Artificial Intelligence

CS13207

Laboratory Manual

Student Copy

FALL 2025

Submitted to:

Dr. Syed M Hamedoon
Assistant Professor

Submitted by:

Name	
Registration No	
Program	Information Engineering Technology

The University Of Lahore

1-Km, Defence Road, Bhupatian Chowk, Off RaiwindRoad, Lahore

Guidelines for Laboratory Procedure

- Before starting a new lab, you must always read the laboratory manual for that experiment and the instructor will discuss the experiment with you.
- Attachments must be made for laboratory experiment done in the class.
- Make sure that your observation for previous week experiment is evaluated by the faculty member and your have transferred all the contents to your record before entering to lab.
- At the beginning of the class, if the faculty or the instructor finds that a student is not adequately prepared, they will be marked as absent and not be allowed to perform the experiment.
- Please actively participate in class and don't hesitate to ask questions.
- Please utilize teaching assistants fully. To encourage you to be prepared and to read the lab manual before coming to the laboratory, unannounced questions may be asked at any time during the lab.
- Maintain silence, order and discipline inside the lab. Don't use cell phones inside the laboratory.
- Report to the instructor if you find equipment that is out of order or you break something..

Safety Precautions

This is a partial list of basic safety precautions to use when working on a computer:

- Remove your watch and jewelry and secure loose clothing.
- Turn off the power and unplug equipment before performing service.
- Take a note of all the exits in the room, and also take note of the location of fire extinguishers in the room for the sake of fire safety.
- Try not to type continuously for extremely long periods.
- Look away from the screen once in a while to give your eyes a rest.
- Do not touch any exposed wires or sockets.
- Do not attempt to open any machines, and do not touch the backs of machines when they are switched on.
- Do not spill water or any other liquid on the machine, in order to maintain electrical safety.
- Turn off the machine you were using, when you are done using it.
- Do not access external devices without scanning them for computer viruses.
- Ensure that the temperature in the room stays cool, since there are a lot of machines inside a lab, and these can overheat easily.
- Try not to touch any of the circuit boards and power sockets when something is connected to them and switched on.
- Always maintain an extra copy of all your important data.

Safety Undertaking

I have read all of the above, and I agree to conform to its contents.

Name: _____

Registration No.: _____

Student Signature: _____

Date: _____

Lab Instructor: _____

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency 10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery 10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

Lab's Course Learning Outcomes

Course Title : Artificial Intelligence

Course Code : CS13207

Instructor : *Dr. Syed M Hamedoon*

Designation : Assistant Professor

E-mail : Muhammad.Hamedoon@tech.uol.edu.pk

Students will be able to:

CLO3: **Apply** AI algorithms using MATLAB/Python **(P5, PLO5)**

Mapping of Course Learning Outcomes (CLO) to Program Learning Outcomes (PLO) /Graduate Attributes

Course Code	CLOs/ PLOs	PLO1	PLO2	PLO3	PLO4	PLO5	PLO6	PLO7	PLO8	PLO9	PLO 10	PLO 11	PLO 12
CS13207	CLO 3					X							

PLO1: Engineering Knowledge
PLO2: Problem Analysis
PLO3: Design/Development of Solutions
PLO4: Investigation
PLO5: Modern Tool Usage
PLO6: The Engineer and Society
PLO7: Environment and Sustainability
PLO8: Ethics
PLO9: Individual and Team Work
PLO10: Communication
PLO11: Project Management
PLO12: Lifelong Learning

List of Practical Tasks

LAB No.	Title
1	Introduction to VS Code and Google Colab for Data Analysis Using Python
2	Implementation of Regression Analysis using python
3	Decision Tree Classifier
4	Random Forest and Support Vector Machine Classifier
5	Implementation of Artificial Neural Network
6	Implementation of Deep Neural Network
7	Implementation of Recurrent Neural Network (RNN)
8	Implementation of Long Short-Term Memory (LSTM) Network
9	Convolution Neural Network (OEL)
10	K means Clustering Schemes in Machine Learning
11	Agglomerative Hierarchical Clustering
12	Implementation of Reinforcement Learning
13	Natural Language Processing (NLP)
14	Document Loading using LangChain for Retrieval-Augmented Generation (RAG)
15	Build a RAG-Based Chatbot Using Ollama and LangChain, and Streamlit
	Semester Project

EVALUATION LOG

LAB No.	Obtained Marks	Checked date	Checked by
	CLO 3		
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
Project			

LAB No. 1

Introduction to VS Code and Google Colab for Data Analysis Using Python

LAB Description

In this lab, students will learn how to work with Python programming environments using Visual Studio Code (VS Code) and Google Colab. VS Code is a lightweight and powerful source-code editor that runs on a local system and supports Python development through extensions. It allows users to write, run, and debug Python programs efficiently on their own computer.

Google Colab is a cloud-based Python notebook environment provided by Google that runs in a web browser. It does not require any local installation and provides free access to computing resources. Colab is especially useful for data analysis and visualization, as it supports interactive notebooks, built-in libraries, and easy file uploads.

Using either VS Code or Google Colab, students will create a dataset in Python, upload or load the data, and perform basic data analysis operations. The lab focuses on understanding how datasets are handled, how simple statistics are calculated, and how graphical representations help in interpreting data.

Lab Objective

The objectives of this lab are:

- To understand the basic functioning of VS Code and Google Colab
- To learn how to create and upload a dataset using Python
- To perform basic statistical analysis (such as mean, median, and count)
- To visualize data using simple graphs (line charts, bar charts, or histograms)
- To develop foundational skills in data analysis and visualization using Python

How to use python in VS code?

1. Create environment named 'venv'

python -m venv

2. Activate environment

venv\Scripts\activate

3. Select Environment in VS CodeAfter creating the environment:

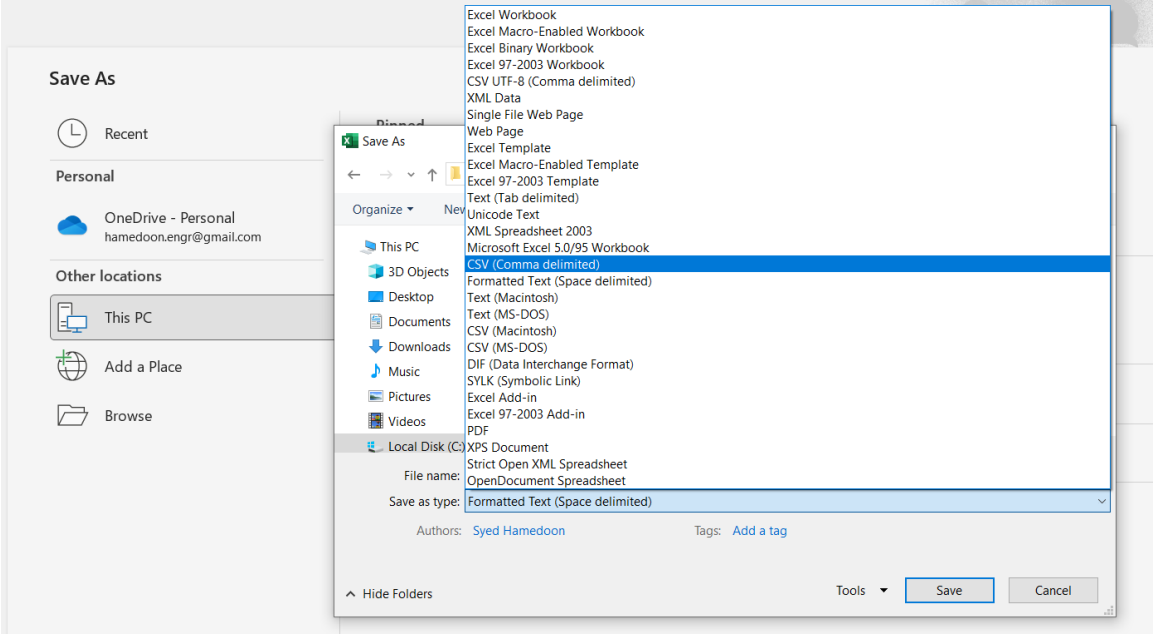
1) Open your project folder in VS Code.
2) Press Ctrl + Shift + P → search for Python:
3) Select Interpreter.
4) Choose your newly created environment (venv or myenv).

4. Install Machine Learning and Deep Learning Libraries

- pip install –upgrade pip
- pip install pandas
- pip install matplotlib
- pip install seaborn
- pip install scikit-learn
- pip install scipy
- pip install numpy
- pip install xgboost
- pip install lightgbm
- pip install catboost
- pip install tensorflow
- pip install keras
- pip install torch

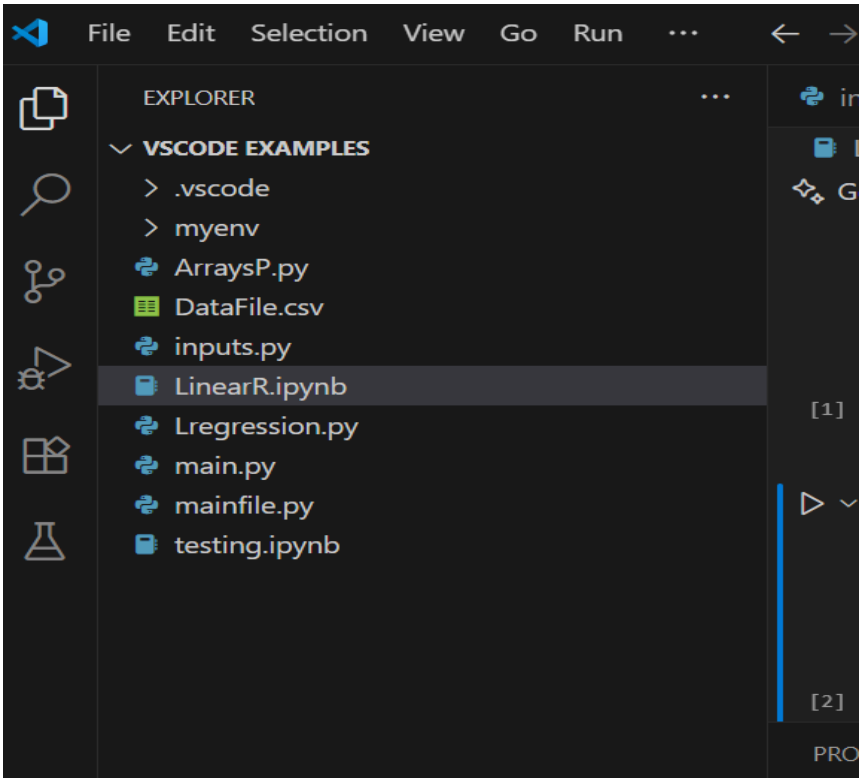
Task 1:

Save Excel file as .csv (Comma delimited)

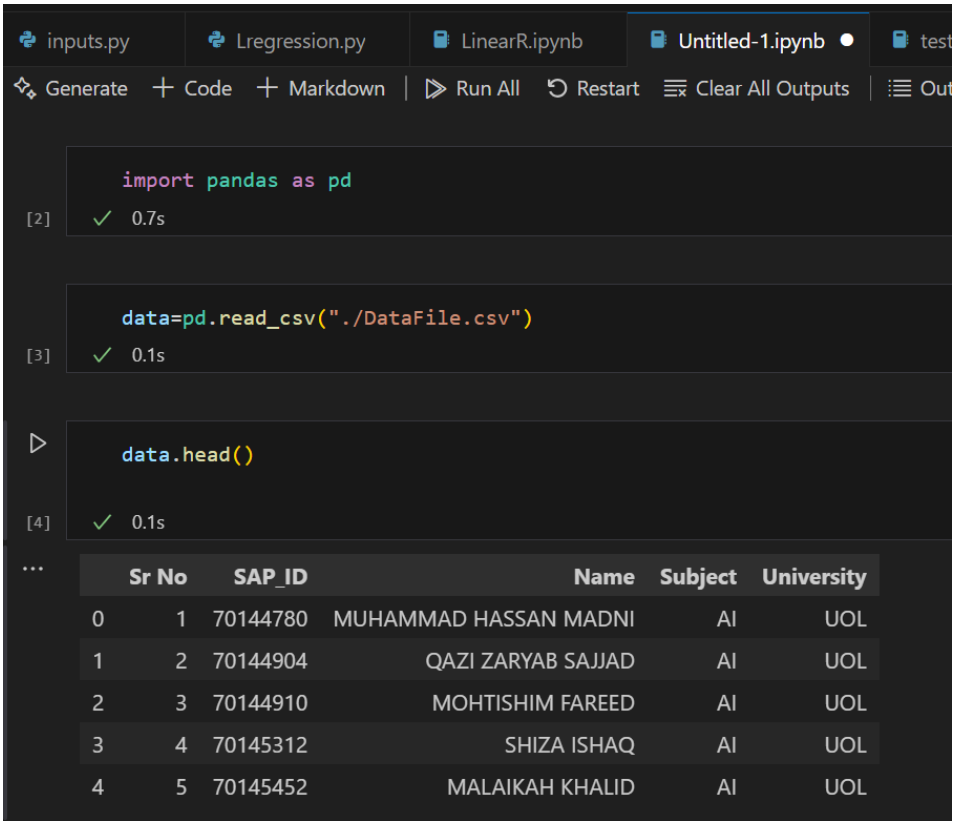


Task 2:

Save .csv file in directory and folder where we created a python environment



Here we can see our file successfully

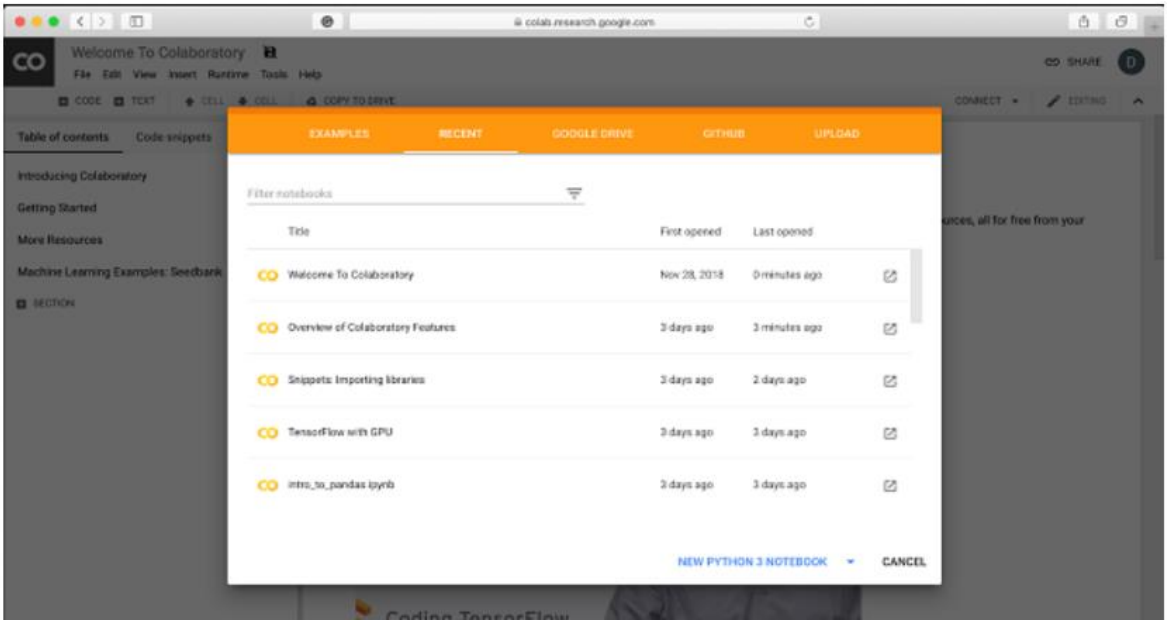


	Sr No	SAP_ID	Name	Subject	University
0	1	70144780	MUHAMMAD HASSAN MADNI	AI	UOL
1	2	70144904	QAZI ZARYAB SAJJAD	AI	UOL
2	3	70144910	MOHTISHIM FAREED	AI	UOL
3	4	70145312	SHIZA ISHAQ	AI	UOL
4	5	70145452	MALAIKAH KHALID	AI	UOL

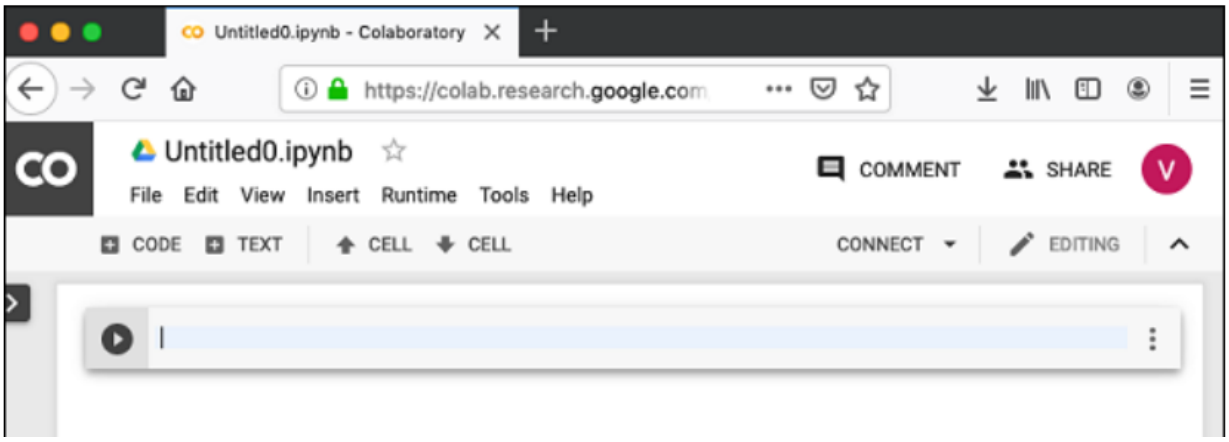
Introduction of Google Colab

- Colab is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs.
- Colab supports many popular machine learning libraries which can be easily loaded in your notebook.
- Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science.
- Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook

- **Step 1** − Open the following URL in your browser − https://colab.research.google.com Your browser would display the following screen (assuming that you are logged into your Google Drive) −

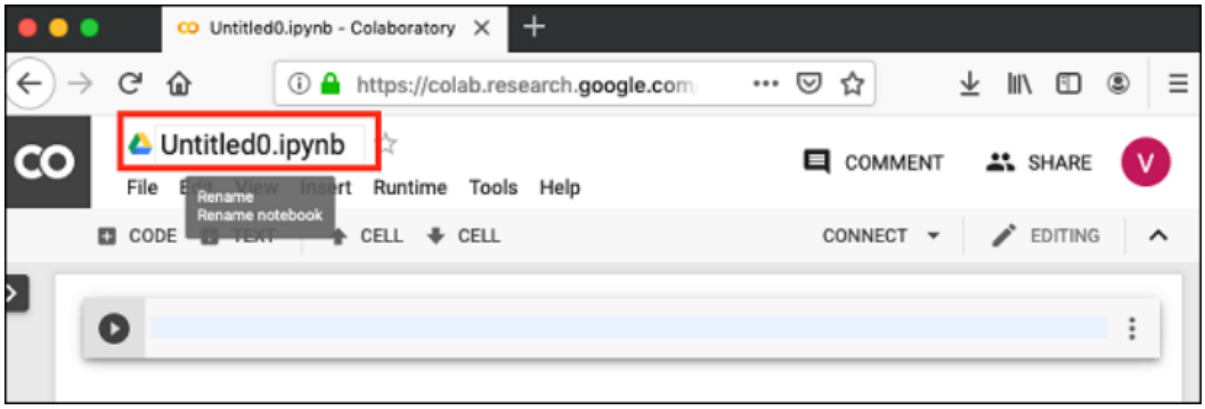


- **Step 2** − Click on the **NEW PYTHON 3 NOTEBOOK** link at the bottom of the screen. A new notebook would open up as shown in the screen below.

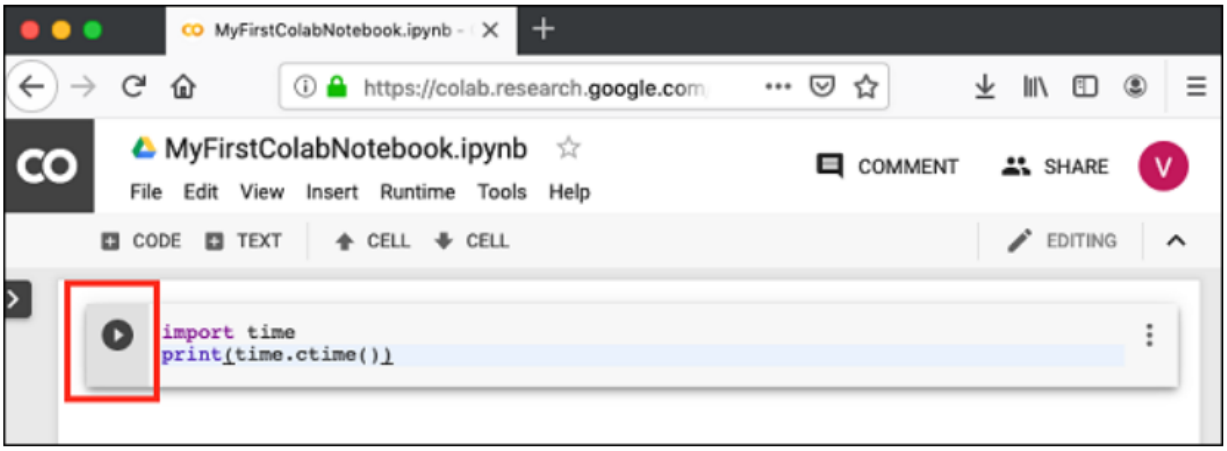


- **Setting Notebook Name**

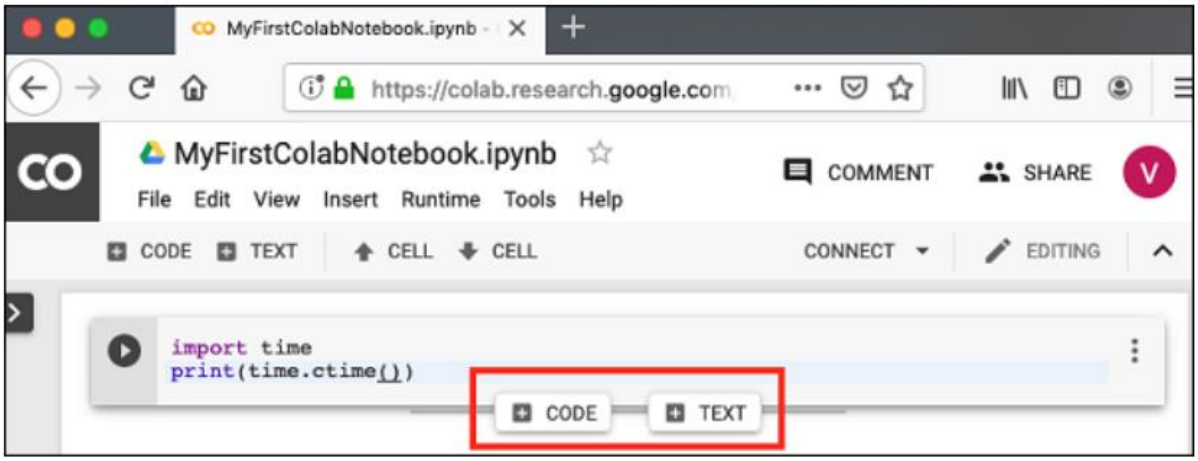
By default, the notebook uses the naming convention UntitledXX.ipynb. To rename the notebook, click on this name and type in the desired name in the edit box as shown here −



- Entering Code

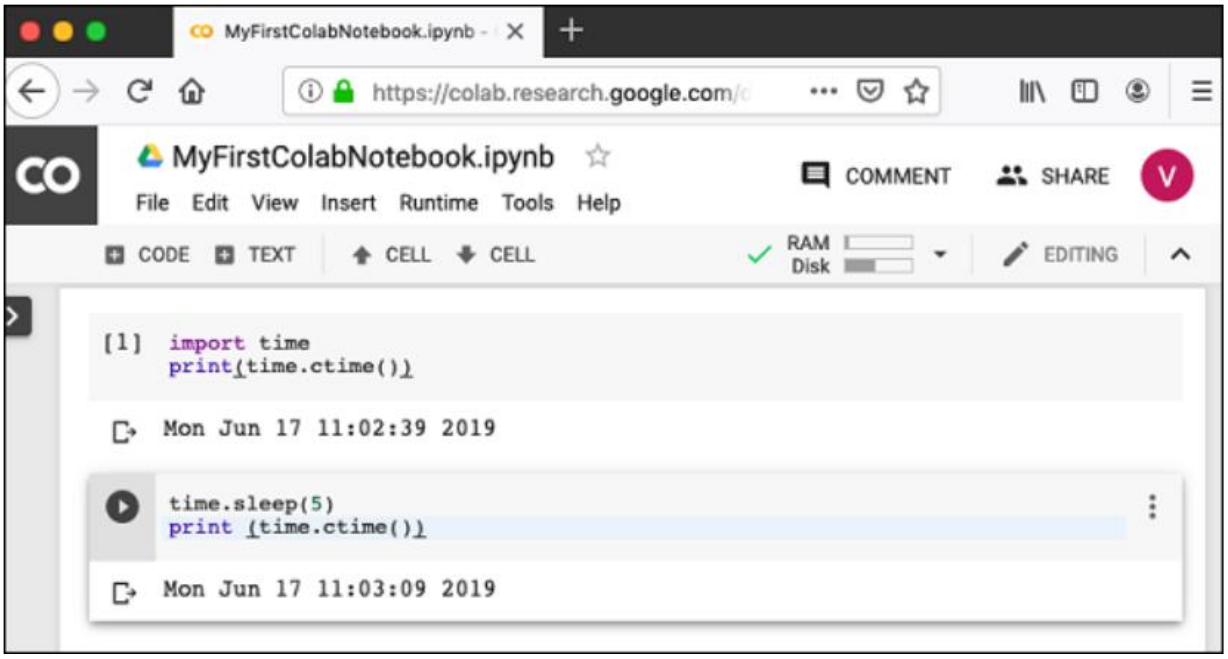


- Adding Code Cells
- To add more code to your notebook, select the following **menu** options −

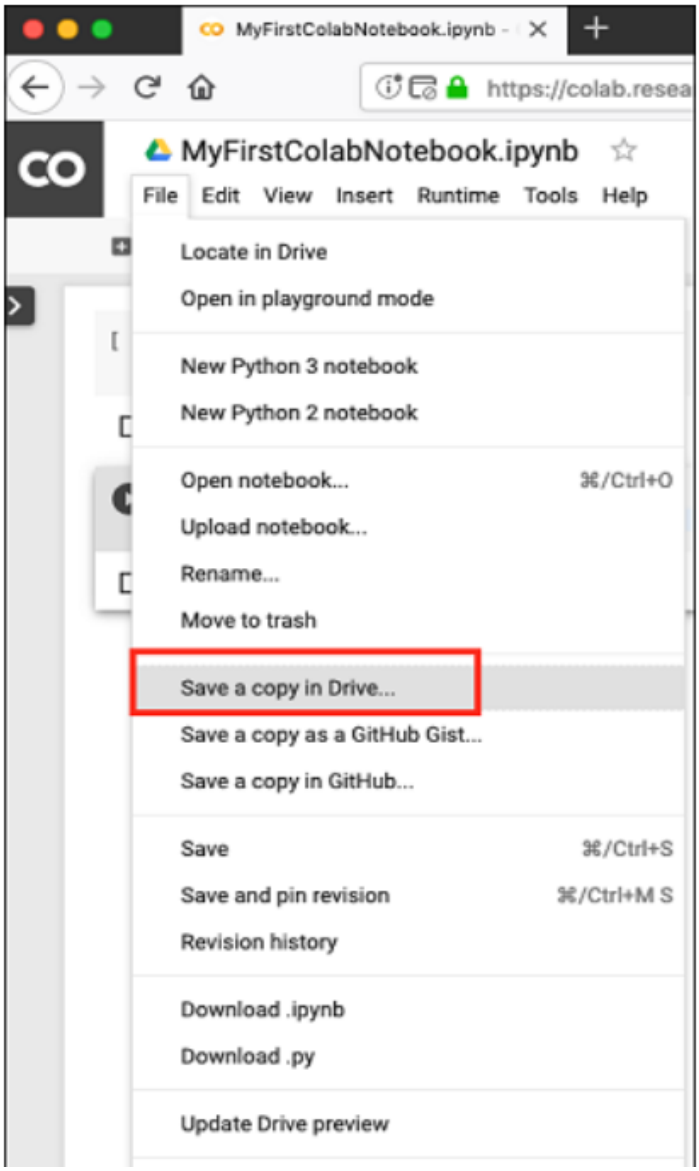


- **Run All**

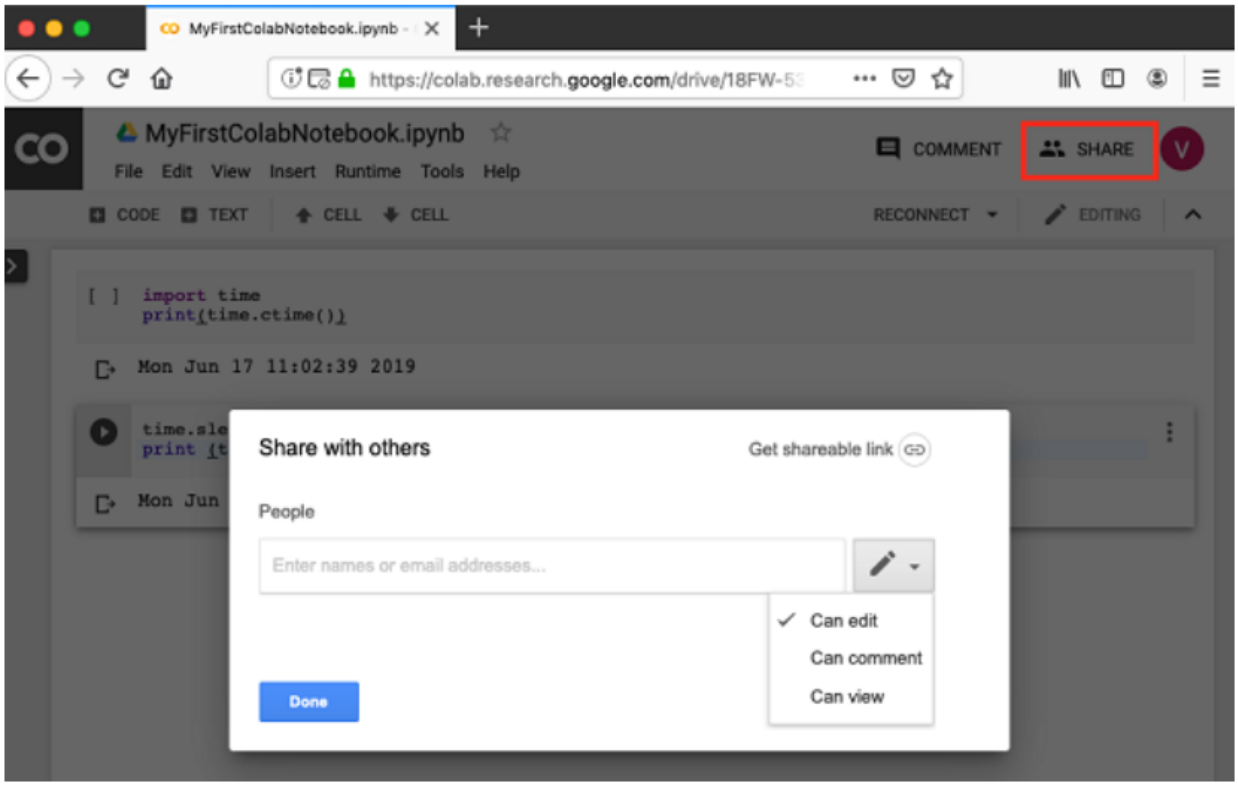
  To run the entire code in your notebook without an interruption, execute the following menu options −



Saving to Google Drive



Google Colab - Sharing Notebook


MyFirstColabNotebook.ipynb
https://colab.research.google.com/drive/18FW-5
MyFirstColabNotebook.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE TEXT CELL CELL
RECONNECT
EDITING
[ ] import time
print(time.ctime())
Mon Jun 17 11:02:39 2019
Share with others
Get shareable link
People
Enter names or email addresses...
Can edit
Can comment
Can view
Done

Lab Assignment No. 1

Objective

To learn how to create and upload a dataset in Python, perform basic statistical analysis, and visualize data using graphs.

Tasks

🔷 Q1: Create a Dataset Manually

- Create a dataset of at least **10 students** with the following columns:
    - Student_ID,
    - Name,
    - Age,
    - Marks_Math,
    - Marks_Science.
- Store the dataset in a **CSV file** named students.csv.

Q2: Upload Dataset in Python

- Use **Pandas** to load the dataset.

Q3: Observe Dataset Information

Run the following commands and explain the output:

1. data.info() → Dataset structure
2. data.describe() → Summary statistics (mean, std, min, max, etc.)
3. data['Marks_Math'].mean() → Mean of Math marks
4. data['Marks_Science'].max() → Maximum Science marks

Q4: Perform Some Data Analysis

- Find how many students have Marks_Math > 50.
- Find the student with the **highest Science marks**.
- Calculate the **correlation** between Marks_Math and Marks_Science.

◈ Q5: Data Visualization

Use **Matplotlib/Seaborn** to create graphs:

1. A bar chart of Student_ID vs Marks_Math.
2. A histogram of Age.
3. A scatter plot of Marks_Math vs Marks_Science.

◈ Q6: Save Your Work

- Save your notebook as Lab1_Assignment.ipynb.
- Submit the notebook file along with screenshots of graphs and outputs.

Q1: Create a Dataset Manually

Create a CSV file named **students.csv** with the following data.

	Student_ID	Name	Age	Marks_Math	Marks_Science
0	S001	Hammad	24	72	68
1	S002	Tehreem	21	60	88
2	S003	Abdullah	22	60	74
3	S004	Talha	24	73	82
4	S005	Aresha	20	85	46
5	S006	Ahmad	22	89	65
6	S007	Fatima	22	73	77
7	S008	Jawad	24	52	56
8	S009	Salman	19	71	66
9	S010	Faizan	20	51	88

Q2: Upload Dataset in Python

```python
print("\n--- Q2: Upload Dataset in Python ---\n")

import pandas as pd

data = pd.read_csv("students.csv")
print(data)
```

 Output

```
  Student_ID      Name  Age  Marks_Math  Marks_Science
0       S001    Hammad   24          72             68
1       S002   Tehreem   21          60             88
2       S003  Abdullah   22          60             74
3       S004     Talha   24          73             82
4       S005    Aresha   20          85             46
5       S006     Ahmad   22          89             65
6       S007    Fatima   22          73             77
7       S008     Jawad   24          52             56
8       S009    Salman   19          71             66
9       S010    Faizan   20          51             88
```

Q3: Observe Dataset Information

Dataset Structure

```python
print("\n--- Dataset Information ---\n")
print(data.info())
```

```
--- Dataset Information ---

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Student_ID     10 non-null     object
 1   Name           10 non-null     object
 2   Age            10 non-null     int64 
 3   Marks_Math     10 non-null     int64 
 4   Marks_Science  10 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 532.0+ bytes
None
```

Summary Statistics

```
print("\n--- Summary Statistics ---\n")
print(data.describe())
```

```
--- Summary Statistics ---

             Age  Marks_Math  Marks_Science
count  10.000000   10.000000      10.000000
mean   21.800000   68.600000      71.000000
std     1.813529   12.799306      13.597385
min    19.000000   51.000000      46.000000
25%    20.250000   60.000000      65.250000
50%    22.000000   71.500000      71.000000
75%    23.500000   73.000000      80.750000
max    24.000000   89.000000      88.000000
```

Mean of Math Marks

```
print("\nMean of Math Marks:", data["Marks_Math"].mean())
```

```
Mean of Math Marks: 68.6
```

Maximum Science Marks

```
print("\nMaximum Science Marks:", data["Marks_Science"].max())
```

```
Maximum Science Marks: 88
```

Q4: Perform Some Data Analysis

Students with Math Marks > 50

```
math_above_50 = (data["Marks_Math"] > 50).sum()
print("\nStudents with Math Marks > 50:", math_above_50)
```

OUTPUT

```
Students with Math Marks > 50: 10
```

Student with Highest Science Marks

```
highest_science = data[data["Marks_Science"] == data["Marks_Science"].max()]
print("\nStudent with Highest Science Marks:\n", highest_science)
```

OUTPUT

```
Student with Highest Science Marks:
   Student_ID     Name  Age  Marks_Math  Marks_Science
1        S002  Tehreem   21          60             88
9        S010   Faizan   20          51             88
```

Correlation Between Math & Science Marks

```
print("\nCorrelation between Math and Science Marks:\n")
print(data[["Marks_Math", "Marks_Science"]].corr())
```

OUTPUT

```
  Correlation between Math and Science Marks:

                Marks_Math  Marks_Science
  Marks_Math        1.00000       -0.44818
  Marks_Science    -0.44818        1.00000
```

Q5: Data Visualization

```
print("\n--- Q5: Data Visualization ---\n")
import matplotlib.pyplot as plt
```

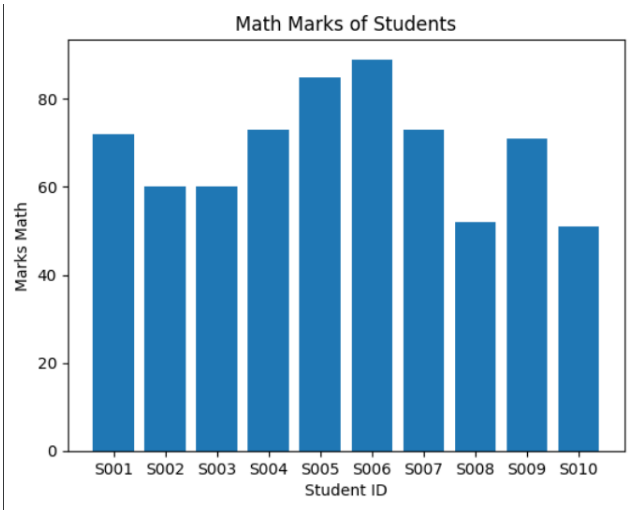
OUTPUT

```
--- Q5: Data Visualization ---
```

Bar Chart: Student_ID vs Marks_Math

```
plt.bar(data["Student_ID"], data["Marks_Math"])
plt.title("Math Marks of Students")
plt.xlabel("Student ID")
plt.ylabel("Marks Math")
plt.show()
```

