```
# Tokenization and padding
tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(list(x_train) + list(x_test))
x_train_seq = tokenizer.texts_to_sequences(x_train)
x_test_seq = tokenizer.texts_to_sequences(x_test)
x_train_pad = pad_sequences(x_train_seq, maxlen=35)
x_test_pad = pad_sequences(x_test_seq, maxlen=35)
```

5. Розробимо та охарактеризуємо модель нейронної мережі

Змоделюємо послідовну архітектуру моделі з вбудовуванням, двонаправленими LSTM шарами, відсівом для регуляризації та щільним шаром для класифікації.

Примітка. Під час розробки ви можете використовувати різну кількість нейронів для кожного шару, різне значення для відсіву. Ви також можете змінювати значення гіперпараметрів для отримання кращого результату.

```
# Нейронна мережа
model = Sequential([
    Embedding(input_dim=20000, output_dim=100, input_length=35),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.5),
    Bidirectional(LSTM(32)),
    Dense(3, activation='softmax')
])
```

Підготуємо модель до навчання, задаючи оптимізатор, функцію втрат та метрики.

```
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 35, 100)           2000000

 bidirectional (Bidirection  (None, 35, 128)           84480
 al)

 dropout (Dropout)           (None, 35, 128)           0

 bidirectional_1 (Bidirecti  (None, 64)                41216
 onal)

 dense (Dense)               (None, 3)                 195

=================================================================
Total params: 2125891 (8.11 MB)
Trainable params: 2125891 (8.11 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 11.9. Характеристика моделі нейронної мережі

Налаштуємо зворотні дзвінки, щоб запобігти надмірному пристосуванню.

```
# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2,
min_lr=0.001)
```

Виконаємо процес навчання та оцінювання роботи моделі.

```
# Model training
history = model.fit(x_train_pad, y_train, epochs=5,
validation_data=(x_test_pad, y_test), callbacks=[early_stopping, reduce_lr])
```

```
Epoch 1/5
859/859 [==============================] - 48s 40ms/step - loss: 0.7694 - accuracy: 0.6497 - val_loss: 0.6524
Epoch 2/5
859/859 [==============================] - 14s 16ms/step - loss: 0.5275 - accuracy: 0.7894 - val_loss: 0.6543
Epoch 3/5
859/859 [==============================] - 13s 15ms/step - loss: 0.3918 - accuracy: 0.8504 - val_loss: 0.7467
Epoch 4/5
859/859 [==============================] - 12s 14ms/step - loss: 0.2951 - accuracy: 0.8926 - val_loss: 0.8033
```

Рисунок 11.10. Процес тренування RNN моделі

6. Візуалізуємо точність навчання та валідації моделі.

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.legend()
plt.show()
```

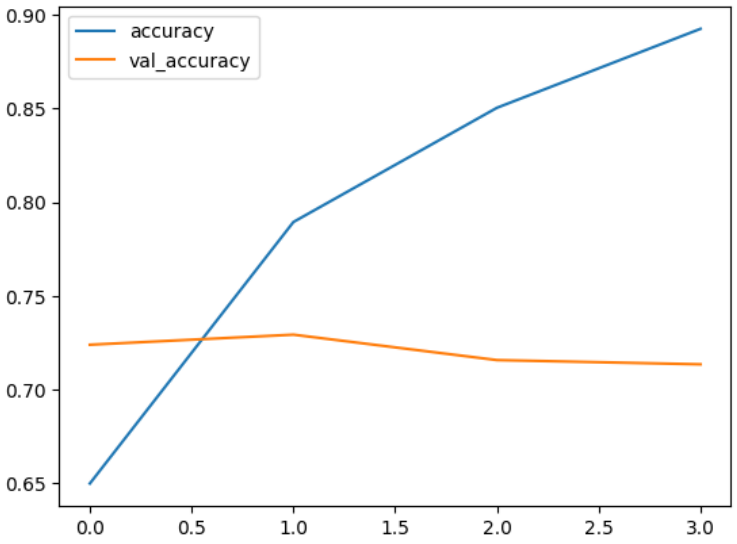


Рисунок 11.11. Графік точності навчання

7. Створимо функцію передбачення настрою в текстовому контексті.

Визначимо функцію для попередньої обробки довільного тексту (токенізація та паддінг), передбачення настрою за допомогою навченої моделі та декодування передбачення у читабельну мітку настрою.

```
def predict_sentiment(text, tokenizer, model):
    seq = tokenizer.texts_to_sequences([text])
    padded_seq = pad_sequences(seq, maxlen=35)
    prediction = model.predict(padded_seq)
    sentiment_idx = np.argmax(prediction)
    sentiments = {0: 'Positive', 1: 'Negative', 2: 'Neutral'}
    return sentiments[sentiment_idx]

# Example 1:
text_to_predict = "I love this product, it's amazing!"
predicted_sentiment = predict_sentiment(text_to_predict, tokenizer, model)
print(f"Predicted sentiment: {predicted_sentiment}")

# Example 2:
text_to_predict = "This is the worst movie I have ever seen."
predicted_sentiment = predict_sentiment(text_to_predict, tokenizer, model)
print(f"Predicted sentiment: {predicted_sentiment}")
```

```
1/1 [==============================] - 2s 2s/step
Predicted sentiment: Positive
1/1 [==============================] - 0s 24ms/step
Predicted sentiment: Negative
```

Рисунок 11.12. Приклад передбачення моделлю RNN

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: **0.8926**

Контрольне запитання №2

Яка основна проблема стандартних RNN при тренуванні на довгих послідовностях?
Відповідь: _____

Контрольне запитання №3

Які дві модифікації RNN розроблені для вирішення цієї проблеми?
Відповідь: _____

Висновки: …

Завдання для виконання лабораторної роботи

1. **Підготовка даних.**
   - Завантажте набір даних для класифікації текстів, який містить текстові зразки та їх відповідні мітки класів (наприклад, позитивний, негативний, нейтральний);
   - Виконайте попередню обробку текстових даних: очищення тексту, видалення зайвих символів, токенізація, конвертація тексту в послідовності та їх подальше вирівнювання (padding).
2. **Розробка моделі RNN.**
   - Створіть модель RNN з використанням шарів Embedding, LSTM або GRU (за вибором) для обробки послідовностей;
   - Додайте необхідні шари для класифікації, такі як Dense, Dropout для запобігання перенавчання;
   - Компілюйте модель, вибравши відповідну функцію втрат та оптимізатор.
3. **Тренування та оцінка моделі.**
   - Тренуйте модель на тренувальному наборі даних, використовуючи валідаційний набір для моніторингу процесу навчання;
   - Використайте колбеки, такі як EarlyStopping та ReduceLROnPlateau, для оптимізації процесу тренування;
   - Оцініть ефективність моделі на тестовому наборі даних, аналізуючи метрики, такі як точність (accuracy).
4. **Аналіз результатів.**
   - Візуалізуйте історію тренувань за допомогою графіків для оцінки змін точності та втрат в процесі тренування;
   - Проведіть декілька тестів з власними текстами для перевірки здатності моделі правильно класифікувати текстові дані.
5. **Експериментування.**
   - Модифікуйте архітектуру моделі, кількість епох, розмір пакету (batch size), оптимізатор та інші параметри тренування для покращення результатів;
   - Спробуйте використати різні стратегії попередньої обробки даних та інженерії ознак для аналізу їх впливу на ефективність моделі.

Примітка. Дана лабораторна робота не має конкретних варіантів, але передбачає, що виконання буде базуватися на принципах креативності та зацікавленості здобувачів.

Посилання на навчальний та тестовий набори даних

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: _____

Контрольне запитання №2

Яка основна проблема стандартних RNN при тренуванні на довгих послідовностях?
Відповідь: _____

Контрольне запитання №3

Які дві модифікації RNN розроблені для вирішення цієї проблеми?
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1vowoil7SsUzJ6Vx3p-xSldiJBkv8jepp?usp=sharing

Лабораторна робота № 12. Побудова та оптимізація нейронних мереж для класифікації зображень

Мета: набуття практичних навичок у побудові та оптимізації згорткових нейронних мереж (Convolutional Neural Networks, CNNs) для задач класифікації зображень. Здобувачі ознайомляться з основними концепціями CNN, навчаться реалізовувати архітектури нейронних мереж у середовищі Google Colab з використанням бібліотеки Keras (або TensorFlow), а також досліджуватимуть вплив різних гіперпараметрів та методів оптимізації на продуктивність моделі. Окрім цього, метою є розвиток навичок командної роботи та розподілу завдань між членами команди.

Теоретичні відомості

Згорткові нейронні мережі (CNN) є особливим класом глибоких нейронних мереж, які продемонстрували виняткову ефективність у задачах обробки зображень, включаючи класифікацію. Основними компонентами CNN є:

- **Згорткові шари (Convolutional Layers)** – шари, які застосовують до вхідного зображення набір фільтрів (ядер згортки), кожен з яких виділяє певну ознаку, наприклад, краї, кути або текстури. Результатом роботи згорткового шару є карта ознак (feature map), яка відображає активації відповідного фільтра на різних ділянках зображення. Важливими гіперпараметрами згорткового шару є розмір ядра (kernel size), кількість фільтрів (number of filters), крок згортки (stride) та доповнення (padding).
- **Шари пулінгу (Pooling Layers)** – шари зменшують просторову розмірність карт ознак, тим самим зменшуючи обчислювальну складність і ризик перенавчання. Найбільш поширеними типами пулінгу є max-pooling та average-pooling. Max-pooling вибирає максимальне значення з кожного підвікна, а average-pooling обчислює середнє значення. Важливими гіперпараметрами шару пулінгу є розмір підвікна (pool size) та крок (stride).
- **Шари повнозв’язного шару (Fully Connected Layers).** Після декількох згорткових та пулінгових шарів карта ознак перетворюється на одновимірний вектор, який подається на вхід повнозв'язного шару. Повнозв’язні шари аналогічні шарам у звичайних нейронних мережах і виконують класифікацію на основі виділених ознак.
- **Функція активації (Activation Function).** Кожен нейрон у мережі має функцію активації, яка вносить нелінійність у модель, дозволяючи їй вивчати складні залежності між даними. Популярними функціями активації є ReLU (Rectified

Linear Unit), Sigmoid та Tanh. ReLU є особливо ефективною для згорткових шарів, оскільки вона вирішує проблему зникаючого градієнта, що часто виникає при навчанні глибоких мереж.

- **Функція втрат (Loss Function).** Функція втрат вимірює розбіжність між передбаченнями моделі та істинними мітками. Для задач класифікації зазвичай використовують категоріальну перехресну ентропію (categorical cross-entropy).
- **Оптимізатор (Optimizer).** Оптимізатор відповідає за оновлення ваг мережі під час навчання з метою мінімізації функції втрат. Популярними оптимізаторами є стохастичний градієнтний спуск (Stochastic Gradient Descent, SGD), Adam та RMSprop.

Оптимізація нейронних мереж:

Для досягнення високої точності класифікації необхідно оптимізувати архітектуру та гіперпараметри нейронної мережі:

- **Підбір архітектури.** Вибір кількості та типів шарів, розміру ядер згортки, кількості фільтрів, розміру пулінгу та інших параметрів.
- **Регуляризація.** Застосування технік регуляризації, таких як L1 та L2 регуляризація, dropout та батч-нормалізація, для запобігання перенавчанню.
- **Аугментація даних.** Штучне розширення навчального набору даних шляхом застосування різних перетворень до зображень, таких як повороти, зсуви, масштабування та відображення, що допомагає підвищити узагальнюючу здатність моделі.
- **Підбір швидкості навчання.** Швидкість навчання (learning rate) є важливим гіперпараметром, який визначає, наскільки сильно оновлюються ваги мережі на кожному кроці оптимізації. Занадто велика швидкість навчання може призвести до нестабільності процесу навчання, а занадто мала – до повільної збіжності. Часто використовують техніки адаптивного підбору швидкості навчання, такі як Adam та RMSprop.
- **Рання зупинка (Early Stopping).** Моніторинг продуктивності моделі на валідаційному наборі даних та припинення навчання, коли продуктивність перестає покращуватися.

Приклад виконання роботи

Завдання

Побудувати та навчити згорткову нейронну мережу для класифікації зображень з набору даних CIFAR-10. Набір CIFAR-10 містить 60000 кольорових зображень розміром 32x32 пікселів, розділених на 10 класів (літак, автомобіль, птах, кіт, олень, собака, жаба, кінь, корабель, вантажівка). 50000 зображень використовуються для навчання, а 10000 – для тестування.

1. Завантажити та попередньо обробити дані CIFAR-10.
2. Визначити архітектуру згорткової нейронної мережі.
3. Скомпілювати модель, вибравши функцію втрат, оптимізатор та метрики.
4. Навчити модель на навчальному наборі даних, використовуючи аугментацію даних.
5. Оцінити продуктивність моделі на тестовому наборі даних.
6. Провести експерименти з різними гіперпараметрами та методами оптимізації та проаналізувати результати.

Контрольне запитання

Які фактори впливають на вибір архітектури згорткової нейронної мережі та як вони впливають на її продуктивність?
Відповідь: _____

Хід роботи

1. Завантаження та попередня обробка даних.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

print(f"Розмір навчального набору даних: {x_train.shape}")
print(f"Розмір тестового набору даних: {x_test.shape}")
```

У цьому коді ми спочатку імпортуємо необхідні бібліотеки, такі як TensorFlow, Keras та ImageDataGenerator. Потім ми завантажуємо набір даних CIFAR-10, використовуючи функцію `cifar10.load_data()`. Після цього ми нормалізуємо значення пікселів зображень, ділячи їх на 255.0, щоб вони знаходилися в діапазоні від 0 до 1. Нарешті, ми перетворюємо мітки класів у формат one-hot encoding, використовуючи функцію `to_categorical()`.

2. Визначення архітектури згорткової нейронної мережі.

```
model = keras.Sequential(
    [
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32, 32, 3)),
        layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(10, activation='softmax')
```

```
    ]
)

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18,496
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	73,856
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 550,570 (2.10 MB)
Trainable params: 550,570 (2.10 MB)
Non-trainable params: 0 (0.00 B)

Рисунок 12.1. Результат виконання завдання №2

Тут ми визначаємо архітектуру згорткової нейронної мережі, використовуючи клас `keras.Sequential`. Модель складається з декількох згорткових шарів, шарів пулінгу, шару вирівнювання, повнозв'язного шару та вихідного шару. Згорткові шари використовують функцію активації ReLU та доповнення 'same', щоб зберегти просторову розмірність. Шари пулінгу використовують max-pooling з розміром вікна 2x2. Шар вирівнювання перетворює вихід з останнього шару пулінгу на одновимірний вектор. Повнозв'язний шар має 128 нейронів та функцію активації ReLU. Для запобігання перенавчанню додано шар Dropout з коефіцієнтом 0.5. Вихідний шар має 10 нейронів (по одному для кожного класу) та функцію активації softmax. Функція `model.summary()` виводить детальну інформацію про архітектуру моделі.

3. Компіляція моделі.

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

На цьому етапі ми компілюємо модель, вказуючи оптимізатор, функцію втрат та метрики. Ми використовуємо оптимізатор Adam, функцію втрат категоріальної перехресної ентропії та метрику точності (accuracy).

4. Навчання моделі з аугментацією даних.

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
)

datagen.fit(x_train)

batch_size = 64
epochs = 50

history = model.fit(datagen.flow(x_train, y_train, batch_size=batch_size),
                    epochs=epochs,
                    validation_data=(x_test, y_test),
                    steps_per_epoch=x_train.shape[0] // batch_size)
```

```
Epoch 46/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 1s 830us/step - accuracy: 0.7031 - loss: 0.7214 - val_accuracy: 0.8022 - val_loss: 0.6127
Epoch 47/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 38s 49ms/step - accuracy: 0.8096 - loss: 0.5654 - val_accuracy: 0.8122 - val_loss: 0.5911
Epoch 48/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 1s 829us/step - accuracy: 0.8594 - loss: 0.4058 - val_accuracy: 0.8149 - val_loss: 0.5850
Epoch 49/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 43s 52ms/step - accuracy: 0.8150 - loss: 0.5569 - val_accuracy: 0.7964 - val_loss: 0.6476
Epoch 50/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 1s 807us/step - accuracy: 0.7812 - loss: 0.5613 - val_accuracy: 0.7956 - val_loss: 0.6507
```

Рисунок 12.3. Результат виконання завдання №3

Тут ми використовуємо клас `ImageDataGenerator` для аугментації даних. Ми застосовуємо випадкові повороти, зсуви та горизонтальні відображення до зображень. Потім ми навчаємо модель, використовуючи метод `model.fit()`, передаючи йому генератор даних `datagen.flow()`, кількість епох, валідаційний набір даних та кількість кроків на епоху. Збільшена кількість епох до 50 дозволяє моделі краще вивчити залежності в даних.

5. Оцінка продуктивності моделі.

```
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"Test loss: {loss:.4f}")
print(f"Test accuracy: {accuracy:.4f}")
```

```
Test loss: 0.6507
Test accuracy: 0.7956
```

Рисунок 12.4. Результат виконання завдання №4

Ми оцінюємо продуктивність навченої моделі на тестовому наборі даних, використовуючи метод `model.evaluate()`. Результати виводяться на екран.

6. Експерименти з гіперпараметрами та аналіз результатів.

На цьому етапі є можливість поекспериментувати з різними гіперпараметрами моделі, такими як кількість згорткових шарів, кількість фільтрів, розмір ядра, швидкість навчання, тип оптимізатора тощо. Також можна спробувати різні методи регуляризації, такі як L1/L2 регуляризація та батч-нормалізація.

Наприклад, змінимо оптимізатор на RMSprop та зменшимо швидкість навчання:

```
model_rmsprop = keras.Sequential(
    [
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32, 32, 3)),
        layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(10, activation='softmax')
    ]
)

model_rmsprop.compile(optimizer=keras.optimizers.RMSprop(learning_rate=0.000
5), # Змінений оптимізатор
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
history_rmsprop = model_rmsprop.fit(datagen.flow(x_train, y_train,
batch_size=batch_size),
                    epochs=epochs,
                    validation_data=(x_test, y_test),
                    steps_per_epoch=x_train.shape[0] // batch_size)

loss_rmsprop, accuracy_rmsprop = model_rmsprop.evaluate(x_test, y_test,
verbose=0)
print(f"Test loss (RMSprop): {loss_rmsprop:.4f}")
print(f"Test accuracy (RMSprop): {accuracy_rmsprop:.4f}")
```

```
Epoch 49/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 37s 47ms/step - accuracy: 0.8000 - loss: 0.6356 - val_accuracy: 0.7982 - val_loss: 0.6543
Epoch 50/50
781/781 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8750 - loss: 0.4678 - val_accuracy: 0.8059 - val_loss: 0.6477
Test loss (RMSprop): 0.6477
Test accuracy (RMSprop): 0.8059
```

Рисунок 12.5. Результат виконання завдання №5

В данному прикладі, ми визначили нову модель `model_rmsprop` з ідентичною архітектурою попередній моделі, але змінили оптимізатор на RMSprop. Також ми встановили швидкість навчання на рівні 0.0005, що є меншим за замовчування значення для Adam. Зменшення швидкості навчання може допомогти покращити стабільність процесу навчання та потенційно досягти кращої точності.

7. Аналіз результатів.

Після проведення експериментів з різними гіперпараметрами та методами оптимізації необхідно проаналізувати отримані результати. Слід звернути увагу на те, як змінилися показники точності та втрат на тестовому наборі даних. Також корисно побудувати графіки залежності точності та втрат від кількості епох навчання для різних конфігурацій моделі, що дозволить візуалізувати процес навчання та виявити можливі проблеми, такі як перенавчання або недонавчання.

Наприклад, порівняємо графіки навчання для моделі з оптимізатором Adam та моделі з оптимізатором RMSprop та зменшеною швидкістю навчання.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
```

```
plt.title('Adam: Accuracy vs. Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Adam: Loss vs. Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history_rmsprop.history['accuracy'], label='Train Accuracy')
plt.plot(history_rmsprop.history['val_accuracy'], label='Validation
Accuracy')
plt.title('RMSprop: Accuracy vs. Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history_rmsprop.history['loss'], label='Train Loss')
plt.plot(history_rmsprop.history['val_loss'], label='Validation Loss')
plt.title('RMSprop: Loss vs. Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

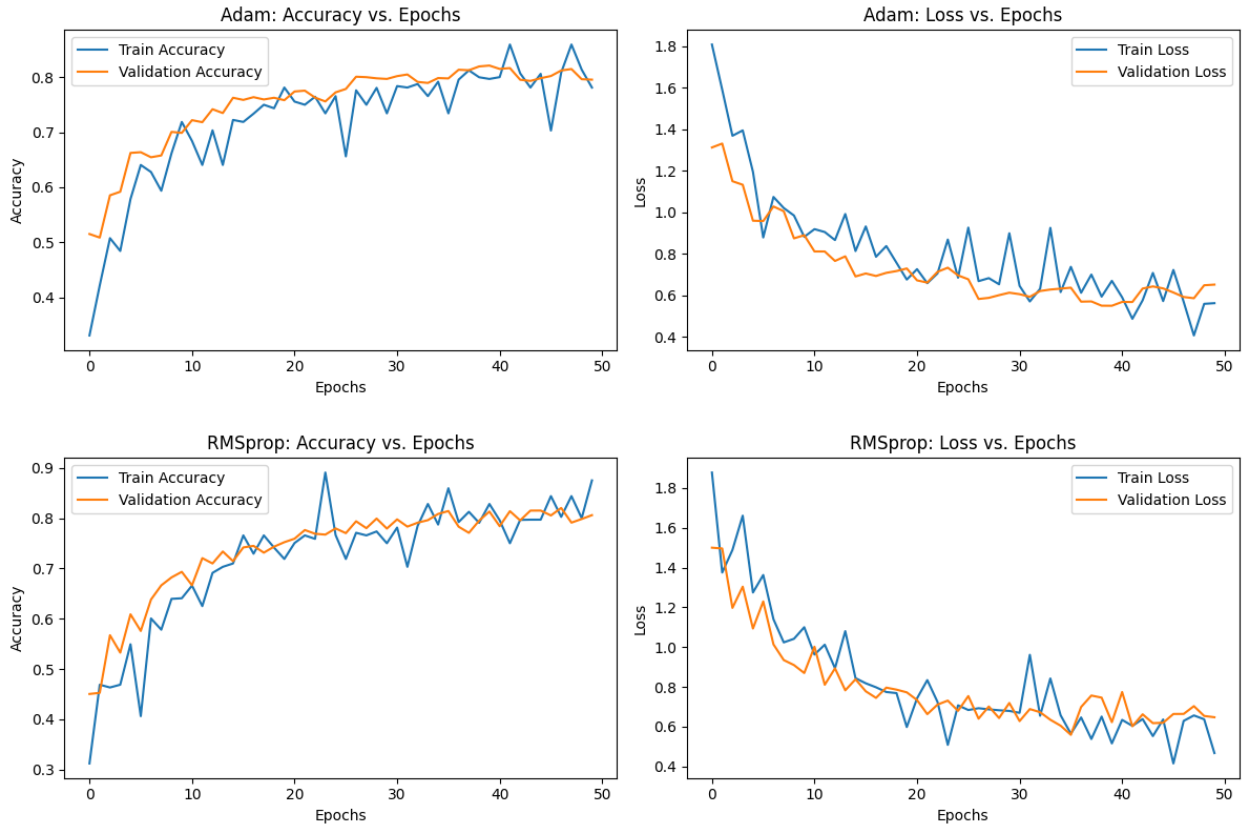


Рисунок 12.6. Результат виконання завдання №6

Цей код генерує графіки, які відображають зміну точності (accuracy) та функції втрат (loss) протягом епох навчання для обох моделей (з оптимізаторами Adam та RMSprop) і дозволяє візуально порівняти їхню продуктивність та стабільність процесу навчання.

Порівнюючи графіки та числові значення точності та втрат, Ви повинні зробити висновки щодо впливу різних гіперпараметрів та методів оптимізації на продуктивність моделі. Наприклад, Ви можете виявити, що модель з оптимізатором RMSprop та зменшеною швидкістю навчання досягає кращої точності на тестовому наборі даних та демонструє більш стабільний процес навчання порівняно з моделлю з оптимізатором Adam.

Важливі аспекти для обговорення:

- **Вплив кількості згорткових шарів та фільтрів.** Як змінюється продуктивність моделі при збільшенні або зменшенні кількості згорткових шарів та фільтрів? Чи спостерігається перенавчання або недонавчання?
- **Вплив розміру ядра згортки.** Як впливає розмір ядра згортки на здатність моделі виділяти різні ознаки?
- **Вплив швидкості навчання.** Як впливає швидкість навчання на швидкість збіжності та стабільність процесу навчання?
- **Вплив методів регуляризації.** Як впливають методи регуляризації (dropout, L1/L2

регуляризація, батч-нормалізація) на здатність моделі до узагальнення?

- **Вплив аугментації даних.** Як впливає аугментація даних на продуктивність моделі та її здатність до узагальнення?
- **Вибір оптимізатора.** Який оптимізатор (Adam, RMSprop, SGD тощо) є найбільш ефективним для даної задачі та архітектури?

Контрольне запитання

Які фактори впливають на вибір архітектури згорткової нейронної мережі та як вони впливають на її продуктивність?
Відповідь: _____

Висновки: …

Завдання для виконання лабораторної роботи

Кожній команді (2-3 здобувачі) необхідно розробити та навчити згорткову нейронну мережу для класифікації зображень з набору даних CIFAR-100. Набір CIFAR-100 схожий на CIFAR-10, але містить 100 класів замість 10. Кожен клас містить 600 зображень. 500 зображень використовуються для навчання, а 100 – для тестування. Класи згруповані в 20 суперкласів. Кожне зображення має мітку «fine» (клас, до якого воно належить) та мітку «coarse» (суперклас, до якого воно належить).

Розподіл завдань між членами команди (приклад):

- **Здобувач 1.** Відповідає за завантаження та попередню обробку даних CIFAR-100, включаючи нормалізацію та аугментацію даних. Досліджує різні методи аугментації та їх вплив на продуктивність.
- **Здобувач 2.** Відповідає за розробку архітектури згорткової нейронної мережі. Експериментує з різною кількістю згорткових та повнозв'язних шарів, кількістю фільтрів, розмірами ядер та функціями активації.
- **Здобувач 3.** Відповідає за навчання та оптимізацію моделі. Експериментує з різними оптимізаторами, швидкостями навчання та методами регуляризації. Використовує техніку ранньої зупинки.

Завдання по пунктам:

1. Завантаження та попередня обробка даних (**Здобувач 1**):

1.1. Завантажити набір даних CIFAR-100.

1.2. Нормалізувати дані, поділивши значення пікселів на 255.0.

1.3. Перетворити мітки класів у формат one-hot encoding.

1.4. Розділити дані на навчальний, валідаційний та тестовий набори (наприклад, 80%, 10%, 10%).

1.5. Реалізувати аугментацію даних, використовуючи `ImageDataGenerator`. Дослідити різні параметри аугментації, такі як повороти, зсуви, масштабування, відображення, зміна яскравості та контрастності.

2. Розробка архітектури нейронної мережі (**Здобувач 2**):

2.1. Створити базову архітектуру згорткової нейронної мережі, використовуючи шари `Conv2D`, `MaxPooling2D`, `Flatten`, `Dense` та `Dropout`.

2.2. Експериментувати з різною кількістю згорткових шарів (наприклад, від 2 до 5).

2.3. Експериментувати з різною кількістю фільтрів у кожному згортковому шарі (наприклад, від 32 до 128).

2.4. Експериментувати з різними розмірами ядер згортки (наприклад, 3x3, 5x5).

2.5. Експериментувати з різними функціями активації (ReLU, Leaky ReLU, ELU).

2.6. Використати шар `Dropout` після одного або декількох повнозв'язних шарів для запобігання перенавчанню.

2.7. Дослідити можливість використання батч-нормалізації.

2.8. Спробувати різні конфігурації повнозв'язних шарів (кількість шарів та кількість нейронів).

3. Навчання та оптимізація моделі (**Здобувач 3**):

3.1. Скомпілювати модель, вибравши оптимізатор (Adam, RMSprop, SGD), функцію втрат (categorical cross-entropy) та метрики (accuracy).

3.2. Навчити модель на навчальному наборі даних, використовуючи аугментацію даних та валідаційний набір.

3.3. Експериментувати з різними швидкостями навчання (наприклад, 0.001, 0.0001).

3.4. Використовувати техніку ранньої зупинки, щоб запобігти перенавчанню.

3.5. Дослідити вплив L1/L2 регуляризації на продуктивність моделі.

3.6. Зберегти найкращу модель на основі точності на валідаційному наборі.

4. Оцінка та аналіз результатів (**Вся команда**):

4.1. Оцінити продуктивність найкращої моделі на тестовому наборі даних.

4.2. Побудувати матрицю невідповідностей (confusion matrix) для аналізу помилок класифікації.

4.3. Побудувати графіки залежності точності та втрат від кількості епох навчання для різних конфігурацій моделі.

4.4. Проаналізувати результати та зробити висновки щодо впливу різних гіперпараметрів та методів оптимізації на продуктивність моделі.

4.5. Обговорити, які класи або суперкласи розпізнаються найкраще, а які найгірше, та чому.

4.6. Запропонувати можливі шляхи покращення продуктивності моделі.

Варіанти для виконання завдання

За бажанням Ви маєте можливість експерементувати будь-яким іншим чином (що не вказаний в межах завдання), задля досягнення кращих результатів.

№	Кількість	Кількість фільтрів	Розмір ядра	Функція активації	Оптимізатор	Швидкість навчання	Регуляризація	Аугментація
1	3	32, 64, 128	3x3	ReLU	Adam	0.001	Dropout (0.3)	Повороти, зсуви, горизонтальне відображення
2	4	16, 32, 64, 128	3x3	Leaky ReLU	RMSprop	0.0005	L2 (0.001)	Повороти, зсуви, масштабування
3	2	64, 128	5x5	ELU	SGD	0.01	Dropout (0.5), Batch Normalization	Повороти, зсуви, зміна яскравості
4	3	32, 64, 128	3x3	ReLU	Adam	0.0001	Dropout (0.4)	Зсуви, горизонтальне та вертикальне відображення
5	5	16, 32, 64, 64, 128	3x3	ReLU	Adam	0.001	L1 (0.0005)	Повороти, масштабування, зміна контрастності
6	2	32, 64	5x5	Leaky ReLU	RMSprop	0.0005	Batch Normalization	Горизонтальне відображення, зміна яскравості та контрастності
7	4	32, 32, 64, 64	3x3	ELU	SGD	0.01	Dropout (0.25)	Повороти, зсуви, масштабування, горизонтальне відображення
8	3	64, 128, 256	3x3	ReLU	Adam	0.0001	L2 (0.0001)	Зсуви, масштабування, зміна яскравості
9	4	16, 32, 64, 128	5x5	Leaky ReLU	RMSprop	0.001	Dropout (0.5)	Повороти, зсуви, горизонтальне та вертикальне відображення
10	2	64, 128	3x3	ReLU	Adam	0.005	L1 (0.001)	Масштабування, зміна яскравості та контрастності
11	3	32, 64, 128	3x3	ELU	SGD	0.001	Batch Normalization	Повороти, зсуви, горизонтальне відображення
12	5	8, 16, 32, 64, 128	3x3	ReLU	Adam	0.0001	Dropout (0.4)	Зсуви, масштабування, зміна контрастності
13	2	32, 64	5x5	Leaky ReLU	RMSprop	0.0005	L2 (0.0005)	Горизонтальне відображення, зміна яскравості та контрастності
14	4	32, 32, 64, 128	3x3	ReLU	Adam	0.001	Dropout (0.3)	Повороти, зсуви, масштабування, горизонтальне відображення
15	3	64, 128, 128	5x5	ELU	RMSprop	0.0001	L1 (0.0001)	Зсуви, масштабування, зміна яскравості
16	4	16, 32, 64, 128	3x3	ReLU	SGD	0.01	Dropout (0.5)	Повороти, зсуви, горизонтальне та вертикальне відображення

17	2	64, 128	3x3	Leaky ReLU	Adam	0.005	Batch Normalization	Масштабування, зміна яскравості та контрастності
18	3	32, 64, 128	5x5	ReLU	RMSprop	0.001	Dropout (0.2)	Повороти, горизонтальне відображення
19	5	8, 16, 32, 64, 128	3x3	ELU	Adam	0.0001	L2 (0.001)	Зсуви, зміна контрастності
20	2	32, 64	3x3	ReLU	SGD	0.01	Dropout (0.4)	Горизонтальне та вертикальне відображення
21	4	32, 32, 64, 64	3x3	Leaky ReLU	RMSprop	0.0005	L1 (0.0005)	Повороти, зсуви, масштабування
22	3	64, 128, 256	5x5	ReLU	Adam	0.001	Batch Normalization	Зсуви, масштабування, зміна яскравості
23	4	16, 32, 64, 128	3x3	ELU	Adam	0.0001	Dropout (0.5)	Повороти, зсуви, горизонтальне та вертикальне відображення
24	2	64, 128	3x3	ReLU	SGD	0.001	L2 (0.0001)	Масштабування, зміна яскравості та контрастності
25	3	32, 64, 128	5x5	Leaky ReLU	RMSprop	0.001	Dropout (0.3)	Повороти, зсуви, горизонтальне відображення
26	5	8, 16, 32, 64, 128	3x3	ReLU	Adam	0.0005	L1 (0.001)	Зсуви, масштабування, зміна контрастності
27	2	32, 64	3x3	ELU	RMSprop	0.0001	Batch Normalization	Горизонтальне відображення, зміна яскравості та контрастності
28	4	32, 32, 64, 128	3x3	ReLU	SGD	0.01	Dropout (0.4)	Повороти, зсуви, масштабування, горизонтальне відображення
29	3	64, 128, 128	3x3	Leaky ReLU	Adam	0.001	L2 (0.0005)	Зсуви, масштабування, зміна яскравості
30	4	16, 32, 64, 128	5x5	ReLU	RMSprop	0.0005	Dropout (0.5)	Повороти, зсуви, горизонтальне та вертикальне відображення

Welcome to Google Colab

https://colab.research.google.com/drive/1fhpBkHnnvPuF6U9ZJSTM7y1H6IKbT-Tu?usp=sharing

Лабораторна робота № 13. Використання генеративних змагальних нейронних мереж (GANs) для створення зображень

Мета: ознайомитися з принципами роботи та можливостями генеративних моделей на прикладі Stable Diffusion, навчитися генерувати та аналізувати зображення, а також експериментувати з різними параметрами моделі.

Завдання для всіх варіантів

1. ***Завдання 1:*** *дослідіть документацію встановлених бібліотек та поясніть, для чого кожна з них використовується в цьому проєкті.*
2. ***Завдання 2:*** *знайдіть інші моделі на Hugging Face, які можна використати замість "`runwayml/stable-diffusion-v1-5`". Спробуйте завантажити та порівняти їх.*
3. ***Завдання 3:*** *модифікуйте функцію `show_images`, щоб вона могла відображати довільну кількість зображень у сітці (наприклад, 3x3 або 4x4).*
4. ***Завдання 4:*** *додайте ще один параметр для контролю генерації зображень (наприклад: Width та Height, Negative Prompt, температуру тощо) та відповідний віджет.*
5. ***Завдання 5:*** *реалізуйте функцію для інтерполяції між трьома промптами та створіть відповідний віджет.*
6. ***Завдання 6:*** *додайте інші методи аналізу зображень (наприклад, гістограма кольорів або виявлення об'єктів) та відповідні кнопки.*
7. ***Завдання 7:*** *реалізуйте функцію для завантаження користувацького зображення та його аналізу.*

Базова структура роботи

Блок 1: Встановлення та імпорт бібліотек

```
!pip install transformers torch torchvision ipywidgets matplotlib pillow
!pip install git+https://github.com/huggingface/diffusers.git
!pip install opencv-python

import torch
import torchvision.transforms as transforms
from torchvision.utils import make_grid
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
from diffusers import StableDiffusionPipeline, DPMSolverMultistepScheduler
import ipywidgets as widgets
from IPython.display import display, clear_output
import cv2
import io
import base64
import glob
import os
```

Завдання 1: *дослідіть документацію встановлених бібліотек та поясніть, для чого кожна з них використовується в цьому проєкті.*

Блок 2: Завантаження моделі

```
model_id = "runwayml/stable-diffusion-v1-5"
pipe = StableDiffusionPipeline.from_pretrained(model_id,
torch_dtype=torch.float16)
pipe.scheduler =
DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
pipe = pipe.to("cuda")
```

Завдання 2: *знайдіть інші моделі на Hugging Face, які можна використати замість "**runwayml/stable-diffusion-v1-5**". Спробуйте завантажити та порівняти їх.*

Блок 3: Функції для генерації та відображення зображень

```
def generate_image(prompt, num_images=1, guidance_scale=7.5,
num_inference_steps=50, seed=None):
    if seed is not None:
        generator = torch.Generator("cuda").manual_seed(seed)
    else:
        generator = None
    with torch.no_grad():
        images = pipe(prompt, num_images_per_prompt=num_images,
guidance_scale=guidance_scale,
                      num_inference_steps=num_inference_steps,
generator=generator).images
    return images
```

```
def show_images(images):
    fig, axes = plt.subplots(1, len(images), figsize=(15, 5))
    for i, img in enumerate(images):
        if len(images) == 1:
            axes.imshow(img)
            axes.axis('off')
        else:
            axes[i].imshow(img)
            axes[i].axis('off')
    plt.tight_layout()
    plt.show()

def slerp(t, v0, v1, DOT_THRESHOLD=0.9995):
    dot = torch.sum(v0 * v1 / (torch.norm(v0) * torch.norm(v1)))
    if abs(dot) > DOT_THRESHOLD:
        v2 = (1 - t) * v0 + t * v1
    else:
        theta_0 = torch.acos(dot)
        sin_theta_0 = torch.sin(theta_0)
        theta_t = theta_0 * t
        sin_theta_t = torch.sin(theta_t)
        s0 = torch.sin(theta_0 - theta_t) / sin_theta_0
        s1 = sin_theta_t / sin_theta_0
        v2 = s0 * v0 + s1 * v1
    return v2
```

Завдання 3: *модифікуйте функцію* ***show_images****, щоб вона могла відображати довільну кількість зображень у сітці (наприклад, 3x3 або 4x4).*

Блок 4: Інтерфейс для генерації зображень

```
output_images = widgets.Output()

def generate_images_button_clicked(b):
    with output_images:
        clear_output(wait=True)
        images = generate_image(prompt.value, num_images.value,
guidance_scale.value, num_inference_steps.value, seed_widget.value)
        show_images(images)
        generate_button.images = images  # зберігаємо згенеровані зображення

prompt = widgets.Text(value="A portrait of a smiling person",
description="Prompt:", style={'description_width': 'initial'})
num_images = widgets.IntSlider(min=1, max=4, step=1, value=1,
description="Number of images:", style={'description_width': 'initial'})
guidance_scale = widgets.FloatSlider(min=1, max=20, step=0.5, value=7.5,
description="Guidance scale:", style={'description_width': 'initial'})
num_inference_steps = widgets.IntSlider(min=10, max=100, step=10, value=50,
description="Inference steps:", style={'description_width': 'initial'})
seed_widget = widgets.IntText(value=42, description="Seed:",
style={'description_width': 'initial'})
random_seed_button = widgets.Button(description="Random Seed")
```

```
generate_button = widgets.Button(description="Generate Images")

def on_random_seed_click(b):
    seed_widget.value = np.random.randint(0, 1000000)

random_seed_button.on_click(on_random_seed_click)
generate_button.on_click(generate_images_button_clicked)
```

Завдання 4: *додайте ще один параметр для контролю генерації зображень (наприклад: Width та Height, Negative Prompt, температуру тощо) та відповідний віджет.*

Блок 5: Функції для інтерполяції промптів

```
output_interpolation = widgets.Output()

def interpolate_prompts(prompt1, prompt2, steps, guidance_scale,
num_inference_steps, seed=None, transition_speed=1.0):
    images = []
    for i in range(steps + 1):
        t = (i / steps) ** transition_speed
        interpolated_prompt = f"{prompt1} {(1-t):.2f}, {prompt2} {t:.2f}"
        if seed is not None:
            seed_i = seed + i  # змінюємо seed для кожного кроку, щоб
отримати різні, але послідовні зображення
        else:
            seed_i = None
        image = generate_image(interpolated_prompt, 1, guidance_scale,
num_inference_steps, seed_i)[0]
        images.append(image)
    return images

def interpolate_prompts_button_clicked(b):
    with output_interpolation:
        clear_output(wait=True)
        seed = np.random.randint(0, 1000000) if seed_widget_interp.value ==
0 else seed_widget_interp.value
        images = interpolate_prompts(prompt1.value, prompt2.value,
steps.value,
                                     guidance_scale_interp.value,
num_inference_steps_interp.value,
                                     seed, transition_speed.value)
        show_images(images)
        interpolate_button.images = images  # зберігаємо інтерпольовані
зображення

prompt1 = widgets.Text(value="A portrait of a young person",
description="Prompt 1:", style={'description_width': 'initial'})
prompt2 = widgets.Text(value="A portrait of an old person",
description="Prompt 2:", style={'description_width': 'initial'})
steps = widgets.IntSlider(min=2, max=20, step=1, value=10,
description="Steps:", style={'description_width': 'initial'})
```

```
guidance_scale_interp = widgets.FloatSlider(min=1, max=20, step=0.5,
value=7.5, description="Guidance scale:", style={'description_width':
'initial'})
num_inference_steps_interp = widgets.IntSlider(min=10, max=100, step=10,
value=50, description="Inference steps:", style={'description_width':
'initial'})
seed_widget_interp = widgets.IntText(value=0, description="Seed (0 for
random):", style={'description_width': 'initial'})
transition_speed = widgets.FloatSlider(min=0.1, max=2.0, step=0.1,
value=1.0, description="Transition speed:", style={'description_width':
'initial'})
interpolate_button = widgets.Button(description="Interpolate Prompts")

interpolate_button.on_click(interpolate_prompts_button_clicked)
```

Завдання 5: *реалізуйте функцію для інтерполяції між трьома промптами та створіть відповідний віджет.*

Блок 6: Функції для обробки та аналізу зображень

```
def image_to_base64(image):
    buffered = io.BytesIO()
    image.save(buffered, format="PNG")
    return base64.b64encode(buffered.getvalue()).decode()

def analyze_image(image):
    img_array = np.array(image)
    gray = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY)
    edges = cv2.Canny(gray, 100, 200)
    plt.figure(figsize=(10, 5))
    plt.subplot(121), plt.imshow(image), plt.title('Original')
    plt.subplot(122), plt.imshow(edges, cmap='gray'), plt.title('Edge
Detection')
    plt.show()

def analyze_image_button_clicked(b):
    if hasattr(generate_button, 'images') and len(generate_button.images) >
0:
        analyze_image(generate_button.images[0])
    else:
        print("No image generated yet. Please generate an image first.")

analyze_button = widgets.Button(description="Analyze Last Generated Image")
analyze_button.on_click(analyze_image_button_clicked)
```

Завдання 6: *додайте інші методи аналізу зображень (наприклад, гістограма кольорів або виявлення об'єктів) та відповідні кнопки.*

Блок 7: Функції для збереження та завантаження зображень/відео

```
def save_image_button_clicked(b):
    if hasattr(generate_button, 'images') and len(generate_button.images) >
0:
        generate_button.images[0].save("generated_image.png")
        print("Image saved as 'generated_image.png'")
    else:
        print("No image generated yet. Please generate an image first.")

save_button = widgets.Button(description="Save Last Generated Image")
save_button.on_click(save_image_button_clicked)

def create_smooth_video(images, output_path, fps):
    height, width, _ = np.array(images[0]).shape
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

    for i in range(len(images)-1):
        img1 = cv2.cvtColor(np.array(images[i]), cv2.COLOR_RGB2BGR)
        img2 = cv2.cvtColor(np.array(images[i+1]), cv2.COLOR_RGB2BGR)

        flow = cv2.calcOpticalFlowFarneback(cv2.cvtColor(img1,
cv2.COLOR_BGR2GRAY),
                                            cv2.cvtColor(img2,
cv2.COLOR_BGR2GRAY),
                                            None, 0.5, 3, 15, 3, 5, 1.2, 0)

        for j in range(fps):
            t = j / fps
            warped = cv2.remap(img1, (flow[..., 0] * t +
np.arange(width)).astype(np.float32),
                               (flow[..., 1] * t + np.arange(height)[:,
np.newaxis]).astype(np.float32),
                               cv2.INTER_LINEAR)
            out.write(warped)

    out.release()

def create_video_button_clicked(b):
    if hasattr(interpolate_button, 'images') and
len(interpolate_button.images) > 0:
        output_path = "interpolation_video.mp4"
        create_smooth_video(interpolate_button.images, output_path,
fps.value)
        print(f"Video saved as '{output_path}'")
    else:
        print("No interpolated images yet. Please interpolate prompts
first.")