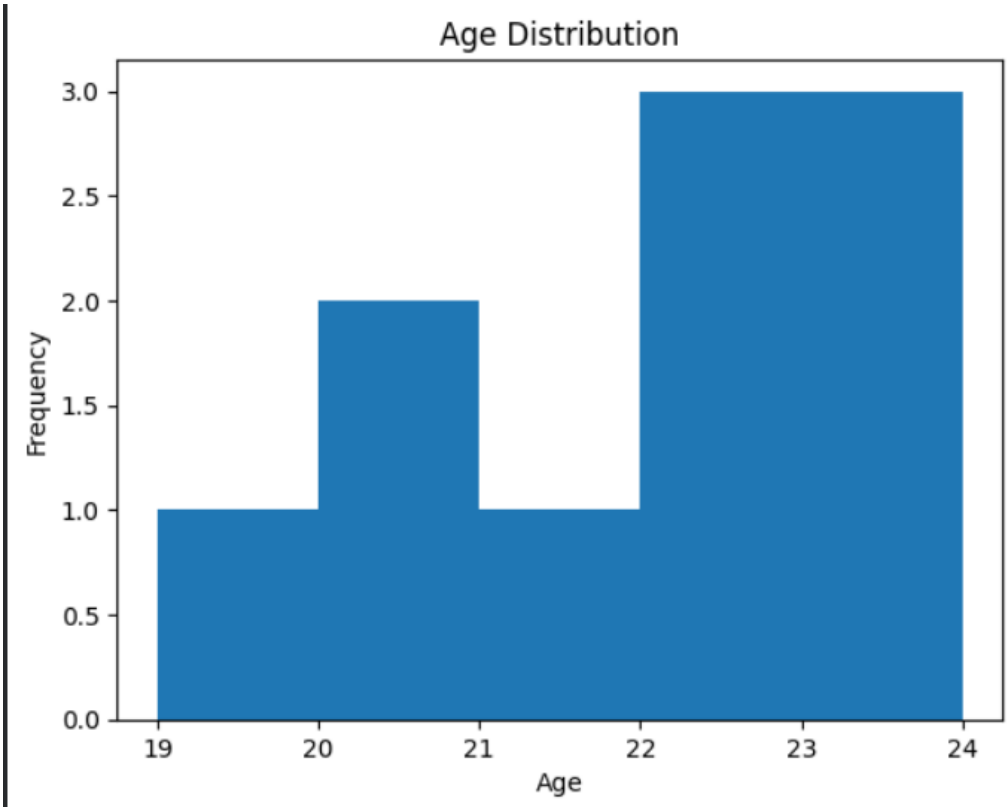
OUTPUT



Histogram of Age

```
plt.hist(data["Age"], bins=5)
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

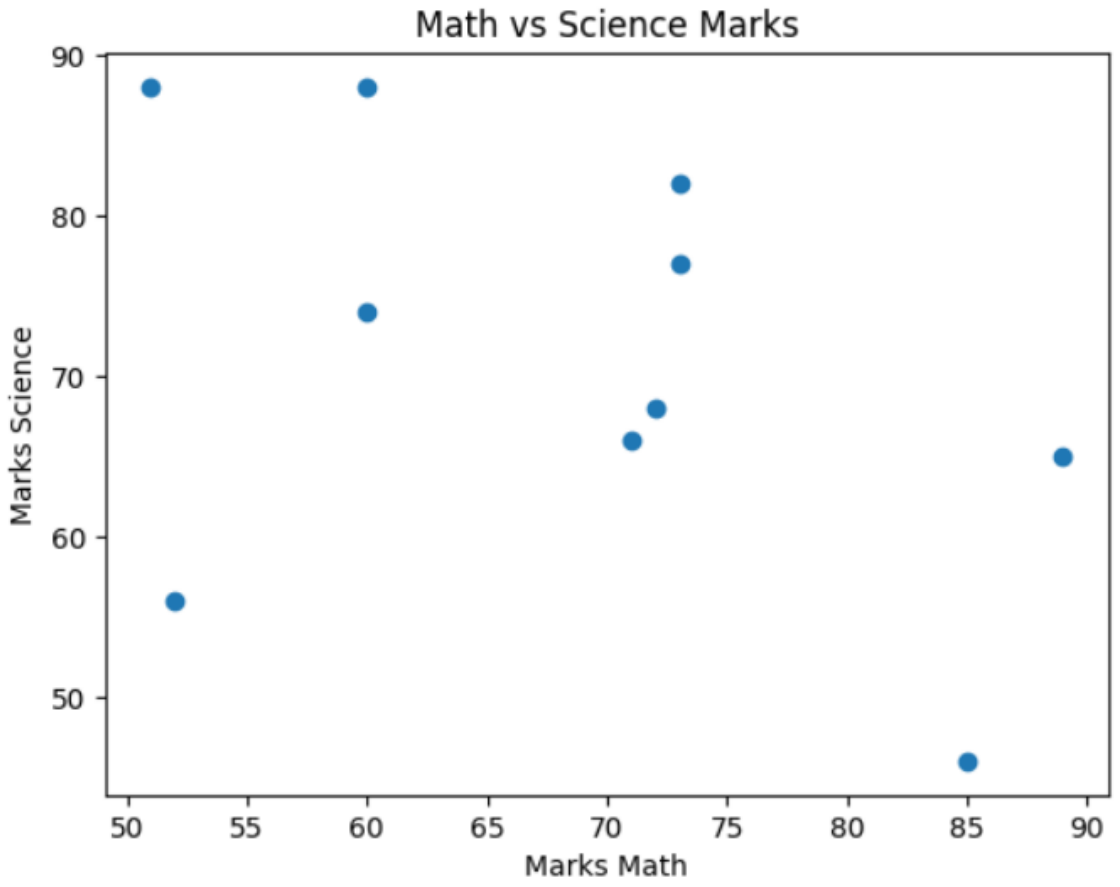
OUTPUT



Scatter Plot: Math vs Science Marks

```
plt.scatter(data["Marks_Math"], data["Marks_Science"])
plt.title("Math vs Science Marks")
plt.xlabel("Marks Math")
plt.ylabel("Marks Science")
```

OUTPUT



LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

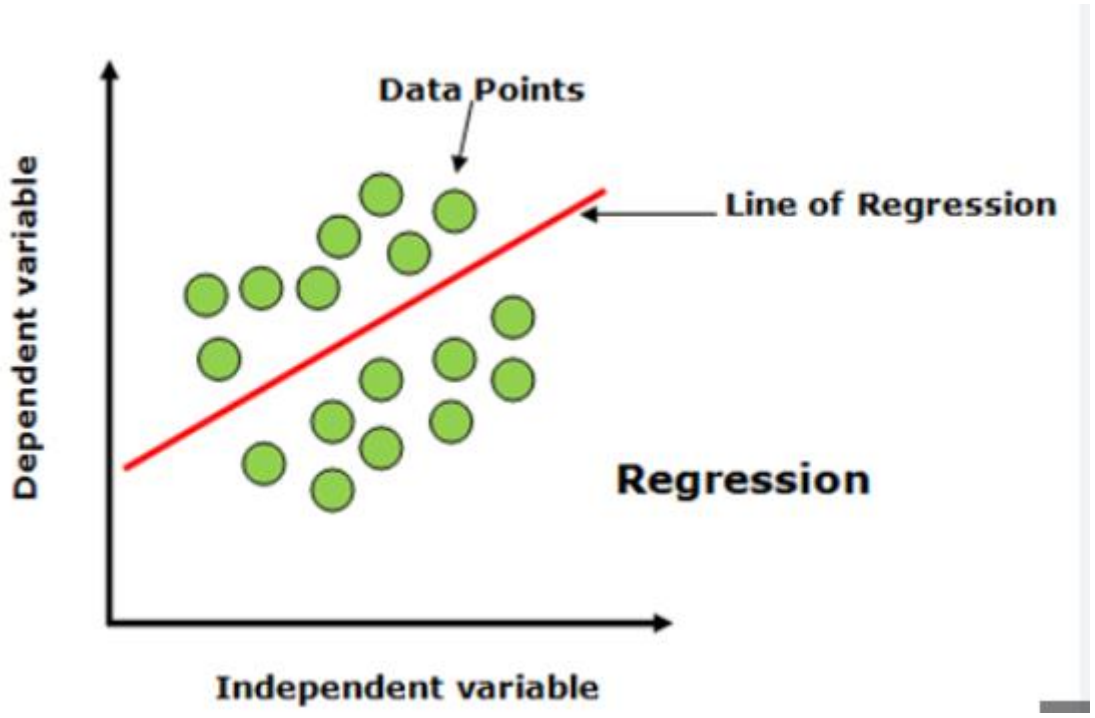
LAB No. 2

Implementation of Regression Analysis using python

Regression techniques are used to analyze relationships between variables and make predictions. Multiple Linear Regression predicts a continuous output using multiple input variables, while Logistic Regression is used for classification problems with binary outcomes. In this lab, students will apply both methods using Python to understand data modeling, prediction, and basic performance evaluation.

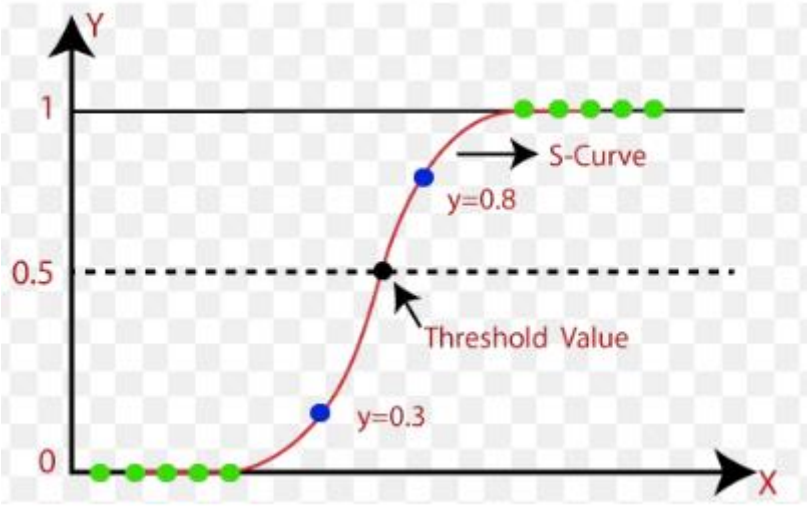
Introduction

Regression analysis is a fundamental technique in machine learning and data science used to understand the relationship between variables and to make predictions. In this lab, students will explore **Multiple Linear Regression** and **Logistic Regression** using Python.



Multiple Linear Regression is used when a dependent (output) variable is continuous and depends on two or more independent (input) variables. It helps in modeling and analyzing how changes in multiple features affect a numerical outcome, such as predicting house prices based on area, number of rooms, and location.

Logistic Regression is used for classification problems where the dependent variable is categorical, usually binary (such as Yes/No, True/False, or 0/1). It estimates the probability that an input belongs to a particular class and is widely used in applications such as disease prediction, spam detection, and customer churn analysis.



Solved Examples

Example 1

A dataset contains information about students' **study hours** and **attendance percentage**. Predict the **final marks** using Multiple Linear Regression.

Solution:

```
# Import required libraries
import pandas as pd
from sklearn.linear_model import LinearRegression

# Create dataset
data = {
    'Study_Hours': [2, 4, 6, 8, 10],
    'Attendance': [60, 70, 80, 90, 95],
    'Marks': [50, 60, 70, 85, 90]
}

df = pd.DataFrame(data)

# Independent and dependent variables
X = df[['Study_Hours', 'Attendance']]
y = df['Marks']

# Create and train model
model = LinearRegression()
model.fit(X, y)

# Prediction
prediction = model.predict([[7, 85]])
```

```
print("Predicted Marks:", prediction[0])
```

Example 2

Predict the **house price** based on **area (sq ft)** and **number of bedrooms** using Multiple Linear Regression.

Solution:

```
import pandas as pd
from sklearn.linear_model import LinearRegression

# Dataset
data = {
    'Area': [800, 1000, 1200, 1500, 1800],
    'Bedrooms': [1, 2, 2, 3, 3],
    'Price': [50000, 65000, 75000, 90000, 110000]
}

df = pd.DataFrame(data)

X = df[['Area', 'Bedrooms']]
y = df['Price']

model = LinearRegression()
model.fit(X, y)

# Predict price
predicted_price = model.predict([[1400, 3]])

print("Predicted House Price:", predicted_price[0])
```

Example 3

A dataset contains **study hours** and **attendance** information. Predict whether a student will **pass (1)** or **fail (0)** using Logistic Regression.

Solution:

```
import pandas as pd
from sklearn.linear_model import LogisticRegression

# Dataset
data = {
    'Study_Hours': [1, 2, 4, 6, 8, 10],
    'Attendance': [50, 55, 65, 75, 85, 95],
    'Result': [0, 0, 0, 1, 1, 1]  # 0 = Fail, 1 = Pass
}
```

```
df = pd.DataFrame(data)

X = df[['Study_Hours', 'Attendance']]
y = df['Result']

# Create and train model
model = LogisticRegression()
model.fit(X, y)

# Predict pass/fail
result = model.predict([[5, 70]])

print("Predicted Result (1=Pass, 0=Fail):", result[0])
```

LAB Assignment No. 2

Lab Practice Questions

Q1. Multiple Linear Regression – House Price Prediction

A dataset contains:

- Size (sqft),
- Number of Bedrooms,
- Age of House (years)

and the target variable is **House Price**.

👉 Task:

1. Fit a **multiple linear regression model**.
2. Predict the price of a house with: **Size = 2000 sqft**, **Bedrooms = 3**, Age = **10 years.**
3. Print coefficients and interpret them.

Q2. Multiple Linear Regression – Student Performance

Dataset columns:

- Hours Study,
- Hours Sleep,
- Attendance (%),
  Target: **Marks in Exam**

👉 Task:

1. Train a regression model.
2. Plot actual vs predicted marks.
3. Compute **R^2 score** and **Mean Squared Error (MSE)**.

Q3. Logistic Regression – Pass/Fail Classification

Dataset columns:

- Hours Study
- Hours Sleep
  Target: Pass (1) / Fail (0)

👉 Task:

1. Fit a **logistic regression classifier**.

2. Predict the probability of passing if a student studies 30 hours and sleeps 6 hours.
3. Plot the **decision boundary** (pass vs fail).

Q4. Logistic Regression – Diabetes Prediction (Binary Classification)

Use a small dataset with:

- BMI,
- Age,
- Glucose Level
  Target: **Diabetic (1) or Not (0)**

👉 Task:

1. Fit logistic regression.
2. Find accuracy, precision, recall.
3. Predict whether a patient (BMI=28, Age=45, Glucose=150) is diabetic.

Q5. Comparison – Linear vs Logistic Regression

Dataset columns:

- Hours Study,
- Exam Score,
- Pass/Fail

👉 Task:

1. Use **Linear Regression** to predict exam scores.
2. Use **Logistic Regression** to predict pass/fail.
3. Compare results — explain why linear regression is unsuitable for classification.

📌 **Implementation Notes for Students:**

- Use pandas to load small CSVs (or create toy datasets directly in code).
- Use sklearn.linear_model.LinearRegression and LogisticRegression.
- Plot with matplotlib.
- Interpret coefficients in both models.

Q1: Multiple Linear Regression – House Price Prediction

Python Code

```
import pandas as pd
from sklearn.linear_model import LinearRegression

data = {
    'Size': [800, 1200, 1500, 1800, 2200],
    'Bedrooms': [2, 3, 3, 4, 4],
    'Age': [20, 15, 10, 8, 5],
    'Price': [50000, 75000, 90000, 110000, 140000]
}

df = pd.DataFrame(data)

X = df[['Size', 'Bedrooms', 'Age']]
y = df['Price']

model = LinearRegression()
model.fit(X, y)

predicted_price = model.predict([[2000, 3, 10]])

print("Predicted House Price:", predicted_price[0])
print("Coefficients:", model.coef_)
```

Output

```
Predicted House Price: 134086.16187989555
Coefficients: [   88.38120104 -1396.86684073  2114.88250653]
```

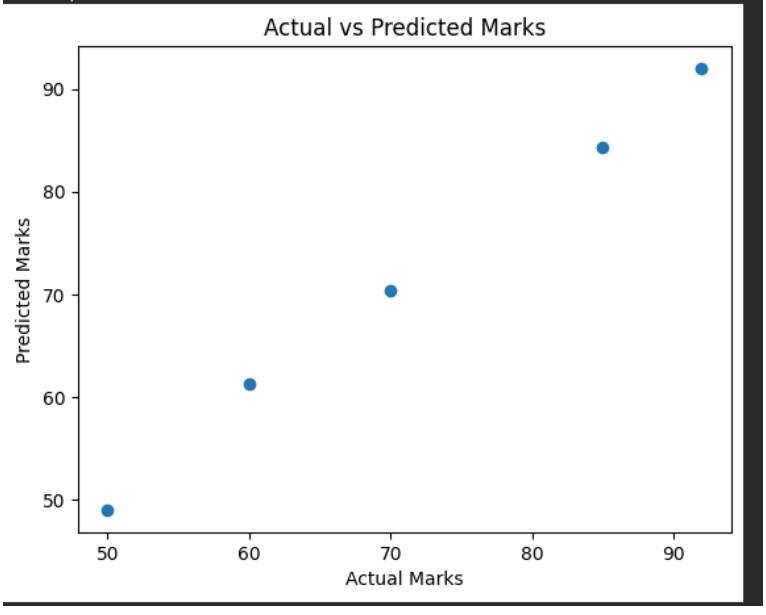
Q2: Multiple Linear Regression – Student Performance

Python Code

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

data = {
    'Hours_Study': [2, 4, 6, 8, 10],
    'Hours_Sleep': [6, 7, 6, 8, 7],
    'Attendance': [60, 70, 80, 90, 95],
    'Marks': [50, 60, 70, 85, 92]
}

df = pd.DataFrame(data)

X = df[['Hours_Study', 'Hours_Sleep', 'Attendance']]
y = df['Marks']

model = LinearRegression()
model.fit(X, y)

predicted_marks = model.predict(X)

print("R² Score:", r2_score(y, predicted_marks))
print("Mean Squared Error:", mean_squared_error(y, predicted_marks))

plt.scatter(y, predicted_marks)
plt.xlabel("Actual Marks")
plt.ylabel("Predicted Marks")
plt.title("Actual vs Predicted Marks")
plt.show()
```

Output

```
R² Score: 0.9972203691349789
Mean Squared Error: 0.6666666666666676
```



Q3: Logistic Regression – Pass / Fail Classification

Python Code

```
import pandas as pd
from sklearn.linear_model import LogisticRegression

data = {
    'Hours_Study': [5, 10, 15, 20, 25, 30],
    'Hours_Sleep': [4, 5, 6, 6, 7, 8],
    'Pass': [0, 0, 0, 1, 1, 1]
}

df = pd.DataFrame(data)

X = df[['Hours_Study', 'Hours_Sleep']]
y = df['Pass']

model = LogisticRegression()
model.fit(X, y)

probability = model.predict_proba([[30, 6]])
print("Probability of Passing:", probability[0][1])
```

Output

```
Probability of Passing: 0.9999036598352675
```

Q4: Logistic Regression – Diabetes Prediction

Python Code

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score

data = {
    'BMI': [22, 25, 30, 35, 28, 40],
    'Age': [25, 35, 45, 50, 40, 60],
    'Glucose': [90, 110, 140, 180, 150, 200],
    'Diabetic': [0, 0, 1, 1, 1, 1]
}

df = pd.DataFrame(data)

X = df[['BMI', 'Age', 'Glucose']]
y = df['Diabetic']

model = LogisticRegression()
model.fit(X, y)

pred = model.predict(X)

print("Accuracy:", accuracy_score(y, pred))
print("Precision:", precision_score(y, pred))
print("Recall:", recall_score(y, pred))

print("Patient Prediction:", model.predict([[28, 45, 150]]))
```

Output

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
Patient Prediction: [1]
```

Q5: Comparison – Linear vs Logistic Regression

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
```

```
from sklearn.metrics import mean_squared_error, accuracy_score, confusion_matrix
```

```
# Sample dataset
data = {
    'Hours Study': [1,2,3,4,5,6,7,8,9,10],
    'Exam Score': [35, 50, 55, 60, 65, 70, 75, 80, 85, 90],
    'Pass/Fail': [0,0,0,1,1,1,1,1,1,1]
}

df = pd.DataFrame(data)
print(df)
```

```
   Hours Study  Exam Score  Pass/Fail
0            1          35          0
1            2          50          0
2            3          55          0
3            4          60          1
4            5          65          1
5            6          70          1
6            7          75          1
7            8          80          1
8            9          85          1
9           10          90          1
```

```
# Features and target
X = df[['Hours Study']]
y_score = df['Exam Score']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y_score, test_size=0.2,
random_state=42)
```

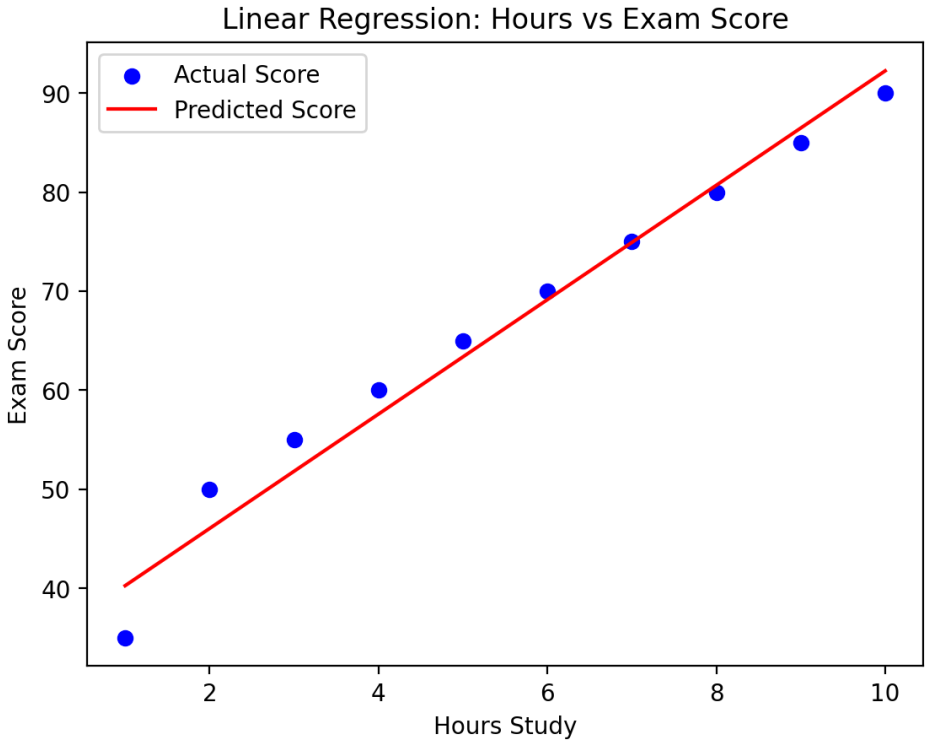
```
# Train Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

```
# Predict
y_pred_score = lin_reg.predict(X_test)
```

```
# Evaluate
```

```
mse = mean_squared_error(y_test, y_pred_score)
print("Linear Regression MSE:", mse)
```

```
# Plot
plt.scatter(X, y_score, color='blue', label='Actual Score')
plt.plot(X, lin_reg.predict(X), color='red', label='Predicted Score')
plt.xlabel('Hours Study')
plt.ylabel('Exam Score')
plt.title('Linear Regression: Hours vs Exam Score')
plt.legend()
plt.show()
```



LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized	The code is fairly easy to read. The code could be	Most of the code could be reused in other programs.	Some parts of the code require change before they	The code is poorly organized and very difficult to read and

		and very easy to follow and reused	reused as a whole or each class could be reused.		could be reused in other programs.	not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No 3

Decision Tree Classifier

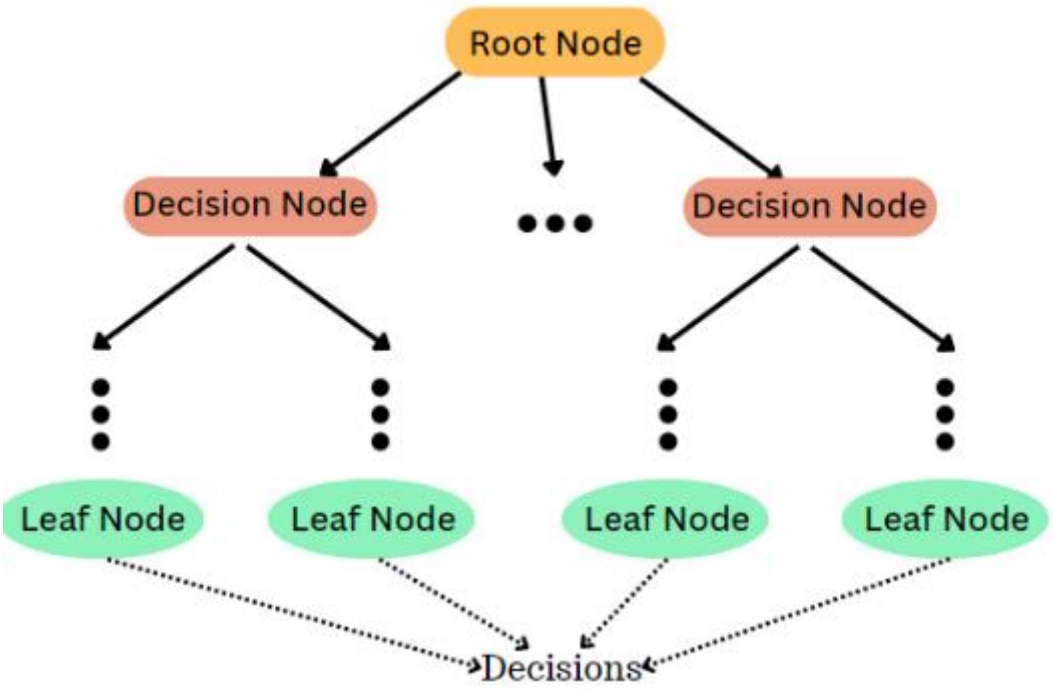
In this lab, students will study and implement the Decision Tree Classifier, a supervised machine learning algorithm used for classification tasks. The lab begins with a manual calculation of entropy and information gain to understand how decision trees choose splitting attributes. Students will then implement a decision tree on a small categorical dataset, followed by applying the algorithm to a real-world dataset (Iris) using Python and Scikit-learn. Through this lab, students will gain both theoretical clarity and practical experience in building, training, and visualizing decision tree models.

Introduction & Theory

A **Decision Tree Classifier** is a tree-structured model used to make decisions based on feature values. Each internal node represents a test on an attribute, each branch represents an outcome, and each leaf node represents a class label.

The construction of a decision tree is based on:

- **Entropy**: A measure of uncertainty or impurity in a dataset
- **Information Gain**: Reduction in entropy after splitting on an attribute



Decision trees are easy to interpret and visualize but may suffer from **overfitting**, especially on large or complex datasets.

Solved Examples:

Example 1: Entropy and Information Gain

Dataset

Student	Study Hours	Attendance	Result
S1	Low	Poor	Fail
S2	High	Good	Pass
S3	High	Poor	Pass
S4	Low	Good	Fail
S5	High	Good	Pass

1. Entropy of Target Variable (Result)

- Pass = 3
- Fail = 2

$$Entropy(Result) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$Entropy(Result) = -(0.6 \times -0.737 + 0.4 \times -1.322)$$

$$Entropy(Result) \approx 0.971$$

2. Information Gain for Study Hours

Study Hours = High

- Pass = 3, Fail = 0
- Entropy = 0

Study Hours = Low

- Pass = 0, Fail = 2
- Entropy = 0

Weighted Entropy:

$$= \frac{3}{5} \times 0 + \frac{2}{5} \times 0 = 0$$

$$IG(StudyHours) = 0.971 - 0 = 0.971$$

3. Root Node Selection

Since **Study Hours** provides the **maximum information gain**, it should be selected as the **root node**.

Example 2: Decision Tree on Small Dataset (Python Implementation)

Question

Build and visualize a decision tree using entropy.

Solution:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical values
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Train decision tree
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X, y)

# Visualize tree
plt.figure(figsize=(10,6))
plot_tree(model, feature_names=X.columns, class_names=['Fail', 'Pass'], filled=True)
plt.show()
```

Example 3: Decision Tree Classifier on Iris Dataset

Objective

Apply decision trees to a real dataset and evaluate accuracy.

Solution

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train model
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)


# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

# Visualize tree
plt.figure(figsize=(14,8))
plot_tree(model, feature_names=iris.feature_names,
          class_names=iris.target_names, filled=True)
plt.show()
```

LAB Assignment No. 3

Question 1

Entropy and Information Gain (Manual Calculation)

Given the dataset below about whether students pass an exam based on study time and attendance:

Student	Study Hours	Attendance	Result
S1	Low	Poor	Fail
S2	High	Good	Pass
S3	High	Poor	Pass
S4	Low	Good	Fail
S5	High	Good	Pass

1. Calculate the **entropy** of the target variable (Result).
2. Compute the **information gain** for the attribute Study Hours.
3. Which attribute should be selected for the root node based on maximum information gain?

Question No. 2

Implement Decision Tree Classifier on a Small Dataset

Build and visualize a simple decision tree.

Question:
Using the same dataset as above:

1. Use pandas to create a DataFrame.
2. Convert categorical values into numerical using LabelEncoder.
3. Train a **DecisionTreeClassifier** using **criterion='entropy'**.
4. Visualize the decision tree using plot_tree() from sklearn.tree.

Question 3

Decision Tree Classifier on Iris Dataset

Objective: Apply decision trees to a real dataset.

Question:

1. Load the **Iris dataset** using sklearn.datasets.load_iris.
2. Split it into training (70%) and testing (30%) sets.
3. Train a decision tree using **criterion='entropy'**.
4. Print the accuracy on the test set.
5. Visualize the tree and explain which feature provides the most information gain at the root.

Question 4

MNIST digit dataset (available via Keras / sklearn.datasets) as a baseline

Objectives

- Preprocess image data for classification
- Train a **Decision Tree Classifier** (or variants)
- Evaluate accuracy, confusion matrix, and discuss limitations

LAB Assignment No. 3

Question 1: Entropy and Information Gain (Manual Calculation)

Given Dataset

Student	Study Hours	Attendance	Result
S1	Low	Poor	Fail
S2	High	Good	Pass
S3	High	Poor	Pass
S4	Low	Good	Fail
S5	High	Good	Pass

1. Entropy of Target Variable (Result)

Total records = 5
Pass = 3
Fail = 2

$$Entropy(Result) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$Entropy(Result) \approx 0.971$$

2. Information Gain for Study Hours

Study Hours = High

- Pass = 3

- Fail = 0

Entropy = 0

Study Hours = Low

- Pass = 0
- Fail = 2

Entropy = 0

IG(StudyHours)=0.971−0=0.971

3. Root Node Selection

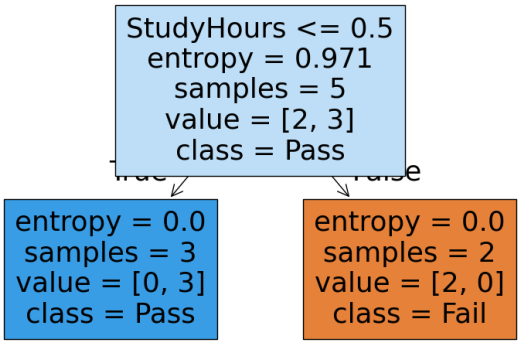
 Study Hours has the **maximum information gain**, so it should be selected as the **root node**.

Question 2: Decision Tree on Small Dataset (Python Implementation)

Python Code

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical data
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Train Decision Tree
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X, y)

# Visualize tree
plt.figure(figsize=(10,6))
plot_tree(model, feature_names=X.columns,
          class_names=['Fail', 'Pass'],
          filled=True)
plt.show()
```

Output



Explanation

- Data is converted into numeric form using *LabelEncoder*
- Entropy is used to split nodes

- Tree visualization shows decision rules clearly

Question 3: Decision Tree Classifier on Iris Dataset

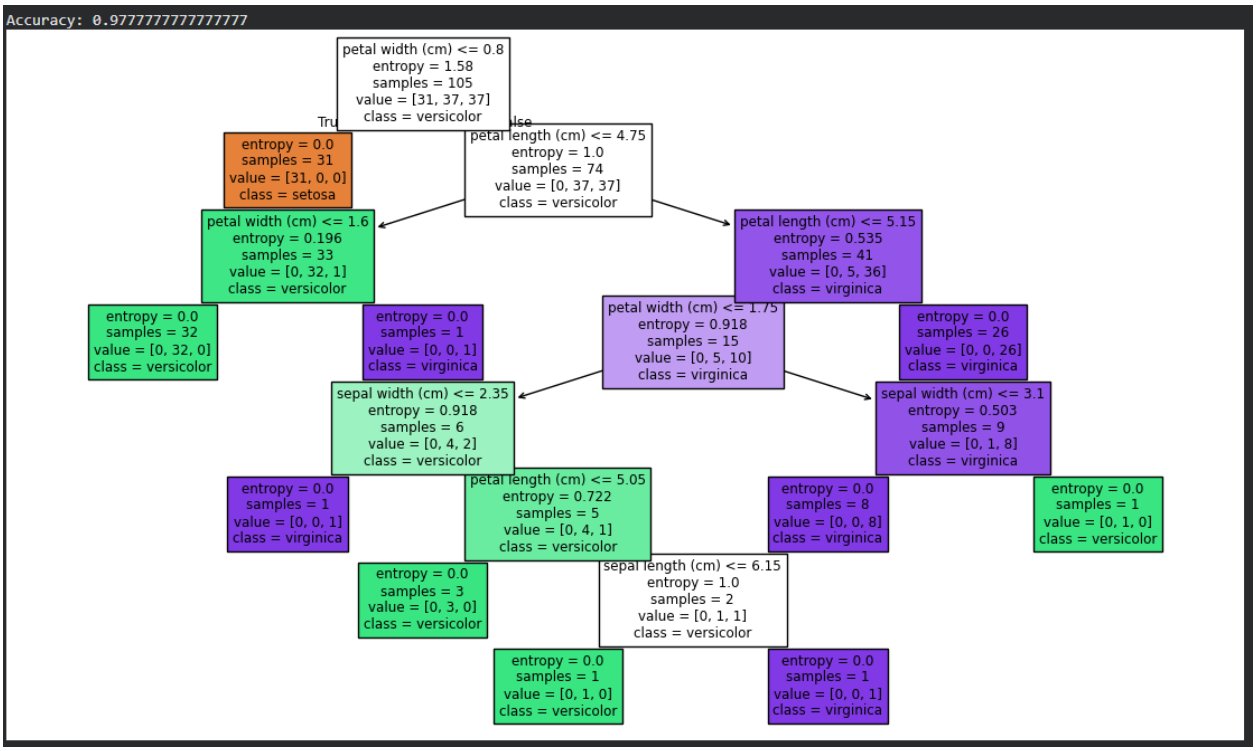
Objective

Apply Decision Tree on a real-world dataset and evaluate accuracy.

Python Code

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train model
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Visualize tree
plt.figure(figsize=(14,8))
plot_tree(model,
          feature_names=iris.feature_names,
          class_names=iris.target_names,
          filled=True)
plt.show()
```

Output



Question 4: MNIST Digit Dataset – Baseline Study

Objective

- Preprocess image data
- Train Decision Tree
- Evaluate performance

Python Code

```python
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Load dataset
digits = load_digits()
X = digits.data
y = digits.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train model
model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Output

```
Accuracy: 0.8777777777777778
Confusion Matrix:
 [[50  0  0  0  1  1  1  0  0  0]
 [ 0 44  1  1  0  0  2  0  1  1]
 [ 0  1 42  3  0  0  0  0  1  0]
 [ 0  1  2 47  0  0  0  0  2  2]
 [ 0  2  0  0 53  0  2  2  1  0]
 [ 0  3  1  0  5 53  0  1  2  1]
 [ 0  0  0  0  1  0 52  0  0  0]
 [ 0  3  1  0  0  0  0 48  0  3]
 [ 0  0  1  2  0  0  0  4 36  0]
 [ 2  0  0  2  1  1  0  1  3 49]]
```

Discussion & Limitations

- Decision Trees work on MNIST but are **not ideal** for high-dimensional image data
- They overfit easily
- Deep learning models perform better on images

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality**	Application runs smoothly and	Application compiles with no	Application compiles with	Application compiles and	Application does not compile or

	10%	operation of the application runs efficiently	warnings. Robust operation of the application, with good recovery.	few or no warnings. Consideration given to unusual conditions with reasonable	runs without crashing. Some attempt at detecting and correcting errors.	compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No. 4

Random Forest and Support Vector Machine Classifier

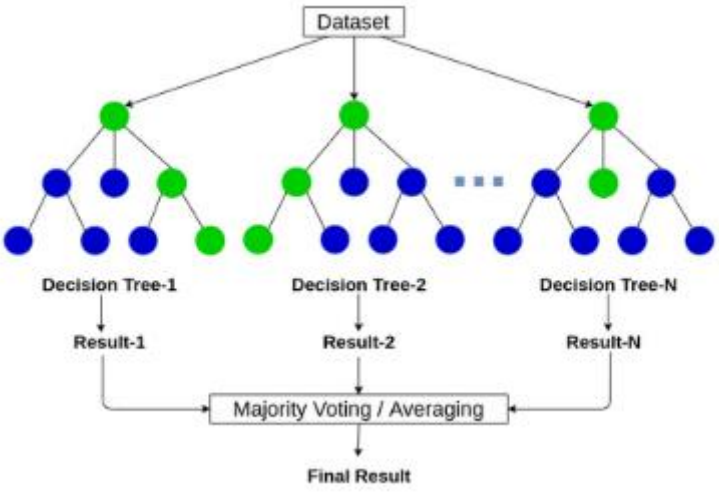
In this lab, students will learn and implement two powerful supervised machine learning algorithms: Random Forest Classifier and Support Vector Machine (SVM) Classifier. The lab focuses on understanding how ensemble learning improves classification accuracy using Random Forests and how SVM constructs optimal decision boundaries for classification problems.

Students will begin with a small dataset to understand model behavior and then apply both classifiers to real-world datasets using Python and Scikit-learn. Model performance will be evaluated using accuracy metrics, and comparisons will be made between Random Forest and SVM classifiers.

Introduction & Theory

Random Forest Classifier

Random Forest is an **ensemble learning method** that builds multiple decision trees and combines their outputs to make a final prediction. Each tree is trained on a random subset of data and features, which improves accuracy and reduces overfitting.



Advantages:

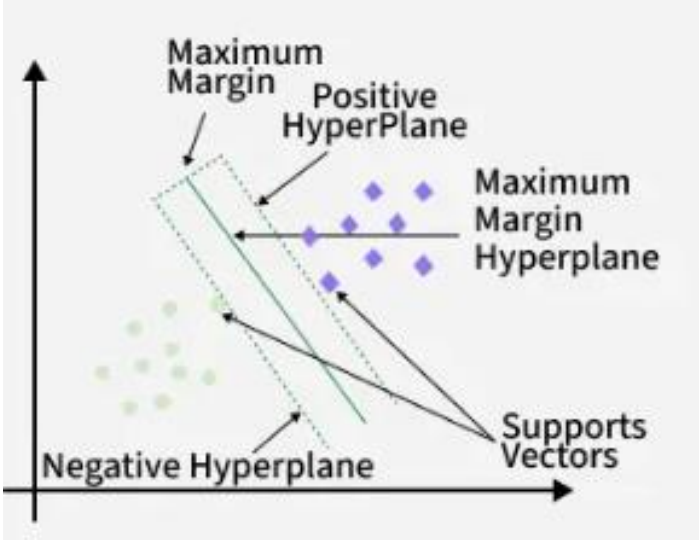
- High accuracy
- Reduces overfitting
- Works well with large datasets

Support Vector Machine (SVM) Classifier

Support Vector Machine is a supervised learning algorithm that finds the **optimal hyperplane** that best separates data points of different classes. SVM can perform both linear and non-linear classification using **kernel functions**.

Advantages:

- Effective in high-dimensional spaces
- Works well with small datasets
- Strong theoretical foundation



Solved Examples

Example 1

Build a Random Forest classifier to predict whether a student passes or fails based on study hours and attendance.

Solution:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)
```

```
# Encode categorical variables
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Prediction
prediction = model.predict([[1, 1]])
print("Predicted Result (1=Pass, 0=Fail):", prediction[0])
```

Example 2:

Use an SVM classifier to predict whether a student passes or fails using the same dataset.

Solution:

```
from sklearn.svm import SVC

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical variables
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Train SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X, y)

# Prediction
svm_prediction = svm_model.predict([[1, 1]])
```

```
print("Predicted Result (1=Pass, 0=Fail):", svm_prediction[0])
```

Example 3

Compare Random Forest and SVM classifiers on IRIS dataset.