create_video_button = widgets.Button(description="Create Smooth Video")
create_video_button.on_click(create_video_button_clicked)
```

```
fps = widgets.IntSlider(min=10, max=60, step=1, value=30,
description="FPS:", style={'description_width': 'initial'})
```

Завдання 7: *реалізуйте функцію для завантаження користувацького зображення та його аналізу.*

Блок 8: Головний інтерфейс

```
# @title Головний інтерфейс
tab = widgets.Tab()
tab.children = [
    widgets.VBox([
        widgets.HBox([prompt, num_images]),
        widgets.HBox([guidance_scale, num_inference_steps]),
        widgets.HBox([seed_widget, random_seed_button]),
        generate_button,
        output_images,
        widgets.HBox([analyze_button, save_button])
    ]),
    widgets.VBox([
        widgets.HBox([prompt1, prompt2]),
        widgets.HBox([steps, guidance_scale_interp,
num_inference_steps_interp]),
        widgets.HBox([seed_widget_interp, transition_speed]),
        interpolate_button,
        output_interpolation,
        widgets.HBox([fps, create_video_button])
    ])
]
tab.set_title(0, "Generate Images")
tab.set_title(1, "Interpolate Prompts")
display(tab)
```

Рисунок 13.1. Результат використання SD 1.5 в базовому прикладі

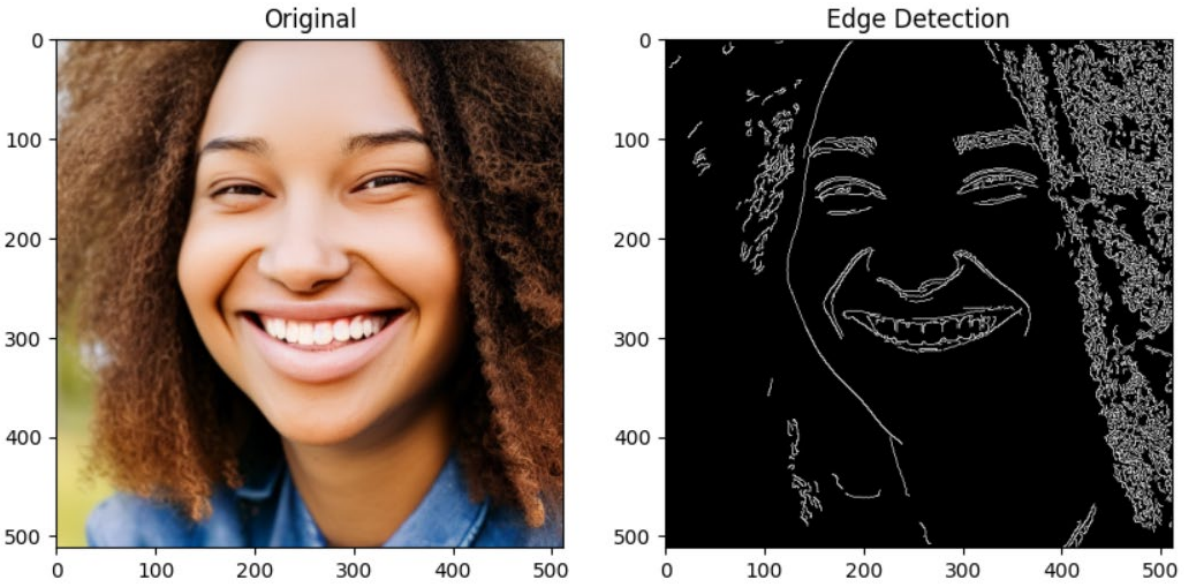


Рисунок 13.2. Результат аналізу зображення в базовому прикладі

Рисунок 13.3. Результат інтерполяції зображень в базовому прикладі

Приклад виконання роботи

Блок 1: Встановлення та імпорт бібліотек

```
!pip install transformers torch torchvision ipywidgets matplotlib pillow
!pip install git+https://github.com/huggingface/diffusers.git
!pip install opencv-python

import torch
import torchvision.transforms as transforms
from torchvision.utils import make_grid
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
from diffusers import StableDiffusionPipeline, DPMSolverMultistepScheduler
import ipywidgets as widgets
from IPython.display import display, clear_output
import cv2
import io
import base64
import glob
```

```
import os
```

Завдання 1: *дослідіть документацію встановлених бібліотек та поясніть, для чого кожна з них використовується в цьому проєкті.*

Відповідь:

`torch` – ...

`torchvision.transforms` – ...

...

Блок 2: Завантаження моделі

```
model_id = "runwayml/stable-diffusion-v1-5"
pipe = StableDiffusionPipeline.from_pretrained(model_id,
torch_dtype=torch.float16)
pipe.scheduler =
DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
pipe = pipe.to("cuda")
```

Завдання 2: *знайдіть інші моделі на Hugging Face, які можна використати замість "**`runwayml/stable-diffusion-v1-5`**". Спробуйте завантажити та порівняти їх.*

Відповідь:

```
models = {
    "SD 1.5": "runwayml/stable-diffusion-v1-5",
    "SD 2.1": "stabilityai/stable-diffusion-2-1"
}

current_model = "SD 1.5"
pipe = StableDiffusionPipeline.from_pretrained(models[current_model],
torch_dtype=torch.float16)
pipe.scheduler =
DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
pipe = pipe.to("cuda")

def change_model(model_name):
    global pipe, current_model
    current_model = model_name
    pipe = StableDiffusionPipeline.from_pretrained(models[model_name],
torch_dtype=torch.float16)
    pipe.scheduler =
DPMSolverMultistepScheduler.from_config(pipe.scheduler.config)
    pipe = pipe.to("cuda")
```

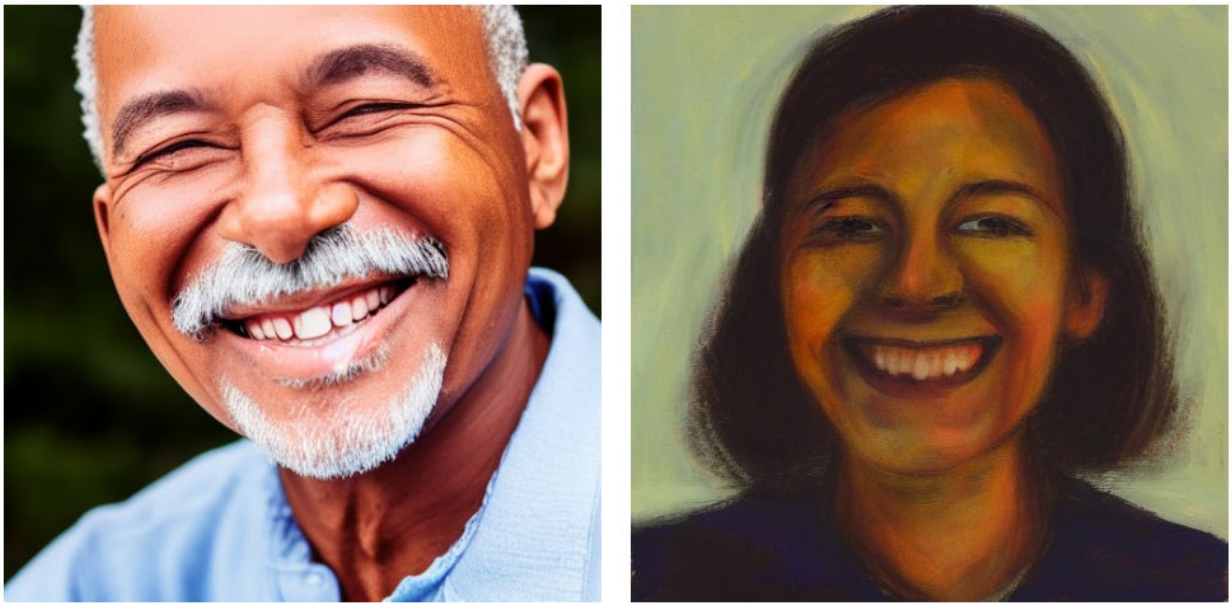


Рисунок 13.4. Результат використання SD 1.5 (за однакових параметрів)

Рисунок 13.5. Результат використання SD 2.1 (за однакових параметрів)

Блок 3: Функції для генерації та відображення зображень

```
def generate_image(prompt, num_images=1, guidance_scale=7.5,
num_inference_steps=50, seed=None):
    if seed is not None:
        generator = torch.Generator("cuda").manual_seed(seed)
    else:
        generator = None
    with torch.no_grad():
        images = pipe(prompt, num_images_per_prompt=num_images,
guidance_scale=guidance_scale,
                      num_inference_steps=num_inference_steps,
generator=generator).images
    return images
```

```
def show_images(images):
    fig, axes = plt.subplots(1, len(images), figsize=(15, 5))
    for i, img in enumerate(images):
        if len(images) == 1:
            axes.imshow(img)
            axes.axis('off')
        else:
            axes[i].imshow(img)
            axes[i].axis('off')
    plt.tight_layout()
    plt.show()

def slerp(t, v0, v1, DOT_THRESHOLD=0.9995):
    dot = torch.sum(v0 * v1 / (torch.norm(v0) * torch.norm(v1)))
    if abs(dot) > DOT_THRESHOLD:
        v2 = (1 - t) * v0 + t * v1
    else:
        theta_0 = torch.acos(dot)
        sin_theta_0 = torch.sin(theta_0)
        theta_t = theta_0 * t
        sin_theta_t = torch.sin(theta_t)
        s0 = torch.sin(theta_0 - theta_t) / sin_theta_0
        s1 = sin_theta_t / sin_theta_0
        v2 = s0 * v0 + s1 * v1
    return v2
```

Завдання 3: *модифікуйте функцію* ***`show_images`****, щоб вона могла відображати довільну кількість зображень у сітці (наприклад, 3x3 або 4x4).*

Відповідь:

```
def show_images(images, rows=None, cols=None):
    n = len(images)
    if n == 0:
        print("No images to display")
        return

    if rows is None and cols is None:
        cols = int(np.ceil(np.sqrt(n)))
        rows = int(np.ceil(n / cols))
    elif rows is None:
        rows = int(np.ceil(n / cols))
    elif cols is None:
        cols = int(np.ceil(n / rows))

    fig, axes = plt.subplots(rows, cols, figsize=(cols*5, rows*5))

    if n == 1:
        axes = np.array([axes])  # Перетворюємо в масив для уніфікації
обробки
    else:
        axes = axes.flatten()
```

```
    for i, img in enumerate(images):
        if i < n:
            axes[i].imshow(img)
            axes[i].axis('off')
        else:
            fig.delaxes(axes[i])

    for i in range(n, len(axes)):
        fig.delaxes(axes[i])

    plt.tight_layout()
    plt.show()
```

Блок 4: Інтерфейс для генерації зображень

```
output_images = widgets.Output()

def generate_images_button_clicked(b):
    with output_images:
        clear_output(wait=True)
        images = generate_image(prompt.value, num_images.value,
guidance_scale.value, num_inference_steps.value, seed_widget.value)
        show_images(images)
        generate_button.images = images  # зберігаємо згенеровані зображення

prompt = widgets.Text(value="A portrait of a smiling person",
description="Prompt:", style={'description_width': 'initial'})
num_images = widgets.IntSlider(min=1, max=4, step=1, value=1,
description="Number of images:", style={'description_width': 'initial'})
guidance_scale = widgets.FloatSlider(min=1, max=20, step=0.5, value=7.5,
description="Guidance scale:", style={'description_width': 'initial'})
num_inference_steps = widgets.IntSlider(min=10, max=100, step=10, value=50,
description="Inference steps:", style={'description_width': 'initial'})
seed_widget = widgets.IntText(value=42, description="Seed:",
style={'description_width': 'initial'})
random_seed_button = widgets.Button(description="Random Seed")
generate_button = widgets.Button(description="Generate Images")

def on_random_seed_click(b):
    seed_widget.value = np.random.randint(0, 1000000)

random_seed_button.on_click(on_random_seed_click)
generate_button.on_click(generate_images_button_clicked)
```

Завдання 4: *додайте ще один параметр для контролю генерації зображень (наприклад: Width та Height, Negative Prompt, температуру тощо) та відповідний віджет.*

Відповідь:

```
def generate_image(prompt, negative_prompt="", num_images=1,
guidance_scale=7.5, num_inference_steps=50, seed=None, width=512,
height=512, temperature=1.0):
    if seed is not None:
        generator = torch.Generator("cuda").manual_seed(seed)
    else:
        generator = None

    with torch.no_grad():
        # Генеруємо латентний простір з типом float16
        latents = torch.randn((num_images, pipe.unet.config.in_channels,
height // 8, width // 8),
                              generator=generator, device="cuda",
dtype=torch.float16) * temperature

        # Перетворюємо промпти в тензори з типом float16
        text_inputs = pipe.tokenizer(
            prompt,
            padding="max_length",
            max_length=pipe.tokenizer.model_max_length,
            truncation=True,
            return_tensors="pt",
        )
        text_input_ids = text_inputs.input_ids.to(pipe.device)
        prompt_embeds = pipe.text_encoder(text_input_ids)[0].half()

        # Обробляємо негативний промпт
        if negative_prompt:
            uncond_input = pipe.tokenizer(
                negative_prompt,
                padding="max_length",
                max_length=text_input_ids.shape[-1],
                truncation=True,
                return_tensors="pt",
            )
            uncond_input_ids = uncond_input.input_ids.to(pipe.device)
            negative_prompt_embeds =
pipe.text_encoder(uncond_input_ids)[0].half()
        else:
            # Якщо негативний промпт не вказано, створюємо пустий ембединг
            negative_prompt_embeds = torch.zeros_like(prompt_embeds)

        # Переконуємось, що форми співпадають
        if prompt_embeds.shape[1] != negative_prompt_embeds.shape[1]:
            negative_prompt_embeds = negative_prompt_embeds.repeat(1,
prompt_embeds.shape[1], 1)

        images = pipe(prompt_embeds=prompt_embeds,
                      negative_prompt_embeds=negative_prompt_embeds,
                      num_images_per_prompt=num_images,
                      guidance_scale=guidance_scale,
```

```
                        num_inference_steps=num_inference_steps,
                        latents=latents,
                        width=width,
                        height=height).images
    return images

output_images = widgets.Output()

def generate_images_button_clicked(b):
    with output_images:
        clear_output(wait=True)
        images = generate_image(prompt.value, negative_prompt.value,
num_images.value,
                                guidance_scale.value,
num_inference_steps.value, seed_widget.value,
                                width.value, height.value,
temperature.value)
        show_images(images)
        generate_button.images = images

prompt = widgets.Text(value="A portrait of a smiling person",
description="Prompt:", style={'description_width': 'initial'})
negative_prompt = widgets.Text(value="", description="Negative Prompt:",
style={'description_width': 'initial'})
num_images = widgets.IntSlider(min=1, max=9, step=1, value=1,
description="Number of images:", style={'description_width': 'initial'})
guidance_scale = widgets.FloatSlider(min=1, max=20, step=0.5, value=7.5,
description="Guidance scale:", style={'description_width': 'initial'})
num_inference_steps = widgets.IntSlider(min=10, max=100, step=10, value=50,
description="Inference steps:", style={'description_width': 'initial'})
seed_widget = widgets.IntText(value=42, description="Seed:",
style={'description_width': 'initial'})
width = widgets.IntSlider(min=256, max=1024, step=64, value=512,
description="Width:", style={'description_width': 'initial'})
height = widgets.IntSlider(min=256, max=1024, step=64, value=512,
description="Height:", style={'description_width': 'initial'})
temperature = widgets.FloatSlider(min=0.1, max=2.0, step=0.1, value=1.0,
description="Temperature:", style={'description_width': 'initial'})

random_seed_button = widgets.Button(description="Random Seed")
generate_button = widgets.Button(description="Generate Images")

def on_random_seed_click(b):
    seed_widget.value = np.random.randint(0, 1000000)

random_seed_button.on_click(on_random_seed_click)
generate_button.on_click(generate_images_button_clicked)
```

Блок 5: Функції для інтерполяції промптів

```
output_interpolation = widgets.Output()

def interpolate_prompts(prompt1, prompt2, steps, guidance_scale,
num_inference_steps, seed=None, transition_speed=1.0):
    images = []
    for i in range(steps + 1):
        t = (i / steps) ** transition_speed
        interpolated_prompt = f"{prompt1} {(1-t):.2f}, {prompt2} {t:.2f}"
        if seed is not None:
            seed_i = seed + i  # змінюємо seed для кожного кроку, щоб
отримати різні, але послідовні зображення
        else:
            seed_i = None
        image = generate_image(interpolated_prompt, 1, guidance_scale,
num_inference_steps, seed_i)[0]
        images.append(image)
    return images

def interpolate_prompts_button_clicked(b):
    with output_interpolation:
        clear_output(wait=True)
        seed = np.random.randint(0, 1000000) if seed_widget_interp.value ==
0 else seed_widget_interp.value
        images = interpolate_prompts(prompt1.value, prompt2.value,
steps.value,
                                     guidance_scale_interp.value,
num_inference_steps_interp.value,
                                     seed, transition_speed.value)
        show_images(images)
        interpolate_button.images = images  # зберігаємо інтерпольовані
зображення

prompt1 = widgets.Text(value="A portrait of a young person",
description="Prompt 1:", style={'description_width': 'initial'})
prompt2 = widgets.Text(value="A portrait of an old person",
description="Prompt 2:", style={'description_width': 'initial'})
steps = widgets.IntSlider(min=2, max=20, step=1, value=10,
description="Steps:", style={'description_width': 'initial'})
guidance_scale_interp = widgets.FloatSlider(min=1, max=20, step=0.5,
value=7.5, description="Guidance scale:", style={'description_width':
'initial'})
num_inference_steps_interp = widgets.IntSlider(min=10, max=100, step=10,
value=50, description="Inference steps:", style={'description_width':
'initial'})
seed_widget_interp = widgets.IntText(value=0, description="Seed (0 for
random):", style={'description_width': 'initial'})
transition_speed = widgets.FloatSlider(min=0.1, max=2.0, step=0.1,
value=1.0, description="Transition speed:", style={'description_width':
'initial'})
interpolate_button = widgets.Button(description="Interpolate Prompts")
```

```
interpolate_button.on_click(interpolate_prompts_button_clicked)
```

Завдання 5: *реалізуйте функцію для інтерполяції між трьома промптами та створіть відповідний віджет.*

Відповідь:

```
output_interpolation = widgets.Output()

def interpolate_prompts(prompt1, prompt2, prompt3, steps, guidance_scale,
num_inference_steps, seed=None, transition_speed=1.0):
    images = []
    total_steps = steps * 2

    for i in range(total_steps + 1):
        t = (i / total_steps) ** transition_speed
        if i <= steps:
            interpolated_prompt = f"{prompt1} {(1-t*2):.2f}, {prompt2}
{t*2:.2f}"
        else:
            t = (i - steps) / steps
            interpolated_prompt = f"{prompt2} {(1-t):.2f}, {prompt3}
{t:.2f}"

        if seed is not None:
            seed_i = seed + i
        else:
            seed_i = None
        image = generate_image(interpolated_prompt, num_images=1,
guidance_scale=guidance_scale,
                               num_inference_steps=num_inference_steps,
seed=seed_i)[0]
        images.append(image)
    return images

def interpolate_prompts_button_clicked(b):
    with output_interpolation:
        clear_output(wait=True)
        seed = np.random.randint(0, 1000000) if seed_widget_interp.value ==
0 else seed_widget_interp.value
        images = interpolate_prompts(prompt1.value, prompt2.value,
prompt3.value, steps.value,
                                     guidance_scale_interp.value,
num_inference_steps_interp.value,
                                     seed, transition_speed.value)
        show_images(images)
        interpolate_button.images = images

prompt1 = widgets.Text(value="A portrait of a young person",
description="Prompt 1:", style={'description_width': 'initial'})
prompt2 = widgets.Text(value="A portrait of a middle-aged person",
description="Prompt 2:", style={'description_width': 'initial'})
```

```
prompt3 = widgets.Text(value="A portrait of an old person",
description="Prompt 3:", style={'description_width': 'initial'})
steps = widgets.IntSlider(min=2, max=20, step=1, value=10,
description="Steps per transition:", style={'description_width': 'initial'})
guidance_scale_interp = widgets.FloatSlider(min=1, max=20, step=0.5,
value=7.5, description="Guidance scale:", style={'description_width':
'initial'})
num_inference_steps_interp = widgets.IntSlider(min=10, max=100, step=10,
value=50, description="Inference steps:", style={'description_width':
'initial'})
seed_widget_interp = widgets.IntText(value=0, description="Seed (0 for
random):", style={'description_width': 'initial'})
transition_speed = widgets.FloatSlider(min=0.1, max=2.0, step=0.1,
value=1.0, description="Transition speed:", style={'description_width':
'initial'})
interpolate_button = widgets.Button(description="Interpolate Prompts")

interpolate_button.on_click(interpolate_prompts_button_clicked)
```

Блок 6: Функції для обробки та аналізу зображень

```
def image_to_base64(image):
    buffered = io.BytesIO()
    image.save(buffered, format="PNG")
    return base64.b64encode(buffered.getvalue()).decode()

def analyze_image(image):
    img_array = np.array(image)
    gray = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY)
    edges = cv2.Canny(gray, 100, 200)
    plt.figure(figsize=(10, 5))
    plt.subplot(121), plt.imshow(image), plt.title('Original')
    plt.subplot(122), plt.imshow(edges, cmap='gray'), plt.title('Edge
Detection')
    plt.show()

def analyze_image_button_clicked(b):
    if hasattr(generate_button, 'images') and len(generate_button.images) >
0:
        analyze_image(generate_button.images[0])
    else:
        print("No image generated yet. Please generate an image first.")

analyze_button = widgets.Button(description="Analyze Last Generated Image")
analyze_button.on_click(analyze_image_button_clicked)
```

Завдання 6: *додайте інші методи аналізу зображень (наприклад, гістограма кольорів або виявлення об'єктів) та відповідні кнопки.*

Відповідь:

```
import traceback

def analyze_image(image):
    try:
        img_array = np.array(image)
        print(f"Image shape: {img_array.shape}")
        print(f"Image dtype: {img_array.dtype}")

        # Edge detection
        gray = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY)
        edges = cv2.Canny(gray, 100, 200)

        # Color histogram
        color = ('r','g','b')
        plt.figure(figsize=(20, 15))
        plt.subplot(221)
        plt.title("Color Histogram")
        for i, col in enumerate(color):
            histr = cv2.calcHist([img_array],[i],None,[256],[0,256])
            plt.plot(histr, color = col)
            plt.xlim([0,256])

        # Display results
        plt.subplot(222)
        plt.imshow(image)
        plt.title('Original')
        plt.axis('off')

        plt.subplot(223)
        plt.imshow(edges, cmap='gray')
        plt.title('Edge Detection')
        plt.axis('off')

        plt.subplot(224)
        plt.imshow(img_array)
        plt.title('RGB Image')
        plt.axis('off')

        plt.tight_layout()
        plt.show()

        print("Image analysis completed successfully.")
    except Exception as e:
        print(f"An error occurred during image analysis: {str(e)}")
        print("Traceback:")
        traceback.print_exc()

def analyze_image_button_clicked(b):
    if hasattr(generate_button, 'images') and len(generate_button.images) >
0:
        print("Starting image analysis...")
```

```
        analyze_image(generate_button.images[0])
    else:
        print("No image generated yet. Please generate an image first.")
```

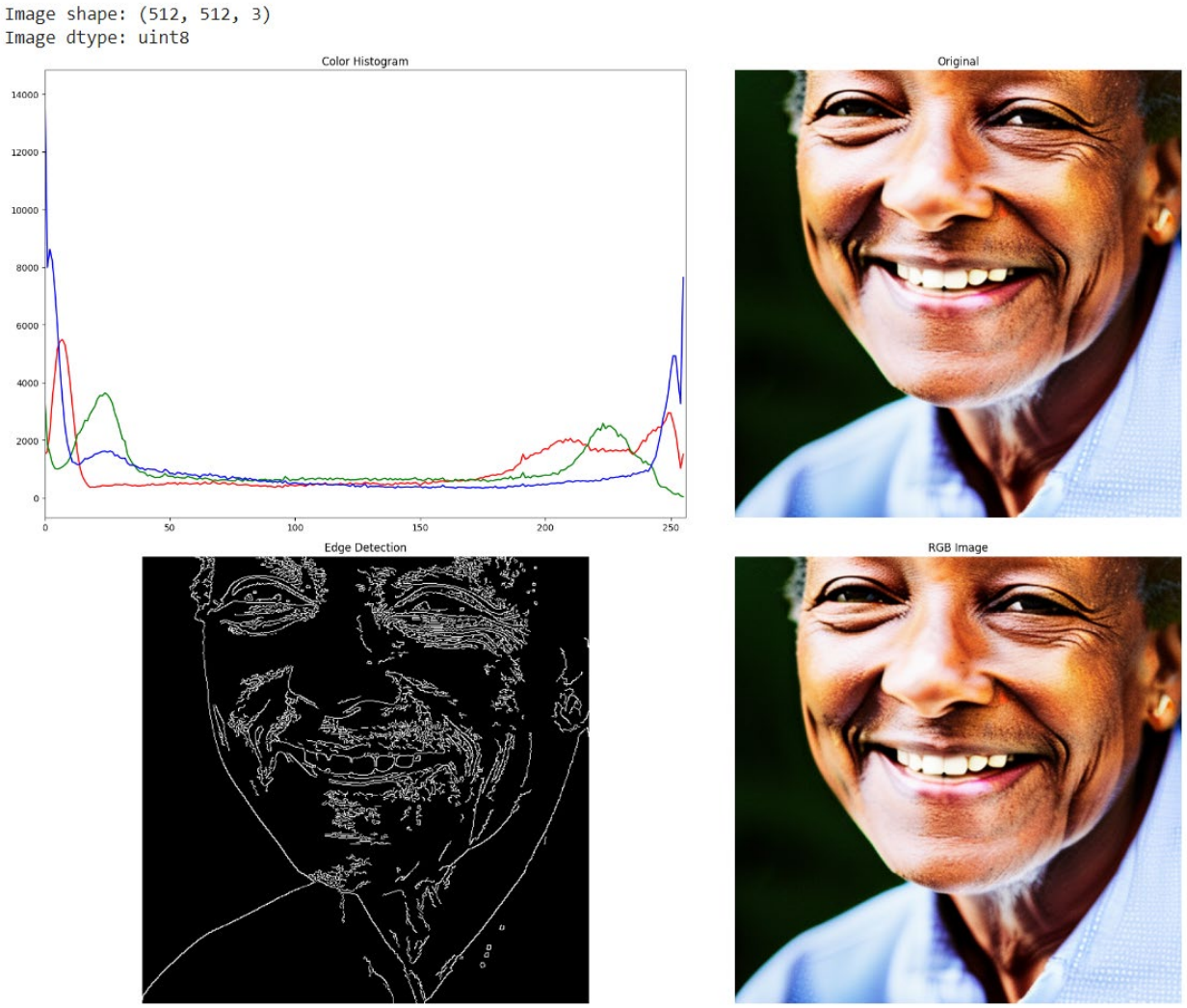


Рисунок 13.6. Приклад розширеного аналізу зображень

Блок 7: Функції для збереження та завантаження зображень/відео

```
def save_image_button_clicked(b):
    if hasattr(generate_button, 'images') and len(generate_button.images) >
0:
        generate_button.images[0].save("generated_image.png")
        print("Image saved as 'generated_image.png'")
    else:
        print("No image generated yet. Please generate an image first.")

save_button = widgets.Button(description="Save Last Generated Image")
save_button.on_click(save_image_button_clicked)

def create_smooth_video(images, output_path, fps):
    height, width, _ = np.array(images[0]).shape
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))
```

```
    for i in range(len(images)-1):
        img1 = cv2.cvtColor(np.array(images[i]), cv2.COLOR_RGB2BGR)
        img2 = cv2.cvtColor(np.array(images[i+1]), cv2.COLOR_RGB2BGR)

        flow = cv2.calcOpticalFlowFarneback(cv2.cvtColor(img1,
cv2.COLOR_BGR2GRAY),
                                            cv2.cvtColor(img2,
cv2.COLOR_BGR2GRAY),
                                            None, 0.5, 3, 15, 3, 5, 1.2, 0)

        for j in range(fps):
            t = j / fps
            warped = cv2.remap(img1, (flow[..., 0] * t +
np.arange(width)).astype(np.float32),
                               (flow[..., 1] * t + np.arange(height)[:,
np.newaxis]).astype(np.float32),
                               cv2.INTER_LINEAR)
            out.write(warped)

    out.release()

def create_video_button_clicked(b):
    if hasattr(interpolate_button, 'images') and
len(interpolate_button.images) > 0:
        output_path = "interpolation_video.mp4"
        create_smooth_video(interpolate_button.images, output_path,
fps.value)
        print(f"Video saved as '{output_path}'")
    else:
        print("No interpolated images yet. Please interpolate prompts
first.")

create_video_button = widgets.Button(description="Create Smooth Video")
create_video_button.on_click(create_video_button_clicked)

fps = widgets.IntSlider(min=10, max=60, step=1, value=30,
description="FPS:", style={'description_width': 'initial'})
```

Завдання 7: *реалізуйте функцію для завантаження користувацького зображення та його аналізу.*

Відповідь:

```
def upload_and_analyze_image(b):
    clear_output()
    print("Please upload an image file.")
    uploaded = files.upload()

    for filename in uploaded.keys():
        content = uploaded[filename]
        image = Image.open(io.BytesIO(content))
        analyze_image(image)
```

```
upload_button = widgets.Button(description="Upload and Analyze Image")
upload_button.on_click(upload_and_analyze_image)
```

Зміни у графічному інтерфейсі:

```
model_dropdown = widgets.Dropdown(options=list(models.keys()),
value=current_model, description="Model:")
model_dropdown.observe(lambda change: change_model(change.new),
names='value')