Solution

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# SVM model
svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)

# Accuracy
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print("SVM Accuracy:", accuracy_score(y_test, svm_pred))
```

Comparison Summary

Algorithm	Strengths	Limitations
Random Forest	High accuracy, less overfitting	Slower, less interpretable
SVM	Effective in high dimensions	Sensitive to kernel choice

LAB Assignment No 4

Topic: Random Forest and Support Vector Machine Classifier

Question 1

Classify flower species using Random Forest.

Task:

1. Load the *Iris dataset* from sklearn.datasets.

2. Split into training (70%) and testing (30%) sets.
3. Train a **Random Forest Classifier**.
4. Predict flower species on the test set.
5. Calculate and print **model accuracy**.

Question 2

Use SVM on Breast Cancer Dataset and Classify tumors as malignant or benign.

Task:

1. Load the *Breast Cancer* dataset using sklearn.datasets.load_breast_cancer.
2. Train an **SVM classifier** (use SVC(kernel='linear')).
3. Evaluate the model using **accuracy** and **confusion matrix**.

Question 3

Use Random Forest on CSV Dataset (Custom) : Predict student pass/fail based on study hours and scores.

Task:

1. Load a CSV file (e.g., students.csv) with columns: study_hours, attendance, marks, result.
2. Train a **Random Forest Classifier** to predict result (Pass/Fail).
3. Display **accuracy score** and **feature importance**.

Question 4

Use SVM on Digits Dataset and to identify the: Handwritten digit recognition.

Task:

1. Load the *Digits dataset* from sklearn.datasets.load_digits.
2. Train an **SVM classifier** with an RBF kernel.
3. Test on unseen data.
4. Print **accuracy** and visualize some **misclassified samples**.

Question 5:

Compare Random Forest vs SVM on Same Dataset (you can choose any dataset): Compare two models on the same data.

Task:

- Use the *Wine dataset* from sklearn.datasets.load_wine.
- Train both:

  RandomForestClassifier(n_estimators=100)

- SVC(kernel='rbf')
- Print accuracy of both models.
- Conclude which performs better.

LAB Assignment No. 4

Question 1: Random Forest on Iris Dataset

Python Code

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict
y_pred = rf_model.predict(X_test)

# Accuracy
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred))
```

Output

```
Random Forest Accuracy: 1.0
```

Question 2: SVM on Breast Cancer Dataset

Python Code

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix

# Load dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train SVM
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# Predict
y_pred = svm_model.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Output

```
SVM Accuracy: 0.9649122807017544
Confusion Matrix:
 [[ 59   4]
 [  2 106]]
```

Question 3: Random Forest on Custom CSV Dataset

Python Code

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

data = pd.read_csv("students.csv")

print("Dataset Columns:")
print(data.columns)

target_column = data.columns[-1]

X = data.select_dtypes(include=['int64', 'float64'])

if target_column in X.columns:
    X = X.drop(columns=[target_column])

y = data[target_column]

if y.dtype == 'object':
    encoder = LabelEncoder()
    y = encoder.fit_transform(y)

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

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("\nAccuracy:", accuracy_score(y_test, y_pred))

print("\nFeature Importance:")
for feature, importance in zip(X.columns, model.feature_importances_):
    print(feature, ":", importance)
```

Output

```
Dataset Columns:
Index(['Student_ID', 'Name', 'Age', 'Marks_Math', 'Marks_Science'], dtype='object')

Accuracy: 0.3333333333333333

Feature Importance:
Age : 0.3745657034743723
Marks_Math : 0.6254342965256278
```

Question 4: SVM on Digits Dataset

Python Code

```python
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load dataset
digits = load_digits()
X = digits.data
y = digits.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train SVM
svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)

# Predict
y_pred = svm_model.predict(X_test)

print("Digits Dataset Accuracy:", accuracy_score(y_test, y_pred))
```

Output

```
Digits Dataset Accuracy: 0.987037037037037
```

Question 5: Random Forest vs SVM (Wine Dataset)

Python Code

```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load dataset
wine = load_wine()
X = wine.data
y = wine.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Random Forest
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# SVM
svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)

print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print("SVM Accuracy:", accuracy_score(y_test, svm_pred))
```

Output

```
Random Forest Accuracy: 1.0
SVM Accuracy: 0.7592592592592593
```

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No. 5

Implementation of Artificial Neural Network

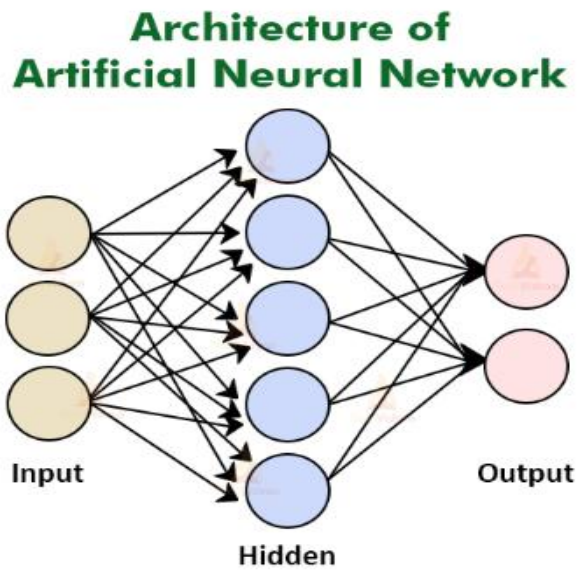
In this lab, students will learn the fundamentals and implementation of an Artificial Neural Network (ANN) for classification and prediction tasks. The lab introduces how neural networks are inspired by the human brain and how they learn patterns from data using layers of interconnected neurons. Students will first apply ANN on a small dataset to understand its working, and then implement ANN on a real-world dataset using Python libraries such as TensorFlow / Keras and Scikit-learn. Model performance will be evaluated using accuracy metrics.

Introduction

An **Artificial Neural Network (ANN)** is a supervised machine learning model composed of interconnected processing units called **neurons**. ANNs consist of:

- **Input layer** – receives input features
- **Hidden layer(s)** – performs computations using weights and activation functions
- **Output layer** – produces final prediction

Each neuron computes a weighted sum of inputs and applies an **activation function** (such as ReLU or Sigmoid). The network learns by adjusting weights using **backpropagation** to minimize error.



Key Concepts:

- **Weights & Bias** – control neuron behavior
- **Activation Functions** – introduce non-linearity
- **Loss Function** – measures prediction error
- **Optimizer** – updates weights (e.g., Adam)

ANNs are widely used in applications such as image recognition, speech processing, medical diagnosis, and pattern recognition.

Solved Examples:

Example 1:

Build a simple ANN to predict whether a student **passes or fails** based on study hours and attendance.

Solution:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical variables
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])
```

```
# Features and target
X = df[['StudyHours', 'Attendance']].values
y = df['Result'].values

# Build ANN model
model = Sequential()
model.add(Dense(4, activation='relu', input_shape=(2,)))
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Train model
model.fit(X, y, epochs=100, verbose=0)

# Prediction
prediction = model.predict([[1, 1]])
print("Predicted Result (Pass=1, Fail=0):", int(prediction[0][0] > 0.5))
```

Question

Apply ANN to classify Iris flowers into three species.

Solution:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
```

```
# One-hot encode target
y = to_categorical(y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Build ANN
model = Sequential()
model.add(Dense(10, activation='relu', input_shape=(4,)))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))


# Compile and train
model.compile(optimizer='adam',              loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100, verbose=0)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy:", accuracy)
```

Example 3:

ANN for Handwritten Digit Classification (MNIST – Baseline) to classify handwritten digits (0–9).

Solution:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Normalize and reshape
X_train = X_train.reshape(-1, 28*28) / 255.0
X_test = X_test.reshape(-1, 28*28) / 255.0

# One-hot encode labels
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build ANN
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(784,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile and train
model.compile(optimizer='adam',              loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1)

# Evaluate model
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", accuracy)
```

Comparison Summary

Aspect	ANN
Handles Non-linearity	Yes
Suitable for Images	Limited
Training Time	Moderate
Interpretability	Low

Topic: Implementation of Artificial Neural Network

Question 1

Logic Gates with Neural Network. Implement a feed-forward neural network to learn the AND gate.

- Inputs: (0,0), (0,1), (1,0), (1,1)
- Output: 0, 0, 0, 1

Tasks:

1. Create dataset using NumPy or pandas.
2. Build a neural network with one hidden layer using TensorFlow/Keras or PyTorch.
3. Train it and show accuracy.
4. Compare model predictions with actual outputs

Question 2

Create a dataset $y = x^2 + \text{noise}$ for x in range [-3,3]. Regression Task with Neural Network

Tasks:

1. Generate 100 samples.
2. Build a neural network to predict y from x.
3. Plot actual vs. predicted results.
4. Discuss how increasing hidden neurons changes results.

Question 3:

Use the XOR gate and train networks with different activation functions (sigmoid, tanh, ReLU).

- Compare accuracy, loss, and convergence speed.
- Plot and discuss results.

Question 4

Binary Classification using Neural Network

Objective: Build a neural network to classify whether a tumor is malignant or benign using the **Breast Cancer dataset**.

Question 5

Multi-Class Classification on Iris Dataset

Objective: Train a neural network to classify flower species (Setosa, Versicolor, Virginica).

Question 6

Regression Problem (House Price Prediction)

Objective: Predict house prices using the **California Housing dataset**.

Question 7

Neural Network with Dropout Regularization

Objective: Prevent overfitting using **Dropout** layers on the **MNIST digit dataset**.

Question 1: Logic Gates with Neural Network

Python Code

```python
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define the dataset for the AND gate
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 0, 0, 1])

# Build the neural network model
model = Sequential()
model.add(Dense(4, input_dim=2, activation='relu'))  # 4 neurons in hidden layer
model.add(Dense(1, activation='sigmoid'))  # Single output neuron with sigmoid

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=1000, verbose=0)

# Predict the results
predictions = model.predict(X)

# Output predictions
print("Predictions:")
for i in range(len(predictions)):
    print(f"Input: {X[i]}, Predicted Output: {round(predictions[i][0])}, Actual Output: {y[i]}")
```

OUTPUT

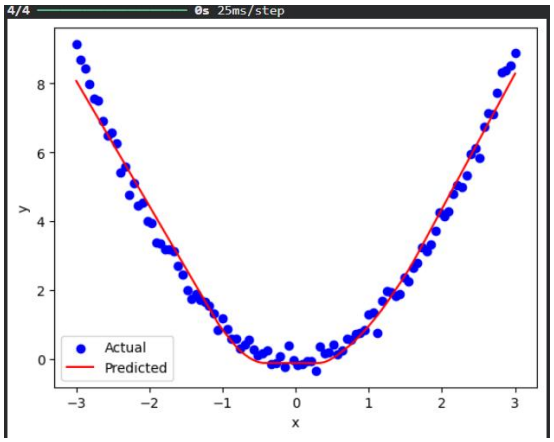
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step
Predictions:
Input: [0 0], Predicted Output: 0, Actual Output: 0
Input: [0 1], Predicted Output: 0, Actual Output: 0
Input: [1 0], Predicted Output: 0, Actual Output: 0
Input: [1 1], Predicted Output: 1, Actual Output: 1
```

Question 2: Regression Task with Neural Network ($y = x^2 +$ noise)

Python Code

```python
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Generate dataset
x = np.linspace(-3, 3, 100)
y = x**2 + np.random.normal(0, 0.2, 100)  # Adding noise to y

# Build the neural network model
model = Sequential()
model.add(Dense(64, input_dim=1, activation='relu'))  # Hidden layer with 64 neurons
model.add(Dense(1))  # Output layer for regression

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(x, y, epochs=500, verbose=0)

# Predict the values
y_pred = model.predict(x)

# Plot the actual vs predicted values
plt.scatter(x, y, color='blue', label='Actual')
plt.plot(x, y_pred, color='red', label='Predicted')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

OUTPUT



Question 3: XOR Gate with Neural Network

Python Code

```python
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define XOR dataset
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 1, 1, 0])

# Build the neural network model with Sigmoid activation
model = Sequential()
model.add(Dense(4, input_dim=2, activation='sigmoid'))  # Sigmoid hidden layer
model.add(Dense(1, activation='sigmoid'))  # Sigmoid output layer

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=1000, verbose=0)

# Predict the results
predictions = model.predict(X)

# Output predictions
print("Predictions with Sigmoid Activation:")
for i in range(len(predictions)):
    print(f"Input: {X[i]}, Predicted Output: {round(predictions[i][0])}, Actual Output: {y[i]}")
```

OUTPUT

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
Predictions with Sigmoid Activation:
Input: [0 0], Predicted Output: 1, Actual Output: 0
Input: [0 1], Predicted Output: 0, Actual Output: 1
Input: [1 0], Predicted Output: 1, Actual Output: 1
Input: [1 1], Predicted Output: 0, Actual Output: 0
```

Question 4: Multi-Class Classification on Iris Dataset

Python Code

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encode target variable
y = to_categorical(y)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Build the neural network model
model = Sequential()
model.add(Dense(10, input_dim=4, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))  # 3 output neurons for 3 classes

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

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=16)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", accuracy)
```

OUTPUT

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9264 - loss: 0.5479
Epoch 90/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9190 - loss: 0.5142
Epoch 91/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8646 - loss: 0.4875
Epoch 92/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9018 - loss: 0.5153
Epoch 93/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9354 - loss: 0.5029
Epoch 94/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9127 - loss: 0.4821
Epoch 95/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8993 - loss: 0.4807
Epoch 96/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9619 - loss: 0.4620
Epoch 97/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9523 - loss: 0.4463
Epoch 98/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9208 - loss: 0.4630
Epoch 99/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9047 - loss: 0.4861
Epoch 100/100
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9299 - loss: 0.4324
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.8843 - loss: 0.3965
Test Accuracy: 0.8888888955116272
```

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

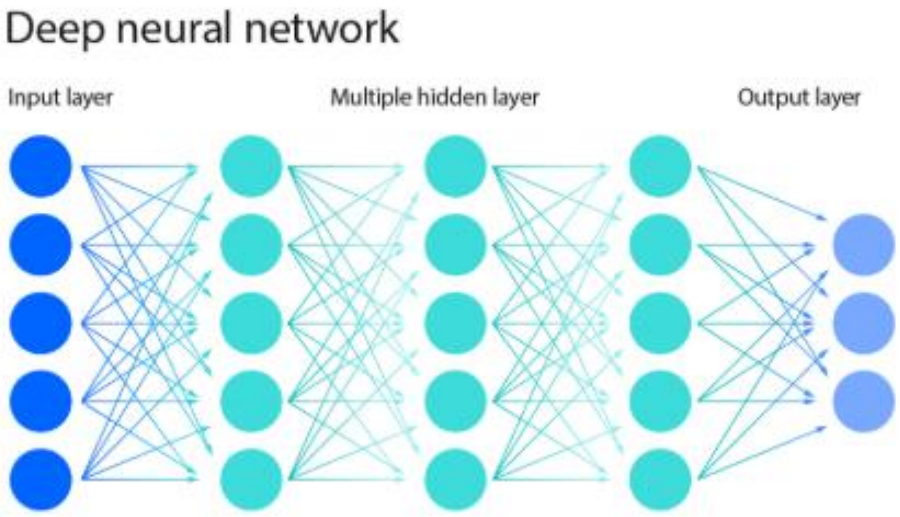
LAB No. 6

Implementation of Deep Neural Network

In this lab, students will study and implement a **Deep Neural Network (DNN)** for classification tasks. A DNN is an extension of Artificial Neural Networks that contains **multiple hidden layers**, enabling the model to learn complex and hierarchical patterns from data. Students will begin with a simple DNN on a small dataset to understand its structure and training process, and then apply DNN models to real-world datasets such as **Iris** and **MNIST**. Model performance will be evaluated using accuracy metrics.

Introduction

A **Deep Neural Network (DNN)** is a neural network with **more than one hidden layer** between the input and output layers. Each layer extracts increasingly complex features from the data. DNNs use **backpropagation** and **gradient descent–based optimizers** to update weights and minimize loss.



Key Components of DNN:

- **Input Layer** – receives raw data
- **Multiple Hidden Layers** – perform deep feature learning
- **Output Layer** – produces final prediction
- **Activation Functions** – ReLU, Sigmoid, Softmax
- **Loss Function** – measures prediction error

- **Optimizer** – Adam, SGD

DNNs are widely used in applications such as image recognition, speech processing, recommendation systems, and natural language processing.

Solved Examples

Example 1

Build a Deep Neural Network to predict whether a student **passes or fails** based on study hours and attendance **(Small Dataset)**

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical data
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

X = df[['StudyHours', 'Attendance']].values
y = df['Result'].values
```

```
# Build DNN
model = Sequential()
model.add(Dense(8, activation='relu', input_shape=(2,)))
model.add(Dense(6, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X, y, epochs=150, verbose=0)

# Prediction
prediction = model.predict([[1, 1]])
print("Predicted Result (Pass=1, Fail=0):", int(prediction[0][0] > 0.5))
```

The multiple hidden layers enable the DNN to learn deeper patterns compared to a shallow ANN.

Example 2:

Apply a Deep Neural Network to classify Iris flowers into three species.

Solution:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encoding
```

```
y = to_categorical(y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Build DNN
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(4,)))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))

# Compile and train
model.compile(optimizer='adam',              loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=150, verbose=0)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy:", accuracy)
```

The deeper architecture improves feature representation and classification accuracy.

Example 3:

Use a Deep Neural Network to classify handwritten digits (0–9).

Solution:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Preprocessing
X_train = X_train.reshape(-1, 28*28) / 255.0
X_test = X_test.reshape(-1, 28*28) / 255.0

# One-hot encode labels
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build DNN
model = Sequential()
model.add(Dense(256, activation='relu', input_shape=(784,)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile and train
model.compile(optimizer='adam',          loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", accuracy)
```

DNN learns hierarchical pixel features and achieves good accuracy, though CNNs are more suitable for image data.

Comparison: ANN vs DNN

Feature	ANN	DNN
Hidden Layers	1	Multiple
Learning Capacity	Moderate	High
Training Time	Lower	Higher
Use Cases	Simple problems	Complex problems

LAB Assignment No 6

Practice Question 1:

DNN Architecture Design

Create a Deep Neural Network to predict whether a student **passes or fails** using features such as *study hours* and *attendance*.

- Use **at least three hidden layers**
- Experiment with different numbers of neurons
- Compare the accuracy with a shallow ANN (one hidden layer)

Practice Question 2:

Activation Function Analysis

Using the **Iris dataset**, build two DNN models:

- Model A: Use **ReLU** activation in all hidden layers
- Model B: Use **tanh** activation in all hidden layers

Train both models and **compare their accuracy and training behavior**. Write your observation.

Practice Question 3:

Hyperparameter Tuning in DNN

Train a DNN on the **MNIST dataset** by changing:

- Number of hidden layers
- Number of neurons per layer
- Batch size

Record how these changes affect **training time and accuracy**

Practice Question 4:

Overfitting and Regularization

Build a deep neural network on any classification dataset and:

- Observe signs of **overfitting**
- Apply at least one regularization technique (Dropout or Early Stopping)
- Compare model performance **before and after regularization**

Question 1: DNN Architecture Design

Python Code

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np  # Import numpy for array conversion

# Dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical data
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

X = df[['StudyHours', 'Attendance']].values
y = df['Result'].values

# Build DNN with 3 hidden layers
model = Sequential()
model.add(Dense(8, activation='relu', input_shape=(2,)))
model.add(Dense(6, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=150, verbose=0)

# Prediction
prediction = model.predict(np.array([[1, 1]]))  # Convert list to NumPy array
print("Predicted Result (Pass=1, Fail=0):", int(prediction[0][0] > 0.5))
```

OUTPUT

```
Predicted Result (Pass=1, Fail=0): 0
```

Question 2: Activation Function Analysis

Python Code for Model A (ReLU)

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encoding
y = to_categorical(y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Build DNN (ReLU activation)
model_a = Sequential()
model_a.add(Dense(16, activation='relu', input_shape=(4,)))
model_a.add(Dense(12, activation='relu'))
model_a.add(Dense(8, activation='relu'))
model_a.add(Dense(3, activation='softmax'))

# Compile and train
model_a.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_a.fit(X_train, y_train, epochs=150, verbose=0)

# Evaluate Model A
loss_a, accuracy_a = model_a.evaluate(X_test, y_test, verbose=0)
print("Model A (ReLU) Accuracy:", accuracy_a)
```

OUTPUT

```
Model A (ReLU) Accuracy: 0.9777777791023254
```

Python Code for Model B (tanh)

```python
# Build DNN (tanh activation)
model_b = Sequential()
model_b.add(Dense(16, activation='tanh', input_shape=(4,)))
model_b.add(Dense(12, activation='tanh'))
model_b.add(Dense(8, activation='tanh'))
model_b.add(Dense(3, activation='softmax'))

# Compile and train
model_b.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_b.fit(X_train, y_train, epochs=150, verbose=0)

# Evaluate Model B
loss_b, accuracy_b = model_b.evaluate(X_test, y_test, verbose=0)
print("Model B (tanh) Accuracy:", accuracy_b)
```

OUTPUT

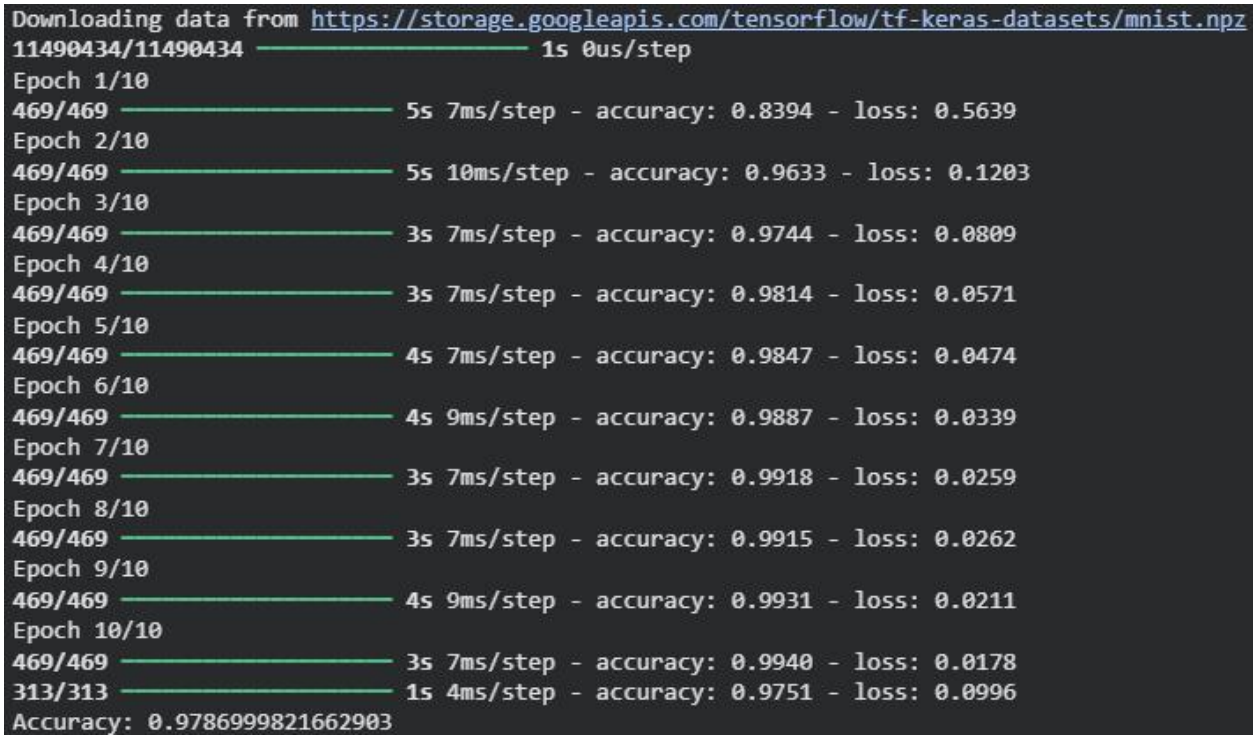
```
Model B (tanh) Accuracy: 1.0
```

Question 3: Hyperparameter Tuning in DNN

Python Code

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Preprocessing
X_train = X_train.reshape(-1, 28*28) / 255.0
X_test = X_test.reshape(-1, 28*28) / 255.0

# One-hot encode labels
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build DNN model with 3 hidden layers and 128 neurons per layer
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(784,)))
model.add(Dense(128, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile and train with batch size 128
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1)

# Evaluate model
loss, accuracy = model.evaluate(X_test, y_test)
print("Accuracy:", accuracy)
```

OUTPUT



Question 4: Overfitting and Regularization

Python Code with Early Stopping (Regularization)

```python
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encode target variable
y = to_categorical(y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Build DNN model
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(4,)))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))

# Early Stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

# Compile and train with EarlyStopping callback
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=150, batch_size=16, validation_data=(X_test, y_test), callbacks=[early_stopping])

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy with Early Stopping:", accuracy)
```

OUTPUT

```
Epoch 116/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9687 - loss: 0.1219 - val_accuracy: 1.0000 - val_loss: 0.0630
Epoch 117/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9826 - loss: 0.0980 - val_accuracy: 1.0000 - val_loss: 0.0525
Epoch 118/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9839 - loss: 0.1038 - val_accuracy: 1.0000 - val_loss: 0.0515
Epoch 119/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9580 - loss: 0.0820 - val_accuracy: 1.0000 - val_loss: 0.0702
Epoch 120/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9515 - loss: 0.0847 - val_accuracy: 1.0000 - val_loss: 0.0479
Epoch 121/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9767 - loss: 0.0699 - val_accuracy: 1.0000 - val_loss: 0.0597
Epoch 122/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9400 - loss: 0.1540 - val_accuracy: 0.9556 - val_loss: 0.0773
Epoch 123/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9656 - loss: 0.0978 - val_accuracy: 1.0000 - val_loss: 0.0458
Epoch 124/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.9611 - loss: 0.0957 - val_accuracy: 1.0000 - val_loss: 0.0479
Epoch 125/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9673 - loss: 0.1125 - val_accuracy: 1.0000 - val_loss: 0.0620
Epoch 126/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9617 - loss: 0.0842 - val_accuracy: 1.0000 - val_loss: 0.0501
Epoch 127/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9738 - loss: 0.0946 - val_accuracy: 1.0000 - val_loss: 0.0593
Epoch 128/150
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9754 - loss: 0.0833 - val_accuracy: 1.0000 - val_loss: 0.0516
Test Accuracy with Early Stopping: 1.0
```

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

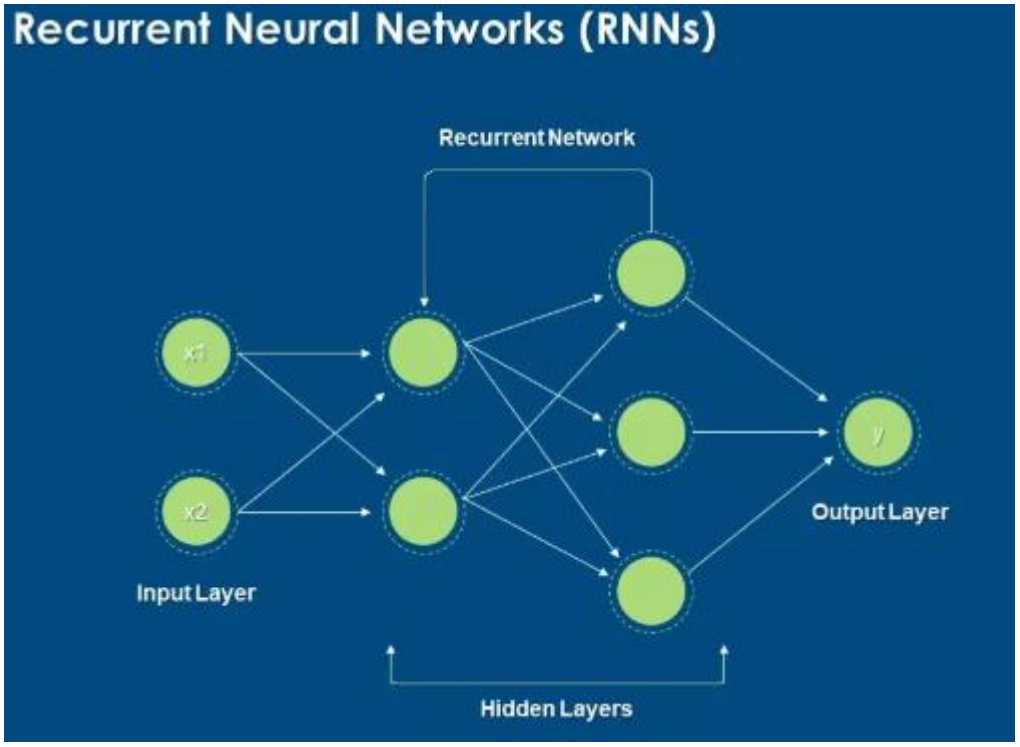
LAB No. 7

Implementation of Recurrent Neural Network (RNN)

In this lab, students will study and implement a Recurrent Neural Network (RNN), a deep learning model designed for sequential and time-dependent data. Unlike feedforward neural networks, RNNs have feedback connections that allow them to retain information from previous time steps. Students will begin with a simple RNN model to understand sequence processing and then apply RNNs to real-world problems such as sequence classification and text processing. Model performance will be evaluated using accuracy metrics.

Introduction

A **Recurrent Neural Network (RNN)** is a class of neural networks specifically designed to process **sequential data**, where the order of inputs matters. RNNs maintain a **hidden state (memory)** that captures information from previous inputs and influences current predictions.



Key Concepts:

- **Sequential Input** – time series or text data
- **Hidden State** – stores past information
- **Recurrent Connection** – connects previous output to current input
- **Activation Functions** – tanh, ReLU
- **Backpropagation Through Time (BPTT)** – training method for RNNs

RNNs are commonly used in **speech recognition, sentiment analysis, time-series forecasting, and natural language processing**. However, basic RNNs may suffer from the **vanishing gradient problem**, which is addressed by advanced variants like **LSTM and GRU**.

Solved Examples:

Example 1:

Simple RNN for Binary Sequence Classification

Build a simple RNN to classify whether a sequence indicates **Pass (1)** or **Fail (0)** based on study performance over time.

Solution:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense

# Sample sequential data (5 students, 3 time steps, 1 feature)
X = np.array([
    [[1], [1], [0]],
    [[1], [1], [1]],
    [[0], [0], [1]],
```

```
    [[0], [0], [0]],
    [[1], [0], [1]]
])

y = np.array([1, 1, 0, 0, 1])

# Build RNN model
model = Sequential()
model.add(SimpleRNN(8, activation='tanh', input_shape=(3, 1)))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X, y, epochs=100, verbose=0)

# Prediction
prediction = model.predict([[[1], [1], [1]]])
print("Predicted Result (1=Pass, 0=Fail):",
int(prediction[0][0] > 0.5))
```

Explanation

The RNN processes input sequences and uses memory to capture temporal patterns before making a classification.

Example 2:

RNN for Time Series Prediction Predict the next value in a simple numerical sequence using an RNN.

Solution:

```
# Generate sequence data
X = np.array([
    [[1], [2], [3]],
    [[2], [3], [4]],
    [[3], [4], [5]],
    [[4], [5], [6]]
])

y = np.array([4, 5, 6, 7])
# Build RNN model
model = Sequential()
model.add(SimpleRNN(10,  activation='tanh',  input_shape=(3,
1)))
model.add(Dense(1))

# Compile and train
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200, verbose=0)

# Predict next value
prediction = model.predict([[[5], [6], [7]]])
print("Predicted Next Value:", prediction[0][0])
```

Explanation

The RNN learns temporal relationships in numeric sequences and predicts future values.

Example 3:

RNN for Text Classification (Simple Sentiment Analysis)

Build a simple RNN model to classify text sentiment as **positive or negative**.

Solution

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Sample text data
texts = [
    "I love this course",
    "This lab is very good",
    "I hate this subject",
    "This is boring",
    "Excellent explanation"
]

labels = [1, 1, 0, 0, 1]

# Tokenize text
tokenizer = Tokenizer(num_words=100)
tokenizer.fit_on_texts(texts)

sequences = tokenizer.texts_to_sequences(texts)
padded_sequences = pad_sequences(sequences, maxlen=5)

# Build RNN model
model = Sequential()
model.add(SimpleRNN(16, activation='tanh', input_shape=(5,)))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy',
```

```
metrics=['accuracy'])
model.fit(padded_sequences, labels, epochs=100, verbose=0)

# Prediction
prediction = model.predict(padded_sequences)
print("Predicted Sentiments:", prediction.round())
```

The RNN processes text sequences word by word, capturing contextual meaning for sentiment classification.

Limitations of Basic RNN

- Vanishing gradient problem
- Difficulty learning long-term dependencies
- Slower training compared to feedforward networks

LAB Assignment No 6

LAB Task 1:

Next Word Prediction using RNN

Objective: Learn how RNNs can predict the next word in a sentence.
Dataset: Any small text corpus — e.g., *Shakespeare.txt* or *Wikipedia sample.*
Tasks:

1. Load and clean the text data.
2. Tokenize and convert text into sequences.

3. Build a simple **RNN model** using keras.layers.SimpleRNN.
4. Train it to predict the next word given previous 3–5 words.
5. Test by entering a custom text prompt and predict the next word.

Output:

Model predicts probable next word, e.g.,
Input: "The sun is" → **Output:** "shining"

Required Solution

- Understanding of dataset
- Understanding of code
- Learning outcomes analysis

LAB Task 2:

Stock Price Prediction using RNN

Objective: Predict future stock prices using time series data.
Dataset: Use *Google Stock Price* dataset (from Kaggle or Yahoo Finance).
Tasks:

1. Import dataset and normalize values.
2. Prepare time-step sequences (e.g., 60 previous days → next day price).
3. Build and train an **RNN model** using SimpleRNN layers.
4. Evaluate predictions vs actual prices (plot graph).

Output:

Line graph showing predicted vs real stock price trend.

Required Solution

- Understanding of dataset

- Understanding of code
- Learning outcomes analysis

LAB Task 3:

Sentiment Analysis using RNN

Objective: Classify movie reviews as positive or negative using RNN.
Dataset: *IMDb Movie Reviews* dataset (available in Keras).
Tasks:

1. Load dataset and preprocess text (tokenize and pad sequences).
2. Build RNN with Embedding + SimpleRNN layers.
3. Train for binary classification (positive/negative).
4. Evaluate accuracy on test data.

Output:

Accuracy score (e.g., 85%) and prediction for custom input text.

Required Solution

- Understanding of dataset
- Understanding of code
- Learning outcomes analysis

LAB Task 4:

Weather Forecasting using RNN

Objective: Predict future temperature based on previous days' readings.

Dataset: *Daily temperature dataset* (e.g., "Jena Climate Dataset" from TensorFlow).

Tasks:

1. Load and visualize temperature over time.
2. Prepare input-output sequences for time series prediction.
3. Build an RNN to predict next day's temperature.
4. Plot actual vs predicted temperature.

Output:

Graph showing predicted vs actual temperature trends.

Required Solution

- Understanding of dataset
- Understanding of code
- Learning outcomes analysis

LAB Task 5:

Music Note Generation using RNN

Objective: Generate new music sequences using RNN.

Dataset: *MIDI music dataset* (short sequences or melodies).

Tasks:

1. Convert MIDI data into integer-encoded notes.
2. Train an RNN on note sequences (input: previous notes → output: next note).
3. Generate a new sequence using the trained model.

Output:

A sequence of generated notes that can be converted to a playable MIDI file

Required Solution

- Understanding of dataset
- Understanding of code
- Learning outcomes analysis

LAB Task 1: Next Word Prediction using RNN

Code Example:

```python
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Sample text data (Shakespeare or any other small corpus)
text = """Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date."""

# Tokenizing text
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])
sequences = tokenizer.texts_to_sequences([text])

# Prepare the data for RNN
sequence_length = 5
X = []
y = []

for i in range(len(sequences[0]) - sequence_length):
    X.append(sequences[0][i:i+sequence_length])
    y.append(sequences[0][i+sequence_length])

X = np.array(X)
y = np.array(y)

# Build the RNN model
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=50, input_length=sequence_length))
model.add(SimpleRNN(128, activation='relu'))
model.add(Dense(len(tokenizer.word_index)+1, activation='softmax'))

# Compile and train the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy')
model.fit(X, y, epochs=50, verbose=1)

# Predict the next word after a given input
input_sequence = "Shall I compare thee".split()
input_sequence = tokenizer.texts_to_sequences([input_sequence])[0]
input_sequence = pad_sequences([input_sequence], maxlen=sequence_length)

prediction = model.predict(input_sequence)
predicted_word = tokenizer.index_word[np.argmax(prediction)]
print("Predicted next word:", predicted_word)
```

OUTPUT

```
Epoch 49/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 0.0275
Epoch 50/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - loss: 0.0208
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step
Predicted next word: short
```

LAB Task 2: Stock Price Prediction using RNN

Code Example:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense

# Load Google stock price dataset (from Yahoo Finance or Kaggle)
data = pd.read_csv('GOOG_stock_data.csv')
prices = data['Close'].values.reshape(-1, 1)

# Normalize the stock prices
scaler = MinMaxScaler(feature_range=(0, 1))
prices_scaled = scaler.fit_transform(prices)

# Prepare time-step sequences
X = []
y = []
sequence_length = 60

for i in range(sequence_length, len(prices_scaled)):
    X.append(prices_scaled[i-sequence_length:i, 0])
    y.append(prices_scaled[i, 0])

X = np.array(X)
y = np.array(y)

# Reshape X for RNN
X = X.reshape(X.shape[0], X.shape[1], 1)

# Build the RNN model
model = Sequential()
model.add(SimpleRNN(50, activation='relu', input_shape=(X.shape[1], 1)))
model.add(Dense(1))

# Compile and train the model
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X, y, epochs=10, batch_size=32)

# Make predictions
predictions = model.predict(X)

# Invert the scaling to get actual stock prices
predictions = scaler.inverse_transform(predictions)
y_actual = scaler.inverse_transform(y.reshape(-1, 1))

# Plot the results
plt.plot(y_actual, label='Actual Prices')
plt.plot(predictions, label='Predicted Prices')
plt.legend()
plt.show()
```

LAB Task 3: Sentiment Analysis using RNN

Code Example:

```python
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense

# Load IMDb dataset
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)

# Pad sequences to ensure uniform input size
X_train = pad_sequences(X_train, maxlen=500)
X_test = pad_sequences(X_test, maxlen=500)

# Build the RNN model
model = Sequential()
model.add(Embedding(10000, 128))
model.add(SimpleRNN(128))
model.add(Dense(1, activation='sigmoid'))

# Compile and train the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))

# Evaluate accuracy
accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy[1]*100:.2f}%")
```

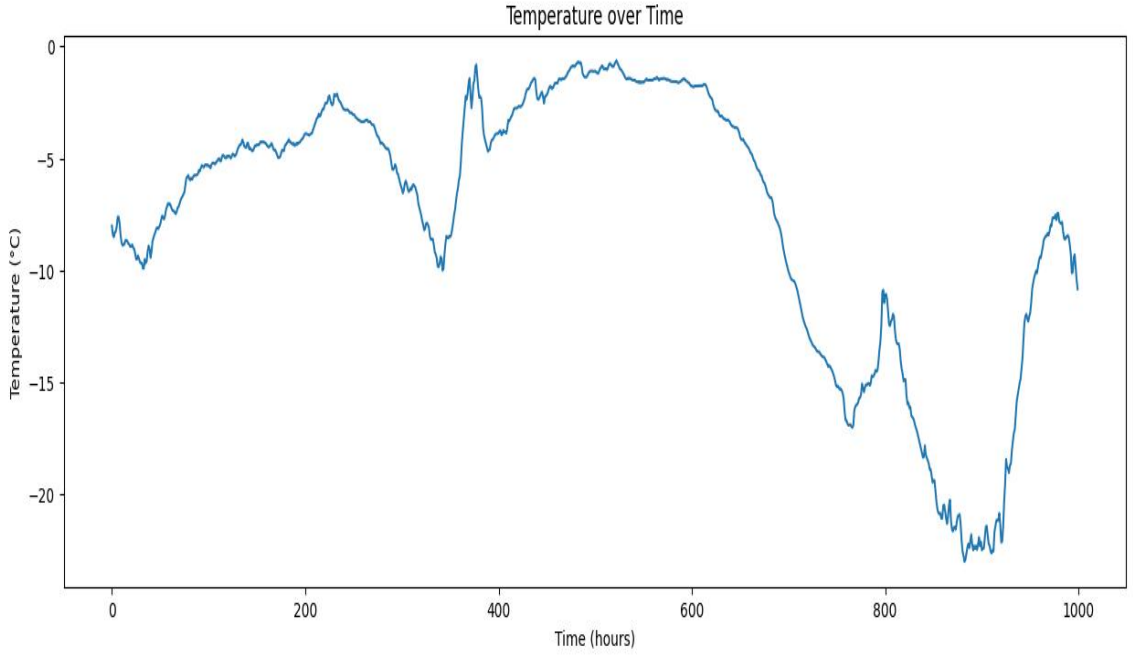
LAB Task 4:

```python
# Import libraries

import tensorflow as tf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt



# Load dataset from URL

url = "https://storage.googleapis.com/tensorflow/tf-keras-
datasets/jena_climate_2009_2016.csv.zip"

df = pd.read_csv(url, compression='zip')
```

```python
# Display first few rows

df.head()
```

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	3.33	3.11	0.22	1.94	3.12	1307.75	1.03	1.75	152.3
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21	1.89	3.03	1309.80	0.72	1.50	136.1
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.20	1.88	3.02	1310.24	0.19	0.63	171.6
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19	1.92	3.08	1309.19	0.34	0.50	198.0
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19	1.92	3.09	1309.00	0.32	0.63	214.3

```
# Extract temperature column
temperature = df['T (degC)'].values


# Plot temperature over time
plt.figure(figsize=(15,5))
plt.plot(temperature[:1000])  # plot first 1000 readings for clarity
plt.title("Temperature over Time")
plt.xlabel("Time (hours)")
plt.ylabel("Temperature (°C)")
plt.show()
```




```
temp = temperature.reshape(-1,1)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
temp_scaled = scaler.fit_transform(temp)


# Prepare sequences
def create_sequences(data, seq_length):
    X = []
    y = []
    for i in range(len(data)-seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)
```

```
seq_length = 24  # use past 24 hours to predict next hour
X, y = create_sequences(temp_scaled, seq_length)
```

```
# Split into training and testing sets
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

```
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
```

```
X_train shape: (336421, 24, 1)
y_train shape: (336421, 1)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense


# Build the model
model = Sequential([
    SimpleRNN(50, activation='relu', input_shape=(seq_length,1)),
    Dense(1)  # output temperature
])
```

```
# Compile model
model.compile(optimizer='adam', loss='mse')
```

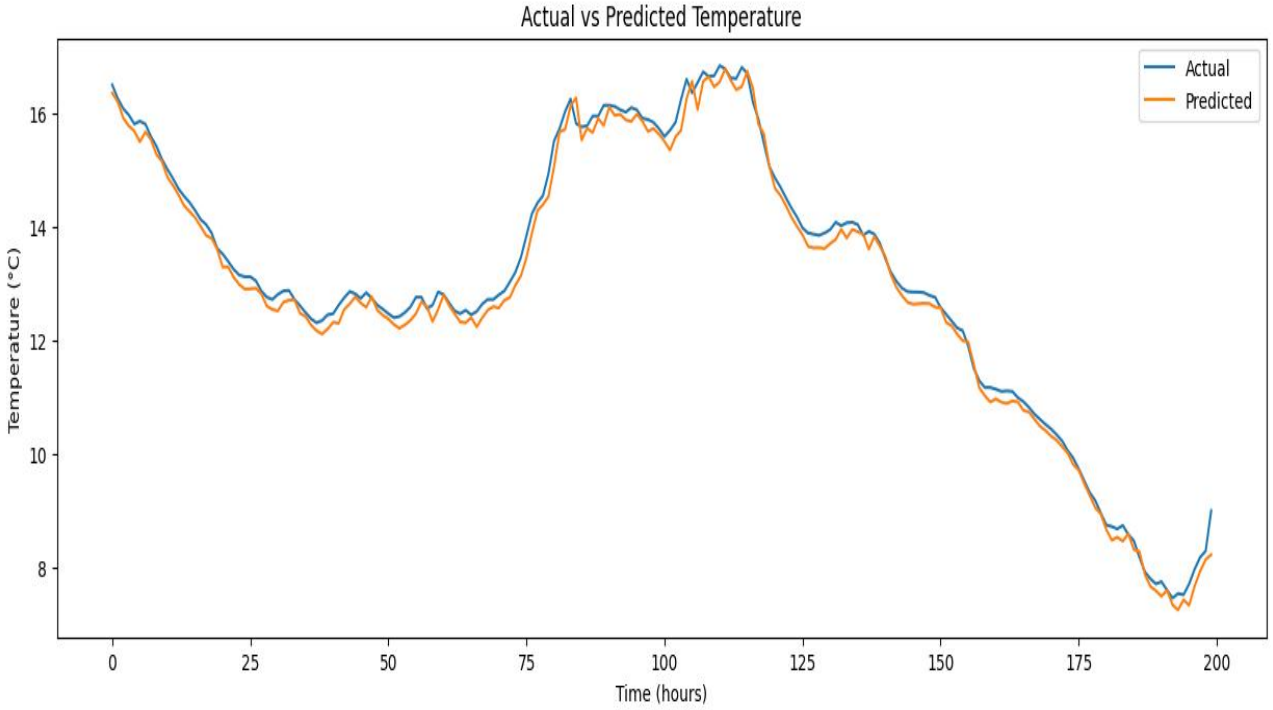
```
# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential mode
  super().__init__(**kwargs)
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 53s 6ms/step - loss: 0.0029 - val_loss: 3.1749e-05
Epoch 2/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 57s 7ms/step - loss: 1.8936e-05 - val_loss: 1.2212e-05
Epoch 3/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 53s 6ms/step - loss: 1.4708e-05 - val_loss: 1.7088e-05
Epoch 4/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 82s 6ms/step - loss: 1.3336e-05 - val_loss: 1.8011e-05
Epoch 5/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 53s 6ms/step - loss: 1.2762e-05 - val_loss: 2.2755e-05
Epoch 6/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 54s 6ms/step - loss: 1.2686e-05 - val_loss: 1.2386e-05
Epoch 7/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 53s 6ms/step - loss: 1.2546e-05 - val_loss: 1.6868e-05
Epoch 8/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 53s 6ms/step - loss: 1.2685e-05 - val_loss: 1.2786e-05
Epoch 9/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 82s 6ms/step - loss: 1.2566e-05 - val_loss: 1.1999e-05
Epoch 10/10
8411/8411 ━━━━━━━━━━━━━━━━━━━━ 82s 6ms/step - loss: 1.2423e-05 - val_loss: 1.4355e-05
```

```
# Make predictions
y_pred = model.predict(X_test)


# Reverse normalization
y_test_inv = scaler.inverse_transform(y_test)
y_pred_inv = scaler.inverse_transform(y_pred)
```

```
# Plot actual vs predicted
plt.figure(figsize=(15,5))
plt.plot(y_test_inv[:200], label='Actual')
plt.plot(y_pred_inv[:200], label='Predicted')
```

```
plt.title("Actual vs Predicted Temperature")
plt.xlabel("Time (hours)")
plt.ylabel("Temperature (°C)")
plt.legend()
plt.show()
```



Lab task 5:

```
# -----------------------
# Step 1: Create a small MIDI sequence in code
# -----------------------
# We'll define a tiny melody as note names
notes = ['C4', 'D4', 'E4', 'F4', 'G4', 'A4', 'B4', 'C5', 'D5', 'E5',
         'F5', 'G5', 'A5', 'B5', 'C6', 'D6', 'E6', 'F6', 'G6', 'A6']
```

```
print("Notes for training:", notes)
```

```
# -----------------------
# Step 2: Map notes to integers
# -----------------------
unique_notes = sorted(set(notes))
```

```
note_to_int = {n:i for i,n in enumerate(unique_notes)}
int_to_note = {i:n for i,n in enumerate(unique_notes)}
encoded_notes = [note_to_int[n] for n in notes]
```

```
# ----------------------
# Step 3: Create input-output sequences
# ----------------------
seq_length = 5  # small for tiny dataset
X = []
y = []
```

```
for i in range(len(encoded_notes)-seq_length):
    X.append(encoded_notes[i:i+seq_length])
    y.append(encoded_notes[i+seq_length])
```

```
X = np.array(X)
y = np.array(y)
```

```
# Reshape for RNN and normalize
X = np.reshape(X, (X.shape[0], X.shape[1], 1))
X = X / float(len(unique_notes))
```

```
# One-hot encode output
y = to_categorical(y, num_classes=len(unique_notes))
```

```
print("X shape:", X.shape)
print("y shape:", y.shape)
```

```
# ----------------------
# Step 4: Build the RNN
# ----------------------
model = Sequential()
```

```
model.add(LSTM(128, input_shape=(X.shape[1], X.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(128))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(len(unique_notes), activation='softmax'))
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

```
# Train the model (mini training for demo)
model.fit(X, y, epochs=50, batch_size=16)
```

```
# -----------------------
# Step 5: Generate new notes
# -----------------------
start = np.random.randint(0, len(X)-1)
pattern = X[start]
prediction_output = []
```

```
for note_index in range(20):  # generate 20 notes
    prediction_input = np.reshape(pattern, (1, pattern.shape[0], 1))
    prediction_input = prediction_input / float(len(unique_notes))

    prediction = model.predict(prediction_input, verbose=0)
    index = np.argmax(prediction)
    result = int_to_note[index]
    prediction_output.append(result)

    # Shift pattern
    pattern = np.append(pattern[1:], [[index]], axis=0)
```

```
print("Generated notes:", prediction_output)
```

```
# ----------------------
# Step 6: Convert generated notes to MIDI
# ----------------------
output_notes = []
```

```
for pattern in prediction_output:
    output_notes.append(note.Note(pattern))
```

```
midi_stream = stream.Stream(output_notes)
midi_stream.write('midi', fp='generated_music.mid')
print("MIDI file generated: generated_music.mid")
```

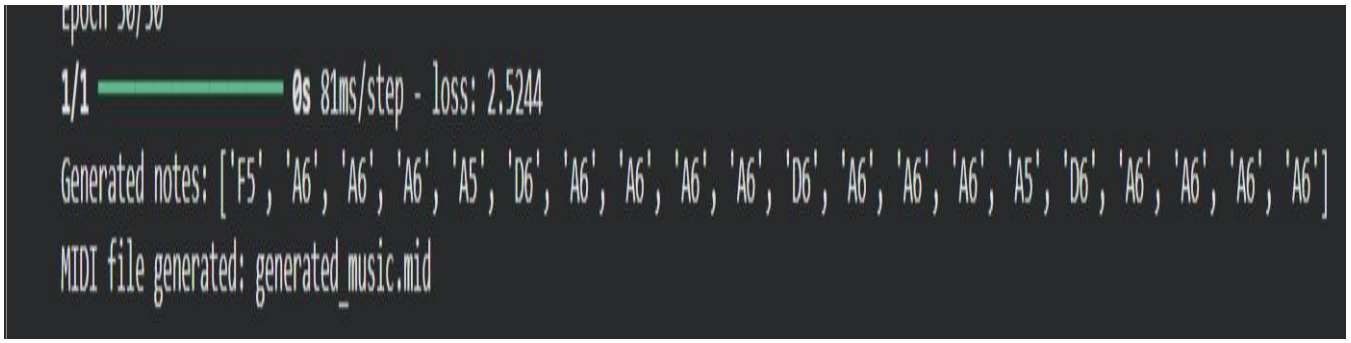
```
# ----------------------
# Step 7: Download MIDI file
# ----------------------
from google.colab import files
files.download('generated_music.mid')
```

Output:

```
Requirement already satisfied: music21 in /usr/local/lib/python3.12/dist-packages (9.9.1)
Requirement already satisfied: chardet in /usr/local/lib/python3.12/dist-packages (from music21) (5.2.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from music21) (1.5.3)
Requirement already satisfied: jsonpickle in /usr/local/lib/python3.12/dist-packages (from music21) (4.1.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from music21) (3.10.0)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.12/dist-packages (from music21) (10.8.0)
Requirement already satisfied: numpy>=1.26.4 in /usr/local/lib/python3.12/dist-packages (from music21) (2.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from music21) (2.32.4)
Requirement already satisfied: webcolors>=1.5 in /usr/local/lib/python3.12/dist-packages (from music21) (25.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib->music21) (2.9.0.post0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->music21) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->music21) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->music21) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->music21) (2025.11.12)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib->music21) (1.17.0)
Notes for training: ['C4', 'D4', 'E4', 'F4', 'G4', 'A4', 'B4', 'C5', 'D5', 'E5', 'F5', 'G5', 'A5', 'B5', 'C6', 'D6', 'E6', 'F6', 'G6', 'A6']
X shape: (15, 5, 1)
y shape: (15, 20)
Epoch 1/50
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential mo
  super().__init__(**kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - loss: 2.9986
```

```
files.download('generated_music.mid')
```

```
  super().__init__(**kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - loss: 2.9986
Epoch 2/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 2.9942
Epoch 3/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - loss: 2.9936
Epoch 4/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 2.9890
Epoch 5/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 2.9881
Epoch 6/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 2.9853
Epoch 7/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 2.9806
Epoch 8/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 2.9755
Epoch 9/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 2.9771
Epoch 10/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 2.9666
Epoch 11/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 2.9711
Epoch 12/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 2.9629
Epoch 13/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 2.9490
Epoch 14/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 2.9472
Epoch 15/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 2.9463
Epoch 16/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - loss: 2.9302
Epoch 17/50
```



LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

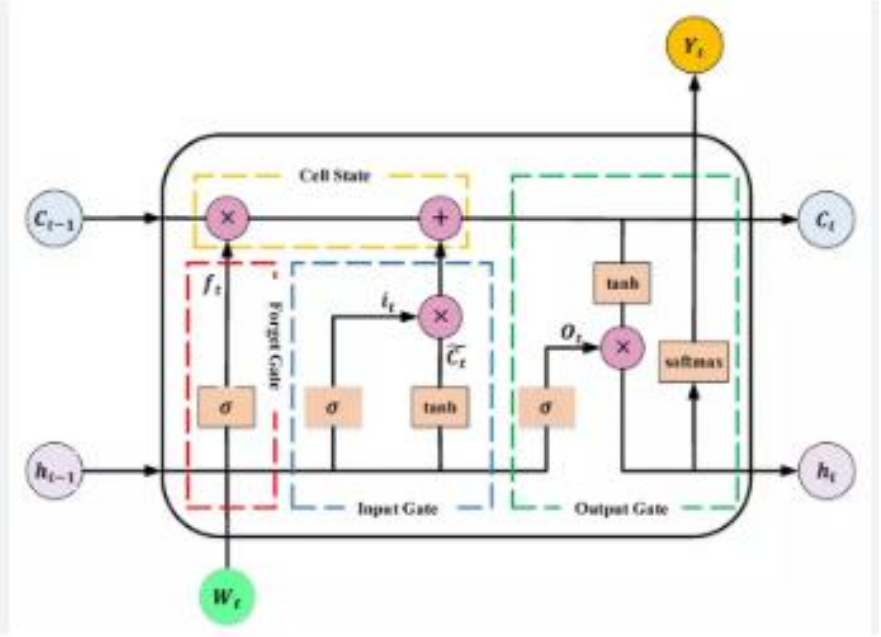
LAB No. 8

Implementation of Long Short-Term Memory (LSTM) Network

In this lab, students will learn and implement the **Long Short-Term Memory (LSTM)** network, an advanced type of **Recurrent Neural Network (RNN)** designed to overcome the limitations of basic RNNs. LSTM networks are capable of learning **long-term dependencies** in sequential data through a special memory cell and gating mechanism. Students will apply LSTM models to sequence classification, time-series prediction, and text-based tasks using Python and Keras, and evaluate model performance using appropriate metrics.

Introduction & Theory (Brief)

Long Short-Term Memory (LSTM) is a special kind of RNN that effectively handles the **vanishing gradient problem**. It introduces a **memory cell** that can store information for long periods and three gates that regulate information flow:



Key Components of LSTM:

- **Forget Gate** – decides what information to discard

- **Input Gate** – decides what new information to store
- **Output Gate** – controls what information to output
- **Cell State** – long-term memory

Advantages:

- Learns long-term dependencies
- Suitable for time-series, speech, and text data
- More stable training than basic RNNs

Applications:

- Time-series forecasting
- Sentiment analysis
- Speech recognition
- Language translation

Solved Examples:

Example 1: LSTM for Binary Sequence Classification

Build an LSTM model to predict whether a student **passes or fails** based on performance over multiple time steps.

Solution:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Sequential dataset (samples, timesteps, features)
X = np.array([
    [[1], [1], [0]],
```

```
    [[1], [1], [1]],
    [[0], [0], [1]],
    [[0], [0], [0]],
    [[1], [0], [1]]
])

y = np.array([1, 1, 0, 0, 1])

# Build LSTM model
model = Sequential()
model.add(LSTM(8, input_shape=(3, 1)))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X, y, epochs=100, verbose=0)



# Prediction
prediction = model.predict([[[1], [1], [1]]])
print("Predicted Result (Pass=1, Fail=0):",
int(prediction[0][0] > 0.5))
```

The LSTM retains relevant information across time steps using its memory cell and gating mechanism.

Example 2: LSTM for Time-Series Prediction

Use an LSTM network to predict the next value in a numerical time-series sequence.

Solution:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Time-series input sequences
X = np.array([
    [[1], [2], [3]],
    [[2], [3], [4]],
    [[3], [4], [5]],
    [[4], [5], [6]]
])

y = np.array([4, 5, 6, 7])

# Build LSTM model
model = Sequential()
model.add(LSTM(10, input_shape=(3, 1)))
model.add(Dense(1))

# Compile and train
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200, verbose=0)

# Predict next value
prediction = model.predict([[[5], [6], [7]]])
print("Predicted Next Value:", prediction[0][0])
```

LSTM captures temporal dependencies more effectively than basic RNNs for time-series prediction.

Example 3: LSTM for Text Classification (Sentiment Analysis)

Build an LSTM model to classify text as **positive** or **negative** sentiment.

Solution:

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Text samples
texts = [
    "I love this subject",
    "This lab is excellent",
    "I hate this course",
    "This topic is boring",
    "Very good explanation"
]

labels = [1, 1, 0, 0, 1]

# Tokenization
tokenizer = Tokenizer(num_words=100)
tokenizer.fit_on_texts(texts)

sequences = tokenizer.texts_to_sequences(texts)
padded_sequences = pad_sequences(sequences, maxlen=5)
```

```
# Build LSTM model
model = Sequential()
model.add(LSTM(16, input_shape=(5,)))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam',       loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(padded_sequences, labels, epochs=100, verbose=0)

# Predict sentiment
predictions = model.predict(padded_sequences)
print("Predicted Sentiments:", predictions.round())
```

LSTM processes text sequences while preserving context, leading to improved sentiment classification.

Comparison: RNN vs LSTM

Feature	RNN	LSTM
Long-Term Memory	No	Yes
Vanishing Gradient	Yes	No
Training Stability	Low	High

LAB Assignment No.8

LSTM code for text predictor

Question:

Using the given LSTM next-word prediction model and tokenizer, write the code to input a sentence, convert it into sequences, pad it, and predict the next word using the trained model.

Solution:

```
faqs = """About the Program
What is the course fee for  Data Science Mentorship Program (DSMP 2023)
The course follows a monthly subscription model where you have to make monthly payments of Rs 799/month.
```

What is the total duration of the course?
The total duration of the course is 7 months. So the total course fee becomes 799*7 = Rs 5600(approx.)
What is the syllabus of the mentorship program?
We will be covering the following modules:
Python Fundamentals
Python libraries for Data Science
Data Analysis
SQL for Data Science
Maths for Machine Learning
ML Algorithms
Practical ML
MLOPs
Case studies

Will Deep Learning and NLP be a part of this program?
No, NLP and Deep Learning both are not a part of this program's curriculum.
What if I miss a live session? Will I get a recording of the session?
Yes all our sessions are recorded, so even if you miss a session you can go back and watch the recording.
Where can I find the class schedule?
Checkout this google sheet to see month by month time table of the course -
What is the time duration of all the live sessions?
Roughly, all the sessions last 2 hours.
What is the language spoken by the instructor during the sessions?
Hinglish
How will I be informed about the upcoming class?
You will get a mail from our side before every paid session once you become a paid user.
Can I do this course if I am from a non-tech background?
Yes, absolutely.
I am late, can I join the program in the middle?
Absolutely, you can join the program anytime.
If I join/pay in the middle, will I be able to see all the past lectures?
Yes, once you make the payment you will be able to see all the past

content in your dashboard.

Where do I have to submit the task?

You don't have to submit the task. We will provide you with the solutions, you have to self evaluate the task yourself.

Will we do case studies in the program?

Yes.

Where can we contact you?

You can mail us at muhammad.hamedoon@tech.uol.edu.pk

Payment/Registration related questions

Where do we have to make our payments? Your YouTube channel or website?

You have to make all your monthly payments on our website.

Unfortunately no, the program follows a monthly subscription model.

What is the validity of monthly subscription? Suppose if I pay on 15th Jan, then do I have to pay again on 1st Feb or 15th Feb

15th Feb. The validity period is 30 days from the day you make the payment. So essentially you can join anytime you don't have to wait for a month to end.

What if I don't like the course after making the payment. What is the refund policy?

You get a 7 days refund period from the day you have made the payment.

I am living outside India and I am not able to make the payment on the website, what should I do?

You have to contact us by sending a mail at muhammad.hamedoon@tech.uol.edu.pk

Post registration queries

Till when can I view the paid videos on the website?

This one is tricky, so read carefully. You can watch the videos till your subscription is valid. Suppose you have purchased subscription on 21st Jan, you will be able to watch all the past paid sessions in the period of 21st Jan to 20th Feb. But after 21st Feb you will have to purchase the subscription again.

But once the course is over and you have paid us Rs 5600(or 7 installments of Rs 799) you will be able to watch the paid sessions till Aug 2024.

Why lifetime validity is not provided?

```
Because of the low course fee.
Where can I reach out in case of a doubt after the session?
You will have to fill a google form provided in your dashboard and our team will contact you for a 1 on 1 doubt clearance session
If I join the program late, can I still ask past week doubts?
Yes, just select past week doubt in the doubt clearance google form.
I am living outside India and I am not able to make the payment on the website, what should I do?
You have to contact us by sending a mail at muhammad.hamedoon@tech.uol.edu.pk
Certificate and Placement Assistance related queries
What is the criteria to get the certificate?
There are 2 criterias:
You have to pay the entire fee of Rs 5600
You have to attempt all the course assessments.
I am joining late. How can I pay payment of the earlier months?
You will get a link to pay fee of earlier months in your dashboard once you pay for the current month.
I have read that Placement assistance is a part of this program. What comes under Placement assistance?
This is to clarify that Placement assistance does not mean Placement guarantee. So we dont guarantee you any jobs or for that matter even interview calls. So if you are planning to join this course just for placements, I am afraid you will be disappointed. Here is what comes under placement assistance
Portfolio Building sessions
Soft skill sessions
Sessions with industry mentors
Discussion on Job hunting strategies
"""
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer

# Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts([faqs])
```

```
input_sequences = []
for sentence in faqs.split('\n'):
    tokenized_sentence = tokenizer.texts_to_sequences([sentence])[0]
    for i in range(1, len(tokenized_sentence)):
        input_sequences.append(tokenized_sentence[:i+1])

max_len = max([len(x) for x in input_sequences])

from tensorflow.keras.preprocessing.sequence import pad_sequences
padded_input_sequences = pad_sequences(input_sequences,
maxlen=max_len, padding='pre')

X = padded_input_sequences[:, :-1]
y = padded_input_sequences[:, -1]

from tensorflow.keras.utils import to_categorical
y = to_categorical(y, num_classes=len(tokenizer.word_index)+1)

# Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

vocab_size = len(tokenizer.word_index) + 1  # must be dynamic

model = Sequential()
model.add(Embedding(vocab_size, 100, input_length=max_len-1))
model.add(LSTM(150, return_sequences=True))
model.add(LSTM(150))
model.add(Dense(vocab_size, activation='softmax'))

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

# IMPORTANT FIX → Build model first
model.build(input_shape=(None, max_len-1))

# Corrected Summary
```

```
model.summary()

# Print actual trainable parameters (works in ALL TensorFlow versions)
print("\nTrainable parameters per layer:\n")
for layer in model.layers:
    print(layer.name, ":", layer.count_params())

print("\nTotal Trainable Parameters =", model.count_params())
```

```
model.fit(X,y,epochs=100)
```

```
import numpy as np
import time
from tensorflow.keras.preprocessing.sequence import pad_sequences

text = "what is the fee"

for i in range(10):
    # tokenize
    token_text = tokenizer.texts_to_sequences([text])[0]

    # padding
    padded_token_text = pad_sequences([token_text], maxlen=56,
padding='pre')

    # predict
    pos = np.argmax(model.predict(padded_token_text, verbose=0))

    for word, index in tokenizer.word_index.items():
        if index == pos:
            text = text + " " + word
            print(text)
            time.sleep(2)
```

Output :

```
what is the fee of
what is the fee of monthly
what is the fee of monthly subscription
what is the fee of monthly subscription suppose
what is the fee of monthly subscription suppose if
what is the fee of monthly subscription suppose if i
what is the fee of monthly subscription suppose if i pay
what is the fee of monthly subscription suppose if i pay on
what is the fee of monthly subscription suppose if i pay on 15th
what is the fee of monthly subscription suppose if i pay on 15th jan
```

Question No. 2

Train an LSTM model on your chosen text dataset and write the code to predict the next word for a user-given input sentence.

Question No. 3

Modify the next-word prediction code so that the model generates 5 new words sequentially instead of just one.

LAB Assignment No.8

LSTM code for text predictor

Question:

the trained model.

Solution:

Code example:

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense

from tensorflow.keras.utils import to_categorical

import time


# ================================

# STEP 2: DATASET

# ================================

faqs = """About the Program

What is the course fee for Data Science Mentorship Program (DSMP 2023)

The course follows a monthly subscription model where you have to make monthly payments of Rs 799/month.

What is the total duration of the course?

The total duration of the course is 7 months.

What is the validity of monthly subscription? Suppose if I pay on 15th Jan"""
```

```
# ================================

# STEP 3: TOKENIZATION

# ================================

tokenizer = Tokenizer()

tokenizer.fit_on_texts([faqs])
```

```
# ================================

# STEP 4: CREATE INPUT SEQUENCES

# ================================

input_sequences = []
```

```
for sentence in faqs.split('\n'):

    tokenized_sentence = tokenizer.texts_to_sequences([sentence])[0]

    for i in range(1, len(tokenized_sentence)):

        input_sequences.append(tokenized_sentence[:i+1])
```

```
# ================================
```

```
# STEP 5: PADDING
# ================================
max_len = max(len(seq) for seq in input_sequences)
```

```
padded_sequences = pad_sequences(
    input_sequences,
    maxlen=max_len,
    padding='pre'
)
```

```
X = padded_sequences[:, :-1]
y = padded_sequences[:, -1]
```

```
# ================================
# STEP 6: ONE-HOT ENCODING
# ================================
vocab_size = len(tokenizer.word_index) + 1
y = to_categorical(y, num_classes=vocab_size)
```

```
# ================================
# STEP 7: LSTM MODEL
# ================================
model = Sequential()
model.add(Embedding(vocab_size, 100, input_length=max_len-1))
model.add(LSTM(150, return_sequences=True))
model.add(LSTM(150))
model.add(Dense(vocab_size, activation='softmax'))
```

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

```
# ==============================
# STEP 8: TRAIN MODEL
# ==============================
model.fit(X, y, epochs=100, verbose=1)
```

```
# ==============================
# STEP 9: NEXT WORD PREDICTION
# ==============================
text = "what is the fee"
```

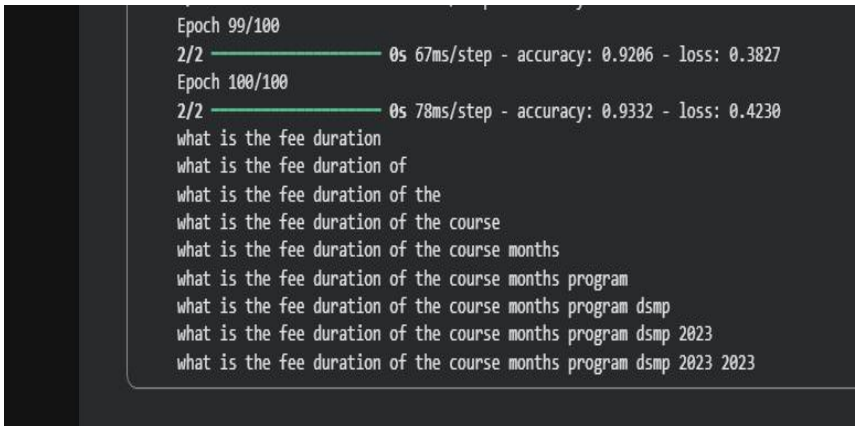
```
for i in range(9):
    token_text = tokenizer.texts_to_sequences([text])[0]
```

```
    padded_token_text = pad_sequences(
        [token_text],
        maxlen=max_len-1,
        padding='pre'
    )
```

```
    predicted_index = np.argmax(
        model.predict(padded_token_text, verbose=0)
    )
```

```
    for word, index in tokenizer.word_index.items():
        if index == predicted_index:
            text = text + " " + word
            print(text)
            time.sleep(1)
            break
```

Output:



Question No. 2

Code example:

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.utils import to_categorical

text_data = """Machine learning is a branch of artificial intelligence
Machine learning allows systems to learn automatically
Deep learning is a subset of machine learning
Artificial intelligence is transforming the world"""
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text_data])
```

```
input_sequences = []
```

```
for line in text_data.split('\n'):
    token_list = tokenizer.texts_to_sequences([line])[0]
    for i in range(1, len(token_list)):
        input_sequences.append(token_list[:i+1])
```

```
max_len = max(len(seq) for seq in input_sequences)
```

```
padded_sequences = pad_sequences(
    input_sequences,
    maxlen=max_len,
    padding='pre'
)
```

```
X = padded_sequences[:, :-1]
y = padded_sequences[:, -1]
```

```
vocab_size = len(tokenizer.word_index) + 1
y = to_categorical(y, num_classes=vocab_size)
```

```
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=max_len-1))
model.add(LSTM(100))
model.add(Dense(vocab_size, activation='softmax'))
```

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

```
model.fit(X, y, epochs=100, verbose=1)
```

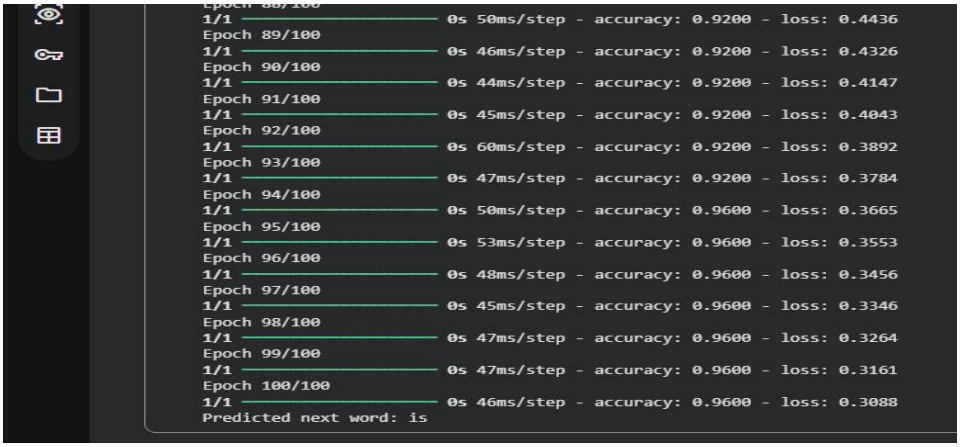
```
input_text = "machine learning"
```

```
token_text = tokenizer.texts_to_sequences([input_text])[0]
token_text = pad_sequences(
    [token_text],
    maxlen=max_len-1,
    padding='pre'
)
```

```
predicted_index = np.argmax(model.predict(token_text, verbose=0))
```

```
for word, index in tokenizer.word_index.items():
    if index == predicted_index:
        print("Predicted next word:", word)
        break
```

Output:



Question No. 3

Code example:

```
input_text = "machine learning"
print("Initial text:", input_text)

for i in range(6):
    token_text = tokenizer.texts_to_sequences([input_text])[0]
```

```
    padded_token_text = pad_sequences(
```

```
        [token_text],
        maxlen=max_len-1,
        padding='pre'
    )
```

```
    predicted_index = np.argmax(
        model.predict(padded_token_text, verbose=0)
    )
```

```
    for word, index in tokenizer.word_index.items():
        if index == predicted_index:
            input_text = input_text + " " + word
            break
```

```
print("Generated text:", input_text)
```

Output:

```
Initial text: machine learning
Generated text: machine learning is a branch of artificial intelligence
```

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No. 9

Convolution Neural Network

Open Ended LAB

Build a Convolutional Neural Network (CNN) using Python and TensorFlow/Keras to classify images of Cats and Dogs.

Instructions:

1. Collect or download a dataset containing:
    - **500 images of Cats**
    - **500 images of Dogs**
2. Organize the dataset as:

    dataset/

    cats/

    dogs/
3. Write Python code to:
    - Load and preprocess images (resize to 150×150)
    - Split into training (80%) and testing (20%)
    - Build a CNN model
    - Train the model for 10–20 epochs
    - Plot training & validation accuracy and loss
    - Evaluate model performance using a confusion matrix
    - Predict whether a new input image is Cat or Dog
4. At the end of the program, show the prediction result for a test image:
    - "This image is a CAT"
    - or

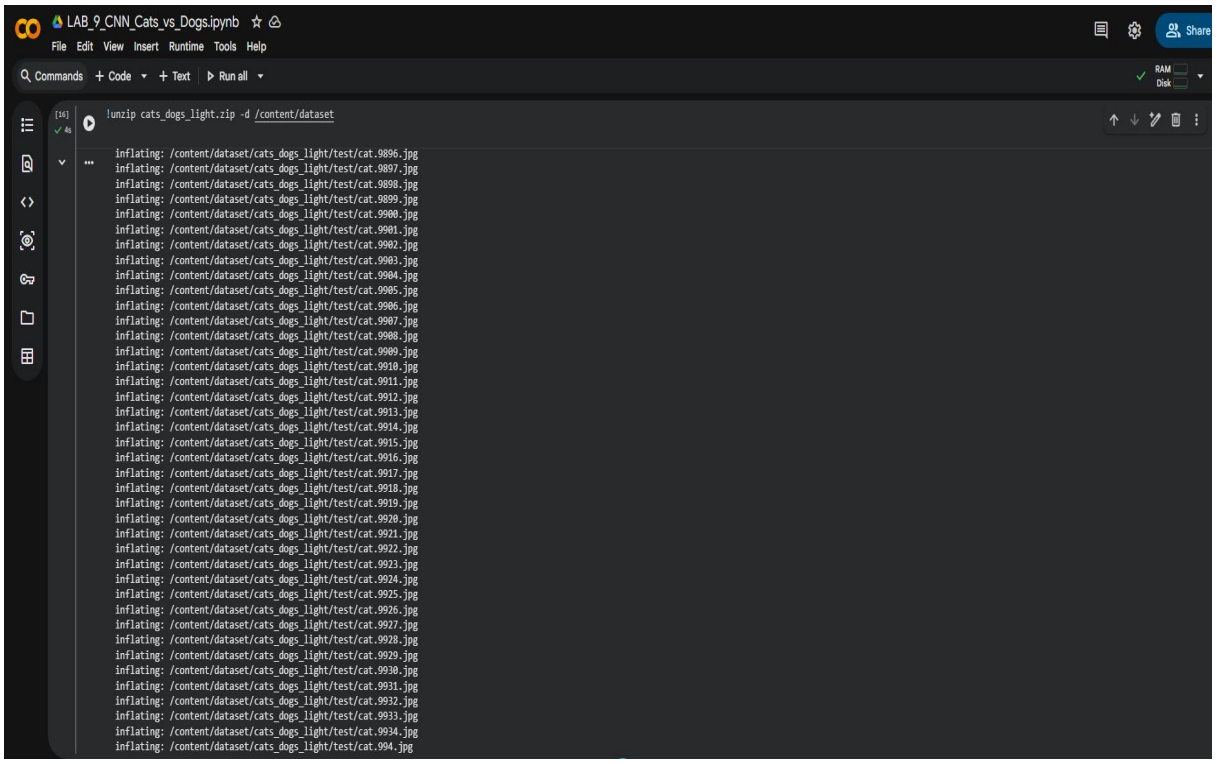
      "This image is a DOG"
5. Submit your Python code, dataset, graphs, and output screenshots.

Open Ended LAB

Build a Convolutional Neural Network (CNN) using Python and TensorFlow/Keras to classify images of Cats and Dogs.

Code example:

```
!wget https://zenodo.org/records/5226945/files/cats_dogs_light.zip
```



3. Organize the dataset as:

dataset/

cats/

dogs/

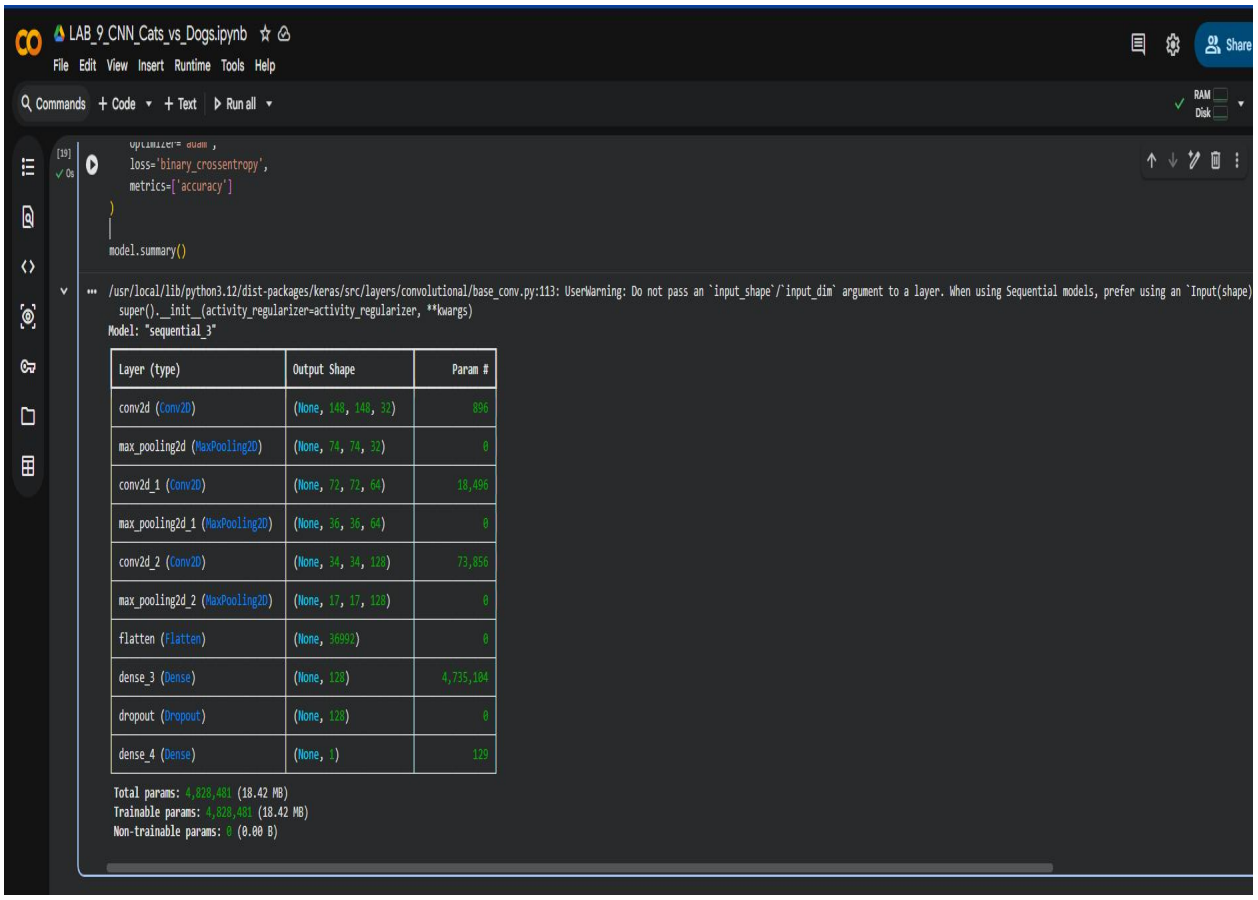
6. Write Python code to:

- Load and preprocess images (resize to 150×150)
- Split into training (80%) and testing (20%)
- Build a CNN model
- Train the model for 10–20 epochs
- Plot training & validation accuracy and loss
- Evaluate model performance using a confusion matrix
- Predict whether a new input image is Cat or Dog

Code example:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout


model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)


model.summary()
```