tab = widgets.Tab()
tab.children = [
    widgets.VBox([
        model_dropdown,
        widgets.HBox([prompt, negative_prompt]),
        widgets.HBox([num_images, guidance_scale, num_inference_steps]),
        widgets.HBox([seed_widget, random_seed_button]),
        widgets.HBox([width, height, temperature]),
        generate_button,
        output_images,
        widgets.HBox([analyze_button, upload_button])
    ]),
    widgets.VBox([
        widgets.HBox([prompt1, prompt2, prompt3]),
        widgets.HBox([steps, guidance_scale_interp,
num_inference_steps_interp]),
        widgets.HBox([seed_widget_interp, transition_speed]),
        interpolate_button,
        output_interpolation
    ])
]
tab.set_title(0, "Generate Images")
tab.set_title(1, "Interpolate Prompts")
display(tab)
```

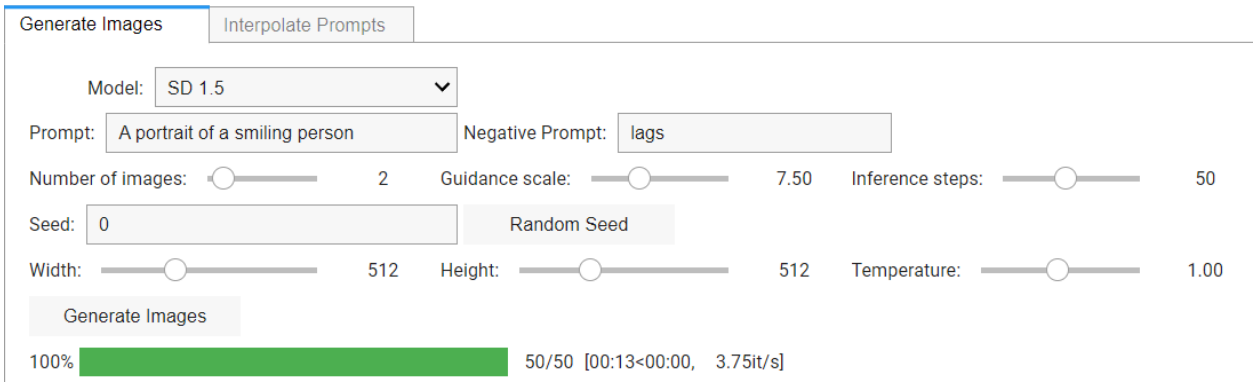


Рисунок 13.7. Вигляд інтерфейсу №1

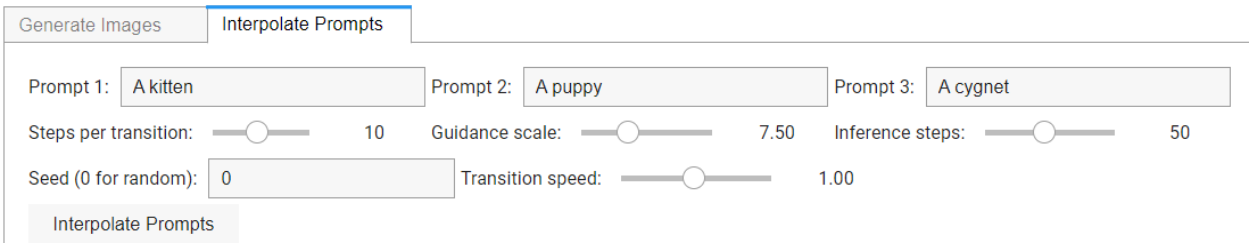


Рисунок 13.8. Вигляд інтерфейсу №2

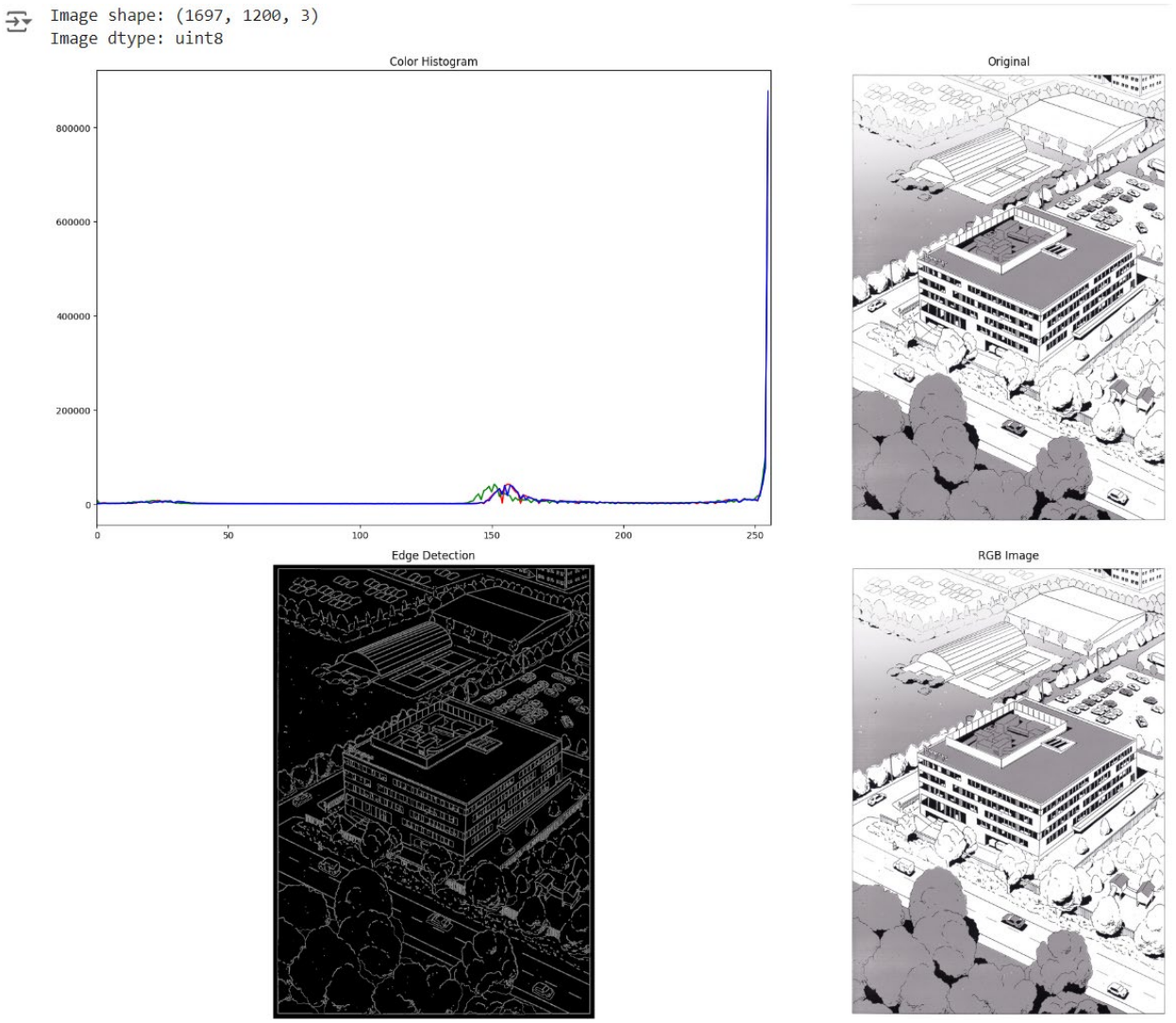


Рисунок 13.9. Приклад аналізу зображення, що завантажується

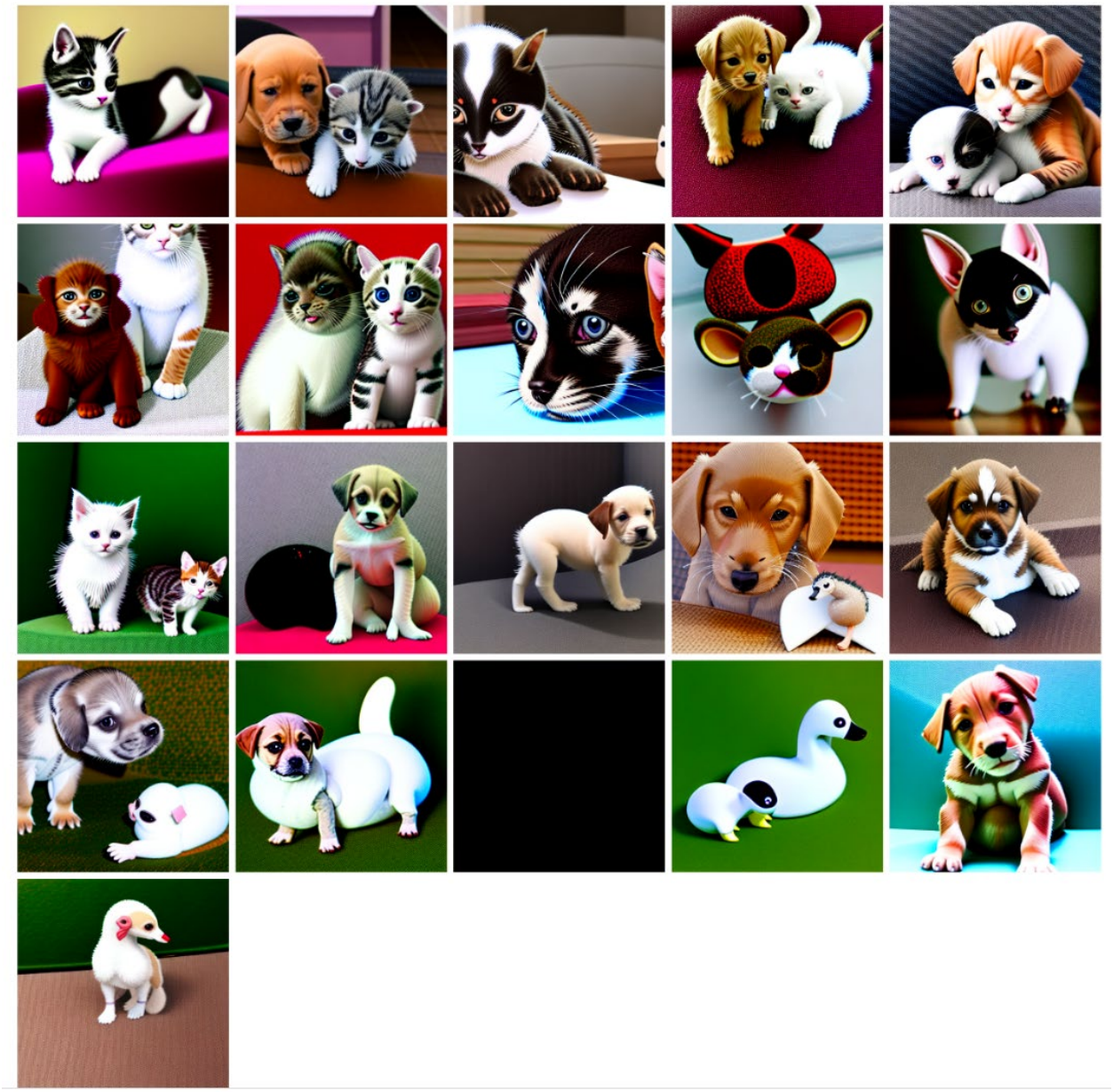


Рисунок 13.10. Приклад інтерполяції зображень за 3 промптами

Висновки: …

Welcome to Google Colab

https://colab.research.google.com/drive/1x2T9VPvZntyxEp45KsNist6yJJe4bqz4?usp=sharing

Лабораторна робота № 14. Практичне застосування reinforcement learning у симуляціях

Мета: ознайомитись з принципами налаштування, навчання та оптимізації агентів навчання з підкріпленням через веб-інтерфейс. Отримати практичний досвід налаштування різних гіперпараметрів та розуміння їх впливу на ефективність навчання.

Теоретичні відомості

Параметри середовища

Параметр	Опис	Вплив на навчання
Grid Size	Розмір сітки середовища (наприклад, 4x4).	Визначає доступний простір для агента; впливає на складність навігації та пошук оптимальної стратегії.
Number of Coins	Кількість монет, які агент може зібрати в середовищі.	Чим більше монет, тим більше можливостей для отримання нагород і тренування на основі зібраних даних.
Number of Obstacles	Кількість статичних перешкод на сітці.	Впливає на складність навігації, змушуючи агента ухилятися та стратегічно обирати маршрути.
Dynamic Obstacles	Наявність динамічних перешкод, які змінюють своє положення.	Підвищує складність завдання, стимулюючи агента швидше адаптуватися до змінюваних умов середовища.

Конфігурація нагород та покарань

Параметр	Опис	Вплив на навчання
Coin Collected Reward	Нагорода, яку агент отримує за збирання монети.	Підвищує мотивацію агента шукати монети; впливає на швидкість навчання та оптимізацію траєкторій.
Collision Penalty	Штраф за зіткнення з перешкодою.	Заохочує агента уникати перешкод; зменшує кількість випадкових дій та покращує обережність агента.
Step Penalty	Штраф за кожен крок.	Сприяє ефективності рухів, мотивуючи агента скоротити час досягнення цілей.
Completion Reward	Нагорода за завершення епізоду.	Збільшує мотивацію агента досягти цілі, завершивши епізод; покращує загальну результативність навчання.
Timeout Penalty	Штраф за перевищення ліміту часу епізоду.	Мотивує агента прискорити проходження епізоду, щоб уникнути штрафу, що підвищує ефективність навчання.

Параметри агента

Параметр	Опис	Вплив на навчання
Learning Rate	Швидкість навчання.	Визначає, наскільки швидко агент оновлює свої знання; надто висока або низька швидкість може погіршити навчання.
Batch Size	Розмір вибірки даних, що використовується під час навчання.	Впливає на стабільність оновлень параметрів, оскільки більше вибірок дозволяє обробляти більше інформації за раз.
Gamma (Discount)	Коефіцієнт знижки для майбутніх нагород.	Визначає важливість майбутніх винагород порівняно з поточними, впливаючи на довготривалу стратегію агента.
Initial Epsilon	Початкове значення епсілон для ε-жадібної стратегії.	Впливає на початкову ступінь випадковості в діях агента; поступово зменшується, щоб перейти до кращих рішень.
Epsilon Decay	Співвідношення зменшення епсілон.	Впливає на темп зменшення випадковості в діях агента; стабілізує навчання після достатньої кількості епізодів.
Minimum Epsilon	Мінімальне значення епсілон.	Гарантує певний рівень випадковості, що дозволяє агенту знаходити нові стратегії, навіть на пізніх етапах.
Memory Size	Розмір буфера пам’яті для збереження досвіду.	Збільшує кількість збережених спостережень, що сприяє якіснішому навчанню та запобігає «забуванню».

Параметри тренування

Параметр	Опис	Вплив на навчання
Episodes	Кількість навчальних епізодів.	Визначає тривалість процесу навчання; більше епізодів дозволяє агенту краще адаптуватися до середовища.
Render Every	Інтервал рендерингу епізодів.	Впливає на частоту візуалізації процесу навчання; корисно для моніторингу прогресу агента та налаштування моделі.

Необхідне програмне забезпечення

1. Git 2.47.0+
2. Python 3.8+
3. Node.js 16+
4. npm 8+
5. Веб-браузер з підтримкою сучасного JavaScript

Завдання для всіх варіантів

Завдання 1. Базове налаштування середовища.

1. Встановіть усі залежності, згідно з інструкцією у репозиторії.

2. Запустіть бекенд та фронтенд частини проєкту.

3. Перевірте роботу веб-інтерфейсу:

3.1. Відкрийте веб-інтерфейс у браузері.

3.2. Переконайтесь, що всі елементи та функції інтерфейсу працюють без помилок.

3.3. Зафіксуйте результат: зробіть скріншот робочого інтерфейсу та додайте його до звіту.

Завдання 2. Експерименти з параметрами середовища.

1. Проведіть серію експериментів із різними налаштуваннями середовища:

*1.1. Розмір сітки: випробуйте значення **від 4 до 12**.*

*1.2. Кількість монет: варіюйте **від 1 до 5**.*

*1.3. Кількість перешкод: змінюйте від **0 до 8**.*

*1.4. Динамічні перешкоди: **включіть** і **вимкніть** динамічні перешкоди, щоб порівняти вплив.*

2. Виконайте експерименти з конфігурацією винагород:

*2.1. Coin Collected Reward: змінюйте **від 0 до 5**.*

*2.2. Collision Penalty: випробуйте значення **від 0.5 до 5**.*

*2.3. Step Penalty: використовуйте значення в діапазоні **від 0 до -0.1**.*

*2.4. Completion Reward: задайте **від 0 до 10**.*

*2.5. Timeout Penalty: варіюйте значення **від 0 до 5**.*

3. Заповніть таблицю результатів для кожної конфігурації:

Конфігурація	Середня винагорода	Час навчання	Примітки
…	…	…	…

4. Проаналізуйте результати:

4.1.Визначте вплив кожного параметра на швидкість навчання та продуктивність агента.

4.2.Додайте графік або діаграму для ілюстрації впливу змінних.

Завдання 3. Оптимізація гіперпараметрів агента.

1. Підберіть оптимальні значення для наступних гіперпараметрів агента:

*1.1. Learning Rate: значення **від 0.0001 до 0.01**.*

*1.2. Batch Size: значення **від 16 до 256**.*

*1.3. Gamma (Discount Factor): значення **від 0.8 до 1**.*

*1.4. Initial Epsilon: початкове значення ε, варіюйте **від 0 до 1**.*

*1.5. Epsilon Decay: швидкість зменшення ε **від 0.9 до 1**.*

*1.6. Minimum Epsilon: встановіть мінімальне значення **від 0.01 до 0.3**.*

*1.7. Memory Size: кількість збережених станів **від 1,000 до 50,000**.*

2. Порівняйте результати з базовими налаштуваннями:

Параметр	Базове значення	Оптимізоване значення	Результат
…	…	…	…

3. Обґрунтуйте вибір кожного параметра:

3.1.Опишіть, як зміни впливають на ефективність навчання та результати.

Завдання 4. Аналіз процесу навчання

*1. Проведіть навчання з різною кількістю епізодів: **100**, **500**, **1000**.*

2. Побудуйте графіки, що відображають процес навчання:

2.1. Графік винагород за епізод.

2.2. Графік втрат під час навчання.

2.3. Середня тривалість епізоду.

3. Проаналізуйте момент збіжності навчання:

3.1. Визначте, скільки епізодів потрібно для стабільного навчання.

3.2. Додайте висновки про мінімальну кількість епізодів для ефективного навчання.

Завдання 5. Додаткове завдання.

1. Розширення агента:

1.1. Модифікуйте `agent.py`*, додавши нову стратегію дослідження.*

1.2. Оцініть, наскільки нова стратегія покращує результати порівняно з базовими.

2. Зміни у середовищі:

2.1. Внесіть зміни до `environment.py`*, додавши нові режими гри.*

2.2. Оцініть, як ці зміни впливають на процес навчання.

3. Вдосконалення моделі:

3.1. Покращіть архітектуру мережі у `model.py`*, додавши нові шари або функції активації.*

3.2. Оцініть, як це впливає на швидкість і точність навчання.

** За необхідності внесіть додаткові зміни в програмний код*

Завдання 1. Базове налаштування середовища.

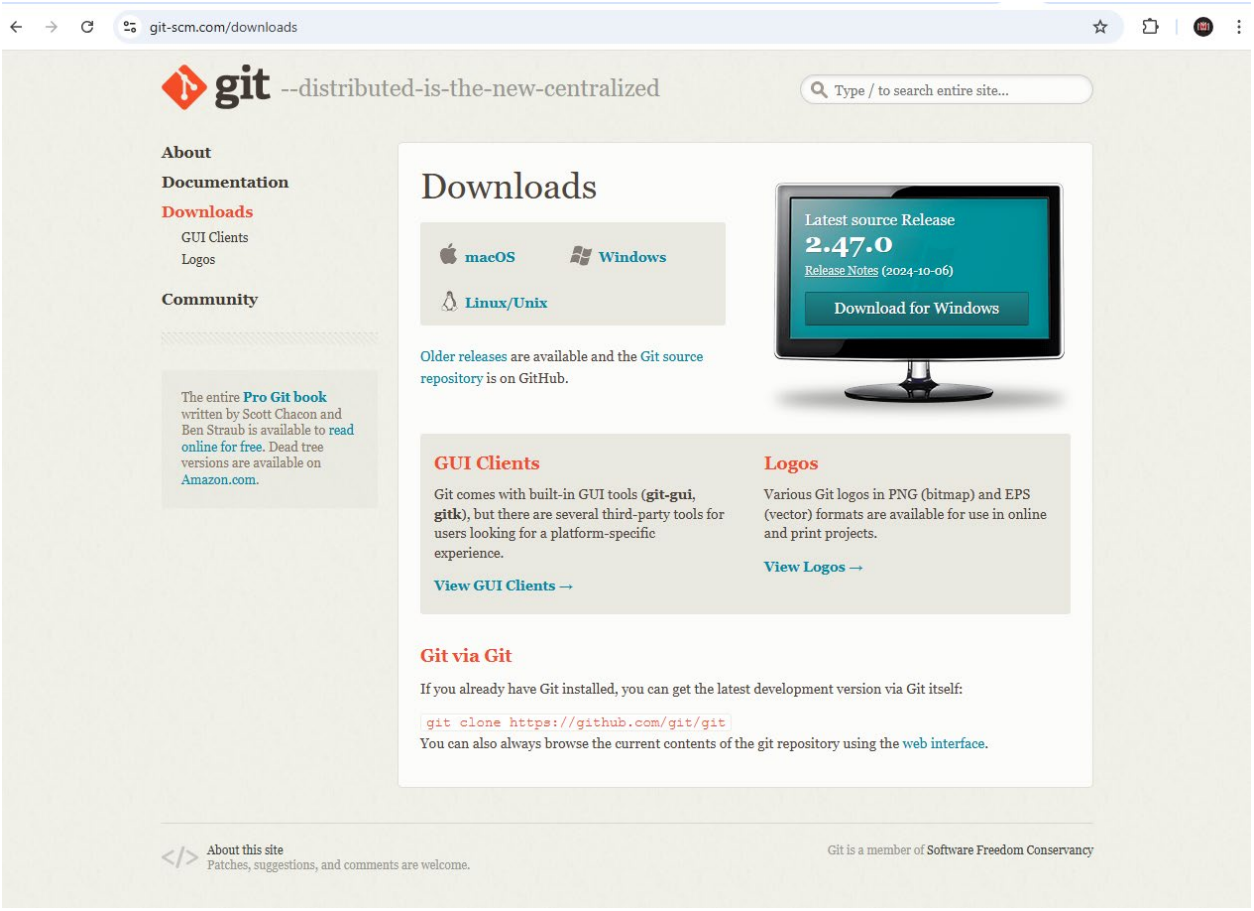


Рисунок 14.1. Встановлення Git

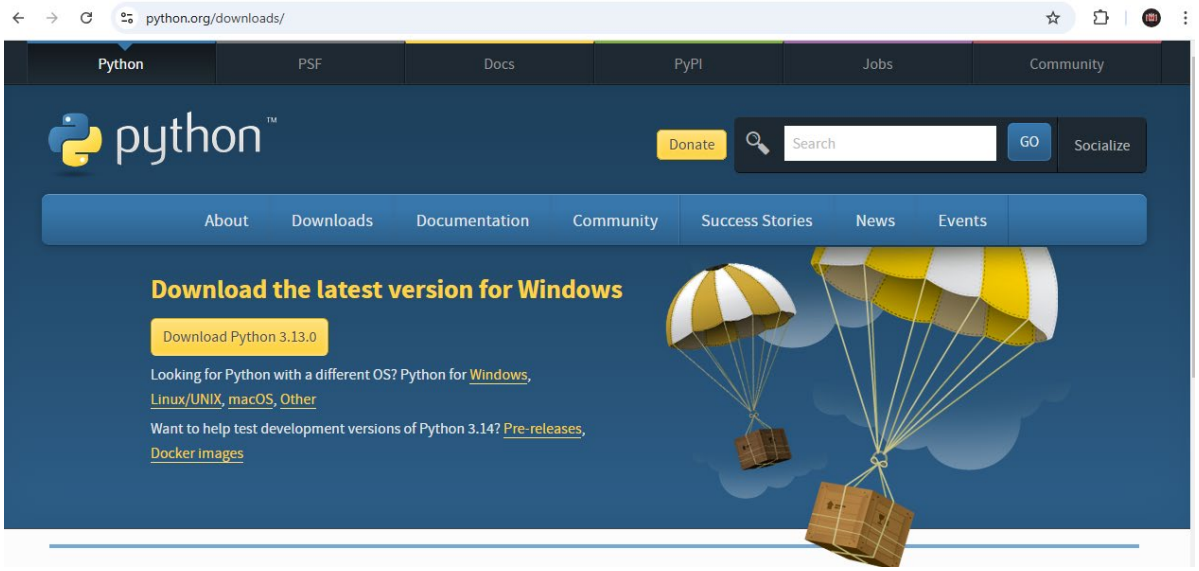


Рисунок 14.2. Встановлення Python

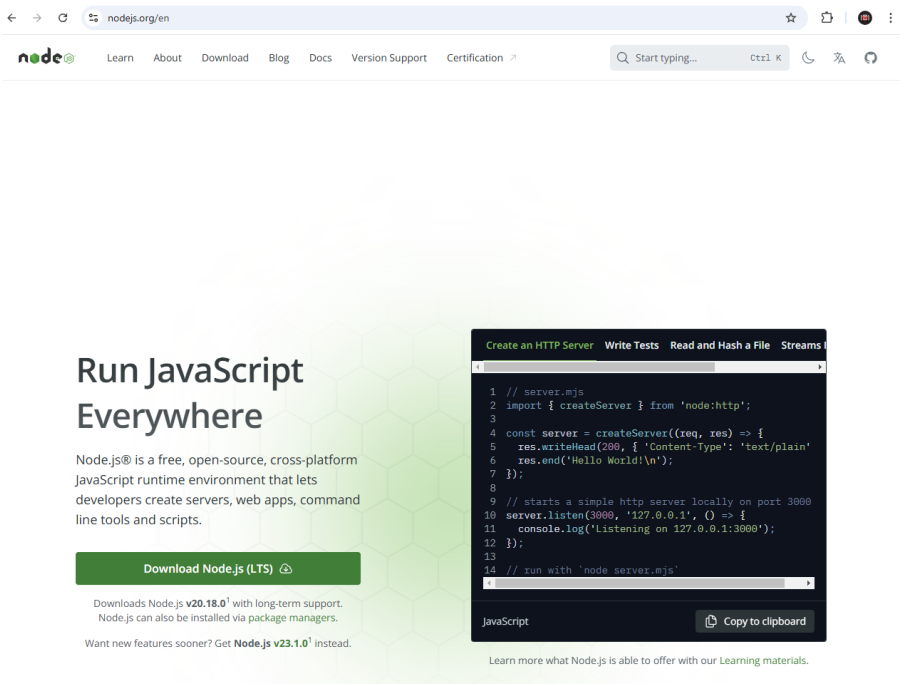


Рисунок 14.3. Встановлення Node.js

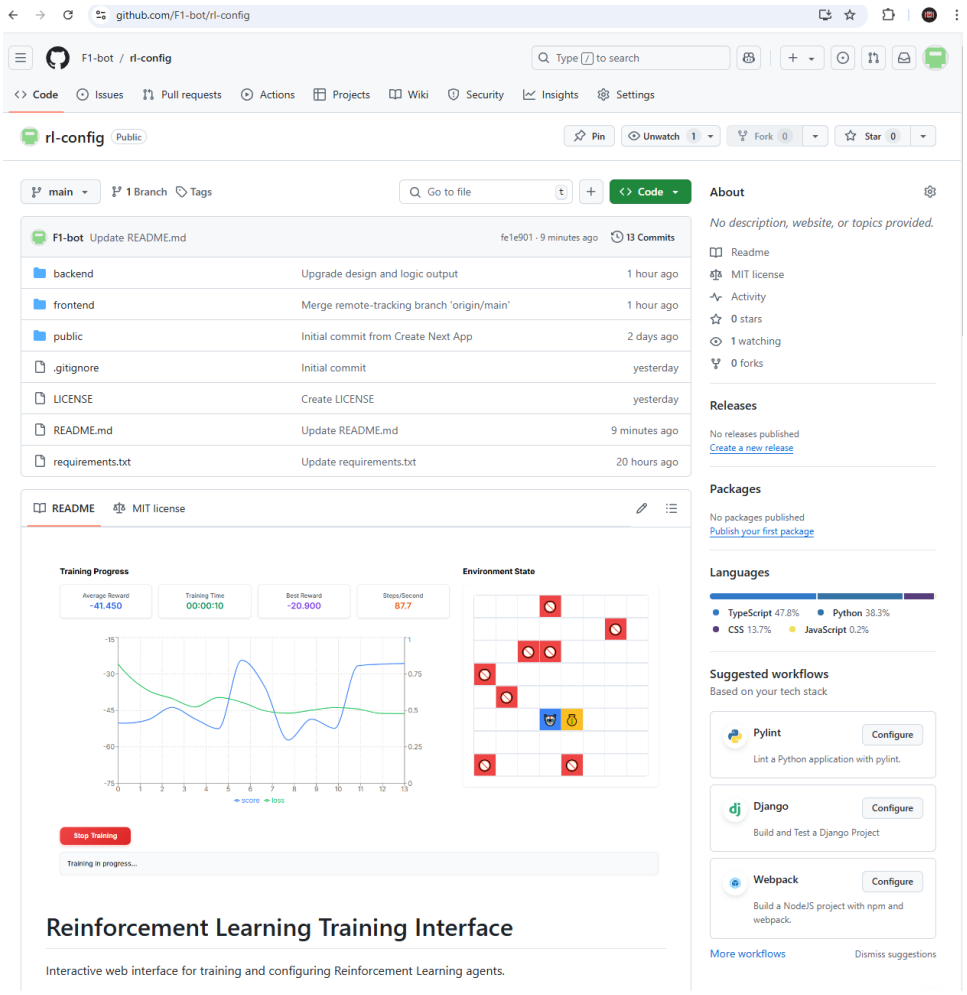


Рисунок 14.4. Вигляд репозиторію rl-config

Клонування репозиторію:

```
git clone https://github.com/F1-bot/rl-config.git
cd rl-config
```

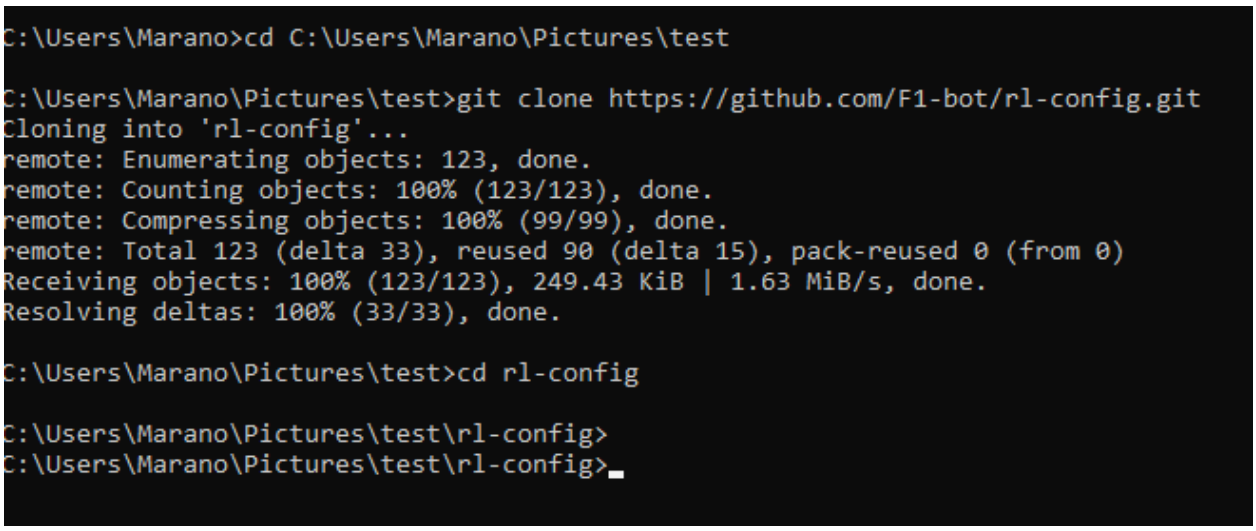


Рисунок 14.5. Клонування репозиторію

Налаштування віртуального середовища Python:

```
python -m venv venv
```

```
venv\Scripts\activate
```

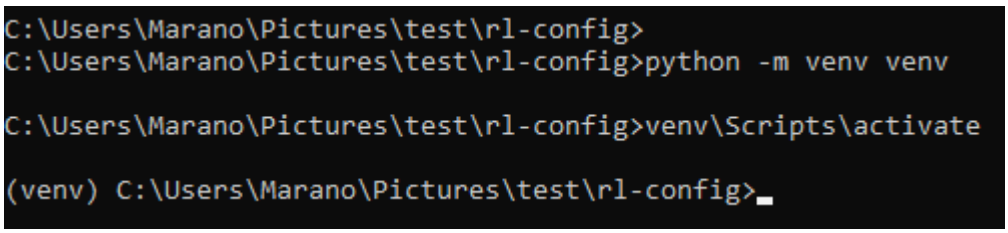


Рисунок 14.6. Створення віртуального середовища Python

Встановлення необхідних залежностей для Python:

```
pip install -r requirements.txt
```

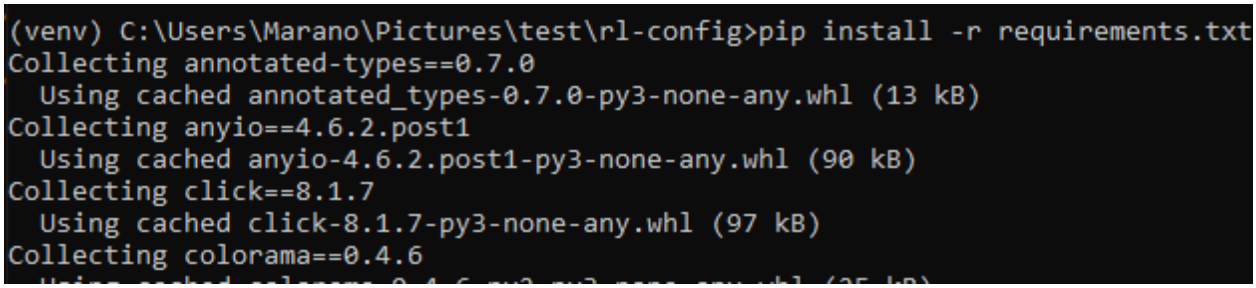


Рисунок 14.7. Завантаження залежностей для Python

Примітка! В межах `requirements.txt` версія torch надана для роботи з CPU. Якщо Ви бажаєте використовувати GPU скористайтесь цієї інформацією задля коректної інсталяції: youtube.com/watch?v=M60_J-jjtn0&ab_channel=AleksandarHaberPhD

Старт роботи сервера fastapi:

```
cd backend
python server.py
```

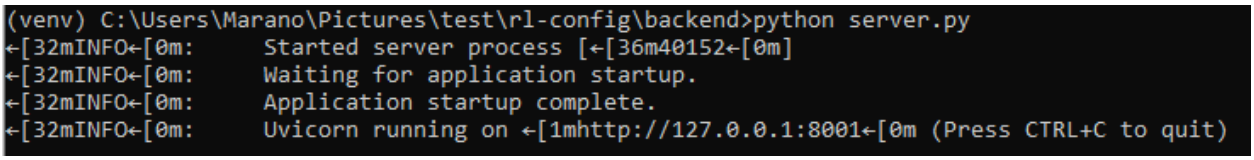


Рисунок 14.8. Запуск сервера fastapi

Встановлення необхідних залежностей для Node.js (у другому терміналі):

```
cd frontend
npm install
```

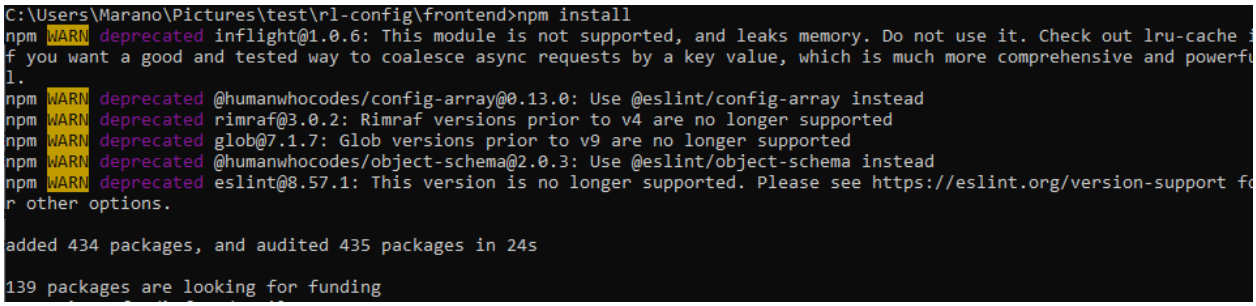


Рисунок 14.9. Завантаження залежностей для Node.js

Запуск веб-інтерфейсу Next.js (у другому терміналі):

```
npm run dev
```

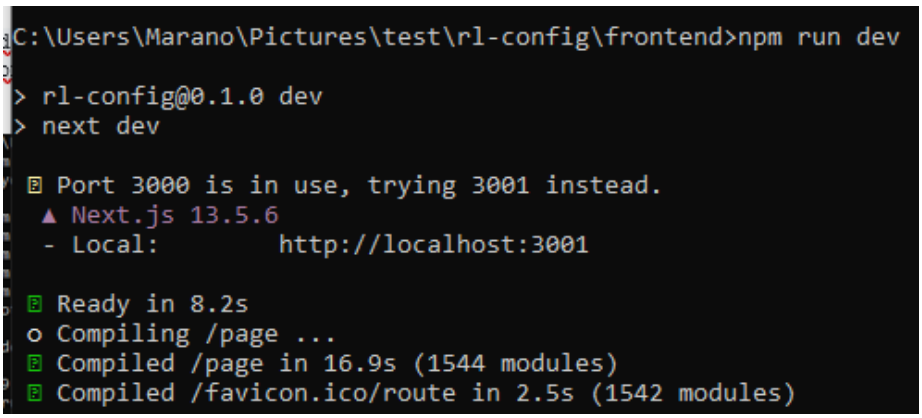


Рисунок 14.10. Запуск веб-інтерфейсу Next.js

Перехід до локального сайту:

```
http://localhost:3000
```

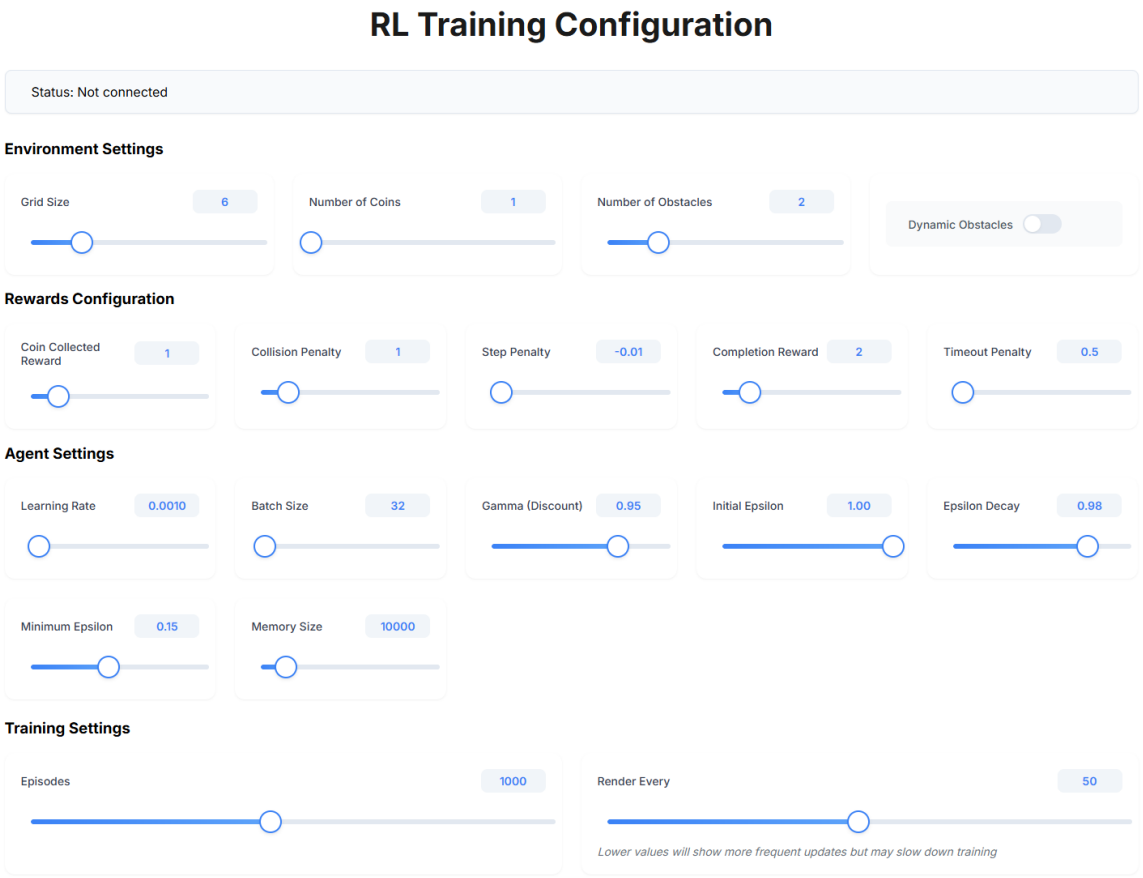


Рисунок 14.11. Вигляд сайту (частина 1)

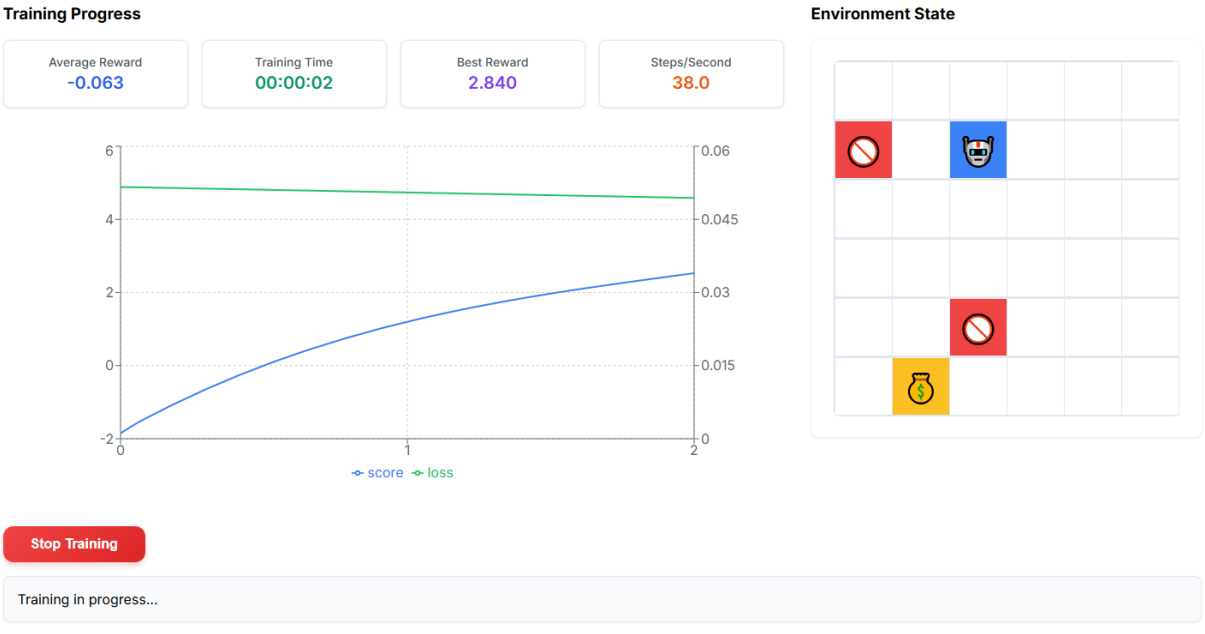


Рисунок 14.12. Вигляд сайту (частина 2)

Завдання 2. Експерименти з параметрами середовища.

Конфігурація		Середня винагорода	Час навчання	Примітки
Environment Settings Grid size: 6 Number of Coins: 1 Number of Obstacles: 2 Dynamic Obstacles: false	**Rewards Configuration** Coin Collected Reward: 2 Collision Penalty: 0.5 Step Penalty: -0.01 Completion Reward: 3 Timeout Penalty: 1	1.560	00:00:27	Best Reward: 5.000 Steps/Second: 85.9
Agent Settings | Default |	**Training Settings** Episodes: 100			
Environment Settings Grid size: 6 Number of Coins: 3 Number of Obstacles: 3 Dynamic Obstacles: true	**Rewards Configuration** Coin Collected Reward: 2 Collision Penalty: 0.5 Step Penalty: -0.01 Completion Reward: 3 Timeout Penalty: 1	2.816	00:00:36	Best Reward: 8.990 Steps/Second: 83.3
Agent Settings | Default |	**Training Settings** Episodes: 100			
Environment Settings Grid size: 10 Number of Coins: 3 Number of Obstacles: 3 Dynamic Obstacles: true	**Rewards Configuration** Coin Collected Reward: 2 Collision Penalty: 0.5 Step Penalty: -0.01 Completion Reward: 3 Timeout Penalty: 1	0.174	00:03:12	Best Reward: 8.860 Steps/Second: 48.7
Agent Settings | Default |	**Training Settings** Episodes: 100			
…	**…**	…	…	…
…	**…**			

Висновки: …

Завдання 3. Оптимізація гіперпараметрів агента.

Параметр	Базове значення	Оптимізоване значення	Результат			
Learning Rate	0.0001	0.0010	Average Reward	5.870	1.796	-69.39%
			Training Time	00:00:25	00:00:38	+34%
			Best Reward	8.970	8.980	+0.11%
			Steps/Second	83.8	84.8	+1.19%
Batch Size	32	96	Average Reward	1.796	1.109	-38.25%
			Training Time	00:00:38	00:00:39	+2.63%
			Best Reward	8.980	8.930	-0.56%
			Steps/Second	84.8	86.3	+1.77%
Memory Size	10000	50000	Average Reward	1.109	0.668	-39.76%
			Training Time	00:00:39	00:00:41	+5.13%
			Best Reward	8.930	8.970	-0.45%
			Steps/Second	86.3	84.1	-2.55%
…	…	…	Average Reward	…	…	…
			Training Time	…	…	…
			Best Reward	…	…	…
			Steps/Second	…	…	…

Висновки: …

Завдання 4. Аналіз процесу навчання.

Рисунок 14.13. Проведення навчання зі 100 епізодами

Рисунок 14.14. Проведення навчання з 500 епізодами

Рисунок 14.15. Проведення навчання з 1000 епізодами

Висновки: …

Завдання 5. Додаткове завдання.

agent.py

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import numpy as np
from collections import deque
import random
from torch.cuda.amp import autocast, GradScaler
from model import SimpleNet
from utils import DEVICE, Experience


class SimplifiedAgent:
    def __init__(self, state_shape, n_actions, learning_rate=1e-3):
        self.state_shape = state_shape  # Повинно бути (7, size, size)
        self.n_actions = n_actions

        # Гіперпараметри
        self.batch_size = 32
        self.gamma = 0.95
        self.epsilon = 1.0
        self.epsilon_decay = 0.98
        self.epsilon_min = 0.15
        self.learning_rate = learning_rate

        # Мережа
        self.policy_net = SimpleNet(state_shape, n_actions).to(DEVICE)
        self.target_net = SimpleNet(state_shape, n_actions).to(DEVICE)
        self.target_net.load_state_dict(self.policy_net.state_dict())

        self.optimizer = optim.Adam(self.policy_net.parameters(),
lr=learning_rate)
        self.memory = deque(maxlen=10000)

    def remember(self, state, action, reward, next_state, done):
        self.memory.append((state, action, reward, next_state, done))

    def get_action(self, state):
        if random.random() < self.epsilon:
            return random.randrange(self.n_actions)

        with torch.no_grad():
            state = torch.FloatTensor(state).unsqueeze(0).to(DEVICE)
            q_values = self.policy_net(state)
            return q_values.argmax(1).item()

    def train(self):
        if len(self.memory) < self.batch_size:
            return None
```

```
        batch = random.sample(self.memory, self.batch_size)
        states, actions, rewards, next_states, dones = zip(*batch)

        states = torch.FloatTensor(np.array(states)).to(DEVICE)
        actions = torch.LongTensor(actions).to(DEVICE)
        rewards = torch.FloatTensor(rewards).to(DEVICE)
        next_states = torch.FloatTensor(np.array(next_states)).to(DEVICE)
        dones = torch.FloatTensor(dones).to(DEVICE)

        current_q_values = self.policy_net(states).gather(1,
actions.unsqueeze(1))
        next_q_values = self.target_net(next_states).max(1)[0].detach()
        expected_q_values = rewards + (1 - dones) * self.gamma *
next_q_values

        loss = F.smooth_l1_loss(current_q_values.squeeze(),
expected_q_values)

        self.optimizer.zero_grad()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(self.policy_net.parameters(), 1.0)
        self.optimizer.step()

        # Оновлюємо target network кожні 10 батчів
        if random.random() < 0.1:
            self.target_net.load_state_dict(self.policy_net.state_dict())

        return loss.item()

    def update_epsilon(self):
        self.epsilon = max(self.epsilon_min, self.epsilon *
self.epsilon_decay)
```

environment.py

```
import numpy as np
import random
from typing import List, Tuple, Dict

class SimplifiedGameEnv:
    def __init__(self, size: int = 6, n_coins: int = 1, n_obstacles: int =
2,
                 dynamic_obstacles: bool = False, rewards: dict = None):
        self.size = size
        self.n_coins = n_coins
        self.n_obstacles = n_obstacles
        self.dynamic_obstacles = dynamic_obstacles
        self.rewards = rewards or {
            'coinCollected': 1,
            'collision': -1,
            'step': -0.01,
            'completion': 2,
```

```
            'timeout': -0.5
        }
        self.action_space_n = 8
        self.observation_space_shape = (7, size, size)
        self.max_steps = size * size
        self.steps = 0
        self.grid = np.zeros((7, size, size))
        self.score = 0
        self.reset()

    def reset(self):
        self.steps = 0
        self.score = 0
        self.grid.fill(0)

        # Розміщення агента в центрі
        self.agent_pos = [self.size // 2, self.size // 2]

        # Розміщення монет та перешкод
        self.coins = []
        self.obstacles = []
        self._place_objects(self.coins, self.n_coins)
        self._place_objects(self.obstacles, self.n_obstacles)

        self.update_grid()
        return self.get_state()

    def _place_objects(self, objects: List, count: int) -> None:
        # Створюємо список всіх можливих позицій
        available_positions = []
        for i in range(self.size):
            for j in range(self.size):
                if ([i, j] != self.agent_pos and
                        [i, j] not in self.coins and
                        [i, j] not in self.obstacles):
                    available_positions.append([i, j])

        # Вибираємо випадкові позиції
        n_positions = min(count, len(available_positions))
        if n_positions > 0:
            selected_positions = random.sample(available_positions,
n_positions)
            objects.extend(selected_positions)

    def update_grid(self) -> None:
        self.grid.fill(0)

        # Базові канали
        self.grid[0, self.agent_pos[0], self.agent_pos[1]] = 1
        for coin in self.coins:
            self.grid[1, coin[0], coin[1]] = 1
        for obs in self.obstacles:
            self.grid[2, obs[0], obs[1]] = 1
```

```
        # Distance map для монет
        if self.coins:
            for i in range(self.size):
                for j in range(self.size):
                    min_dist = min(abs(i - coin[0]) + abs(j - coin[1])
                                   for coin in self.coins)
                    self.grid[3, i, j] = 1 - min_dist / (2 * self.size)

        # Вільний простір
        self.grid[4] = 1 - (self.grid[0] + self.grid[1] + self.grid[2])

        # Додаємо напрямки до найближчої монети
        if self.coins:
            closest_coin = min(self.coins,
                               key=lambda c: abs(c[0] - self.agent_pos[0]) +
                                             abs(c[1] - self.agent_pos[1]))
            dx = closest_coin[0] - self.agent_pos[0]
            dy = closest_coin[1] - self.agent_pos[1]
            dist = max(abs(dx), abs(dy), 1)
            self.grid[5].fill(dx / dist)
            self.grid[6].fill(dy / dist)

    def get_state(self) -> np.ndarray:
        return self.grid.copy()

    def step(self, action: int) -> Tuple[np.ndarray, float, bool, bool,
Dict]:
        self.steps += 1
        old_pos = self.agent_pos.copy()

        # Рух агента
        moves = [(-1, 0), (-1, 1), (0, 1), (1, 1),
                 (1, 0), (1, -1), (0, -1), (-1, -1)]
        dx, dy = moves[action]
        self.agent_pos[0] = np.clip(self.agent_pos[0] + dx, 0, self.size -
1)
        self.agent_pos[1] = np.clip(self.agent_pos[1] + dy, 0, self.size -
1)

        # Перевірка колізій та винагород
        if self.agent_pos in self.obstacles:
            self.agent_pos = old_pos
            reward = self.rewards['collision']
            done = False
        else:
            reward = self.rewards['step']
            done = False

            # Збір монет
            if self.agent_pos in self.coins:
                self.coins.remove(self.agent_pos)
                reward = self.rewards['coinCollected']
```

```
                self.score += 1
                if not self.coins:
                    reward += self.rewards['completion']
                    done = True

            # Перевірка ліміту кроків
            if self.steps >= self.max_steps:
                reward += self.rewards['timeout']
                done = True

        # Динамічні перешкоди
        if self.dynamic_obstacles and not done and random.random() < 0.1:
            self._update_dynamic_obstacles()

        self.update_grid()
        info = {
            "score": self.score,
            "steps": self.steps,
            "coins_left": len(self.coins)
        }

        return self.get_state(), reward, done, False, info

    def _update_dynamic_obstacles(self):
        # Рухаємо кожну перешкоду з певною ймовірністю
        for i, obs in enumerate(self.obstacles):
            if random.random() < 0.3:  # 30% шанс руху
                possible_moves = [
                    (dx, dy) for dx, dy in [(0, 1), (1, 0), (0, -1), (-1,
0)]
                    if 0 <= obs[0] + dx < self.size and 0 <= obs[1] + dy <
self.size
                ]
                if possible_moves:
                    dx, dy = random.choice(possible_moves)
                    new_pos = [obs[0] + dx, obs[1] + dy]
                    # Перевіряємо, чи нова позиція не зайнята
                    if (new_pos != self.agent_pos and
                        new_pos not in self.coins and
                        new_pos not in self.obstacles):
                        self.obstacles[i] = new_pos

    def render(self) -> None:
        grid = [[' ' for _ in range(self.size)] for _ in range(self.size)]

        for obs in self.obstacles:
            grid[obs[0]][obs[1]] = 'X'
        for coin in self.coins:
            grid[coin[0]][coin[1]] = 'O'
        grid[self.agent_pos[0]][self.agent_pos[1]] = 'A'

        print("\n".join([''.join(row) for row in grid]))
        print(f"Score: {self.score}, Steps: {self.steps}")
```

model.py

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
from typing import Tuple

class SimpleNet(nn.Module):
    def __init__(self, input_shape, n_actions):
        super(SimpleNet, self).__init__()

        input_channels = input_shape[0]  # Отримуємо кількість вхідних
каналів

        self.conv = nn.Sequential(
            nn.Conv2d(input_channels, 32, kernel_size=3, stride=1,
padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU()
        )

        # Обчислюємо розмір виходу згорткових шарів
        conv_out_size = self._get_conv_out(input_shape)

        self.fc = nn.Sequential(
            nn.Linear(conv_out_size, 512),
            nn.ReLU(),
            nn.Linear(512, n_actions)
        )

    def _get_conv_out(self, shape):
        o = self.conv(torch.zeros(1, *shape))
        return int(np.prod(o.shape))

    def forward(self, x):
        conv_out = self.conv(x)
        return self.fc(conv_out.view(conv_out.size(0), -1))
```

Висновки: …

Welcome to Github Repository

github.com/F1-bot/rl-config

Лабораторна робота № 15. Дослідження та застосування мультимодальних моделей для аналізу та генерації візуально-текстового контенту

Мета: ознайомитися з принципами роботи та можливостями мультимодальних моделей, навчитися аналізувати зображення, генерувати підписи до них, створювати історії на основі зображень, а також експериментувати з різними моделями та їх параметрами.

Завдання для всіх варіантів

1. ***Завдання 1:*** *використайте різні моделі VQA для відповіді на питання про Ваше зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*
2. ***Завдання 2:*** *застосуйте моделі ViT для генерації описів до заданого зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*
3. ***Завдання 3:*** *реалізуйте простий пошук зображень за текстовим запитом за допомогою моделей CLIP. Знайдіть додаткову інформацію про моделі, що будуть використані.*
4. ***Завдання 4:*** *використовуючи функцію `visual_question_answering`, створіть гру «Візуальний детектив».*
5. ***Завдання 5:*** *використовуючи функцію `image_captioning`, створіть інструмент для генерації історій на основі серії зображень.*

a. ***Завдання 5.1:*** *оберіть 5-7 зображень, які можуть скласти послідовну історію.*
b. ***Завдання 5.2:*** *використайте функцію `image_captioning` для створення підпису до кожного зображення.*
c. ***Завдання 5.3:*** *об'єднайте ці підписи в коротку історію, додаючи сполучні фрази між ними.*

** За необхідності внесіть додаткові зміни в програмний код*

Базова структура роботи

Блок 1. Встановлення та імпорт бібліотек

```
!pip install gradio transformers pillow requests

import torch
from PIL import Image
import requests
from transformers import pipeline, CLIPProcessor, CLIPModel, AutoProcessor,
AutoModelForCausalLM
import gradio as gr
```

Блок 2. Дослідження VQA

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def visual_question_answering(image, question, model_name):
    vqa_pipeline = pipeline("visual-question-answering", model=model_name)
    result = vqa_pipeline(image, question, top_k=1)
    return result[0]['answer']

def process_image1(image, question, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    answer = visual_question_answering(image, question, model_name)
    return f"Відповідь: {answer}"

vqa_models = [
    "Salesforce/blip-vqa-capfilt-large",
    "microsoft/git-base-vqav2",
    "dandelin/vilt-b32-finetuned-vqa",
    "Salesforce/blip-vqa-capfilt-large"
]
iface = gr.Interface(
    fn=process_image1,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть питання"),
        gr.Dropdown(choices=vqa_models, label="Виберіть модель VQA")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей візуального питання-відповіді (VQA)",
    description="Завантажте зображення, введіть питання та виберіть модель
для тестування."
)

iface.launch()
```