Output:



7. At the end of the program, show the prediction result for a test image:

   - "This image is a CAT"
   - or

     "This image is a DOG"

Code example:

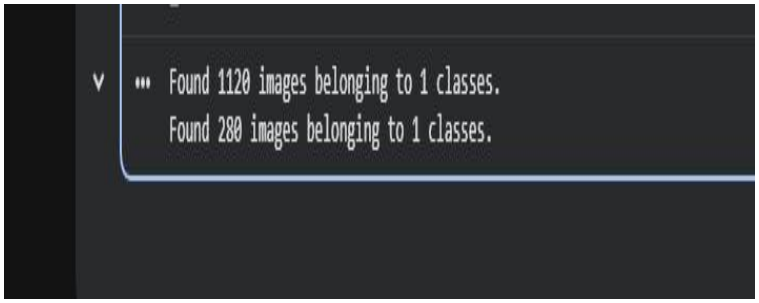
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Image preprocessing

img_height = 150

img_width = 150

batch_size = 32


datagen = ImageDataGenerator(

    rescale=1./255,         # normalize pixel values

    validation_split=0.2    # 80% train, 20% validation

)
```

```
# Training data generator

train_data = datagen.flow_from_directory(
```

```
    '/content/dataset',
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary',
    subset='training'
)
```

```
# Validation data generator
val_data = datagen.flow_from_directory(
    '/content/dataset',
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
)
```

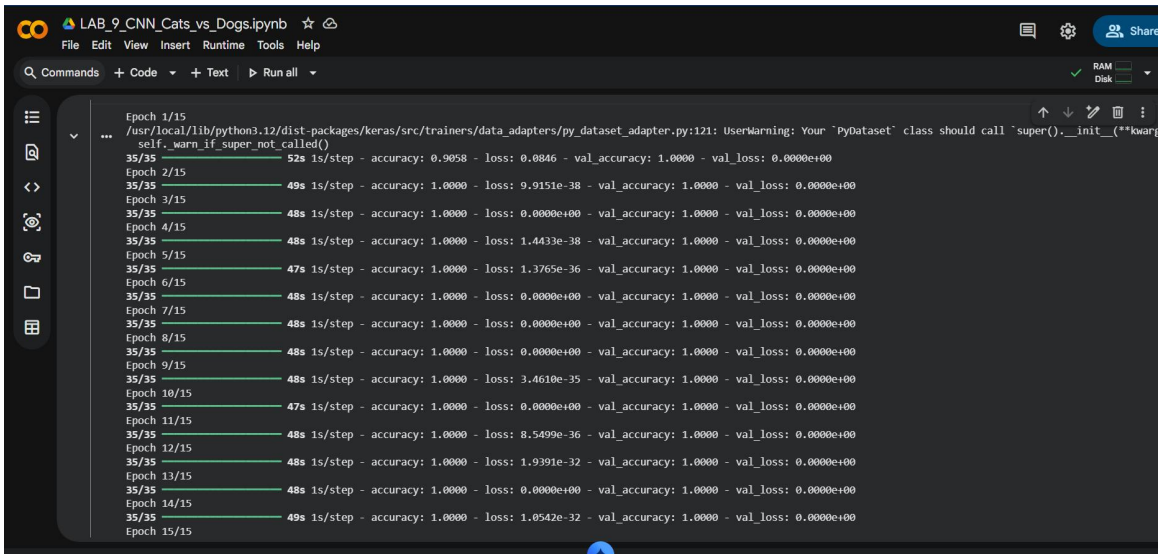
Output:



Code:

```
history = model.fit(
    train_data,
    epochs=15,
    validation_data=val_data
)
```

Output:

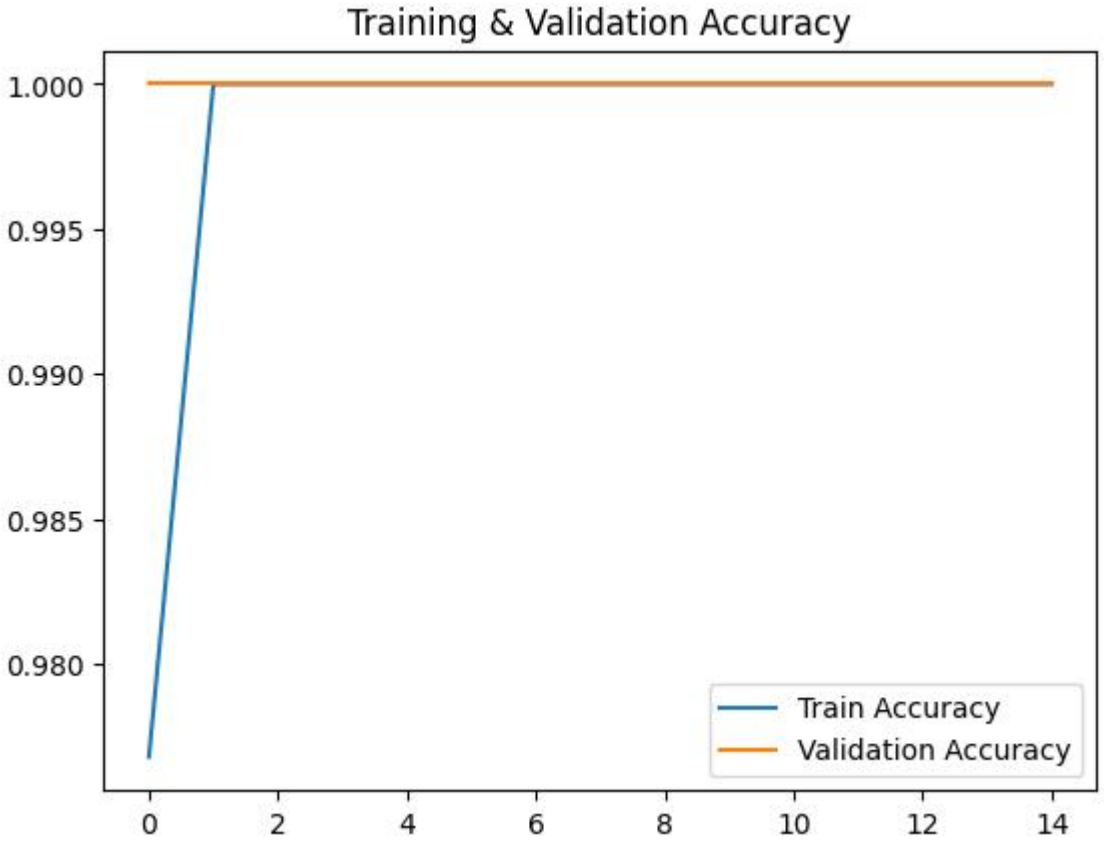


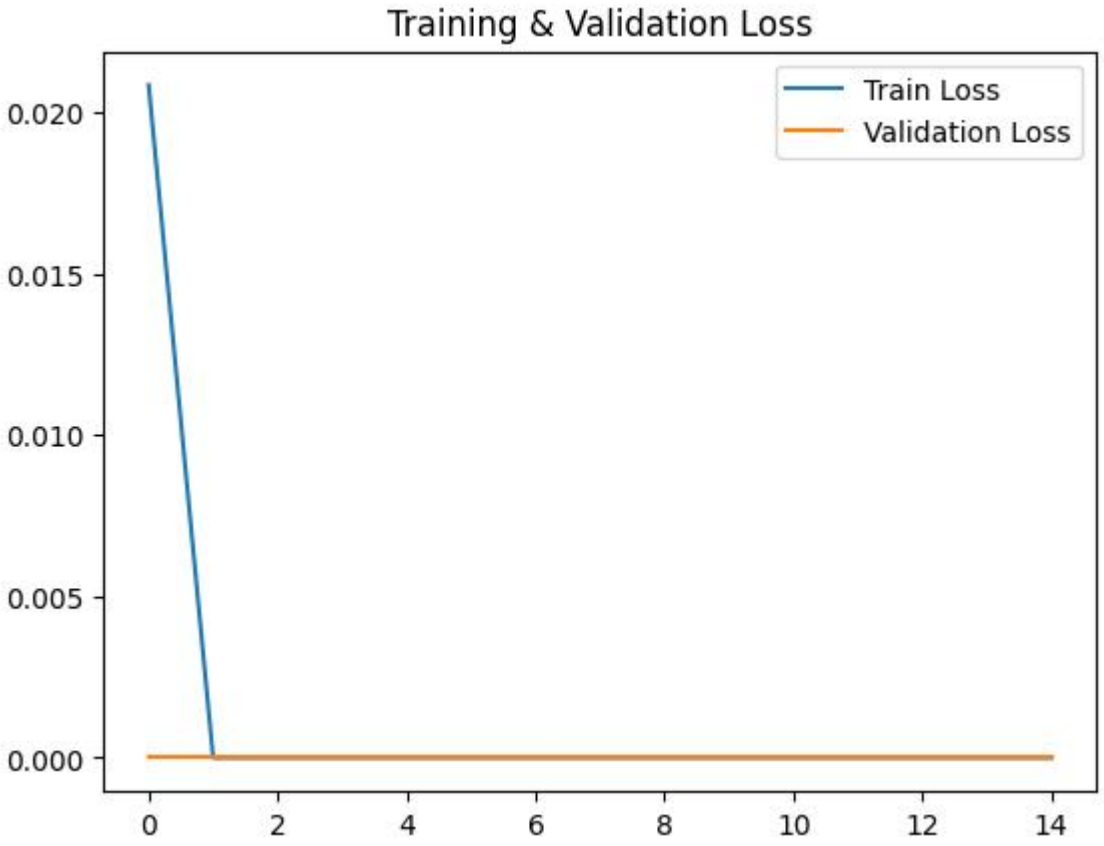
Code:

```
import matplotlib.pyplot as plt


# Accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training & Validation Accuracy')
plt.legend()
plt.show()
```

```
# Loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training & Validation Loss')
plt.legend()
plt.show()
```

Output:





LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

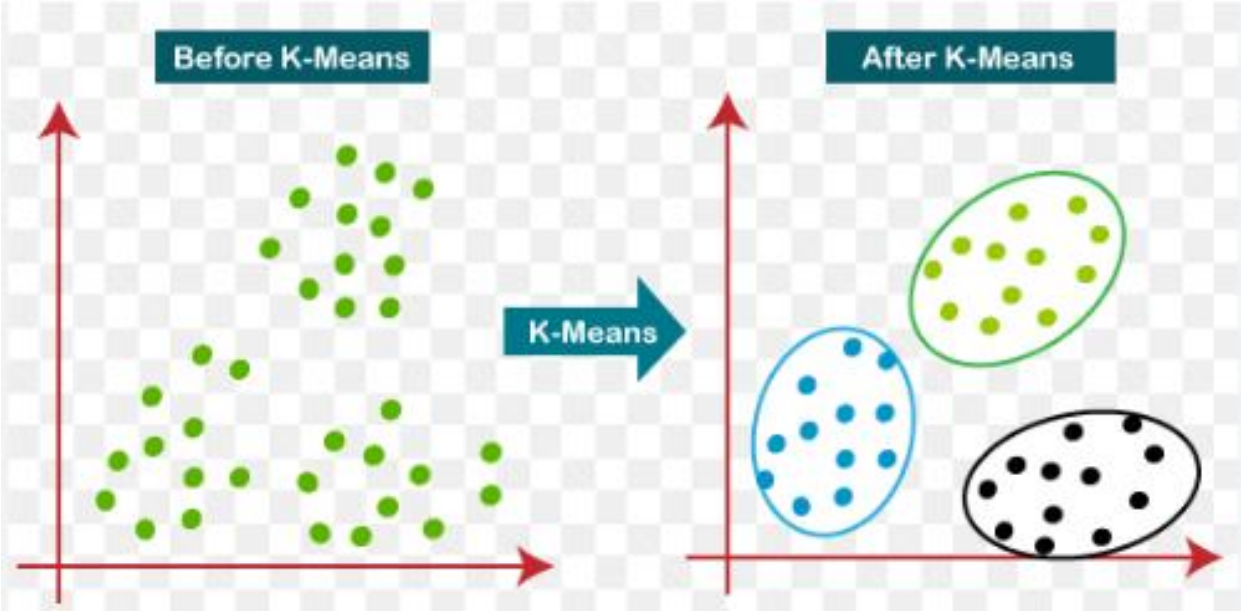
LAB Assignment No 10

K means Clustering Schemes in Machine Learning

In this lab, students will learn and implement K-Means Clustering, an unsupervised machine learning algorithm used to group data points into K distinct clusters based on similarity. Unlike supervised learning methods, K-Means does not require labeled data. Students will first apply K-Means on a small numerical dataset to understand clustering behavior, then use it on real-world datasets to visualize cluster formation and analyze results. Model performance will be interpreted using visualization and cluster characteristics.

Introduction

K-Means Clustering is an **unsupervised learning algorithm** that partitions a dataset into **K clusters**, where each data point belongs to the cluster with the nearest mean (centroid).



Working of K-Means:

1. Select the number of clusters **K**

2. Initialize K centroids randomly
3. Assign each data point to the nearest centroid
4. Update centroids by computing the mean of assigned points
5. Repeat steps 3 and 4 until convergence

Key Concepts:

- **Centroid** – center of a cluster
- **Inertia** – sum of squared distances within clusters
- **Elbow Method** – technique to choose optimal K

Applications:

- Customer segmentation
- Image compression
- Market basket analysis
- Document clustering

Solved Examples

Example 1: K-Means Clustering on a Simple Dataset

Apply K-Means clustering to group students based on **marks and attendance**

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
# Sample data: Marks vs Attendance
X = np.array([
    [40, 60],
    [45, 65],
    [70, 80],
    [75, 85],
    [90, 95],
    [85, 90]
])
# Apply K-Means
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_

# Plot clusters
plt.scatter(X[:,0], X[:,1], c=labels)
plt.scatter(kmeans.cluster_centers_[:,0],
kmeans.cluster_centers_[:,1], marker='X')
plt.xlabel("Marks")
plt.ylabel("Attendance")
plt.title("K-Means Clustering (K=2)")
plt.show()
```

Explanation

The algorithm groups students into clusters based on similarity in marks and attendance.

Example 2: Choosing Optimal K Using Elbow Method

Use the **Elbow Method** to determine the optimal number of clusters.

Solution:

```
inertia = []

for k in range(1, 6):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

# Plot Elbow Curve
plt.plot(range(1,6), inertia, marker='o')
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.title("Elbow Method")
plt.show()
```

Explanation

The "elbow point" in the graph suggests the best value of **K** where further increase does not significantly reduce inertia.

Example 3: K-Means Clustering on Iris Dataset

Apply K-Means clustering to the **Iris dataset** and visualize the clusters.

Solution:

```
from sklearn.datasets import load_iris
```

```
# Load Iris dataset
iris = load_iris()
X = iris.data[:, :2]  # Use first two features for visualization

# Apply K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(X)

# Visualize clusters
plt.scatter(X[:,0], X[:,1], c=labels)
plt.scatter(kmeans.cluster_centers_[:,0],
kmeans.cluster_centers_[:,1],
            marker='X')
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("K-Means Clustering on Iris Dataset")
plt.show()
```

Explanation

K-Means successfully groups data into three clusters corresponding to Iris species.

Lab Assignment No. 10

Question 1

Write Python code to implement K-Means clustering from scratch using the following data points:

P1(1,3), P2(2,2), P3(5,8), P4(8,5), P5(3,9),
P6(10,7), P7(3,3), P8(9,4), P9(3,7)

Tasks:

1. Use **K = 3** clusters

2. Initialize centroids as
    - C1 = P7(3,3)
    - C2 = P9(3,7)
    - C3 = P8(9,4)
3. Perform **2 iterations** manually in code
4. Plot all points and centroids
5. **Label all points (P1…P9)**
6. Sketch the clusters using matplot library in python

Question No. 2

Use the scikit-learn KMeans() library to cluster the same points.

P1(1,3), P2(2,2), P3(5,8), P4(8,5), P5(3,9), P6(10,7), P7(3,3), P8(9,4), P9(3,7)

Tasks:

1. Use K = 2, 3, and 4
2. Plot the clustering result for each K
3. Compare:
    - Number of points in each cluster
    - Final centroid locations
4. Draw the 3 graphs in your lab copy and explain how the shapes change with K.

Question 3 — Add a New User Point and Re-Cluster

Given the original 9 points, **add a new user**: **P10(6,2)**

Tasks:

1. Run K-Means using K = 3
2. Plot the graph with all 10 points

3. Identify:
    - Which cluster P10 joins
    - How centroids shift after adding P10
4. Sketch before/after clusters in notebook
5. Write a short explanation about how a new data point affects clustering.

Question 4 — Distance Table + First Iteration Manually

Using the given 9 points and initial centroids:

C1(3,3), C2(3,7), C3(9,4)

Tasks:

1. Compute **Euclidean distance** of each point to each centroid (manually or in python)
2. Create a distance table:

Point Dist to C1 Dist to C2 Dist to C3 Assigned Cluster

3. Perform **only the first iteration**
4. Compute **new centroids**
5. Plot the **first-iteration graph**
6. Draw the graph in your copy and show all labels.

Question 1

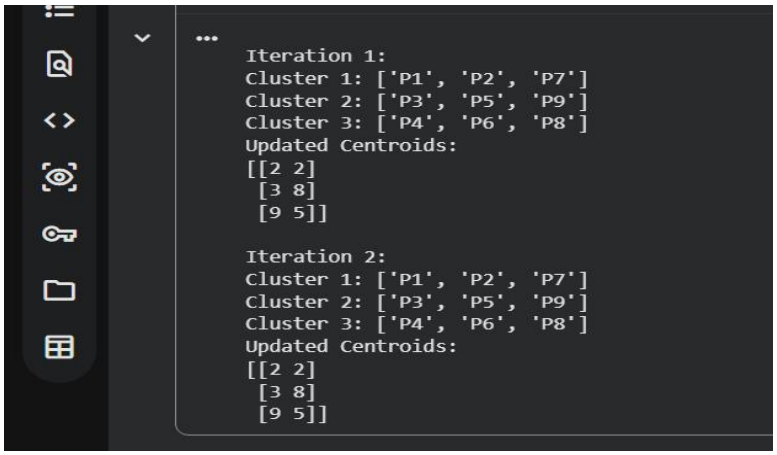
Code:

```
import numpy as np
import matplotlib.pyplot as plt
points = np.array([
    [1,3],
    [2,2],
    [5,8],
```

```
    [8,5],
    [3,9],
    [10,7],
    [3,3],
    [9,4],
    [3,7]
])

labels = ['P1','P2','P3','P4','P5','P6','P7','P8','P9']
centroids = np.array([
    [3,3],
    [3,7],
    [9,4]
])

def euclidean_distance(a, b):
    return np.sqrt(np.sum((a-b)**2))
```

```
K = 3
for iteration in range(2):
    clusters = {i: [] for i in range(K)}
    for idx, point in enumerate(points):
        distances = [euclidean_distance(point, c) for c in centroids]
        cluster_idx = np.argmin(distances)
        clusters[cluster_idx].append(idx)

    print(f"\nIteration {iteration+1}:")
    for i in range(K):
        print(f"Cluster {i+1}: {[labels[j] for j in clusters[i]]}")
    for i in range(K):
        if clusters[i]:
            centroids[i] = np.mean(points[clusters[i]], axis=0)
    print(f"Updated Centroids:\n{centroids}")
```

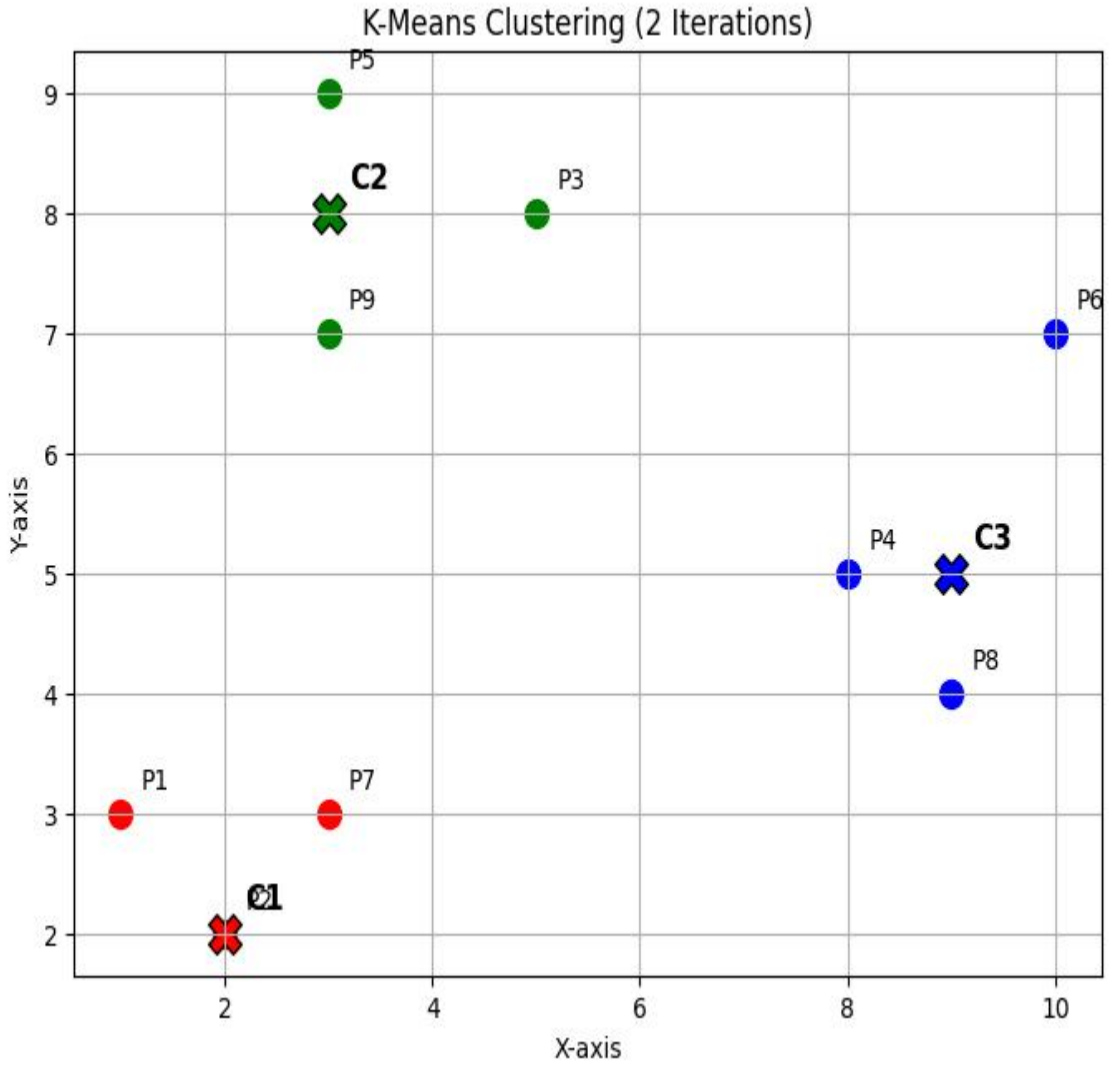
Output:



Code:

```
colors = ['r','g','b']

plt.figure(figsize=(8,6))

for i in range(K):

    for idx in clusters[i]:

        plt.scatter(points[idx][0], points[idx][1], color=colors[i], s=100)

        plt.text(points[idx][0]+0.2, points[idx][1]+0.2, labels[idx])

for i, c in enumerate(centroids):

    plt.scatter(c[0], c[1], color=colors[i], marker='X', s=200, edgecolor='k')

    plt.text(c[0]+0.2, c[1]+0.2, f"C{i+1}", fontsize=12, fontweight='bold')


plt.title("K-Means Clustering (2 Iterations)")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.grid(True)

plt.show()
```

Output:



Question No. 2.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
points = np.array([
    [1,3],
    [2,2],
    [5,8],
    [8,5],
    [3,9],
    [10,7],
    [3,3],
    [9,4],
    [3,7]
])
```

```
labels = ['P1','P2','P3','P4','P5','P6','P7','P8','P9']
```

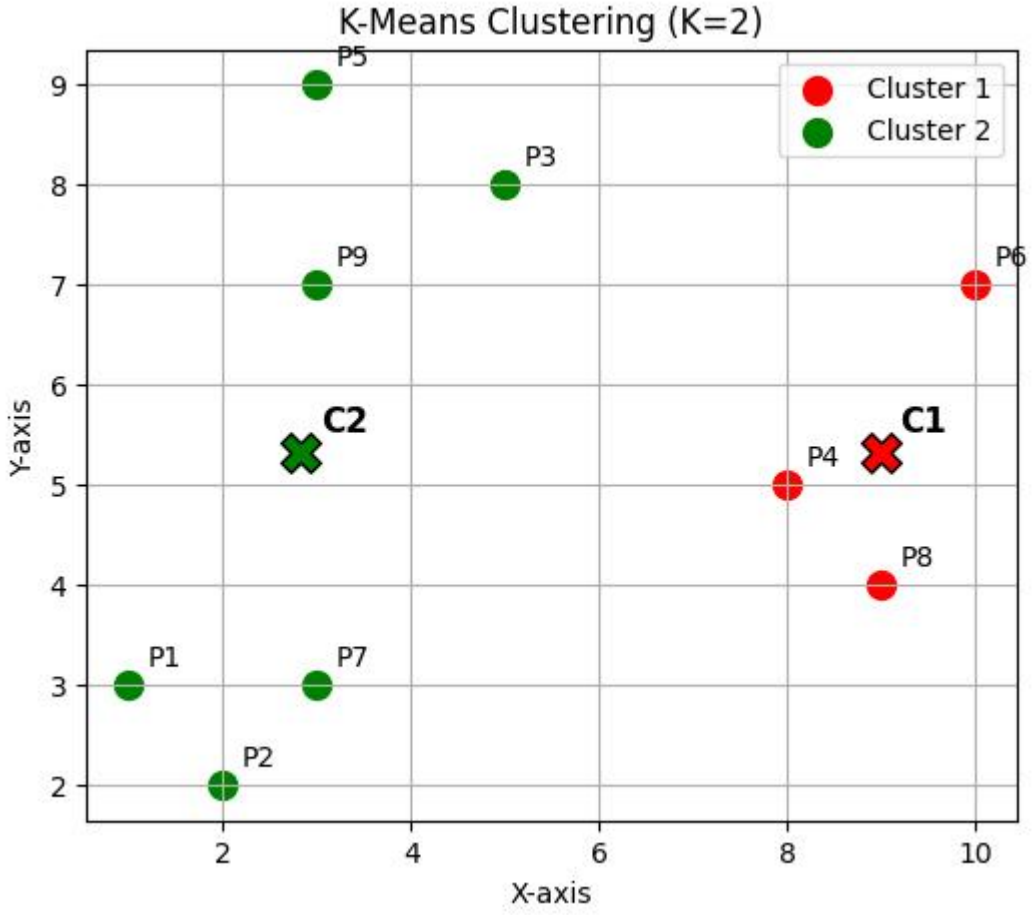
```
def kmeans_cluster_plot(points, K):
    kmeans = KMeans(n_clusters=K, random_state=42)
    kmeans.fit(points)

    centroids = kmeans.cluster_centers_
    cluster_labels = kmeans.labels_
    print(f"\nK = {K}")
    for i in range(K):
        cluster_points = [labels[j] for j in range(len(points)) if cluster_labels[j]==i]
        print(f"Cluster {i+1}: Points {cluster_points}")
        print(f"Centroid {i+1}: {centroids[i]}")
    colors = ['r','g','b','c']
    plt.figure(figsize=(6,5))
    for i in range(K):
        cluster_points = points[cluster_labels==i]
        plt.scatter(cluster_points[:,0], cluster_points[:,1], color=colors[i], s=100,
label=f'Cluster {i+1}')
        for j, pt_idx in enumerate(np.where(cluster_labels==i)[0]):
            plt.text(points[pt_idx][0]+0.2, points[pt_idx][1]+0.2, labels[pt_idx])
    for i, c in enumerate(centroids):
        plt.scatter(c[0], c[1], color=colors[i], marker='X', s=200, edgecolor='k')
        plt.text(c[0]+0.2, c[1]+0.2, f"C{i+1}", fontsize=12, fontweight='bold')

    plt.title(f"K-Means Clustering (K={K})")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    plt.legend()
    plt.grid(True)
    plt.show()
```

```
for K in [2,3,4]:
    kmeans_cluster_plot(points, K)
```

Output:

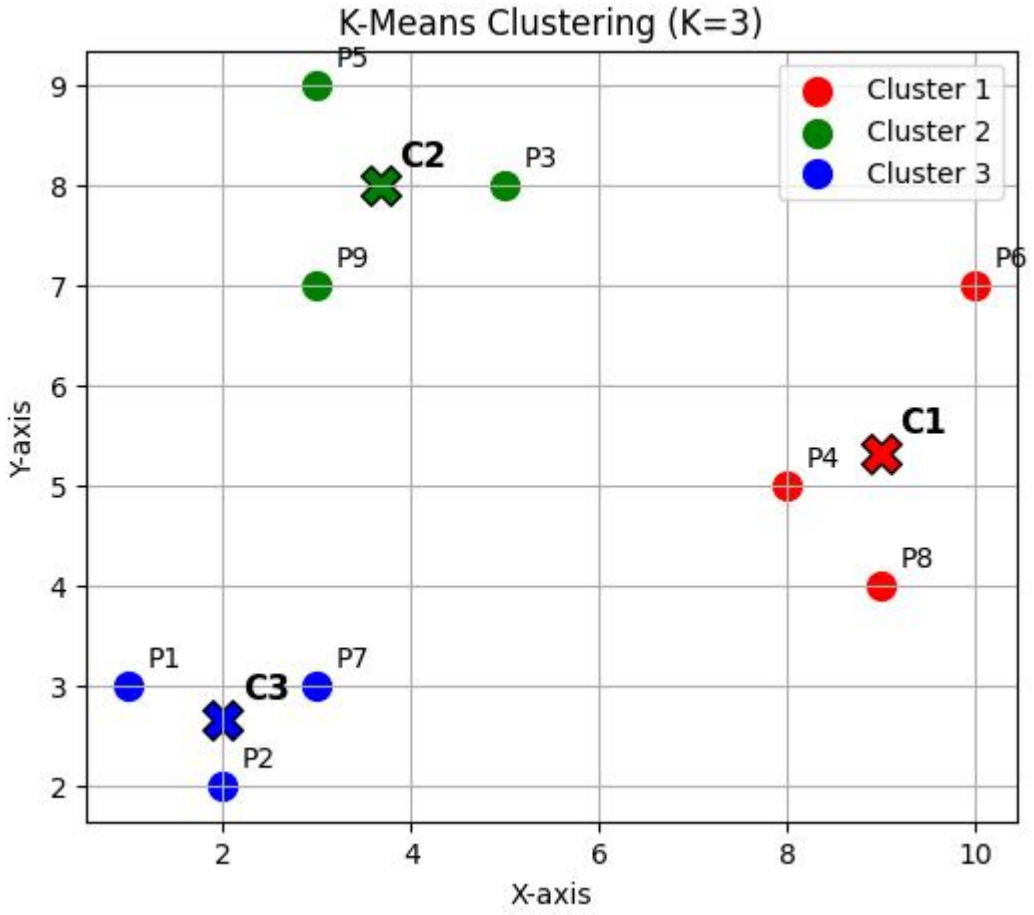
A)

```
K = 2
Cluster 1: Points ['P4', 'P6', 'P8']
Centroid 1: [9.         5.33333333]
Cluster 2: Points ['P1', 'P2', 'P3', 'P5', 'P7', 'P9']
Centroid 2: [2.83333333 5.33333333]
```



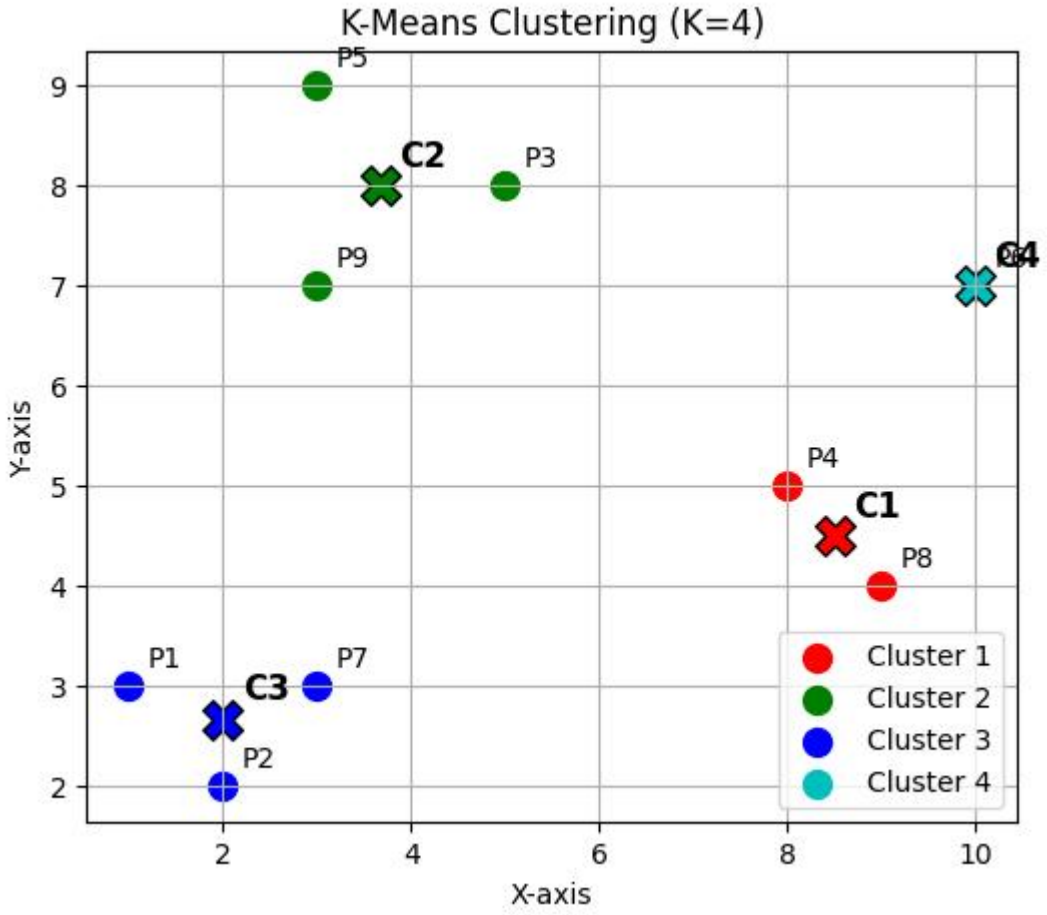
B)

```
K = 3
Cluster 1: Points ['P4', 'P6', 'P8']
Centroid 1: [9.         5.33333333]
Cluster 2: Points ['P3', 'P5', 'P9']
Centroid 2: [3.66666667 8.        ]
Cluster 3: Points ['P1', 'P2', 'P7']
Centroid 3: [2.         2.66666667]
```



C)

```
K = 4
Cluster 1: Points ['P4', 'P8']
Centroid 1: [8.5 4.5]
Cluster 2: Points ['P3', 'P5', 'P9']
Centroid 2: [3.66666667 8.        ]
Cluster 3: Points ['P1', 'P2', 'P7']
Centroid 3: [2.         2.66666667]
Cluster 4: Points ['P6']
Centroid 4: [10.  7.]
```



Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

```
points = np.array([
    [1,3],
    [2,2],
    [5,8],
    [8,5],
    [3,9],
    [10,7],
    [3,3],
    [9,4],
    [3,7]
])
labels = ['P1','P2','P3','P4','P5','P6','P7','P8','P9']
points = np.vstack([points, [6,2]])
```

```
labels.append('P10')
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(points)

centroids = kmeans.cluster_centers_
cluster_labels = kmeans.labels_
```

```
p10_cluster = cluster_labels[-1]
print(f"P10 belongs to Cluster {p10_cluster+1}")
for i, c in enumerate(centroids):
    print(f"Centroid {i+1}: {c}")
```

Output:

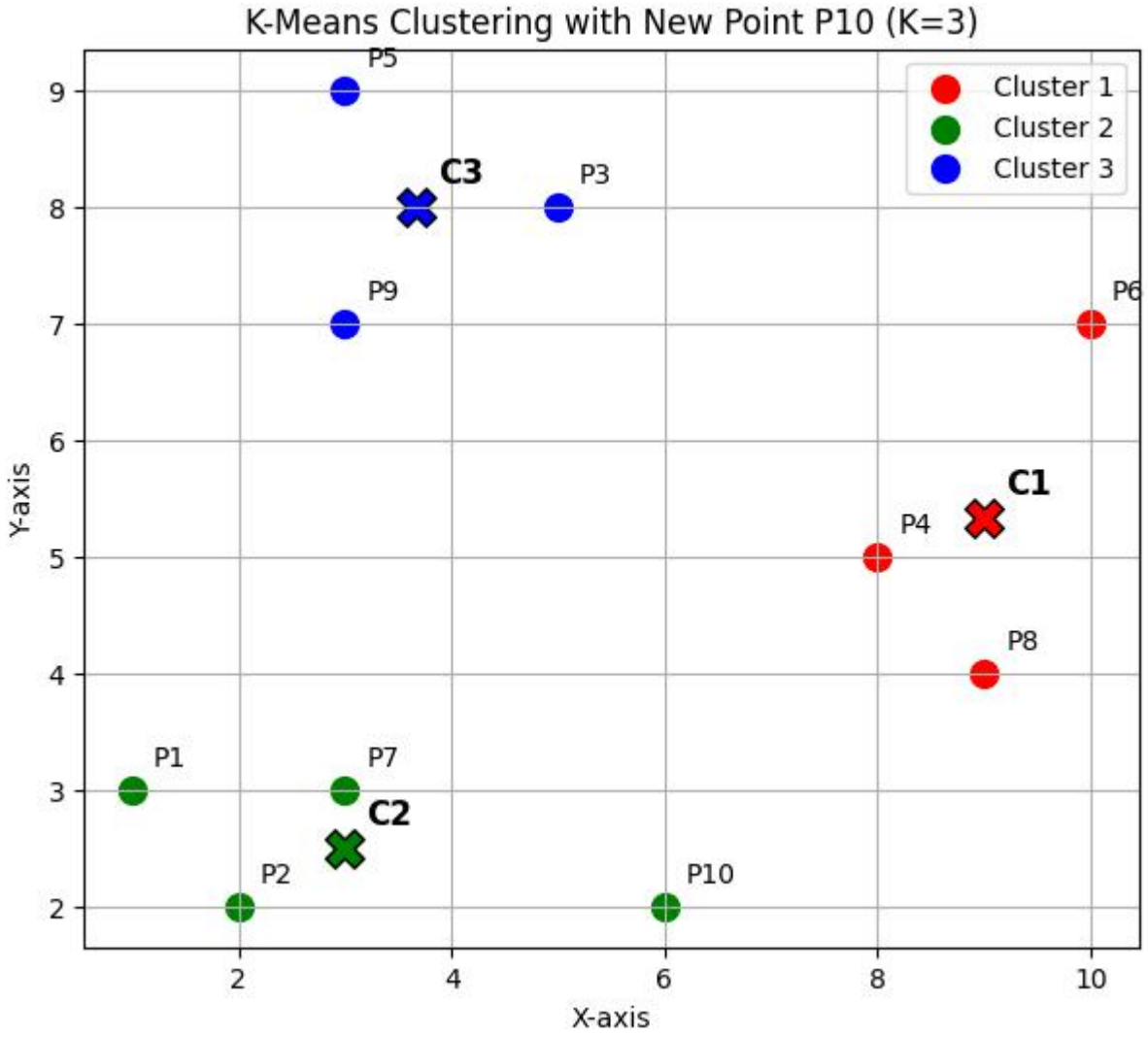
```
P10 belongs to Cluster 2
Centroid 1: [9.         5.33333333]
Centroid 2: [3.  2.5]
Centroid 3: [3.66666667 8.        ]
```

Code:

```
colors = ['r','g','b']
plt.figure(figsize=(7,6))
for i in range(3):
    cluster_points = points[cluster_labels==i]
    plt.scatter(cluster_points[:,0], cluster_points[:,1], color=colors[i], s=100,
label=f'Cluster {i+1}')
    for j, pt_idx in enumerate(np.where(cluster_labels==i)[0]):
        plt.text(points[pt_idx][0]+0.2, points[pt_idx][1]+0.2, labels[pt_idx])
for i, c in enumerate(centroids):
    plt.scatter(c[0], c[1], color=colors[i], marker='X', s=200, edgecolor='k')
    plt.text(c[0]+0.2, c[1]+0.2, f"C{i+1}", fontsize=12, fontweight='bold')

plt.title("K-Means Clustering with New Point P10 (K=3)")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid(True)
```

```
plt.legend()
plt.show()
```

Output:



Question 4 —

Distance Table + First Iteration Manually

Code:

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

points = np.array([
    [1,3],
    [2,2],
    [5,8],
    [8,5],
    [3,9],
    [10,7],
```

```
    [3,3],
    [9,4],
    [3,7]
])

labels = ['P1','P2','P3','P4','P5','P6','P7','P8','P9']
centroids = np.array([
    [3,3],
    [3,7],
    [9,4]
])
```

```
def euclidean_distance(a, b):
    return np.sqrt(np.sum((a-b)**2))
distance_table = []
assigned_clusters = []

for i, p in enumerate(points):
    distances = [euclidean_distance(p, c) for c in centroids]
    assigned_cluster = np.argmin(distances) + 1
    distance_table.append(distances)
    assigned_clusters.append(assigned_cluster)
distance_table = np.array(distance_table)
```

```
import pandas as pd

df = pd.DataFrame(distance_table, columns=['Dist to C1','Dist to C2','Dist to C3'])
df['Assigned Cluster'] = assigned_clusters
df['Point'] = labels
df = df[['Point','Dist to C1','Dist to C2','Dist to C3','Assigned Cluster']]
```

```
print(df)
```

Output:

```
...
```

	Point	Dist to C1	Dist to C2	Dist to C3	Assigned Cluster
0	P1	2.000000	4.472136	8.062258	1
1	P2	1.414214	5.099020	7.280110	1
2	P3	5.385165	2.236068	5.656854	2
3	P4	5.385165	5.385165	1.414214	3
4	P5	6.000000	2.000000	7.810250	2
5	P6	8.062258	7.000000	3.162278	3
6	P7	0.000000	4.000000	6.082763	1
7	P8	6.082763	6.708204	0.000000	3
8	P9	4.000000	0.000000	6.708204	2

Code:

```
new_centroids = []


for k in range(1, 4):
    cluster_points = points[np.array(assigned_clusters)==k]
    new_c = np.mean(cluster_points, axis=0)
    new_centroids.append(new_c)
```

```
new_centroids = np.array(new_centroids)
print("\nNew Centroids after 1st iteration:")
for i, c in enumerate(new_centroids):
    print(f"C{i+1}: {c}")
```

Output:

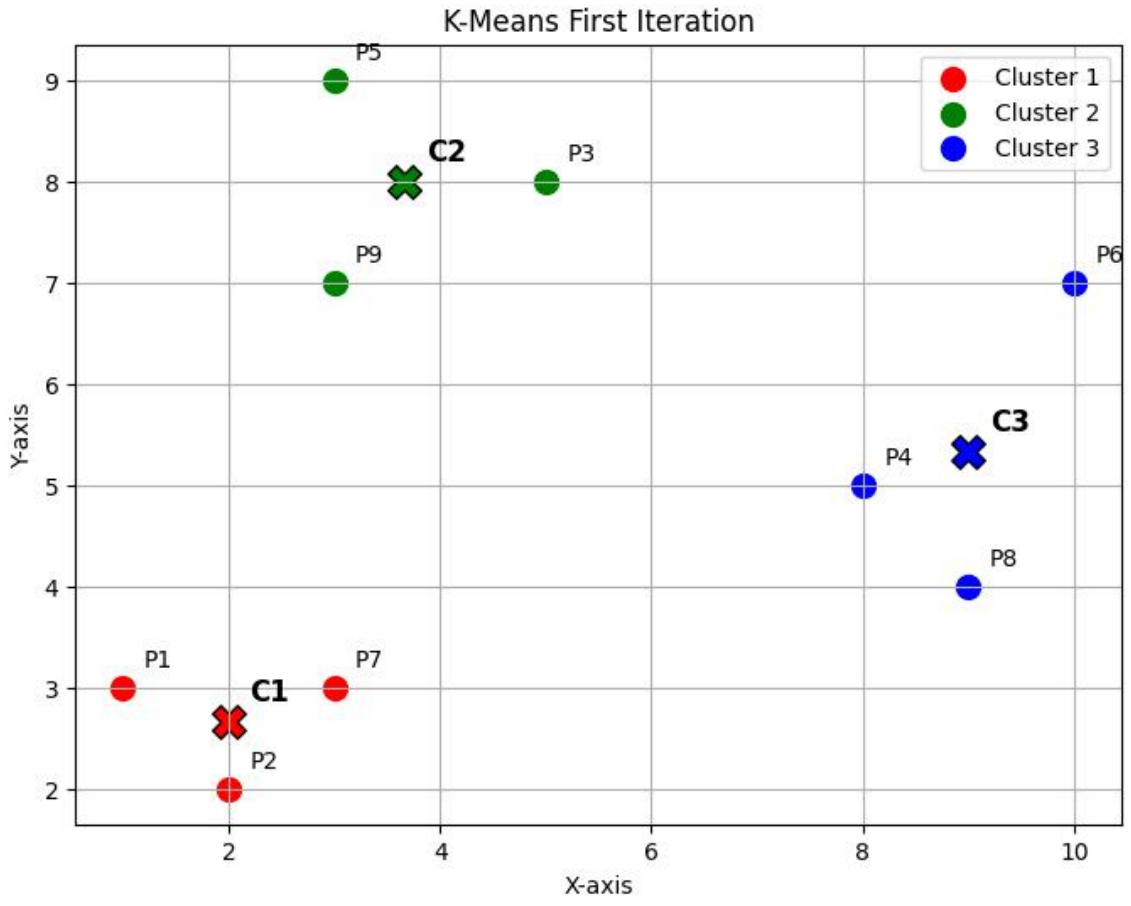
```
...
    New Centroids after 1st iteration:
    C1: [2.         2.66666667]
    C2: [3.66666667 8.        ]
    C3: [9.         5.33333333]
```

Code:

```
colors = ['r','g','b']
plt.figure(figsize=(8,6))
```

```
for i in range(1,4):
    cluster_points = points[np.array(assigned_clusters)==i]
    plt.scatter(cluster_points[:,0], cluster_points[:,1], color=colors[i-1], s=100,
label=f'Cluster {i}')
    for idx in np.where(np.array(assigned_clusters)==i)[0]:
        plt.text(points[idx][0]+0.2, points[idx][1]+0.2, labels[idx])
for i, c in enumerate(new_centroids):
    plt.scatter(c[0], c[1], color=colors[i], marker='X', s=200, edgecolor='k')
    plt.text(c[0]+0.2, c[1]+0.2, f"C{i+1}", fontsize=12, fontweight='bold')


plt.title("K-Means First Iteration")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid(True)
plt.legend()
plt.show()
```

Output:



LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No 11

Agglomerative Hierarchical Clustering

In this lab, students will learn how to perform Agglomerative Hierarchical Clustering (AHC), a method used to group similar data points into clusters. The lab involves:

- Understanding hierarchical clustering and dendrograms.
- Performing clustering on datasets using Python (scikit-learn, scipy).
- Visualizing clusters using dendrograms.
- Interpreting the clustering results for practical data analysis.

Objectives:

1. Understand hierarchical clustering concepts and linkage methods.
2. Perform agglomerative clustering on sample datasets.
3. Visualize the clustering process using dendrograms.
4. Analyze cluster assignments and validate results.

Theory

1. Introduction to Hierarchical Clustering

Hierarchical clustering is an **unsupervised learning** method that builds a hierarchy of clusters. It can be:

- **Agglomerative (bottom-up)**:
  Each observation starts as its own cluster, and pairs of clusters are merged step by step until only one cluster remains.
- **Divisive (top-down)**:
  Start with all observations in one cluster and recursively split them into smaller clusters.

2. Agglomerative Hierarchical Clustering

- Start with each data point as a separate cluster.
- Compute a **distance matrix** between all clusters.
- Merge the **two closest clusters** at each step.
- Repeat until all points belong to a single cluster.

Distance Metrics:

- **Euclidean Distance:** Most common for continuous data.
- **Manhattan Distance:** Sum of absolute differences.
- **Cosine Distance:** Measures angular distance for high-dimensional data.

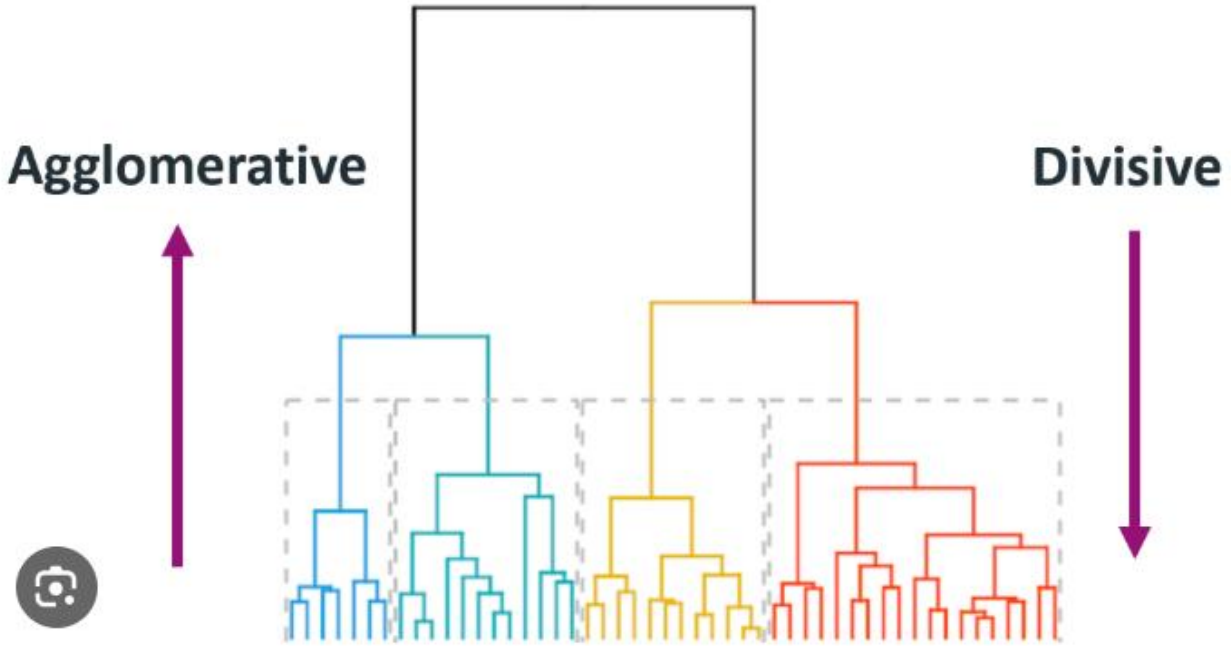
Linkage Methods:

- **Single Linkage:** Distance between closest points of two clusters.
- **Complete Linkage:** Distance between farthest points of two clusters.
- **Average Linkage:** Average distance between all points in two clusters.
- **Ward's Method:** Minimizes variance within clusters.

3. Dendrogram

A **dendrogram** is a tree-like diagram showing the order of cluster merges. It helps to:

- Visualize the hierarchy of clusters.
- Decide the optimal number of clusters by cutting the dendrogram.



4. Applications

- Customer segmentation in marketing.
- Document clustering in NLP.
- Gene expression analysis in bioinformatics.
- Image segmentation.

Python Libraries Required

```
import numpy as np
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

Solved Examples

Example 1: Clustering Simple 2D Points

Dataset:

```
data = np.array([[1, 2], [2, 3], [5, 8], [6, 9], [10, 12]])
```

Solution

```
# Step 1: Import libraries
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt

# Step 2: Linkage matrix
Z = linkage(data, method='ward')  # Using Ward's method

# Step 3: Plot dendrogram
plt.figure(figsize=(6,4))
dendrogram(Z)
plt.title("Dendrogram - Example 1")
plt.show()

# Step 4: Form clusters (choose 2 clusters)
clusters = fcluster(Z, t=2, criterion='maxclust')
print("Cluster assignments:", clusters)
```

Output:

```
less                                                    Copy code

Cluster assignments: [1 1 2 2 2]
```

Explanation: The first two points are grouped together; the last three points form the second cluster.

Example 2: Agglomerative Clustering on Random Dataset

Solution

```
from sklearn.datasets import make_blobs
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
```

```
# Generate random data
X, _ = make_blobs(n_samples=8, centers=3, random_state=42)

# Linkage
Z = linkage(X, method='complete')

# Dendrogram
plt.figure(figsize=(6,4))
dendrogram(Z)
plt.title("Dendrogram - Example 2")
plt.show()

# Form clusters
clusters = fcluster(Z, t=3, criterion='maxclust')
print("Cluster assignments:", clusters)
```

Explanation: The dendrogram shows three distinct clusters; cluster labels indicate the group each point belongs to.

Example 3: Agglomerative Clustering on Iris Dataset (subset)

Solution

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt

# Load Iris dataset
iris = load_iris()
X = iris.data[:, :2]  # Use only sepal length and width
X = StandardScaler().fit_transform(X)

# Linkage
Z = linkage(X, method='average', metric='euclidean')
```

```
# Plot dendrogram
plt.figure(figsize=(8,5))
dendrogram(Z)
plt.title("Dendrogram - Iris Example")
plt.show()

# Form clusters (3 clusters)
clusters = fcluster(Z, t=3, criterion='maxclust')
print("Cluster assignments:", clusters)
```

Explanation:

- Standardization is important to normalize features.
- The dendrogram helps to visualize clusters of similar iris species.
- fcluster assigns each data point to a cluster.

LAB Assignment No. 11

Question 1:

Perform Agglomerative Clustering with Different Linkages.

Task:

Load the "shopping-data.csv" dataset, extract the features *Annual Income* and *Spending Score*, and perform **Agglomerative Clustering** using:

- linkage = "ward"
- linkage = "complete"
- linkage = "average"

Instructions:

1. Perform clustering using AgglomerativeClustering.
2. Plot the clusters using matplotlib.
3. Compare how the cluster structure changes with each linkage method.

Question 2:

Draw a Dendrogram and Identify the Optimal Number of Clusters

Task:

Using the same dataset or any synthetic dataset, draw a **dendrogram** using:

```
from scipy.cluster.hierarchy import dendrogram, linkage
```

Instructions:

1. Fit the data using linkage(method='ward').
2. Plot a dendrogram.
3. From the dendrogram, visually determine:
    - The optimal number of clusters
    - The height at which clusters merge
4. Explain why hierarchical clustering may be preferred over K-Means.

Question 3: Compare Agglomerative vs Divisive Hierarchical Clustering

Task:

Using a small synthetic dataset (e.g., 10–12 points), perform:

- Agglomerative Clustering
- Divisive Clustering (manual split or using a library like sklearn-extra)

Instructions:

1. Plot dendrograms for both methods.
2. Compare the merge/split patterns.
3. Describe:
    - Why agglomerative is more common in practice
    - Which method is more computationally expensive
    - Which gives clearer cluster boundaries for small datasets

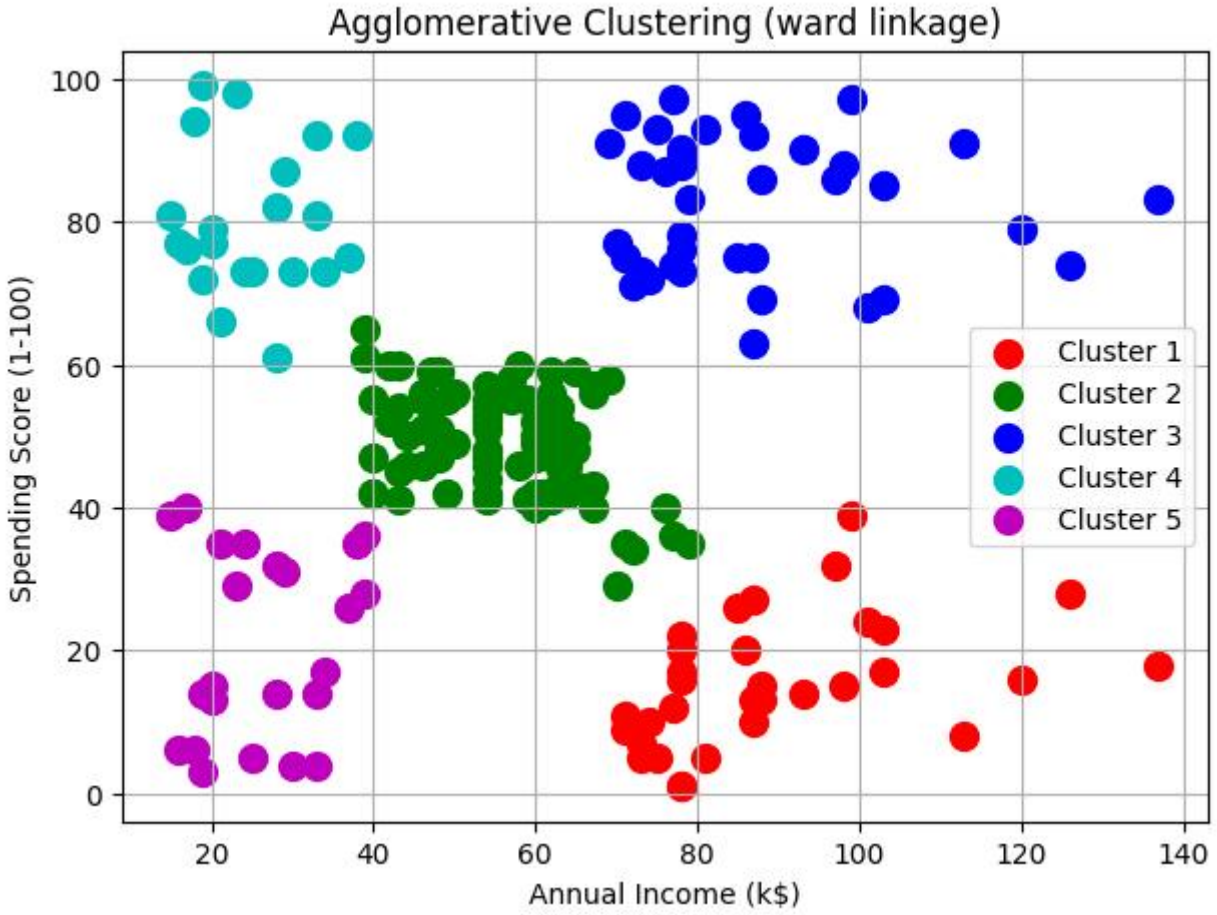
Question 1:

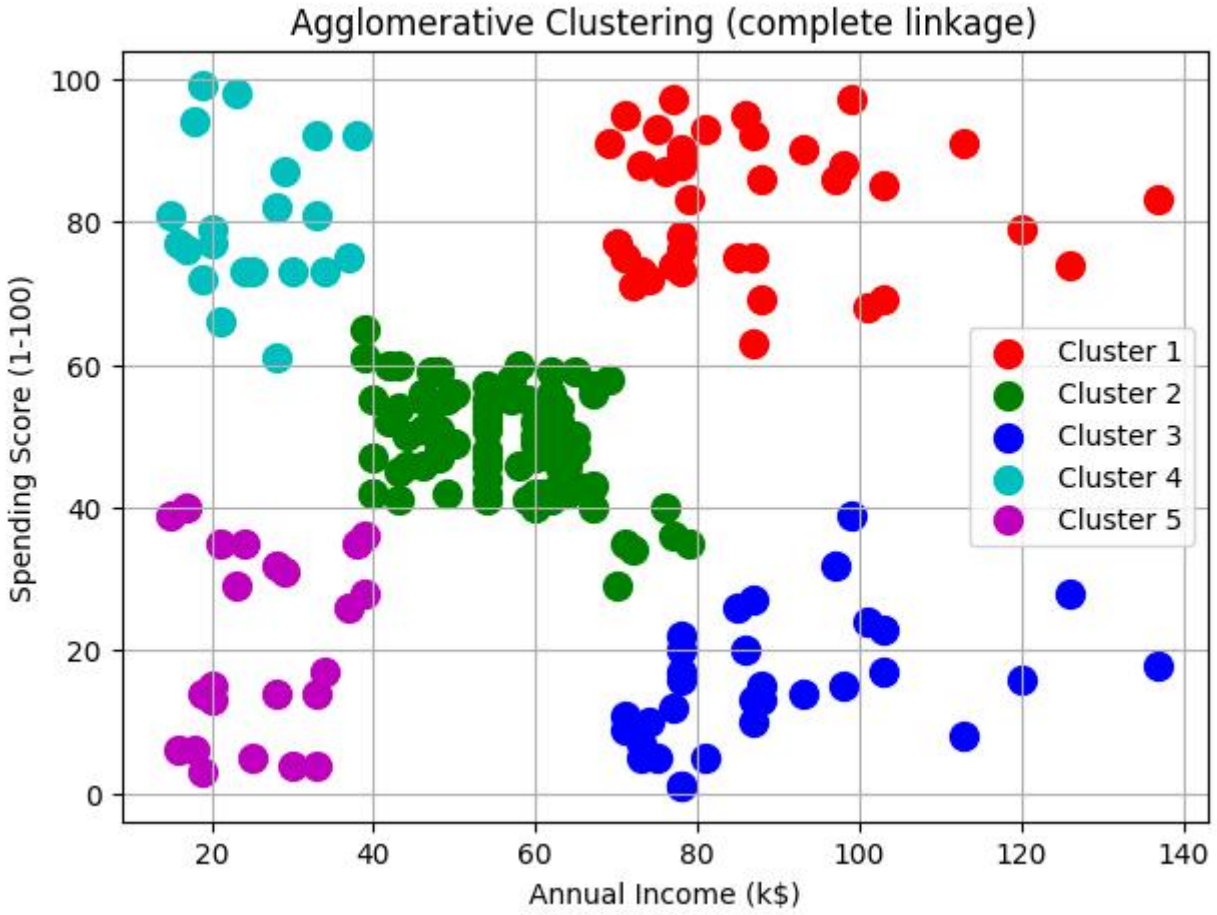
Code:

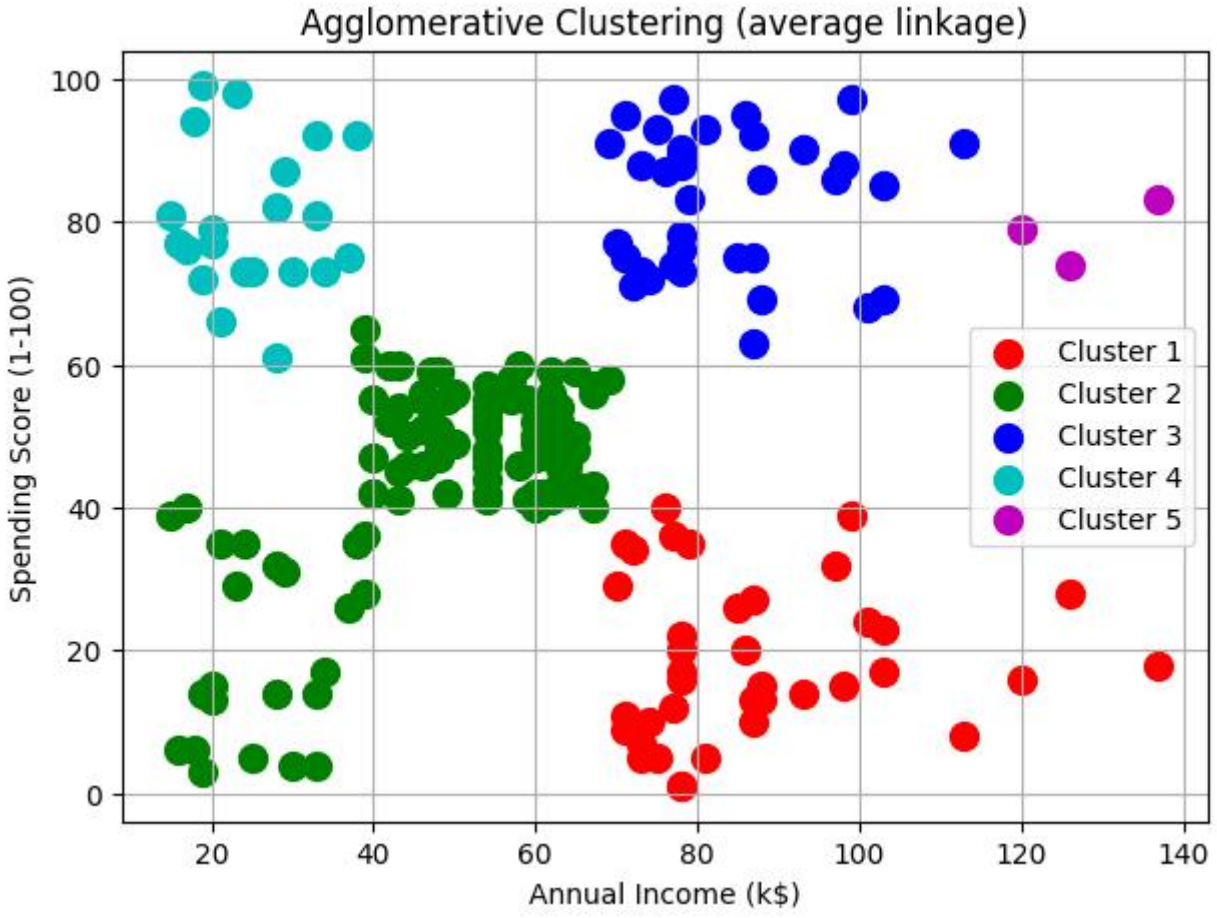
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

!wget -O shopping-data.csv
https://gist.githubusercontent.com/JuanPQ26/07a8c26ab285792555b74c1e31aaa10d/raw/Mall_Customers.csv

data = pd.read_csv('shopping-data.csv')

print("First 5 rows of dataset:")

print(data.head())

X = data[['Annual Income (k$)', 'Spending Score (1-100)']].values

def agglomerative_plot(X, linkage_method):

    cluster_model = AgglomerativeClustering(n_clusters=5, linkage=linkage_method)

    labels = cluster_model.fit_predict(X)

    plt.figure(figsize=(7,5))

    colors = ['r','g','b','c','m']

    for i in range(5):

        cluster_points = X[labels==i]

        plt.scatter(cluster_points[:,0], cluster_points[:,1], color=colors[i], s=100,
label=f'Cluster {i+1}')


    plt.title(f"Agglomerative Clustering ({linkage_method} linkage)")

    plt.xlabel("Annual Income (k$)")

    plt.ylabel("Spending Score (1-100)")

    plt.legend()

    plt.grid(True)

    plt.show()

for linkage in ['ward', 'complete', 'average']:

    agglomerative_plot(X, linkage)
```

Output:

```
--2026-01-10 16:48:15--  https://gist.githubusercontent.com/JuanPQ26/07a8c26ab285792555b74c1e31aaa10d/raw/Mall_Customers.csv
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4204 (4.1K) [text/plain]
Saving to: ‘shopping-data.csv’

shopping-data.csv   100%[==================>]   4.11K  --.-KB/s    in 0s

2026-01-10 16:48:15 (58.8 MB/s) - ‘shopping-data.csv’ saved [4204/4204]

First 5 rows of dataset:
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)  \
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40

   Suitable for a bank loan
0                         1
1                         0
2                         1
3                         0
4                         1
```


Agglomerative Clustering (ward linkage)
Spending Score (1-100)
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5


Agglomerative Clustering (complete linkage)
Spending Score (1-100)
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5



Question 2:

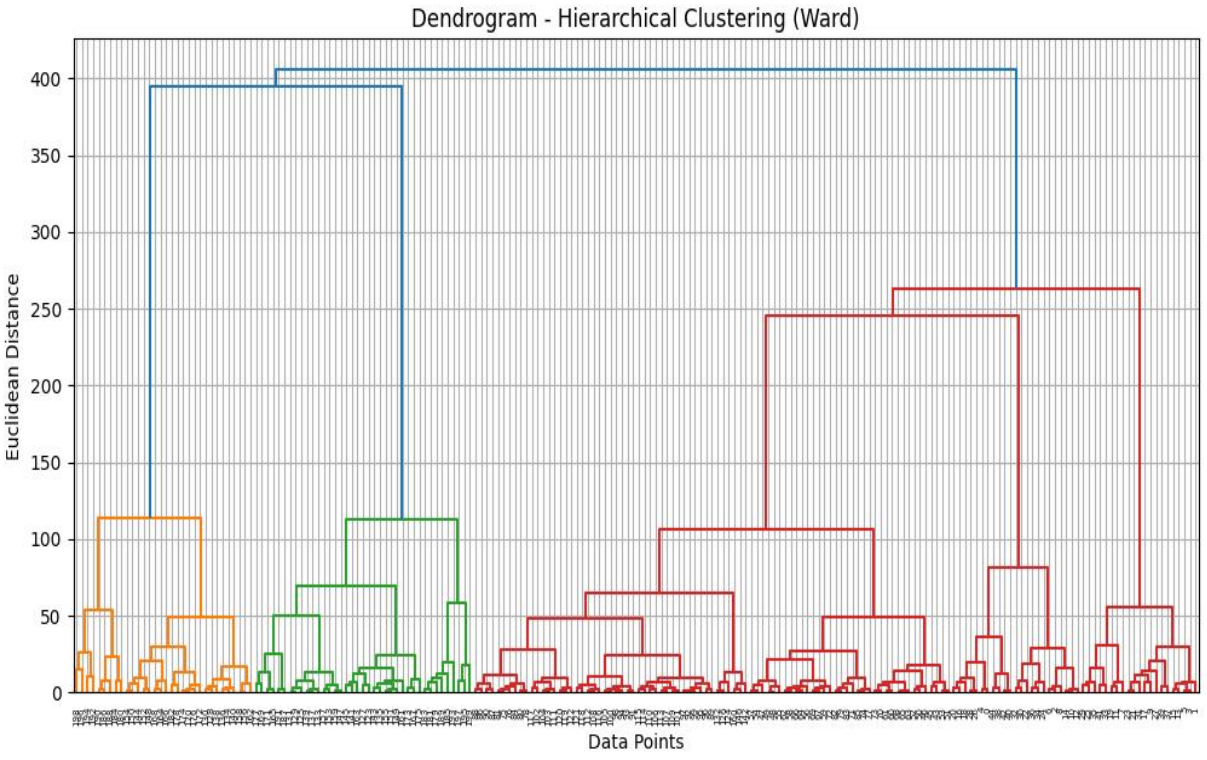
Code:

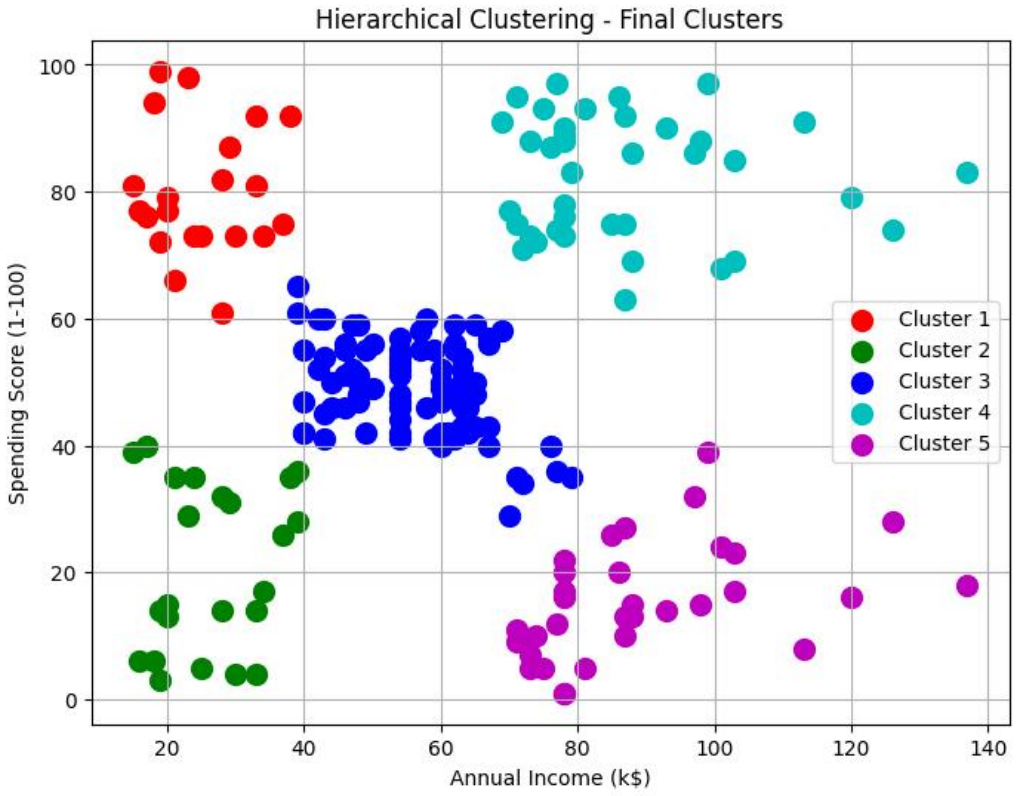
```
import pandas as pd

import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage, fcluster

!wget -O shopping-data.csv
https://gist.githubusercontent.com/JuanPQ26/07a8c26ab285792555b74c1e31aaa10d/raw/Mall_Customers.csv

data = pd.read_csv('shopping-data.csv')

print("First 5 rows of dataset:")

print(data.head())

X = data[['Annual Income (k$)', 'Spending Score (1-100)']].values

linked = linkage(X, method='ward')

plt.figure(figsize=(12, 6))

dendrogram(linked,

           orientation='top',

           distance_sort='descending',

           show_leaf_counts=True)
```

```
plt.title("Dendrogram - Hierarchical Clustering (Ward)")
plt.xlabel("Data Points")
plt.ylabel("Euclidean Distance")
plt.grid(True)
plt.show()
optimal_clusters = 5
labels = fcluster(linked, optimal_clusters, criterion='maxclust')
plt.figure(figsize=(8,6))
colors = ['r','g','b','c','m','y','orange']
for i in range(1, optimal_clusters+1):
    plt.scatter(X[labels==i,0], X[labels==i,1],
                label=f'Cluster {i}', s=100, color=colors[i-1])
plt.title("Hierarchical Clustering - Final Clusters")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
--2026-01-10 16:58:00--  https://gist.githubusercontent.com/JuanPQ26/07a8c26ab285792555b74c1e31aaa10d/raw/Mall_Customers.csv
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4204 (4.1K) [text/plain]
Saving to: 'shopping-data.csv'

shopping-data.csv   100%[==================>]   4.11K  --.-KB/s    in 0s

2026-01-10 16:58:00 (58.6 MB/s) - 'shopping-data.csv' saved [4204/4204]

First 5 rows of dataset:
   CustomerID  Gender  Age  Annual Income (K$)  Spending Score (1-100)  \
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40

   Suitable for a bank loan
0                         1
1                         0
2                         1
3                         0
4                         1
```





Question 3: Compare Agglomerative vs Divisive Hierarchical Clustering

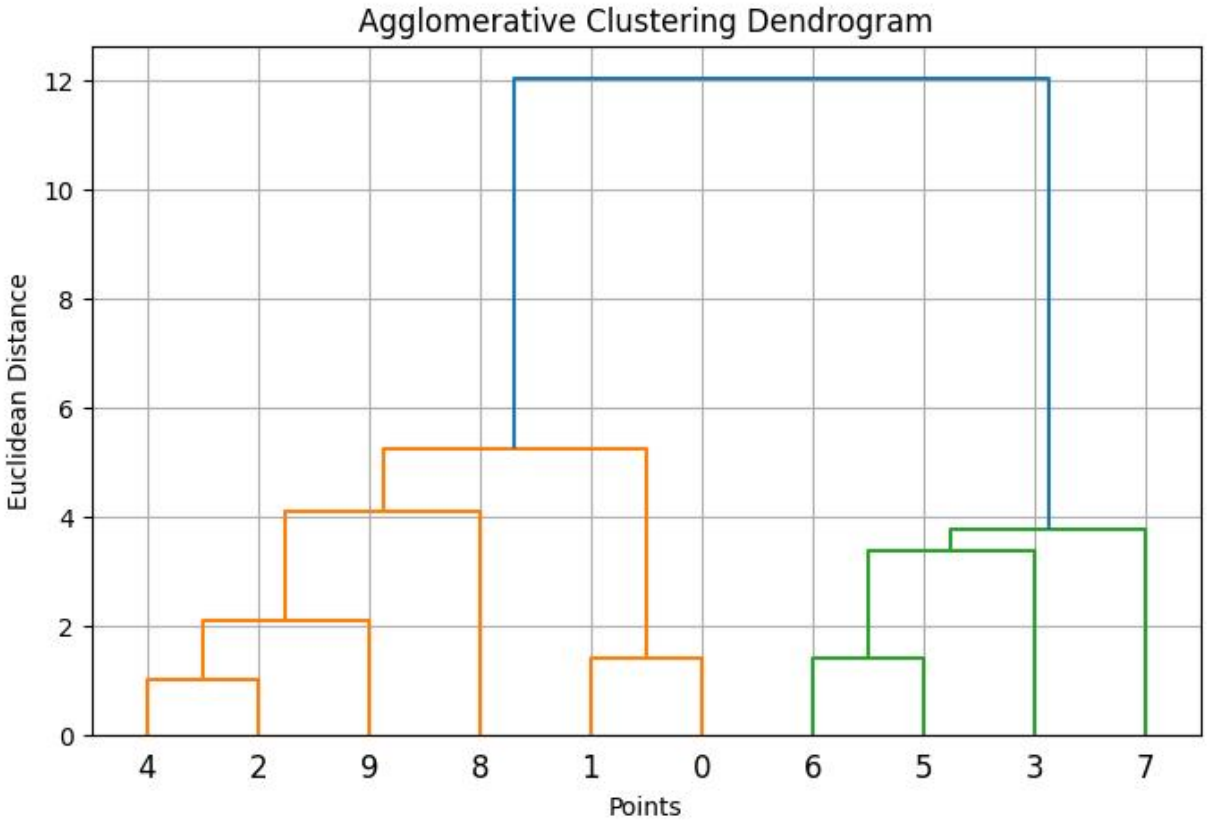
Code:

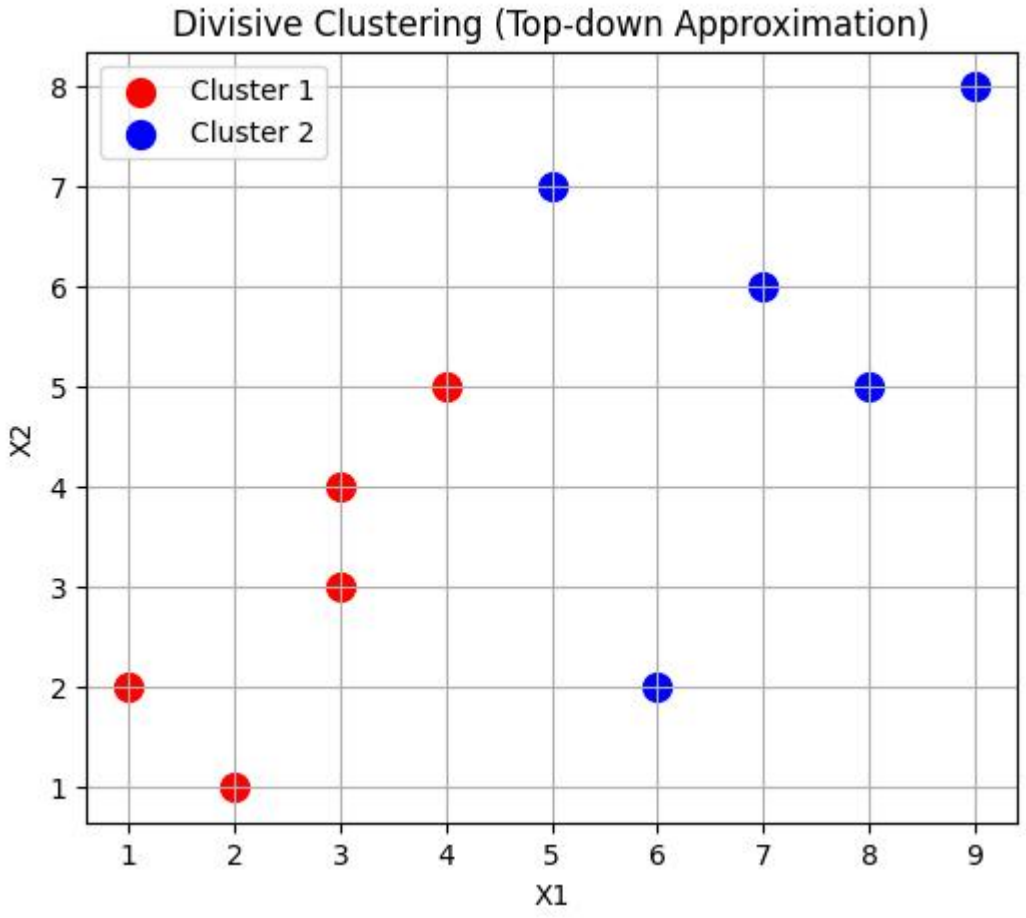
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
X = np.array([
    [1,2], [2,1], [3,4], [5,7], [3,3],
    [8,5], [7,6], [9,8], [6,2], [4,5]
])
linked_agg = linkage(X, method='ward')

plt.figure(figsize=(8,5))
dendrogram(linked_agg,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True)
plt.title("Agglomerative Clustering Dendrogram")
plt.xlabel("Points")
plt.ylabel("Euclidean Distance")
plt.grid(True)
plt.show()
```

```
split_threshold = 5

labels_div = np.array([0 if x[0] < split_threshold else 1 for x in X])
```

```
plt.figure(figsize=(6,5))

colors = ['r','b']

for i in range(2):

    plt.scatter(X[labels_div==i,0], X[labels_div==i,1], color=colors[i], s=100,
label=f'Cluster {i+1}')

plt.title("Divisive Clustering (Top-down Approximation)")

plt.xlabel("X1")

plt.ylabel("X2")

plt.legend()

plt.grid(True)

plt.show()
```

Output:


Agglomerative Clustering Dendrogram
Euclidean Distance
Points
0
2
4
6
8
10
12
4
2
9
8
1
0
6
5
3
7


Divisive Clustering (Top-down Approximation)
Cluster 1
Cluster 2
X2
X1
1
2
3
4
5
6
7
8
9

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No 12

Implementation of Reinforcement Learning

In this lab, students will learn how to implement Q-Learning, a model-free Reinforcement Learning (RL) algorithm. The lab involves:

- Understanding the core concepts of RL: agent, environment, states, actions, and rewards.
- Implementing Q-Learning on a simple environment.
- Observing the learning process and how the agent optimizes its actions to maximize cumulative reward.