Завдання 1: *використайте різні моделі VQA для відповіді на питання про Ваше зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

Блок 3. Створення описів до зображень

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_captioning(image, model_name):
    captioning_pipeline = pipeline("image-to-text", model=model_name)
    result = captioning_pipeline(image)
    return result[0]['generated_text']

def process_image2(image, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    caption = image_captioning(image, model_name)
    return f"Підпис до зображення: {caption}"

vit_models = [
    "nlpconnect/vit-gpt2-image-captioning",
    "microsoft/git-base-coco",
    "Salesforce/blip-image-captioning-base",
    "ydshieh/vit-gpt2-coco-en"
]

iface = gr.Interface(
    fn=process_image2,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Dropdown(choices=vit_models, label="Виберіть ViT модель")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування ViT моделей для створення підписів до зображень",
    description="Завантажте зображення та виберіть модель для створення
підпису."
)
iface.launch()
```

Завдання 2: *застосуйте моделі ViT для генерації описів до заданого зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

Блок 4. Пошук зображень за текстом

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_text_similarity(image, text_queries, model_name):
    model = CLIPModel.from_pretrained(model_name)
    processor = CLIPProcessor.from_pretrained(model_name)

    inputs = processor(text=text_queries, images=image, return_tensors="pt",
padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
    logits_per_image = outputs.logits_per_image
```

```
    probs = logits_per_image.softmax(dim=1)

    return probs.tolist()[0]

clip_models = [
    "openai/clip-vit-large-patch14",
    "laion/CLIP-ViT-H-14-laion2B-s32B-b79K"
]

def process_image_and_text(image, text_queries, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."

    queries = [q.strip() for q in text_queries.split(',')]
    if not queries:
        return "Будь ласка, введіть текстові запити."

    similarities = image_text_similarity(image, queries, model_name)

    result = ""
    for query, similarity in zip(queries, similarities):
        result += f"Подібність до '{query}': {similarity:.4f}\n"

    return result

iface = gr.Interface(
    fn=process_image_and_text,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть текстові запити (розділені комами)"),
        gr.Dropdown(choices=clip_models, label="Виберіть модель CLIP")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей подібності зображення та тексту",
    description="Завантажте зображення, введіть текстові запити та виберіть
модель для оцінки подібності."
)

iface.launch()
```

Завдання 3: *реалізуйте простий пошук зображень за текстовим запитом за допомогою моделей CLIP. Знайдіть додаткову інформацію про моделі, що будуть використані.*

Блок 5. Візуальний детектив

```
model_name = "Salesforce/blip-vqa-capfilt-large"
image_url = "https://www.youngminds.org.uk/media/qkxfwq40/group-of-young-
people-talking.jpg"
questions = [
    ("What is in the foreground?", 1),
    ("How many people do you see in the image?", 2),
    ("What mood does this scene convey?", 3)
```

```
]

score = 0
for question, points in questions:
    answer = visual_question_answering(image_url, question, model_name)

    print(f"Питання: {question}")
    print(f"Відповідь моделі: {answer}")
    user_correct = input("Чи правильна відповідь? (так/ні): ").lower() ==
'так'
    if user_correct:
        score += points
        print(f"Чудово! Ви отримали {points} балів.")
    else:
        print("На жаль, відповідь неправильна.")
    print()

print(f"Ваш загальний рахунок: {score}")
```

Завдання 4: *використовуючи функцію `visual_question_answering`, створіть гру «Візуальний детектив».*

Блок 6. Створення історії за зображеннями

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Функція для генерації зв'язної історії
def generate_coherent_story(captions):
    # Ініціалізуємо модель BLOOM
    model_name = "bigscience/bloom-560m"  # оберіть модель
    model = AutoModelForCausalLM.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    # Створюємо промпт для моделі
    prompt = "Create a coherent story based on the following scenes:\n\n"
    for i, caption in enumerate(captions, 1):
        prompt += f"Scene {i}: {caption}\n"
    prompt += "\nStory:"

    # Генеруємо історію
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids

    attention_mask = torch.ones(input_ids.shape, dtype=torch.long,
device=input_ids.device)

    outputs = model.generate(
        input_ids,
        attention_mask=attention_mask,
        max_length=300,
        num_return_sequences=1,
        temperature=0.7,
```

```
        no_repeat_ngram_size=2,
        do_sample=True
    )
    story = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return story.split("Story:")[1].strip()

# Основний код
image_urls = [
    "https://t3.ftcdn.net/jpg/02/43/12/34/360_F_243123463_zTooub557xEWABDLk0
jJklDyLSGl2jrr.jpg",
    "http://images.cocodataset.org/val2017/000000039769.jpg",
    "https://www.loveyourlandscape.org/media/19953/mighty-oak-tree-from-
below-000056382958_large.jpg"
    # Додайте додаткових зображень
]

model_name = "nlpconnect/vit-gpt2-image-captioning" # оберіть модель


captions = []
for i, url in enumerate(image_urls, 1):
    caption = image_captioning(url, model_name)
    captions.append(caption)
    print(f"Part {i}: {caption}")

print("\nЗгенеровані підписи до зображень:")
for i, caption in enumerate(captions, 1):
    print(f"Part {i}: {caption}")

print("\nГенеруємо зв'язну історію...")
coherent_story = generate_coherent_story(captions)

print("\nЗгенерована зв'язна історія:")
print(coherent_story)
```

Завдання 5: *використовуючи функцію `image_captioning`, створіть інструмент для генерації історій на основі серії зображень.*

Приклад виконання роботи

Блок 1. Встановлення та імпорт бібліотек

```
!pip install gradio transformers pillow requests

import torch
from PIL import Image
import requests
from transformers import pipeline, CLIPProcessor, CLIPModel, AutoProcessor,
AutoModelForCausalLM
import gradio as gr
```

Блок 2. Дослідження VQA

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def visual_question_answering(image, question, model_name):
    vqa_pipeline = pipeline("visual-question-answering", model=model_name)
    result = vqa_pipeline(image, question, top_k=1)
    return result[0]['answer']

def process_image1(image, question, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    answer = visual_question_answering(image, question, model_name)
    return f"Відповідь: {answer}"

vqa_models = [
    "Salesforce/blip-vqa-capfilt-large",
    "microsoft/git-base-vqav2",
    "dandelin/vilt-b32-finetuned-vqa",
    "Salesforce/blip-vqa-capfilt-large"
]

iface = gr.Interface(
    fn=process_image1,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть питання"),
        gr.Dropdown(choices=vqa_models, label="Виберіть модель VQA")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей візуального питання-відповіді (VQA)",
    description="Завантажте зображення, введіть питання та виберіть модель
для тестування."
)

iface.launch()
```

Завдання 1: *використайте різні моделі VQA для відповіді на питання про Ваше зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

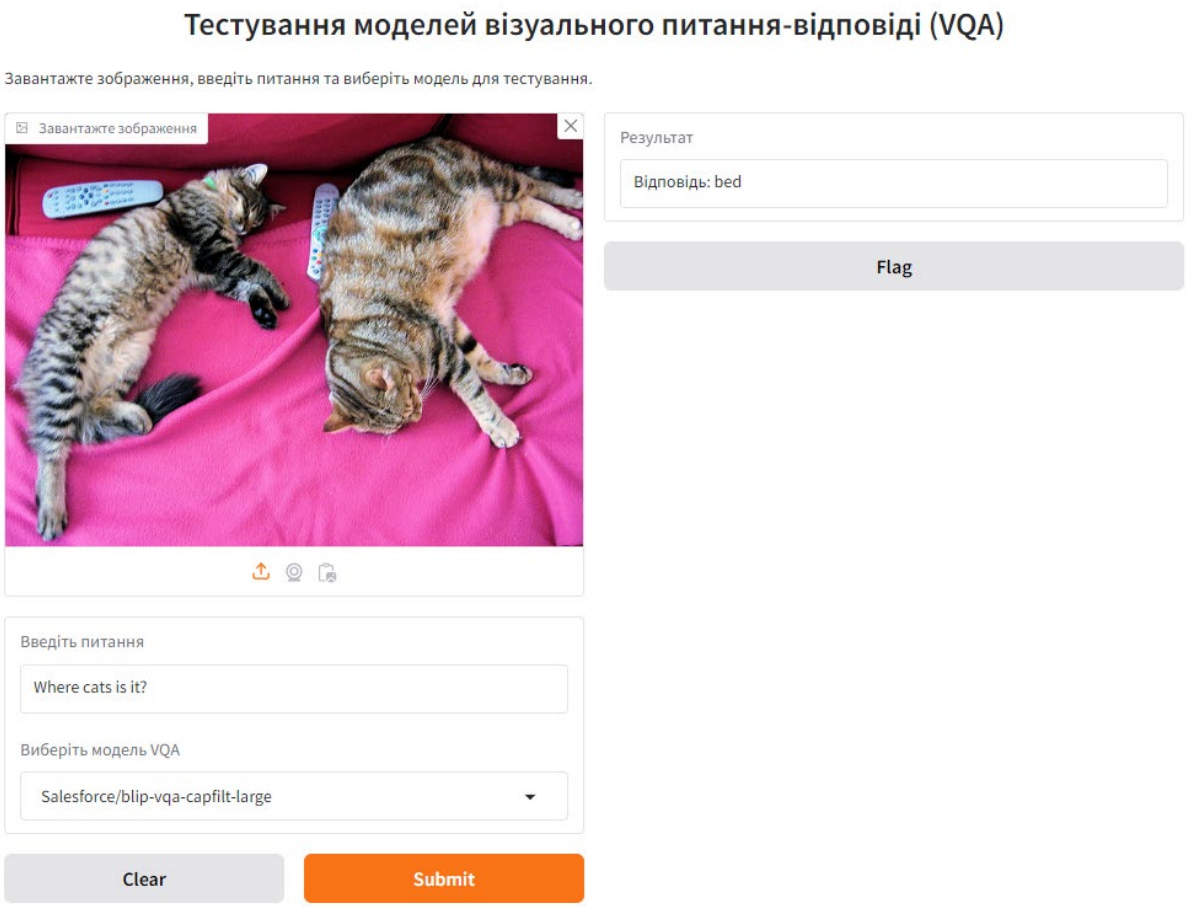


Рисунок 15.1. Результат обробки запитання моделлю blip-vqa-capfilt-large

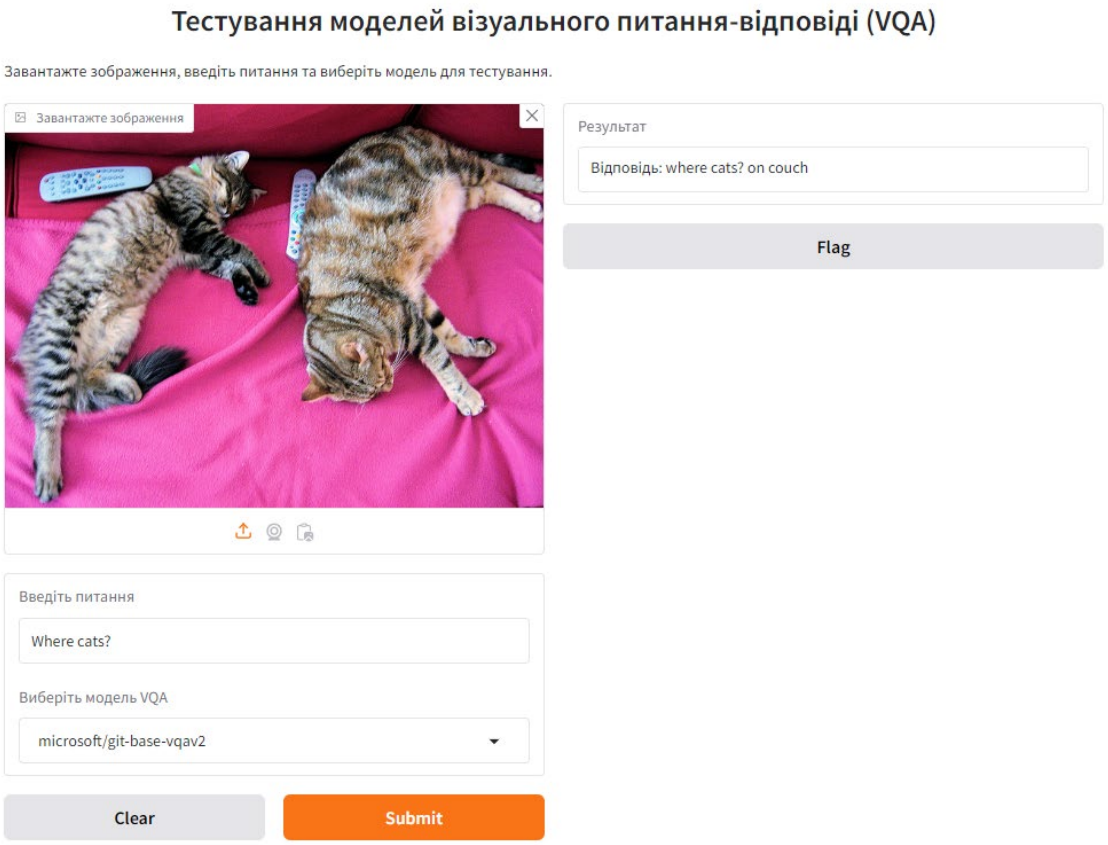


Рисунок 15.2 Результат обробки запитання моделлю git-base-vqav2

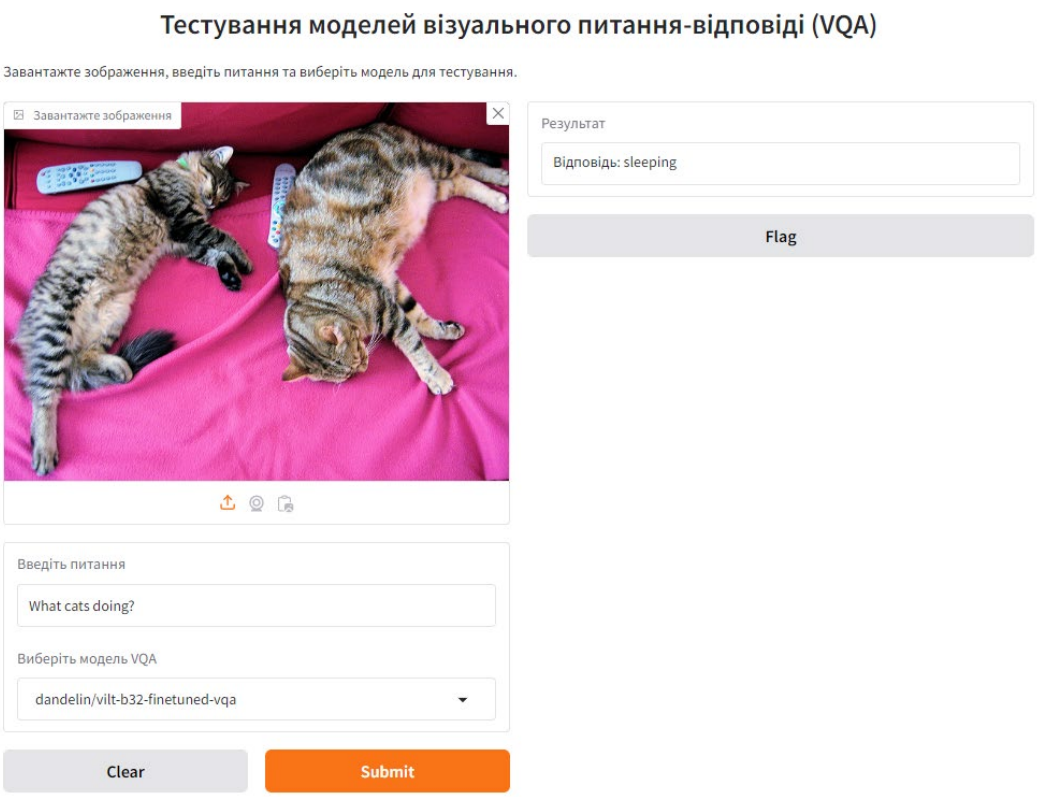


Рисунок 15.3 Результат обробки запитання моделлю vilt-b32-finetuned-vqa

Блок 3. Створення описів до зображень

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_captioning(image, model_name):
    captioning_pipeline = pipeline("image-to-text", model=model_name)
    result = captioning_pipeline(image)
    return result[0]['generated_text']

def process_image2(image, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."
    caption = image_captioning(image, model_name)
    return f"Підпис до зображення: {caption}"

vit_models = [
    "nlpconnect/vit-gpt2-image-captioning",
    "microsoft/git-base-coco",
    "Salesforce/blip-image-captioning-base",
    "ydshieh/vit-gpt2-coco-en"
]

iface = gr.Interface(
    fn=process_image2,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Dropdown(choices=vit_models, label="Виберіть ViT модель")
```

```
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування ViT моделей для створення підписів до зображень",
    description="Завантажте зображення та виберіть модель для створення
підпису."
)
iface.launch()
```

Завдання 2: *застосуйте моделі ViT для генерації описів до заданого зображення. Знайдіть додаткову інформацію про моделі, що будуть використані.*

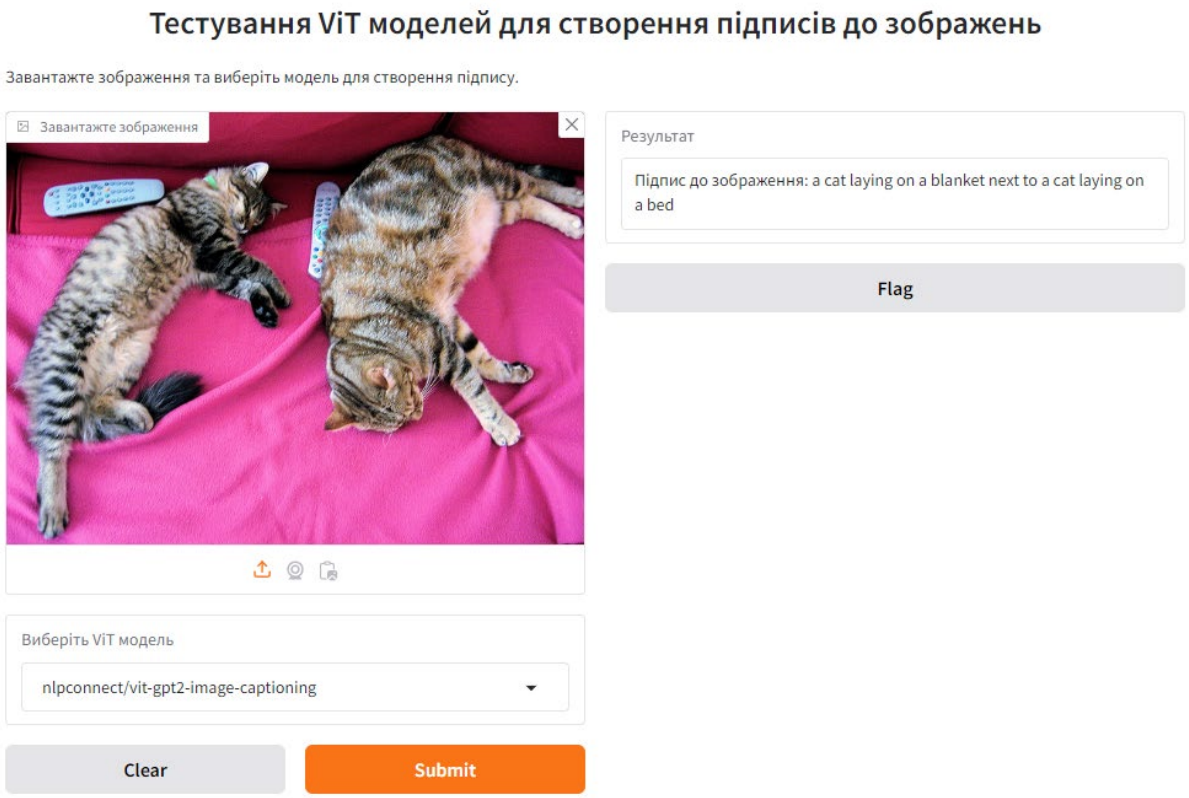


Рисунок 15.4. Результат надання опису моделлю vit-gpt2-image-captioning

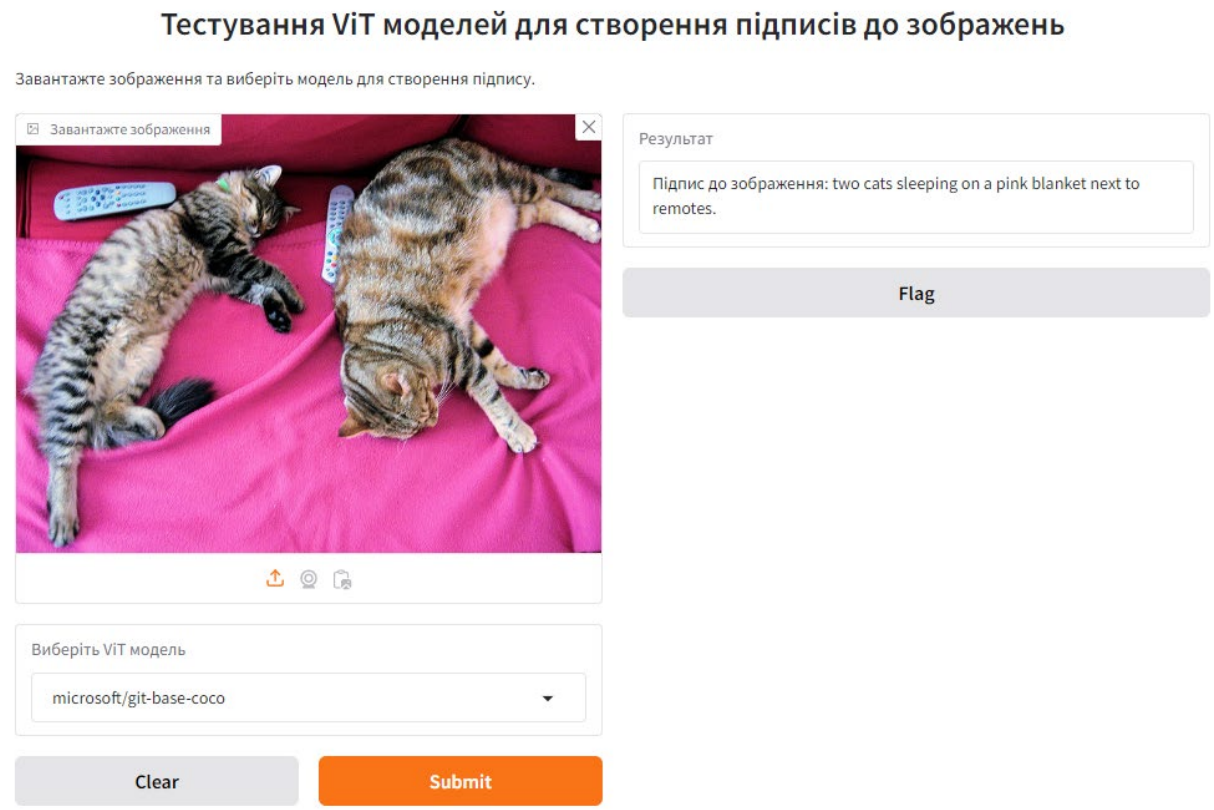


Рисунок 15.5. Результат надання опису моделлю git-base-coco

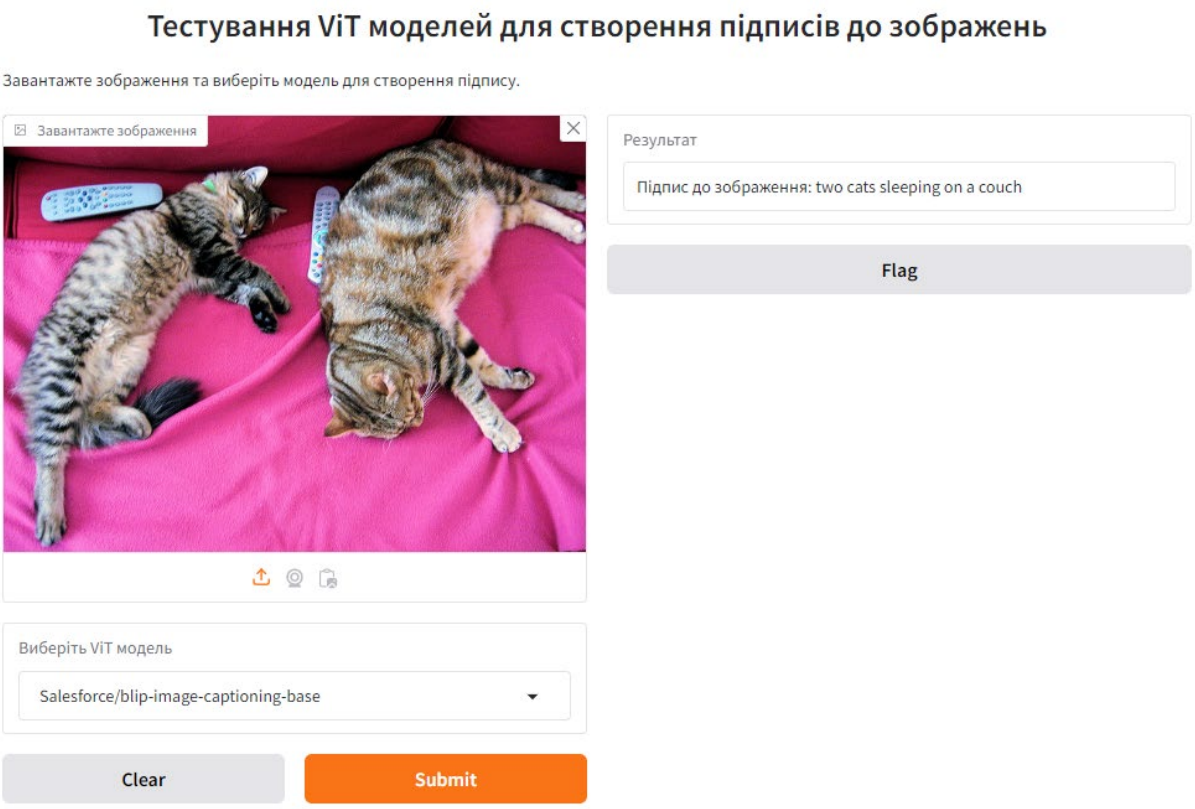


Рисунок 15.6. Результат надання опису моделлю blip-image-captioning-base

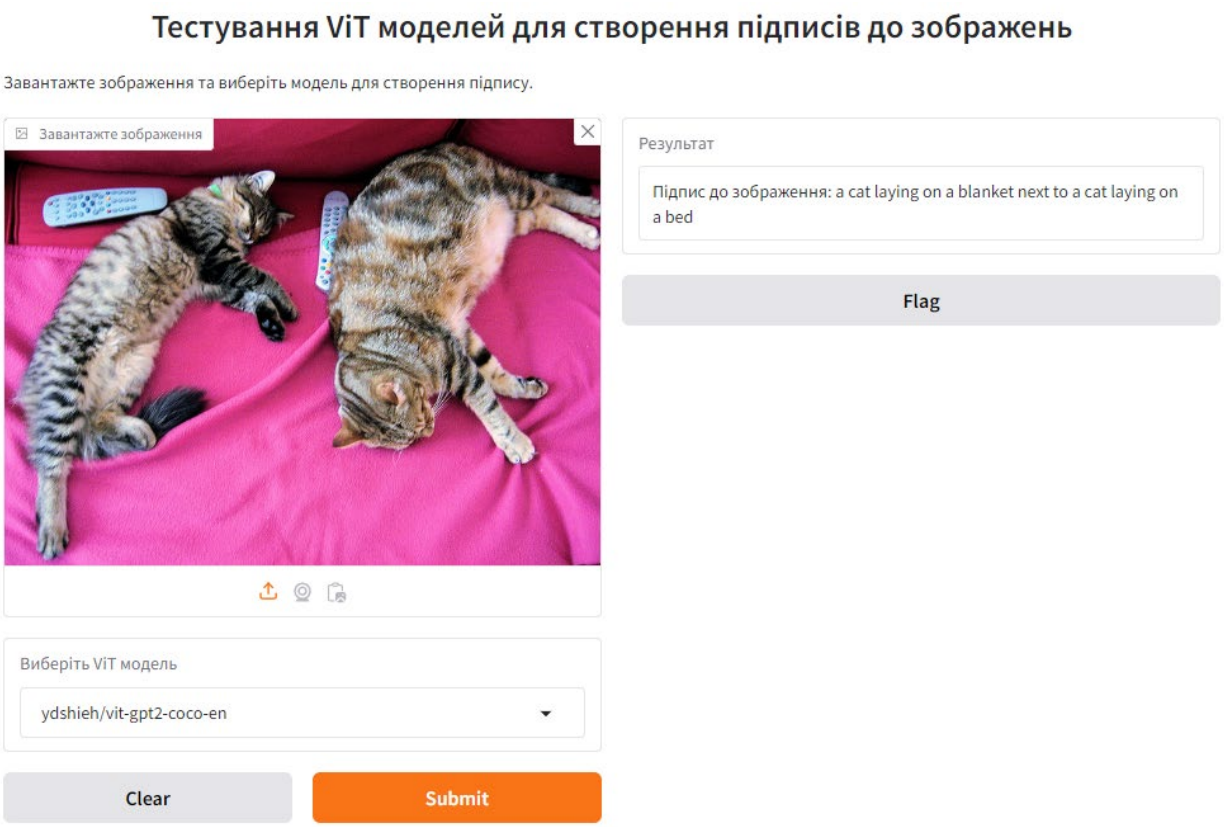


Рисунок 15.7. Результат надання опису моделлю vit-gpt2-coco-en

Блок 4. Пошук зображень за текстом

```
def load_image(url):
    return Image.open(requests.get(url, stream=True).raw)

def image_text_similarity(image, text_queries, model_name):
    model = CLIPModel.from_pretrained(model_name)
    processor = CLIPProcessor.from_pretrained(model_name)

    inputs = processor(text=text_queries, images=image, return_tensors="pt",
padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
    logits_per_image = outputs.logits_per_image
    probs = logits_per_image.softmax(dim=1)

    return probs.tolist()[0]

clip_models = [
    "openai/clip-vit-large-patch14",
    "laion/CLIP-ViT-H-14-laion2B-s32B-b79K"
]

def process_image_and_text(image, text_queries, model_name):
    if image is None:
        return "Будь ласка, завантажте зображення."

    queries = [q.strip() for q in text_queries.split(',')]
```

```
    if not queries:
        return "Будь ласка, введіть текстові запити."

    similarities = image_text_similarity(image, queries, model_name)

    result = ""
    for query, similarity in zip(queries, similarities):
        result += f"Подібність до '{query}': {similarity:.4f}\n"

    return result

iface = gr.Interface(
    fn=process_image_and_text,
    inputs=[
        gr.Image(type="pil", label="Завантажте зображення"),
        gr.Textbox(label="Введіть текстові запити (розділені комами)"),
        gr.Dropdown(choices=clip_models, label="Виберіть модель CLIP")
    ],
    outputs=gr.Textbox(label="Результат"),
    title="Тестування моделей подібності зображення та тексту",
    description="Завантажте зображення, введіть текстові запити та виберіть модель для оцінки подібності."
)

iface.launch()
```

Завдання 3: *реалізуйте простий пошук зображень за текстовим запитом за допомогою моделей CLIP. Знайдіть додаткову інформацію про моделі, що будуть використані.*

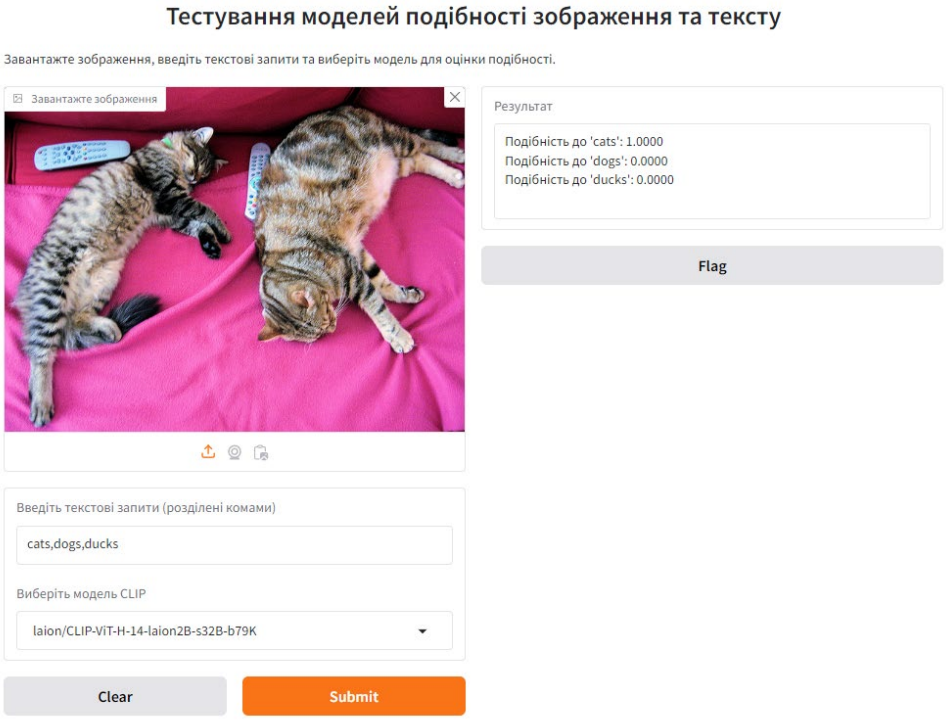


Рисунок 15.8. Результат оцінки подібності моделлю CLIP-ViT-H-14-laion2B-s32B-b79K

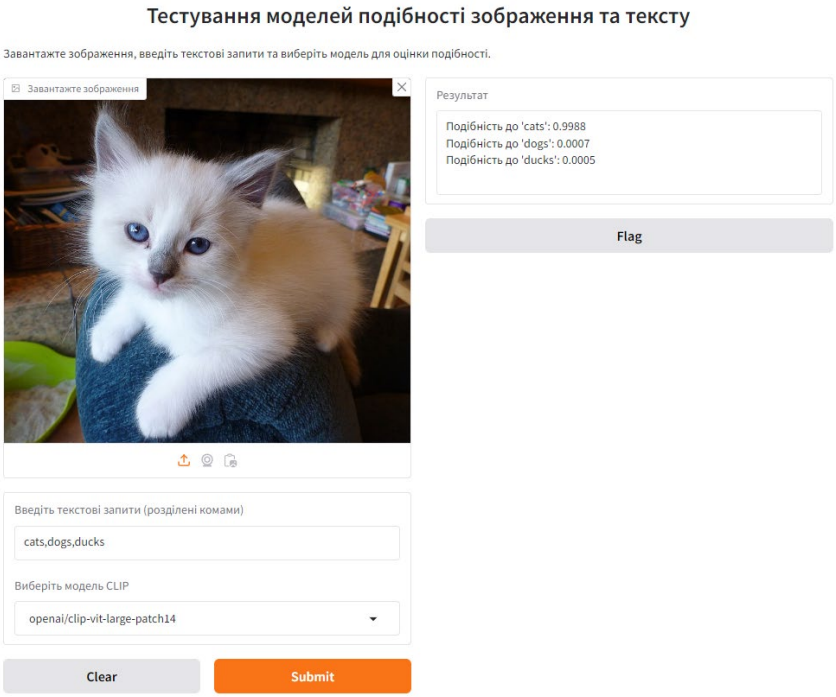


Рисунок 15.9. Результат оцінки подібності моделлю clip-vit-large-patch14

Блок 5. Візуальний детектив

```
model_name = "Salesforce/blip-vqa-capfilt-large"
image_url = "https://www.youngminds.org.uk/media/qkxfwq40/group-of-young-
people-talking.jpg"
questions = [
    ("What is in the foreground?", 1),
    ("How many people do you see in the image?", 2),
    ("What mood does this scene convey?", 3)
]

score = 0
for question, points in questions:
    answer = visual_question_answering(image_url, question, model_name)

    print(f"Питання: {question}")
    print(f"Відповідь моделі: {answer}")
    user_correct = input("Чи правильна відповідь? (так/ні): ").lower() ==
'так'
    if user_correct:
        score += points
        print(f"Чудово! Ви отримали {points} балів.")
    else:
        print("На жаль, відповідь неправильна.")
    print()

print(f"Ваш загальний рахунок: {score}")
```

Завдання 4: *використовуючи функцію `visual_question_answering`, створіть гру «Візуальний детектив».*

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1258: UserWarn
  warnings.warn(
Питання: What is in the foreground?
Відповідь моделі: woman
Чи правильна відповідь? (так/ні): так
Чудово! Ви отримали 1 балів.

Hardware accelerator e.g. GPU is available in the environment, but no `device` argument
Питання: How many people do you see in the image?
Відповідь моделі: 4
Чи правильна відповідь? (так/ні): так
Чудово! Ви отримали 2 балів.

Hardware accelerator e.g. GPU is available in the environment, but no `device` argument
Питання: What mood does this scene convey?
Відповідь моделі: happy
Чи правильна відповідь? (так/ні): так
Чудово! Ви отримали 3 балів.

Ваш загальний рахунок: 6
```

Рисунок 15.10. Результат оцінки роботи «Візуального детективу»

Блок 6. Створення історії за зображеннями

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Функція для генерації зв'язної історії
def generate_coherent_story(captions):
    # Ініціалізуємо модель BLOOM
    model_name = "bigscience/bloom-560m"  # оберіть модель
    model = AutoModelForCausalLM.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    # Створюємо промпт для моделі
    prompt = "Create a coherent story based on the following scenes:\n\n"
    for i, caption in enumerate(captions, 1):
        prompt += f"Scene {i}: {caption}\n"
    prompt += "\nStory:"

    # Генеруємо історію
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids

    attention_mask = torch.ones(input_ids.shape, dtype=torch.long,
device=input_ids.device)

    outputs = model.generate(
        input_ids,
        attention_mask=attention_mask,
        max_length=300,
        num_return_sequences=1,
        temperature=0.7,
        no_repeat_ngram_size=2,
        do_sample=True
    )
    story = tokenizer.decode(outputs[0], skip_special_tokens=True)
```

```
    return story.split("Story:")[1].strip()

# Основний код
image_urls = [
    "https://t3.ftcdn.net/jpg/02/43/12/34/360_F_243123463_zTooub557xEWABDLk0
jJklDyLSGl2jrr.jpg",
    "http://images.cocodataset.org/val2017/000000039769.jpg",
    "https://www.loveyourlandscape.org/media/19953/mighty-oak-tree-from-
below-000056382958_large.jpg"
    # Додайте додаткових зображень
]

model_name = "nlpconnect/vit-gpt2-image-captioning" # оберіть модель


captions = []
for i, url in enumerate(image_urls, 1):
    caption = image_captioning(url, model_name)
    captions.append(caption)
    print(f"Part {i}: {caption}")

print("\nЗгенеровані підписи до зображень:")
for i, caption in enumerate(captions, 1):
    print(f"Part {i}: {caption}")

print("\nГенеруємо зв'язну історію...")
coherent_story = generate_coherent_story(captions)

print("\nЗгенерована зв'язна історія:")
print(coherent_story)
```

Завдання 5: *використовуючи функцію `image_captioning`, створіть інструмент для генерації історій на основі серії зображень.*

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` o
Part 1: a man in a suit and tie smiling
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` o
Part 2: a cat laying on a blanket next to a cat laying on a bed
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` o
Part 3: a tree with a lot of leaves and a few trees

Згенеровані підписи до зображень:
Part 1: a man in a suit and tie smiling
Part 2: a cat laying on a blanket next to a cat laying on a bed
Part 3: a tree with a lot of leaves and a few trees

Генеруємо зв'язну історію...

Згенерована зв'язна історія:
The man sees a couple of cats laying in the bush and he decides to take them to the hospital. He decides that the cat
```

Рисунок 15.11. Результат створення історії за зображеннями

Згенерована історія: «The man sees a couple of cats laying in the bush and he decides to take them to the hospital. He decides that the cat is so small that it is impossible to get it to stay on its

bed. The cat has to climb up onto a branch and then it climbs right into the man's suit. As the couple sits down they see a little tree and they decide to come back to this tree to see what the tree looks like and to have a look at the forest. There are some twigs in this forest that are just too small and this cat cann't get to them. They decide that they are going to try a different method of climbing and the sun is shining and so they climb a ladder instead and climb into a huge tree that has a pool of water. This cat gets to its feet and it has some food on it and starts to eat it. She does not even have to leave the water to touch it, it just gets a taste in its mouth and tries to swallow it for a while. Then, when it gets hungry, the two of them try to go over the pool and find a nearby tree, this time the bigger tree».

Висновки: …

Welcome to Google Colab

https://colab.research.google.com/drive/1TCLyv2gXPiIgNNAxr3ljw2FylkI3Ezj-?usp=sharing

Перелік рекомендованої літератури

1. Булгакова О.С., Зосімов В.В., Поздєєв В.О. Методи та системи штучного інтелекту:теорія та практика. Навчальний посібник. Одеса : ОЛДІ-ПЛЮС, 2020. 356 с.
2. Гавриленко С.Ю., Челак В.В., Горносталь О.А., Зозуля В.Д. Машинне навчання. Лабораторний практикум / С.Ю. Гавриленко, В.В. Челак, О.А. Горносталь, В.Д Зозуля. – Х.: НТУ «ХПІ», 2022.86 с.
3. Кононова К. Ю. Машинне навчання: методи та моделі : підручник. Харків : ХНУ імені В. Н. Каразіна, 2020. 301 с.
4. Методи та системи штучного інтелекту: навч. посіб. / укл. Д.В. Лубко, С.В. Шаров. Мелітополь: ФОП Однорог Т.В., 2019. 264 с.
5. Субботін С.О. Нейронні мережі: навч. посібник / С.О. Субботін, А.О. Олійник; за ред. С.О. Субботіна. Запоріжжя : ЗНТУ, 2014. 132 с.
6. Тимощук П.В. Штучні нейронні мережі. Навчальний посібник / Львів: Видавництво Львівської Політехніки, 2011. 444 с.
7. Федорін І. В. Методи та технології обчислювального інтелекту: Практикум : навч. посіб. КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2022. 317 с.
8. Шаповал Н. В. Методи та системи штучного інтелекту. Комп’ютерний практикум : навч. посіб. Київ : КПІ ім. І. Сікорського, 2022. 45 с.
9. Charu C. Aggarwal Neural Networks and Deep Learning. Springer International Publishing. 2023. 529 p.
10. Dulani Meedeniya Deep Learning. CRC Press LLC, 2023. 200 p.
11. Foster, D. Generative Deep Learning. O’Reilly Media, Inc. 2023. 456 p.
12. Gallatin, K. & Albon, C. Machine Learning with Python Cookbook. Practical Solutions from Preprocessing to Deep Learning. O’Reilly Media, Inc. 2023. 416 p.
13. Hapke, H., & Nerlson, C. Building Machine Learning Pipelines. Automating Model Life Cycles with TensorFlow. O’Reilly Media, Inc. 2020. 367 p.
14. Howard, J., & Gugger, S. Deep Learning for Coders with fastai & PyTorch. AI Applications Without a PhD. O’Reilly Media, Inc. 2020. 624 p.
15. Huyen, C. Designing Machine Learning Systems. An Iterative Process for Production-Ready Applications. O’Reilly Media, Inc. 2022. 389 p.
16. Huyen, C. Designing Machine Learning Systems. An Iterative Process for Production-Ready Applications. O’Reilly Media, Inc. 2022. 389 p.
17. Lakshmanan, V., Görner, M., & Gillard, R. Practical Machine Learning for Computer

Vision. End to-End Machine Learning for images.2021. 481 p.

18. Machine Learning and Its Application to Reacting Flows : ML and Combustion / N. Swaminathan, A. Parente (eds.). Cham : Springer, 2023. 346 p.
19. Moroney, L. AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence. O'Reilly Media, Inc. 2021. 392 p.
20. Zgurovsky M. Z., Zaychenko Y. P. The Fundamentals of Computational Intelligence: System Approach. Springer International Publishing Switzerland, 2016. 375 p.