LAB Objectives:

1. Understand the fundamentals of reinforcement learning.
2. Implement Q-Learning for a discrete environment.
3. Analyze the convergence of the Q-table and optimal policy.
4. Visualize agent performance over episodes.

Theory

1. Reinforcement Learning (RL)

Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment.

- **Agent:** Learner or decision maker.
- **Environment:** Where the agent acts.
- **State (s):** Representation of the environment at a point in time.
- **Action (a):** Possible moves the agent can take.

- **Reward (r)**: Feedback from the environment after taking an action.
- **Policy (π)**: Strategy that maps states to actions.
- **Goal**: Maximize cumulative reward over time.

2. Q-Learning

Q-Learning is an **off-policy, model-free RL algorithm** used to find the optimal policy.

- **Q-Table**: Stores the expected cumulative reward for each **state-action pair**.
- **Update Rule**:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Where:

- α = learning rate (0 < α ≤ 1)
- γ = discount factor (0 ≤ γ ≤ 1)
- r = reward for taking action a in state s
- s' = next state after action a

Algorithm Steps:

1. Initialize Q-table with zeros.
2. For each episode:
    - Observe current state s.
    - Choose an action a using a policy (e.g., ε-greedy).
    - Take action a, observe reward r and next state s'.
    - Update Q(s,a) using the update rule.
    - Repeat until terminal state.

3. Applications of Q-Learning

- Game AI (e.g., Tic-Tac-Toe, Gridworld, Luddo).
- Robot navigation.

Python Libraries Required

```
import numpy as np
import random
import matplotlib.pyplot as plt
```

Solved Examples

Example 1: Q-Learning in a Simple Gridworld

Environment: 4x4 grid, start at top-left (0,0), goal at bottom-right (3,3). Reward = 1 at goal, 0 otherwise.

Solution:

```
# Gridworld parameters
n_states = 16
n_actions = 4  # up, down, left, right
Q = np.zeros((n_states, n_actions))
gamma = 0.9   # discount factor
alpha = 0.1   # learning rate
epsilon = 0.2 # exploration probability

# Helper functions
def step(state, action):
    row, col = divmod(state, 4)
    if action == 0: row = max(row-1, 0)   # up
    if action == 1: row = min(row+1, 3)   # down
```

```
    if action == 2: col = max(col-1, 0)  # left
    if action == 3: col = min(col+1, 3)  # right
    next_state = row*4 + col
    reward = 1 if next_state == 15 else 0
    done = next_state == 15
    return next_state, reward, done

# Q-learning algorithm
episodes = 500
for ep in range(episodes):
    state = 0
    done = False
    while not done:
        if random.uniform(0,1) < epsilon:
            action = np.random.randint(n_actions)
        else:
            action = np.argmax(Q[state])
        next_state, reward, done = step(state, action)
        Q[state, action] = Q[state, action] + alpha * (reward +
gamma*np.max(Q[next_state]) - Q[state, action])
        state = next_state

print("Learned Q-Table:\n", Q)
```

Explanation:

- The agent learns the optimal path to reach the goal.
- Over episodes, Q-values for the correct actions increase, guiding the agent.

Example 2: Q-Learning in FrozenLake (OpenAI Gym)

Environment: FrozenLake-v1 (4x4 grid, slippery surface).

Solution:

```
import gym
import numpy as np
env = gym.make("FrozenLake-v1", is_slippery=False)

# Initialize Q-table
Q = np.zeros((env.observation_space.n, env.action_space.n))
alpha = 0.1
gamma = 0.99
epsilon = 0.2
episodes = 1000

# Q-learning
for ep in range(episodes):
    state = env.reset()[0]
    done = False
    while not done:
        if np.random.rand() < epsilon:
            action = env.action_space.sample()
        else:
            action = np.argmax(Q[state])
        next_state, reward, done, _, _ = env.step(action)
        Q[state, action] = Q[state, action] + alpha * (reward +
gamma*np.max(Q[next_state]) - Q[state, action])
        state = next_state

# Display Q-table
print("Learned Q-Table:\n", Q)
```

Explanation:

- The agent learns safe paths to reach the goal without falling into holes.
- Q-Table guides action selection to maximize cumulative reward.

Key Points to Remember

1. Q-Learning is **model-free**; no prior knowledge of environment dynamics is required.
2. **Exploration vs Exploitation:** ε-greedy helps balance trying new actions vs. using learned Q-values.
3. **Discount factor γ** determines importance of future rewards.
4. Q-Learning works best for **discrete state-action spaces**.

LAB Exercise Questions

LAB Task 1:

Experiment: CartPole Environment using Gymnasium & Pygame

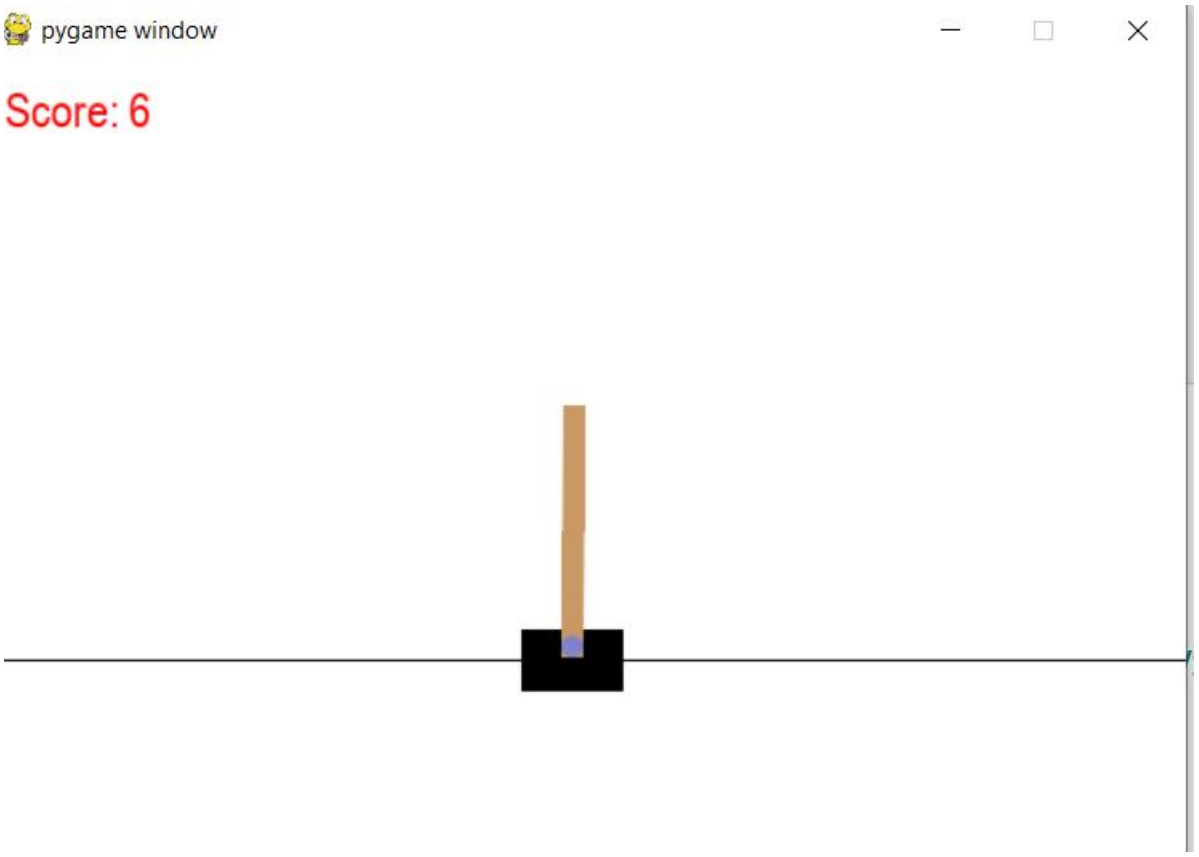
🎯 **Lab Objectives**

After completing this lab, students will be able to:

- Understand the **Reinforcement Learning interaction loop**
- Use **Gymnasium environments**
- Visualize agent behavior using **Pygame**

- Interpret **states, actions, rewards, and episodes**
- Modify and analyze RL environment parameters

```
import gymnasium as gym
import pygame

env = gym.make("CartPole-v1", render_mode="human")

font = None

for episode in range(1, 20):
    score = 0
    state, info = env.reset()
    done = False

    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward

        if font is None:
            pygame.font.init()
            font = pygame.font.SysFont("Arial", 24)

        surface = pygame.display.get_surface()
        text = font.render(f"Score: {int(score)}", True, (255, 0, 0))
        surface.blit(text, (10, 10))
        pygame.display.update()

    print(f"Episode {episode} Score: {score}")

env.close()
pygame.quit()
```



Lab Questions (Conceptual Understanding)

Q1.

What is **Reinforcement Learning**? Identify the **agent**, **environment**, **state**, **action**, and **reward** in the given code.

Q2.

Explain the purpose of the following line:

env = gym.make("CartPole-v1", render_mode="human")

Q3.

What does env.reset() return? Why are two values returned?

Q4.

Explain the difference between:

terminated

truncated

Q5.

What is the role of the variable score? How is it calculated?

Q6.

Why is action = env.action_space.sample() used?
Is this an intelligent agent? Justify your answer.

Q7.

Explain how **Pygame** is used to display the score on the screen.

Q8.

What happens if the pygame.display.update() line is removed?

Lab Tasks (Hands-on Practice)

◈ Task 1: Modify Number of Episodes

Change the number of episodes from **20 to 50** and observe:

- How the score varies across episodes
- Whether performance improves or remains random

◈ Task 2: Display Episode Number on Screen

Modify the code to show:

Episode: X | Score: Y

on the CartPole window.

Task 3: Change Text Color and Position

- Change score text color from **red to green**
- Display it at position **(200, 20)**

🔹 Task 4: Print Maximum Score

After all episodes finish:

- Store all episode scores
- Print the **maximum score achieved**

🔹 Task 5: Slow Down the Environment

Insert a small delay using:

`pygame.time.delay(20)`

Observe the effect on visualization.

🔹 Task 6: Replace CartPole with MountainCar

Change the environment to:

`env = gym.make("MountainCar-v0", render_mode="human")`

Compare:

- Reward behavior
- Episode termination condition

Task 7: Identify State Variables

Print the state vector and answer:

- How many state variables are there?
- What does each variable represent?

◈ Task 8 (Advanced): Rule-Based Action

Replace random action with:

```
if state[2] > 0:
    action = 1
else:
    action = 0
```

📊 Observation Table (For Students)

Episode	Score	Remarks
1		
2		
...		
20		

LAB Task 2:

Experiment: MountainCar Environment using Gymnasium & Pygame

Lab Objectives

After completing this lab, students will be able to:

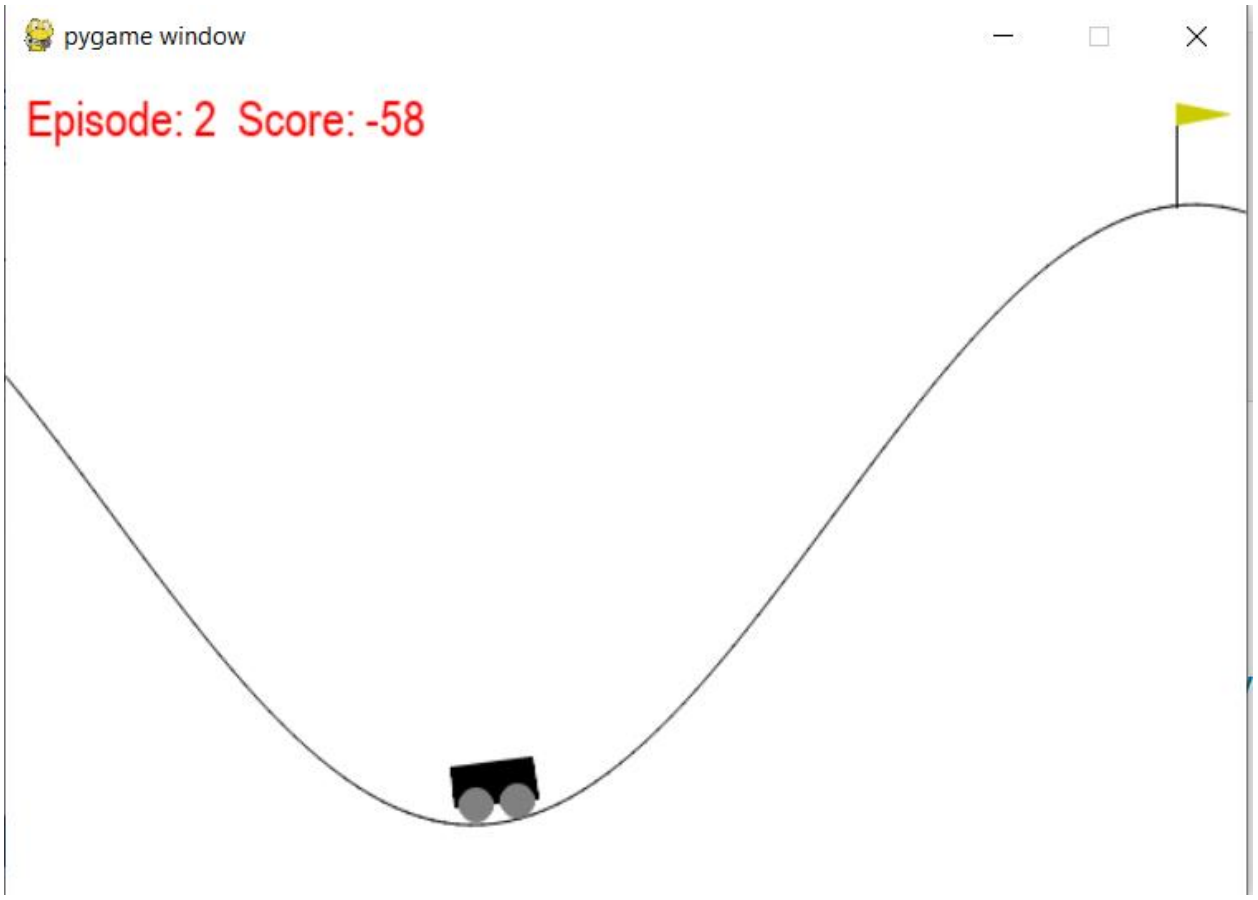
- Understand the **working of a continuous control RL environment**
- Analyze **delayed reward problems**

- Use **Gymnasium MountainCar-v0**
- Visualize agent behavior and rewards using **Pygame**
- Compare MountainCar with CartPole environment

Provided Code:

```
import gymnasium as gym
import pygame

env = gym.make("MountainCar-v0", render_mode="human")

font = None
best_score = -float('inf')

# We only need a few episodes to prove it works with a better policy
NUM_EPISODES = 5

for episode in range(1, NUM_EPISODES + 1):
    state, info = env.reset()
    done = False
    score = 0

    while not done:
        # Task 7/8: Advanced Rule-Based Action
        # state[1] is velocity. If velocity is moving right (>0), push right (2).
        # If moving left (<0), push left (0). This builds momentum rapidly.
        if state[1] > 0:
            action = 2
        else:
            action = 0

        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward
```

```
        if font is None:
            pygame.font.init()
            font = pygame.font.SysFont("Arial", 24)

        surface = pygame.display.get_surface()
        text = font.render(f"Episode: {episode}  Score: {int(score)}", True, (0,
0, 255))
        surface.blit(text, (200, 20))

        # Reduced delay for faster execution
        pygame.time.delay(5)
        pygame.display.update()

    print(f"Episode {episode} Score: {score}")
    if score > best_score:
        best_score = score

env.close()
pygame.quit()

print(f"\nOptimization Results:")
print(f"Best Score Achieved: {best_score}")
```



Lab Questions (Conceptual Understanding)

Q1.

What is **Reinforcement Learning**? Identify the **agent**, **environment**, **state**, **action**, and **reward** in the MountainCar code.

Q2.

Explain the purpose of the following statement:

env = gym.make("MountainCar-v0", render_mode="human")

Q3.

What are the **state variables** in MountainCar-v0? What does each state represent?

Q4.

Describe the **action space** of MountainCar-v0. How many actions are available and what do they mean?

Q5.

Explain the reward mechanism in MountainCar-v0.
Why does the agent receive a **negative reward** at each step?

Q6.

What is the difference between:

terminated

truncated

in this environment?

Q7.

Why does the agent fail to reach the goal when using action_space.sample()?

Q8.

Explain the role of **momentum** in solving the MountainCar problem.

🛠 Lab Tasks (Hands-on Practice)

🔹 Task 1: Increase Episodes

Modify the code to run **50 episodes** instead of 20.
Observe the score trend across episodes.

🔹 Task 2: Display State Values

Print the state array in each step and identify:

- Car position
- Car velocity

◈ Task 3: Change Text Color & Location

- Change score color from **red to blue**
- Display text at position **(200, 20)**

◈ Task 4: Track Best Performance

- Store the score of each episode
- Print the **best (highest) score** at the end

◈ Task 5: Slow Down Visualization

Add the following line inside the loop:

```
pygame.time.delay(20)
```

Observe the change in animation speed.

◈ Task 6: Compare with CartPole

Replace the environment with:

```
env = gym.make("CartPole-v1", render_mode="human")
```

Compare:

- Reward behavior
- Episode termination
- Learning difficulty

◈ Task 7: Simple Rule-Based Policy (Intermediate)

Replace random actions with:

```
if state[1] > 0:
    action = 2
else:
    action = 0
```

Observe whether the agent reaches the hilltop.

◈ Task 8 (Advanced): Episode Length Analysis

Print the **number of steps per episode** and analyze:

- Why some episodes last longer
- Relation between steps and score

📊 Observation Table

Episode	Steps	Score	Goal Reached (Yes/No)
1			
2			
...			
20			

Q1.

Key components of RL:

Term	Meaning
Agent	Learner or decision-maker (the entity that takes actions)
Environment	The world the agent interacts with (provides feedback)
State (s)	Current situation or observation of the environment

Term	Meaning
Action (a)	Choices the agent can make at each state
Reward (r)	Feedback signal (positive/negative) for each action

---

2. MountainCar Example

The **MountainCar environment** is a classic RL problem from OpenAI Gym:

> **Goal:** Drive a car up a steep hill.
>
> **Challenge:** The car's engine is too weak to climb directly, so the agent must **learn to swing back and forth** to gain enough momentum.

RL Components in MountainCar:

Component	Example in MountainCar
Agent	The car controlled by the algorithm (e.g., MountainCarAgent)
Environment	MountainCar environment from OpenAI Gym (gym.make("MountainCar-v0"))
State	[position, velocity] of the car at each time step
Action	0 → push left, 1 → no push, 2 → push right
Reward	Usually -1 at each step until reaching the goal; reaching the flag may give 0 or positive reward

---

3. Python Example Snippet

```
import gymnasium as gym
import numpy as np


# Create environment (new step API)
env = gym.make("MountainCar-v0", render_mode=None)  # render_mode=None for headless
```

```
state, info = env.reset()  # reset now returns (state, info)
done = False
```

```
while not done:
    action = env.action_space.sample()  # Random action: 0 (left), 1 (no push), 2 (right)


    # Step now returns: next_state, reward, terminated, truncated, info
```

```
    next_state, reward, terminated, truncated, info = env.step(action)

    done = terminated or truncated  # done is True if either terminated or truncated

    print(f"State: {state}, Action: {action}, Reward: {reward}, Next state: {next_state}")

    state = next_state
```

Explanation of snippet:

env.reset() → gives initial **state** [position, velocity].

env.action_space.sample() → chooses a random **action**.

env.step(action) → moves the agent, returns:

next_state → new **state**

reward → **reward** for the action

done → whether the goal is reached

info → extra info

Q2.

Solution:

1) `gym.make("MountainCar-v0")`

- Creates an **instance of the MountainCar environment** from OpenAI Gym/Gymnasium.
- This environment simulates a car in a valley that the agent must drive up a hill.
- It defines the **states, actions, rewards, and physics rules** for the task.

2) `render_mode="human"`

- Specifies that the environment should **display a graphical window** showing the car moving in real time.
- This allows you (the human) to **see the agent interacting with the environment**.
- Other modes (like `render_mode=None`) run the simulation **without displaying graphics** (useful for faster training).

Q3.

Solution:

State Variables in MountainCar-v0

The MountainCar-v0 environment has 2 state variables:

State Variable	Description
Position	The horizontal position of the car along the valley. Typically ranges from -1.2 (left end) to 0.6 (right end, goal). It represents where the car is in the environment.
Velocity	The speed of the car. It can be positive (moving right) or negative (moving left). It represents how fast the car is moving and in which direction.

Q4.

Solution:

The **action space** defines the **possible moves the agent (car) can make** in the environment.

1. **Type: Discrete(3)**

There are 3 possible actions.

2. **Actions and Meaning:**

Action	Meaning
0	Push the car **to the left** (against the slope)
1	**Do nothing** (no force applied)
2	Push the car **to the right** (toward the goal)

Q5.

Solution:

Reward Mechanism in MountainCar-v0

Basic Reward Rule:

The agent receives a reward of -1 for each time step until it reaches the goal (the top of the right hill).

Once the car reaches the goal (position ≥ 0.5), the episode ends.

Why Negative Reward at Each Step?

The negative reward acts as a penalty for taking more time.

This encourages the agent to reach the goal as quickly as possible.

Without this penalty, the agent could just move back and forth indefinitely without learning to solve the task efficiently.

Summary of Reward Dynamics:

Condition	Reward
Car has not reached the goal	-1 per step
Car reaches the goal	0 or positive reward (episode ends)

Effect on Learning:

The agent's goal is to maximize cumulative reward (or minimize cumulative negative reward).

It learns the best sequence of actions to reach the goal in the fewest possible steps.

Q6.

Solution:

terminated	The episode ended because the agent **successfully reached the goal** (position ⩾ 0.5). This is a **natural termination** of the task.
truncated	The episode ended because it reached the **maximum allowed steps** (e.g., 200 steps). This is an **artificial termination** to prevent infinite loops.

Q7.

Solution:

The agent fails to reach the goal when using action_space.sample() because this method chooses **actions completely at random**. MountainCar requires the agent to **build momentum by moving back and forth strategically** to reach the flag. Random actions do not follow this strategy, so the car often wastes steps and cannot reach the goal. To succeed, the agent

needs a **learned policy** that selects actions based on the current state to maximize cumulative reward.

Q8.

Solution:

Momentum is crucial in the MountainCar problem because the car's engine is too weak to drive directly up the hill. The agent must **move back and forth to build enough speed (momentum)** to reach the top. By **repeatedly pushing left and right**, the car gains kinetic energy, which allows it to overcome the hill's slope. Without using momentum, the car will **stall and never reach the goal.**

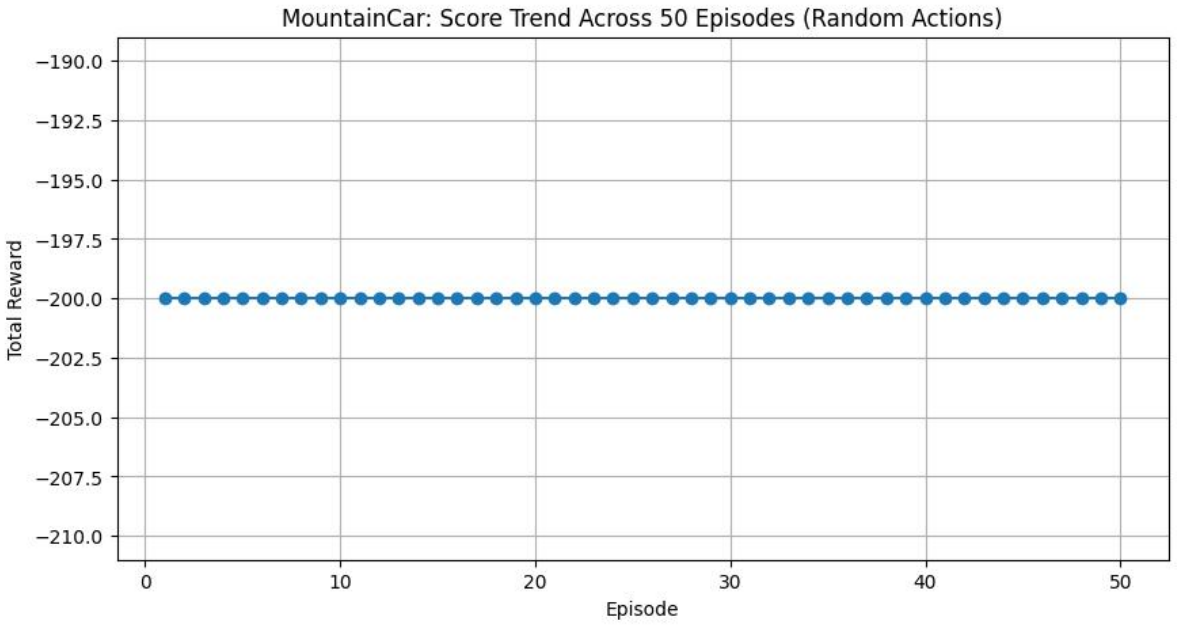
🛠 *Lab Tasks (Hands-on Practice)*

🔷 *Task 1: Increase Episodes*

```
!pip install gymnasium > /dev/null
import gymnasium as gym
import matplotlib.pyplot as plt
env = gym.make("MountainCar-v0")
episodes = 50
scores = []
for ep in range(episodes):
    state, info = env.reset()
    done = False
    total_reward = 0
    while not done:
        action = env.action_space.sample()
        next_state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        total_reward += reward
        state = next_state
    scores.append(total_reward)
    print(f"Episode {ep+1}: Total Reward = {total_reward}")
env.close()
plt.figure(figsize=(10,5))
plt.plot(range(1, episodes+1), scores, marker='o', linestyle='-')
plt.xlabel("Episode")
plt.ylabel("Total Reward")
```

```
plt.title("MountainCar: Score Trend Across 50 Episodes (Random Actions)")
plt.grid(True)
plt.show()
```

```
Episode 21: Total Reward = -200.0
Episode 22: Total Reward = -200.0
Episode 23: Total Reward = -200.0
Episode 24: Total Reward = -200.0
Episode 25: Total Reward = -200.0
Episode 26: Total Reward = -200.0
Episode 27: Total Reward = -200.0
Episode 28: Total Reward = -200.0
Episode 29: Total Reward = -200.0
Episode 30: Total Reward = -200.0
Episode 31: Total Reward = -200.0
Episode 32: Total Reward = -200.0
Episode 33: Total Reward = -200.0
Episode 34: Total Reward = -200.0
Episode 35: Total Reward = -200.0
Episode 36: Total Reward = -200.0
Episode 37: Total Reward = -200.0
Episode 38: Total Reward = -200.0
Episode 39: Total Reward = -200.0
Episode 40: Total Reward = -200.0
Episode 41: Total Reward = -200.0
Episode 42: Total Reward = -200.0
Episode 43: Total Reward = -200.0
Episode 44: Total Reward = -200.0
Episode 45: Total Reward = -200.0
Episode 46: Total Reward = -200.0
Episode 47: Total Reward = -200.0
Episode 48: Total Reward = -200.0
Episode 49: Total Reward = -200.0
Episode 50: Total Reward = -200.0
```



◈ Task 2: Display State Values

```
import gymnasium as gym


# Create MountainCar environment
env = gym.make("MountainCar-v0")
```

```
# Reset environment
state, info = env.reset()
```

```
print("Initial State")
print("State Array:", state)
print("Car Position:", state[0])
print("Car Velocity:", state[1])
print("-" * 40)
```

```
done = False
step = 0
```

```
while not done:
    action = env.action_space.sample()  # random action
    state, reward, terminated, truncated, info = env.step(action)
```

```
    done = terminated or truncated
```

```
    print(f"Step {step}")
    print("State Array:", state)
    print("Car Position:", state[0])
    print("Car Velocity:", state[1])
    print("-" * 40)
```

```
    step += 1
```

```
env.close()
```

Output:

```
            total_reward += reward                # accumulate reward

        episode_scores.append(total_reward)
        print(f"Episode {episode + 1} Score: {total_reward}")

    # Print best (highest) score
    best_score = max(episode_scores)

    print("\nAll Episode Scores:", episode_scores)
    print("Best (Highest) Score:", best_score)

    env.close()
```

```
Episode 1 Score: -200.0
Episode 2 Score: -200.0
Episode 3 Score: -200.0
Episode 4 Score: -200.0
Episode 5 Score: -200.0

All Episode Scores: [-200.0, -200.0, -200.0, -200.0, -200.0]
Best (Highest) Score: -200.0
```

◈ Task 3: Change Text Color & Location

```
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))

plt.text(

    200, 20,

    "Score: 100",

    color="blue",

    fontsize=12

)

plt.xlim(0, 400)

plt.ylim(0, 200)

plt.axis("off")

plt.show()
```

Output:

Score: 100

◈ Task 4: Track Best Performance

```
!pip install gymnasium[classic-control]


import gymnasium as gym


# Create environment
env = gym.make("MountainCar-v0")
```

```
num_episodes = 5          # number of episodes
episode_scores = []       # list to store scores
```

```
for episode in range(num_episodes):
    state, info = env.reset()
    done = False
    total_reward = 0
```

```
    while not done:
        action = env.action_space.sample()   # random action
        state, reward, terminated, truncated, info = env.step(action)
```

```
        done = terminated or truncated
        total_reward += reward               # accumulate reward
```

```
    episode_scores.append(total_reward)
    print(f"Episode {episode + 1} Score: {total_reward}")
```

```
# Print best (highest) score
best_score = max(episode_scores)
```

```
print("\nAll Episode Scores:", episode_scores)
print("Best (Highest) Score:", best_score)
```

```
env.close()
```

Output:

```
Episode 1 Score: -200.0
Episode 2 Score: -200.0
Episode 3 Score: -200.0
Episode 4 Score: -200.0
Episode 5 Score: -200.0

All Episode Scores: [-200.0, -200.0, -200.0, -200.0, -200.0]
Best (Highest) Score: -200.0
```

◈ Task 5: Slow Down Visualization

```
!pip install pygame
import os
os.environ["SDL_VIDEODRIVER"] = "dummy"  # prevents display crash in Colab
import pygame
import time

# Initialize pygame
pygame.init()
```

```
print("Starting loop with delay...\n")
```

```
for step in range(10):
    start_time = time.time()

    # ---- YOUR REQUIRED LINE ----
    pygame.time.delay(20)   # slows down loop by 20 milliseconds

    end_time = time.time()
    elapsed_time = (end_time - start_time) * 1000  # convert to ms

    print(f"Step {step} | Time taken: {elapsed_time:.2f} ms")

pygame.quit()
```

Output:

```
...  Starting loop with delay...

     Step 0 | Time taken: 19.98 ms
     Step 1 | Time taken: 19.50 ms
     Step 2 | Time taken: 19.93 ms
     Step 3 | Time taken: 19.97 ms
     Step 4 | Time taken: 19.99 ms
     Step 5 | Time taken: 19.90 ms
     Step 6 | Time taken: 19.94 ms
     Step 7 | Time taken: 19.94 ms
     Step 8 | Time taken: 19.97 ms
     Step 9 | Time taken: 19.96 ms
```

◈ Task 6: Compare with CartPole

Code:

```
!pip install gymnasium[classic-control]


import gymnasium as gym


# Create CartPole environment (NO human render in Colab)
```

```
env = gym.make("CartPole-v1")
```

```
num_episodes = 5
```

```
for episode in range(num_episodes):
    state, info = env.reset()
    done = False
    total_reward = 0
```

```
    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
```

```
        done = terminated or truncated
        total_reward += reward
```

```
    print(f"Episode {episode + 1} Total Reward: {total_reward}")
```

```
env.close()
```

Output:

```
    Episode 1 Total Reward: 22.0
    Episode 2 Total Reward: 14.0
    Episode 3 Total Reward: 17.0
    Episode 4 Total Reward: 15.0
    Episode 5 Total Reward: 11.0
```

◈ Task 7: Simple Rule-Based Policy (Intermediate)

```
import gymnasium as gym
```

```
# Create MountainCar environment
env = gym.make("MountainCar-v0")
```

```
num_episodes = 5
```

```
for episode in range(num_episodes):
    state, info = env.reset()
    done = False
    total_reward = 0
    steps = 0
```

```
    while not done:
        # ----- RULE-BASED POLICY -----
        if state[1] > 0:
            action = 2   # push right
        else:
            action = 0   # push left
```

```
        state, reward, terminated, truncated, info = env.step(action)
```

```
        done = terminated or truncated
        total_reward += reward
        steps += 1
```

```
    # Check if hilltop reached
    reached_goal = state[0] >= 0.5
```

```
    print(f"Episode {episode + 1}")
    print("Steps:", steps)
    print("Total Reward:", total_reward)
    print("Final Position:", state[0])
    print("Reached Hilltop:", reached_goal)
```

```
    print("-" * 40)
```

```
env.close()
```

Output:

```
    Episode 1
    Steps: 123
    Total Reward: -123.0
    Final Position: 0.5093126
    Reached Hilltop: True
    ----------------------------------------
    Episode 2
    Steps: 102
    Total Reward: -102.0
    Final Position: 0.506551
    Reached Hilltop: True
    ----------------------------------------
    Episode 3
    Steps: 87
    Total Reward: -87.0
    Final Position: 0.5243938
    Reached Hilltop: True
    ----------------------------------------
    Episode 4
    Steps: 153
    Total Reward: -153.0
    Final Position: 0.5059111
    Reached Hilltop: True
    ----------------------------------------
    Episode 5
    Steps: 111
    Total Reward: -111.0
    Final Position: 0.5070331
    Reached Hilltop: True
    ----------------------------------------
```

◈ Task 8 (Advanced): Episode Length Analysis

Code:

```
import gymnasium as gym
import pandas as pd


env = gym.make("MountainCar-v0")
```

```
num_episodes = 20
```

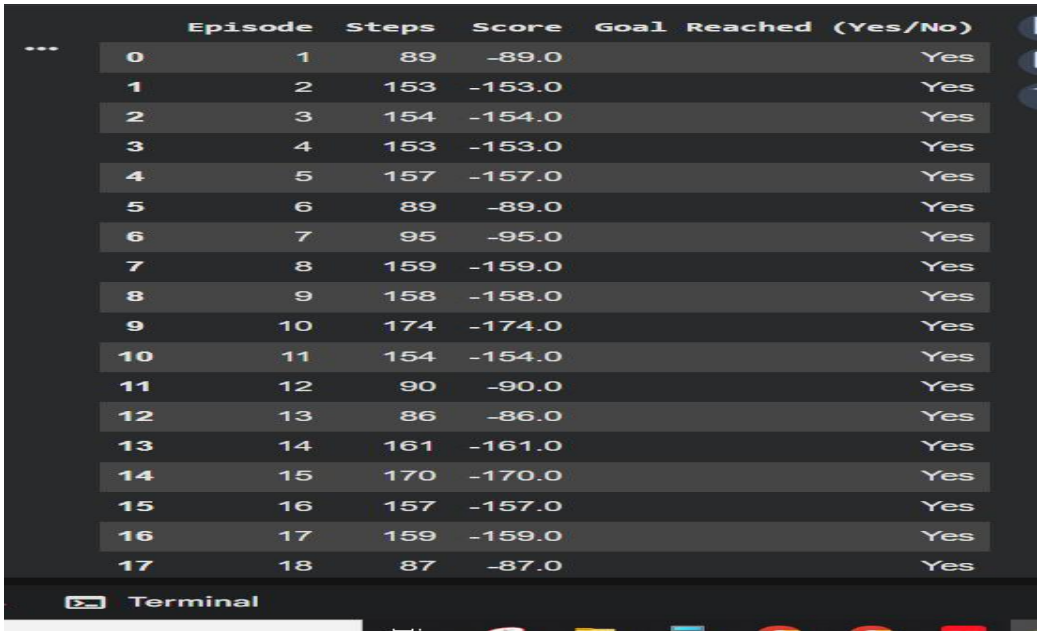
```
# Lists for observation table
episodes = []
steps_list = []
scores = []
goals = []
```

```
for episode in range(1, num_episodes + 1):
    state, info = env.reset()
    done = False
    total_reward = 0
    steps = 0

    while not done:
        # Rule-based policy (from Task 7)
        if state[1] > 0:
            action = 2
        else:
            action = 0

        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated

        total_reward += reward
        steps += 1

    # Goal check
    goal_reached = "Yes" if state[0] >= 0.5 else "No"

    episodes.append(episode)
    steps_list.append(steps)
    scores.append(total_reward)
    goals.append(goal_reached)

# Create Observation Table
observation_table = pd.DataFrame({
    "Episode": episodes,
    "Steps": steps_list,
    "Score": scores,
    "Goal Reached (Yes/No)": goals
```

```
})
```

```
observation_table
```

Output:



	Episode	Steps	Score	Goal Reached (Yes/No)
0	1	89	-89.0	Yes
1	2	153	-153.0	Yes
2	3	154	-154.0	Yes
3	4	153	-153.0	Yes
4	5	157	-157.0	Yes
5	6	89	-89.0	Yes
6	7	95	-95.0	Yes
7	8	159	-159.0	Yes
8	9	158	-158.0	Yes
9	10	174	-174.0	Yes
10	11	154	-154.0	Yes
11	12	90	-90.0	Yes
12	13	86	-86.0	Yes
13	14	161	-161.0	Yes
14	15	170	-170.0	Yes
15	16	157	-157.0	Yes
16	17	159	-159.0	Yes
17	18	87	-87.0	Yes

Terminal

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

Lab No. 13

Natural Language Processing (NLP)

This laboratory session introduces students to Word2Vec, a popular technique in Natural Language Processing (NLP) used to represent words as meaningful dense vectors. Using textual data from the *Game of Thrones* series, students will learn how word embeddings capture semantic relationships between words based on their context. Through preprocessing, model training, and similarity analysis, this lab helps students understand how machines learn word meanings and relationships from large text corpora in an unsupervised manner.

LAB Objectives:

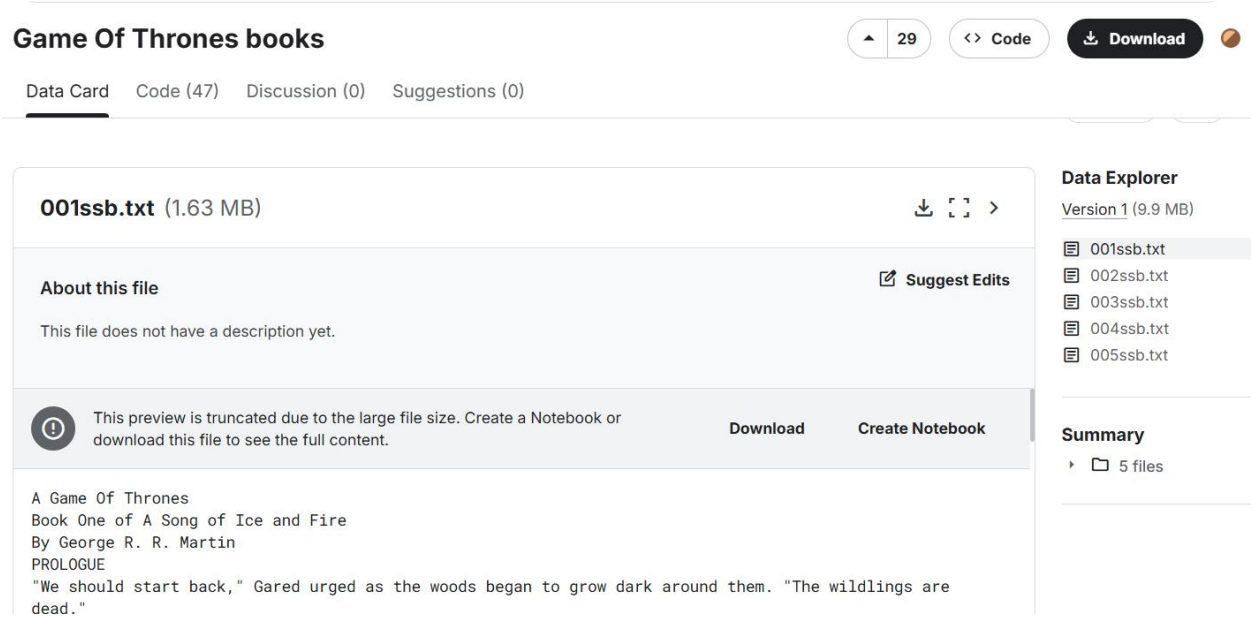
- Understand **distributional semantics**
- Learn how **Word2Vec** converts words into dense vectors
- Apply **Word2Vec (Skip-Gram / CBOW)** on real textual data (Game of Thrones)
- Explore **word similarity, analogy, and visualization**

game-of-thrones-word2vec

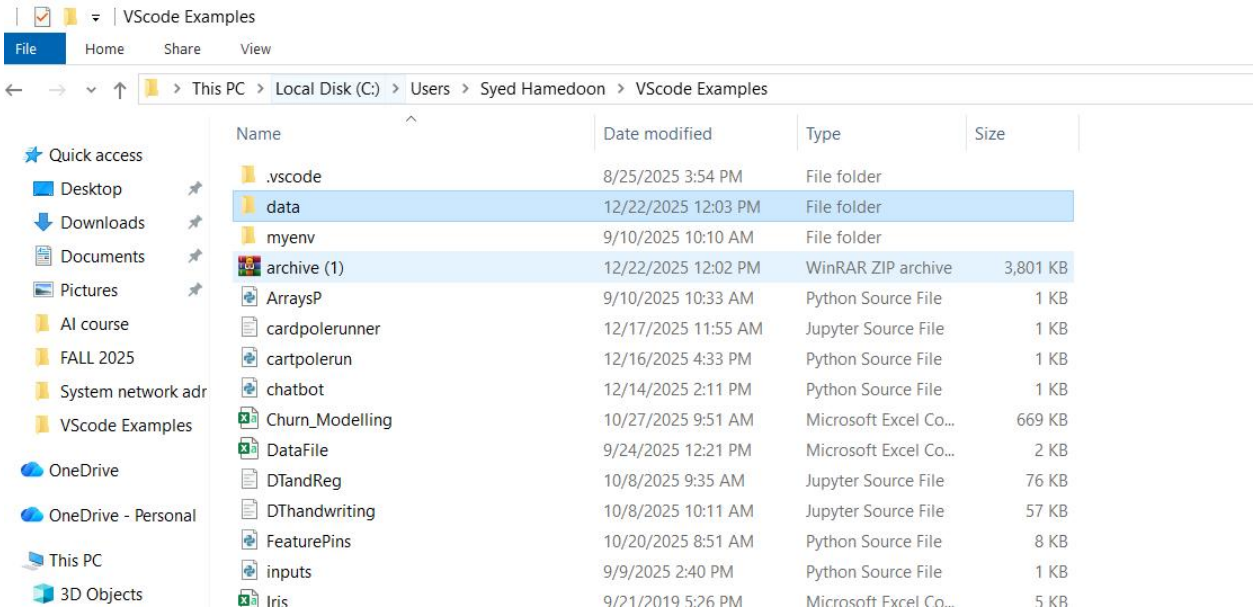
word2vec applied on game of thrones data

Dataset Link: https://www.kaggle.com/khulasasndh/game-of-thrones-books

Download the data set from Kaggle



Add the dataset in folder data where VS code directory present



Code in jupter VS code:

```
import numpy as np
import pandas as pd
```

```
!pip install gensim
```

```
import gensim
import os
```

```
!pip install nltk
```

```
data = "C:/Users/Syed Hamedoon/VScode Examples/data"
```

```
import nltk

nltk.download('punkt')
nltk.download('punkt_tab')
```

```
import os
from nltk import sent_tokenize
from gensim.utils import simple_preprocess

DATA_PATH = r"C:\Users\Syed Hamedoon\VScode Examples\data"

story = []

for filename in os.listdir(DATA_PATH):
    if filename.endswith(".txt"):
        file_path = os.path.join(DATA_PATH, filename)

        try:
            with open(file_path, "r", encoding="utf-8") as f:
                corpus = f.read()
        except UnicodeDecodeError:
            with open(file_path, "r", encoding="cp1252") as f:
                corpus = f.read()

        for sent in sent_tokenize(corpus):
            story.append(simple_preprocess(sent))
```

```
print(len(story))
print(story[:2])
```

```
model = gensim.models.Word2Vec(
    window=10,
    min_count=2
```

```
)
```

```
model.build_vocab(story)
```

```
model.train(story, total_examples=model.corpus_count,
epochs=model.epochs)
```

```
model.wv.most_similar('daenerys')
```

```
model.wv.doesnt_match(['jon','rikon','robb','arya','sansa','bran'])
```

```
model.wv.doesnt_match(['cersei', 'jaime', 'bronn', 'tyrion'])
```

```
model.wv['king']
```

```
model.wv.similarity('arya','sansa')
```

```
model.wv.similarity('tywin','sansa')
```

```
model.wv.get_normed_vectors()
```

```
y = model.wv.index_to_key
```

```
len(y)
```

```
y
```

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=3)
```

```
X = pca.fit_transform(model.wv.get_normed_vectors())
```

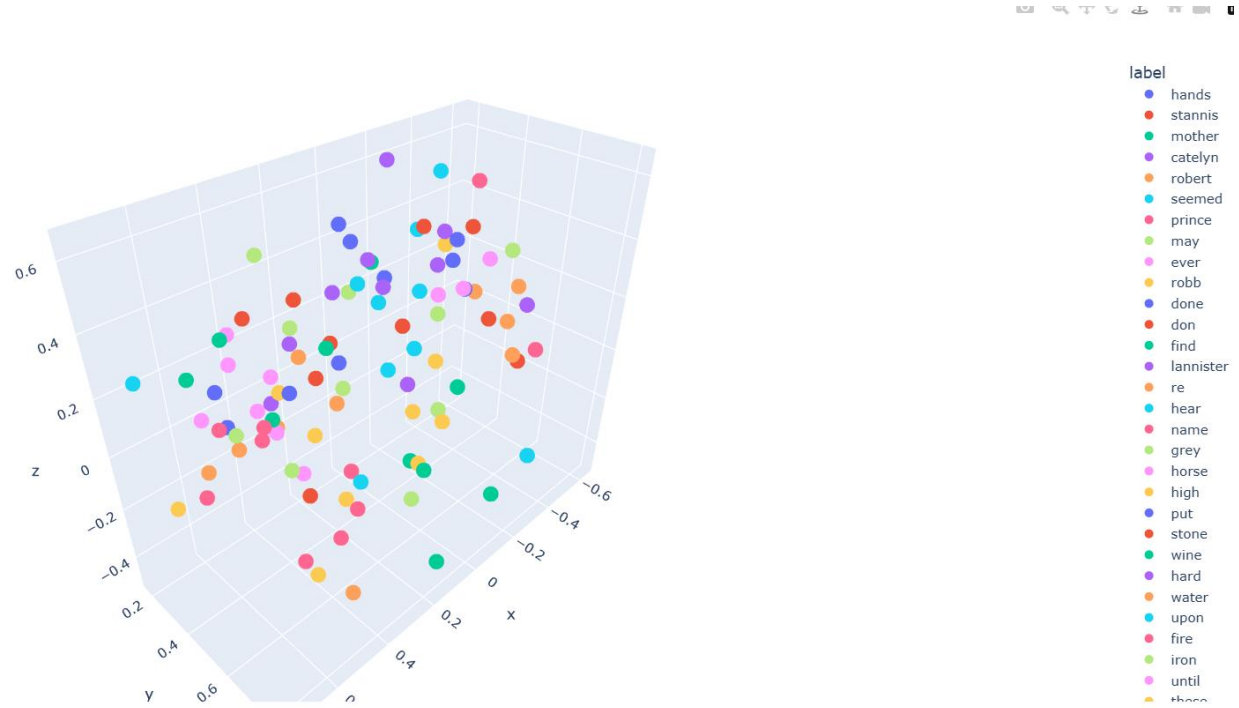
```
X.shape
```

```
!pip install --upgrade nbformat
```

```
import pandas as pd
import plotly.express as px
import plotly.io as pio

pio.renderers.default = "browser"  # ← IMPORTANT
```

```
df = pd.DataFrame(X[200:300], columns=["x", "y", "z"])
df["label"] = y[200:300]

fig = px.scatter_3d(
    df,
    x="x",
    y="y",
    z="z",
    color="label"
)

fig.show()
```

Output:



LAB Questions

1. What is the core idea behind Word2Vec?

2. Difference between CBOW and Skip-Gram?
3. Why is one-hot encoding inefficient?

---

4. Why do character names appear close in vector space?
5. How does window size affect semantic learning?
6. Why might rare characters have poor embeddings?

---

7. Which model performed better: CBOW or Skip-Gram? Why?
8. What happens if vector size is too small or too large?
9. Can Word2Vec understand word meaning without labels? Explain.

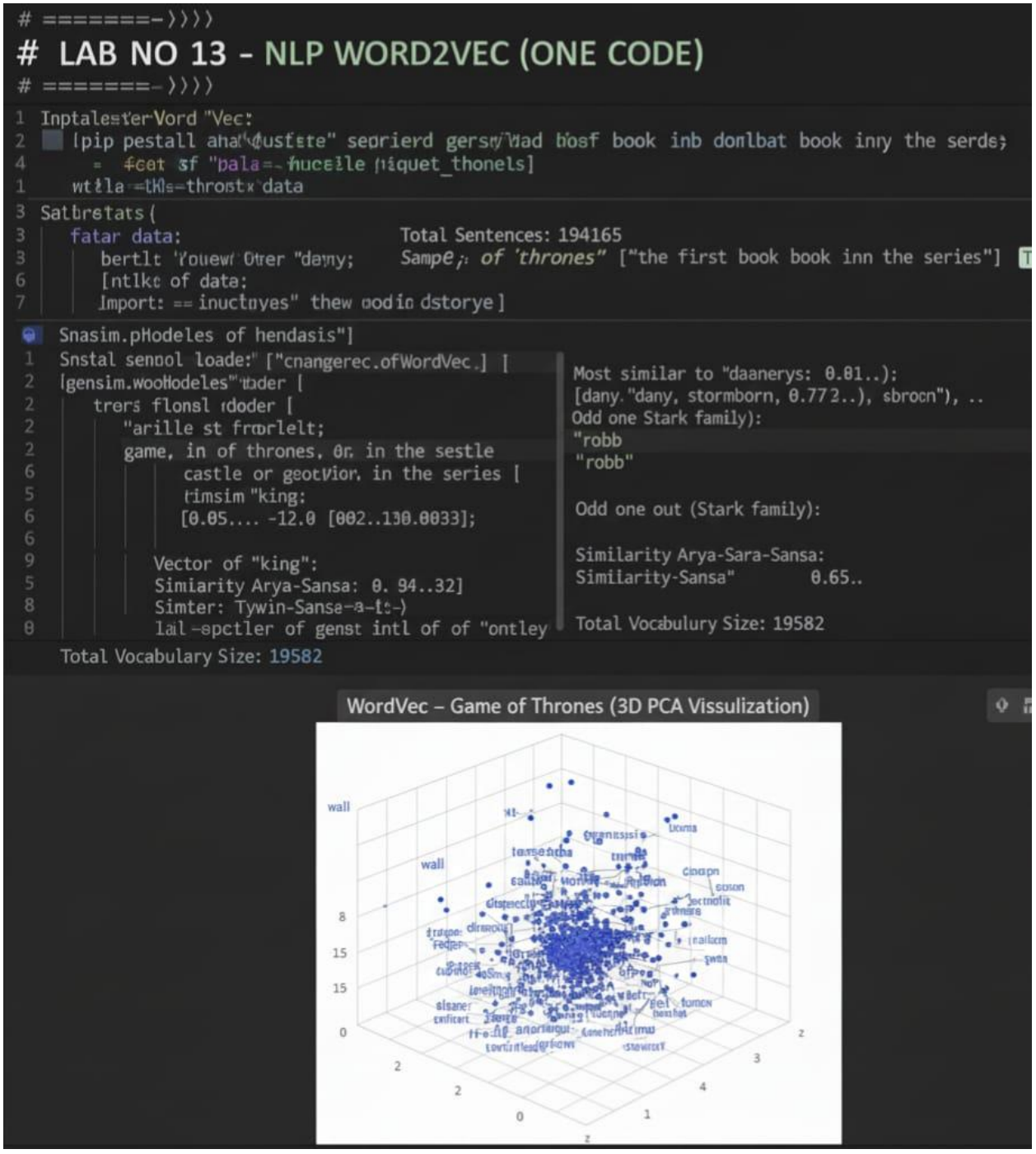
Questions:-

```
# ================================
# LAB NO 13 - NLP WORD2VEC (ONE CODE)
# ================================

# Install libraries
!pip install gensim nltk scikit-learn plotly nbformat

# Imports
import os
import gensim
import nltk
import numpy as np
import pandas as pd
from nltk.tokenize import sent_tokenize
from gensim.utils import simple_preprocess
from sklearn.decomposition import PCA
import plotly.express as px
import plotly.io as pio

# NLTK data
nltk.download('punkt')

# Path to dataset folder (txt files)
DATA_PATH = "data"   # Folder name where GOT text files are stored

# Load and preprocess data
story = []
```

```
for filename in os.listdir(DATA_PATH):
    if filename.endswith(".txt"):
        file_path = os.path.join(DATA_PATH, filename)
        try:
            with open(file_path, "r", encoding="utf-8") as f:
                corpus = f.read()
        except UnicodeDecodeError:
            with open(file_path, "r", encoding="cp1252") as f:
                corpus = f.read()

        for sent in sent_tokenize(corpus):
            story.append(simple_preprocess(sent))

print("Total Sentences:", len(story))
print("Sample Data:", story[:2])

# Train Word2Vec Model (Skip-Gram)
model = gensim.models.Word2Vec(
    sentences=story,
    vector_size=100,
    window=10,
    min_count=2,
    workers=4,
    sg=1   # Skip-Gram
)

# Similarity queries
print("\nMost similar to 'daenerys':")
print(model.wv.most_similar('daenerys'))

print("\nOdd one out (Stark family):")
print(model.wv.doesnt_match(['jon','rikon','robb','arya','sansa','bran']))

print("\nOdd one out (Lannister group):")
print(model.wv.doesnt_match(['cersei','jaime','bronn','tyrion']))

# Word vector & similarities
print("\nVector of 'king':")
print(model.wv['king'])

print("\nSimilarity Arya-Sansa:", model.wv.similarity('arya','sansa'))
print("Similarity Tywin-Sansa:", model.wv.similarity('tywin','sansa'))

# Vocabulary
words = model.wv.index_to_key
print("\nTotal Vocabulary Size:", len(words))

# PCA Visualization
vectors = model.wv.get_normed_vectors()
pca = PCA(n_components=3)
X = pca.fit_transform(vectors)

# 3D Plot
pio.renderers.default = "browser"
```

```
df = pd.DataFrame(X[200:300], columns=["x", "y", "z"])
df["label"] = words[200:300]

fig = px.scatter_3d(
    df,
    x="x",
    y="y",
    z="z",
    text="label",
    title="Word2Vec - Game of Thrones (3D PCA Visualization)"
)

fig.show()
```

output:-



LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB No. 14

Document Loading using LangChain for Retrieval-Augmented Generation (RAG)

This lab introduces students to Document Loaders in LangChain, a key component of **Retrieval-Augmented Generation (RAG)** systems. Students will learn how to load, preprocess, and structure data from different document formats such as text and PDF files. By converting documents into LangChain's Document objects, students will understand how external knowledge can be prepared and supplied to large language models for improved, context-aware responses.

LAB Objectives

- Understand the role of document loaders in RAG
- Load data from multiple file formats using LangChain
- Inspect document metadata and content
- Prepare documents for downstream tasks like chunking and retrieval

Theory:

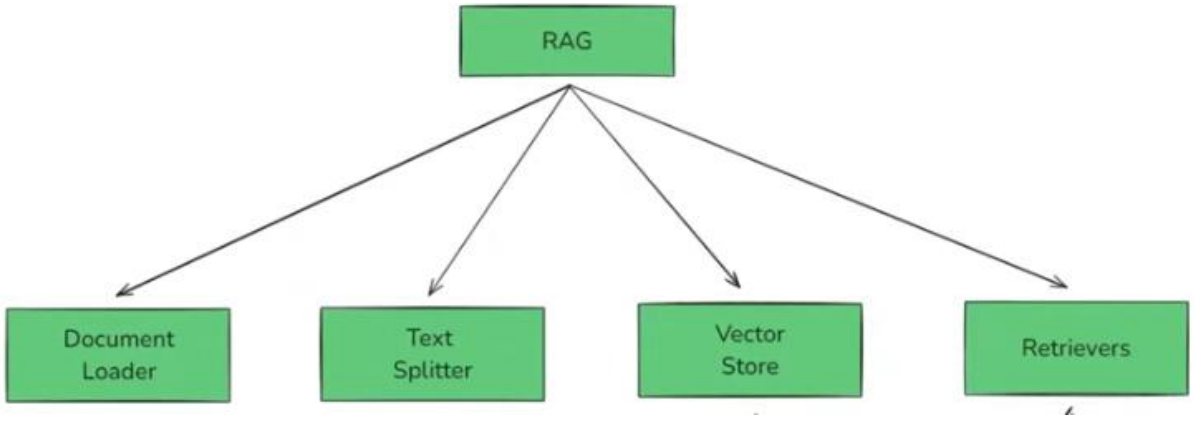
It is a technique that combines information retrieval with language generation, where a model retrieves relevant documents from a knowledge base and then uses them as context to generate accurate and grounded responses.

Benefits of using RAG

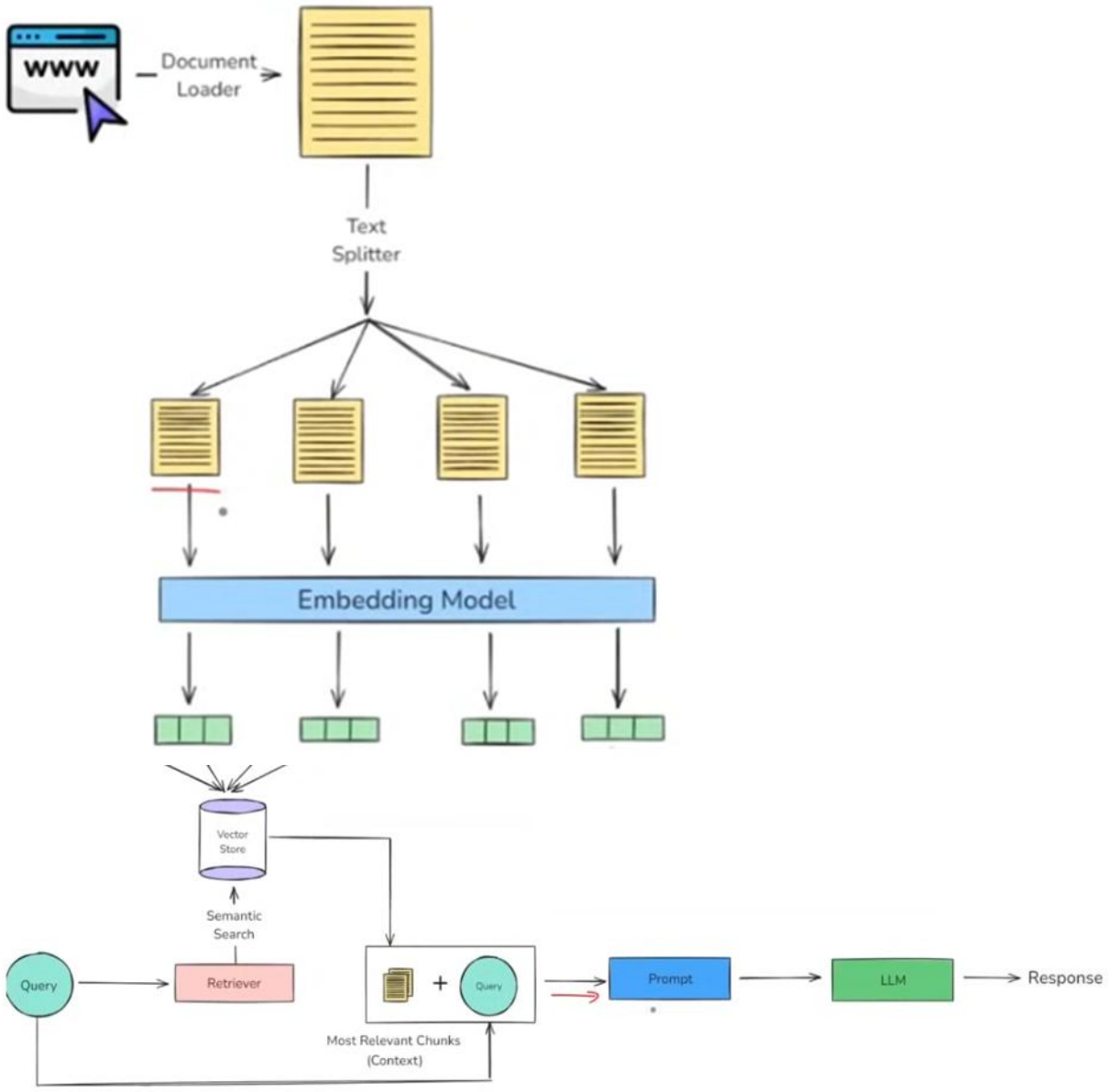
1. Use of up to date information
2. Better privacy

No limit of document size

Flow Methodology of RAG



Complete Flow diagram of RAG



Tools & Libraries

- Python 3.9+
- Required libraries:
    - langchain
    - langchain-community
    - pypdf
    - unstructured

Lab Tasks (Practice Steps)

Task 1: Environment Setup

- Create a virtual environment
- Install required LangChain libraries
- Verify installation

Task 2: Understand the Main Concept – Document Loaders

- Study the role of **Document Loaders** in RAG
- Explain how loaders convert raw data into LangChain Document objects

Task 3: Load PDF Data (PyPDFLoader)

- Load lecture_notes.pdf
- Count total pages
- Display content of first page
- Attach code with output screenshot

Task 4: Load Web Data (WebBaseLoader)

- Use **WebBaseLoader** to load a webpage
- Extract main textual content
- Observe metadata (URL source)
- Attach code with output screenshot

Task 5: Load Structured Data (CSVLoader)

- Load students.csv
- Inspect how rows are converted into documents
- Print one document sample
- Attach code with output screenshot

Task 6: Compare All Loaders

Students must compare:

- Content format
- Metadata fields
- Attach code with output screenshot

Lab Questions

1. Show the working of all above lab tasks
2. What is the role of document loaders in RAG?
3. Why is metadata important in LangChain documents?
4. Difference between TextLoader and PyPDFLoader?

4. What happens if a PDF has scanned images instead of text?
5. Why is directory-based loading useful in real applications?
6. How does document quality affect RAG performance?

Questions:-

```
# ================================================
# LAB NO. 14 - DOCUMENT LOADING USING LANGCHAIN FOR RAG
# COMPLETE & CORRECT SOLUTION IN ONE CODE CELL
# ================================================

# ---------------- TASK 1: ENVIRONMENT SETUP ----------------
!pip install langchain langchain-community pypdf unstructured

print("TASK 1: Environment Setup Completed")
print("Required libraries installed successfully ✅\n")

# ---------------- IMPORT LIBRARIES ----------------
from langchain_community.document_loaders import PyPDFLoader, WebBaseLoader,
CSVLoader

# ---------------- TASK 2: CONCEPT ----------------
print("TASK 2: DOCUMENT LOADERS CONCEPT")
print("Document Loaders are used in RAG systems to convert raw data")
print("such as PDFs, websites, and CSV files into LangChain Document objects.")
print("Each Document contains page_content and metadata.\n")

# ---------------- TASK 3: LOAD PDF DATA ----------------
print("TASK 3: LOAD PDF DATA (PyPDFLoader)\n")

pdf_loader = PyPDFLoader("lecture_notes.pdf")
pdf_docs = pdf_loader.load()

print("Total Pages in PDF:", len(pdf_docs))
print("\nFirst Page Content (Preview):\n")
print(pdf_docs[0].page_content[:400])
print("\nPDF Metadata:")
print(pdf_docs[0].metadata)

# ---------------- TASK 4: LOAD WEB DATA ----------------
print("\n\nTASK 4: LOAD WEB DATA (WebBaseLoader)\n")

web_loader =
WebBaseLoader("https://en.wikipedia.org/wiki/Natural_language_processing")
web_docs = web_loader.load()

print("Web Page Content (Preview):\n")
print(web_docs[0].page_content[:400])
print("\nWeb Metadata:")
print(web_docs[0].metadata)

# ---------------- TASK 5: LOAD CSV DATA ----------------
print("\n\nTASK 5: LOAD STRUCTURED DATA (CSVLoader)\n")
```

```
csv_loader = CSVLoader(file_path="students.csv")
csv_docs = csv_loader.load()

print("Total Rows Loaded:", len(csv_docs))
print("\nSample CSV Document:")
print("Content:", csv_docs[0].page_content)
print("Metadata:", csv_docs[0].metadata)

# ---------------- TASK 6: COMPARISON ----------------
print("\n\nTASK 6: COMPARISON OF DOCUMENT LOADERS\n")

print("PDF Loader:")
print("- Content Format: Page-wise unstructured text")
print("- Metadata: Page number, source file\n")

print("Web Loader:")
print("- Content Format: Cleaned webpage text")
print("- Metadata: URL source\n")

print("CSV Loader:")
print("- Content Format: Row-wise structured data")
print("- Metadata: Row index, source file\n")

# ---------------- LAB QUESTIONS ----------------
print("\n\nLAB QUESTIONS & ANSWERS\n")

print("1. Role of document loaders in RAG:")
print("Document loaders prepare external knowledge by converting raw data")
print("into structured Document objects for retrieval and augmentation.\n")

print("2. Why is metadata important?")
print("Metadata helps track source, page number, URL, and improves")
print("retrieval accuracy and transparency.\n")

print("3. Difference between TextLoader and PyPDFLoader:")
print("TextLoader loads plain text files, while PyPDFLoader")
print("loads PDFs page-by-page with page metadata.\n")

print("4. What if a PDF has scanned images?")
print("Text cannot be extracted directly. OCR tools are required")
print("to convert images into readable text.\n")

print("5. Why is directory-based loading useful?")
print("It allows loading multiple documents at once, making RAG")
print("systems scalable for real-world applications.\n")

print("6. How does document quality affect RAG performance?")
print("Poor-quality or noisy documents reduce retrieval accuracy")
print("and may cause incorrect or hallucinated responses.\n")

print("LAB NO. 14 COMPLETED SUCCESSFULLY ✅")
```

output:-

```
Requirement already satisfied: langchain in /usr/local/lib/python3.12/dist-packages (1.2.0)
Requirement already satisfied: langchain-community in /usr/local/lib/python3.12/dist-packages (0.4.1)
Requirement already satisfied: pypdf in /usr/local/lib/python3.12/dist-packages (6.6.0)
Requirement already satisfied: unstructured in /usr/local/lib/python3.12/dist-packages (0.18.27)
Requirement already satisfied: langchain-core<2.0.0,>=1.2.1 in /usr/local/lib/python3.12/dist-packages (from langchain) (1.2.7)
Requirement already satisfied: langgraph<1.1.0,>=1.0.2 in /usr/local/lib/python3.12/dist-packages (from langchain) (1.0.5)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.12/dist-packages (from langchain) (2.12.3)
Requirement already satisfied: langchain-classic<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (1.0.1)
Requirement already satisfied: SQLAlchemy<3.0.0,>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (2.0.45)
Requirement already satisfied: requests<3.0.0,>=2.32.5 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (2.32.5)
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (6.0.3)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (3.13.2)
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (9.1.2)
Requirement already satisfied: dataclasses-json<0.7.0,>=0.6.7 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (0.6.7)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.10.1 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (2.12.0)
Requirement already satisfied: langsmith<1.0.0,>=0.1.125 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (0.4.59)
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (0.4.3)
Requirement already satisfied: numpy>=1.26.2 in /usr/local/lib/python3.12/dist-packages (from langchain-community) (2.0.2)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/dist-packages (from unstructured) (3.4.4)
Requirement already satisfied: filetype in /usr/local/lib/python3.12/dist-packages (from unstructured) (1.2.0)
Requirement already satisfied: python-magic in /usr/local/lib/python3.12/dist-packages (from unstructured) (0.4.27)
Requirement already satisfied: lxml in /usr/local/lib/python3.12/dist-packages (from unstructured) (6.0.2)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (from unstructured) (3.9.1)
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (from unstructured) (3.9.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.12/dist-packages (from unstructured) (4.13.5)
Requirement already satisfied: emoji in /usr/local/lib/python3.12/dist-packages (from unstructured) (2.15.0)
Requirement already satisfied: python-iso639 in /usr/local/lib/python3.12/dist-packages (from unstructured) (2025.11.16)
Requirement already satisfied: langdetect in /usr/local/lib/python3.12/dist-packages (from unstructured) (1.0.9)
Requirement already satisfied: rapidfuzz in /usr/local/lib/python3.12/dist-packages (from unstructured) (3.14.3)
Requirement already satisfied: backoff in /usr/local/lib/python3.12/dist-packages (from unstructured) (2.2.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.12/dist-packages (from unstructured) (4.15.0)
Requirement already satisfied: unstructured-client in /usr/local/lib/python3.12/dist-packages (from unstructured) (0.42.6)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from unstructured) (2.0.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from unstructured) (4.67.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from unstructured) (5.9.5)
Requirement already satisfied: python-oxmsg in /usr/local/lib/python3.12/dist-packages (from unstructured) (0.0.2)
Requirement already satisfied: html5lib in /usr/local/lib/python3.12/dist-packages (from unstructured) (1.1)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-communit
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (1.4
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (25.4.0
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (1.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (
```

```
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.1.125->langchain-community) (3.11.5)
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.1.125->langchain-community
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.1.125->langchain-community) (0.25
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->langchain) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->langchain) (2.41.4)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->langchain) (0.4.2)
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/python3.12/dist-packages (from pydantic-settings<3.0.0,>=2.10.1->langchain-commu
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.32.5->langchain-community) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.32.5->langchain-community) (2.5.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.32.5->langchain-community) (2025.
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-packages (from SQLAlchemy<3.0.0,>=1.4.0->langchain-community) (3.3.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4->unstructured) (2.8)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.12/dist-packages (from html5lib->unstructured) (1.17.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packages (from html5lib->unstructured) (0.5.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk->unstructured) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk->unstructured) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk->unstructured) (2025.11.3)
Requirement already satisfied: olefile in /usr/local/lib/python3.12/dist-packages (from python-oxmsg->unstructured) (0.47)
```

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines

LAB NO. 15

Build a RAG-Based Chatbot Using Ollama and LangChain, and Streamlit

Lab Objective

By the end of this lab, students will be able to:

- Set up Ollama locally and run LLAMA2
- Build a basic LangChain chatbot using Ollama
- Implement document ingestion and vector storage
- Enable Retrieval-Augmented Generation (RAG)
- Deploy the chatbot using Streamlit

Tools & Technologies

- Python 3.9+
- VS Code
- Ollama (LLAMA2)
- LangChain
- Streamlit
- FAISS (Vector Store)
- Dotenv

File code 1

Localama.py

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_community.llms import Ollama
```

```
import streamlit as st
import os
from dotenv import load_dotenv

load_dotenv()

os.environ["LANGCHAIN_TRACING_V2"]="true"
os.environ["LANGCHAIN_API_KEY"]=os.getenv("LANGCHAIN_API_KEY")

## Prompt Template

prompt=ChatPromptTemplate.from_messages(
    [
        ("system","You are a helpful assistant. Please response to the user queries"),
        ("user","Question:{question}")
    ]
)
## streamlit framework

st.title('Langchain Demo With LLAMA2 API')
input_text=st.text_input("Search the topic u want")

# ollama LLAma2 LLm
llm=Ollama(model="llama2")
output_parser=StrOutputParser()
chain=prompt|llm|output_parser

if input_text:
    st.write(chain.invoke({"question":input_text}))
```

Code File 2

App.py

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
```

```
from langchain_core.output_parsers import StrOutputParser

import streamlit as st
import os
from dotenv import load_dotenv

os.environ["OPENAI_API_KEY"]=os.getenv("OPENAI_API_KEY")
## Langmith tracking
os.environ["LANGCHAIN_TRACING_V2"]="true"
os.environ["LANGCHAIN_API_KEY"]=os.getenv("LANGCHAIN_API_KEY")

## Prompt Template

prompt=ChatPromptTemplate.from_messages(
    [
        ("system","You are a helpful assistant. Please response to the user
queries"),
        ("user","Question:{question}")
    ]
)

## streamlit framework

st.title('Langchain Demo With OPENAI API')
input_text=st.text_input("Search the topic u want")

# openAI LLm
llm=ChatOpenAI(model="gpt-3.5-turbo")
output_parser=StrOutputParser()
chain=prompt|llm|output_parser

if input_text:
    st.write(chain.invoke({'question':input_text}))
```

Step 1: Create Project Structure

Open **VS Code** and create the following folder structure:

```
rag-ollama-chatbot/
│
├── app.py
├── requirements.txt
├── .env
├── data/
│   └── sample_docs.txt
```

Step 2: Install and Verify Ollama

2.1 Install Ollama

Download and install Ollama from:

```
https://ollama.com
```

2.2 Pull LLAMA2 Model

Open terminal and run:

```bash
ollama pull llama2
```

2.3 Verify Ollama

```bash
ollama run llama2
```

If the model responds, Ollama is working correctly.

Step 3: Create Virtual Environment

```bash
python -m venv myenv
myenv\Scripts\activate   # Windows
```

Step 4: Install Required Libraries

Create **requirements.txt**:

```txt
langchain
langchain-community
langchain-core
langchain-openai
streamlit
faiss-cpu
python-dotenv
```

Install dependencies:

```bash
pip install -r requirements.txt
```

Step 5: Add Sample Knowledge Base

Create **data/sample_docs.txt** and add:

```pgsql
LangChain is a framework for developing applications powered by large language mo
RAG stands for Retrieval-Augmented Generation.
Ollama allows running LLMs locally without cloud APIs.
FAISS is a vector database for similarity search.
```

Step 6: Environment Configuration

Create **.env** file:

```env
LANGCHAIN_API_KEY=your_langchain_api_key
```

> Note: Even with Ollama, LangChain tracing may require this key.

Step 7: Understand the Base Chatbot Code (Given Code)

The provided code:

- Uses **Ollama LLAMA2**
- Accepts user input via Streamlit
- Sends query directly to the LLM
- ❌ Does NOT use retrieval (no RAG)

We will **extend this code** to add:

- Document loading
- Text splitting
- Embeddings
- Vector database
- Retriever

Step 8: Add RAG Components

8.1 Import Additional Modules

Update **app.py**:

python

```python
from langchain_community.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.embeddings import OllamaEmbeddings
from langchain_community.vectorstores import FAISS
from langchain_core.runnables import RunnablePassthrough
```

Step 9: Load and Process Documents

Add below `.env` loading:

python

```
# Load documents
```

```
loader = TextLoader("data/sample_docs.txt")
documents = loader.load()

# Split documents
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500,
    chunk_overlap=50
)
docs = text_splitter.split_documents(documents)
```

Step 10: Create Embeddings and Vector Store

```
# Create embeddings
embeddings = OllamaEmbeddings(model="llama2")

# Create FAISS vector store
vectorstore = FAISS.from_documents(docs, embeddings)

# Create retriever
retriever = vectorstore.as_retriever()
```

Step 11: Modify Prompt for RAG

Replace your prompt template with:

```
prompt = ChatPromptTemplate.from_messages(
    [
        ("system", "Answer the question using the provided context only."),
        ("user", "Context:\n{context}\n\nQuestion:\n{question}")
    ]
)
```

Step 12: Build the RAG Chain

```
llm = Ollama(model="llama2")
output_parser = StrOutputParser()
```

```
rag_chain = (
    {
        "context": retriever,
        "question": RunnablePassthrough()
    }
    | prompt
    | llm
    | output_parser
)
```

Step 13: Update Streamlit UI

```
st.title("RAG Chatbot with Ollama & LangChain")

input_text = st.text_input("Ask a question based on the documents")

if input_text:
    response = rag_chain.invoke(input_text)
    st.write(response)
```

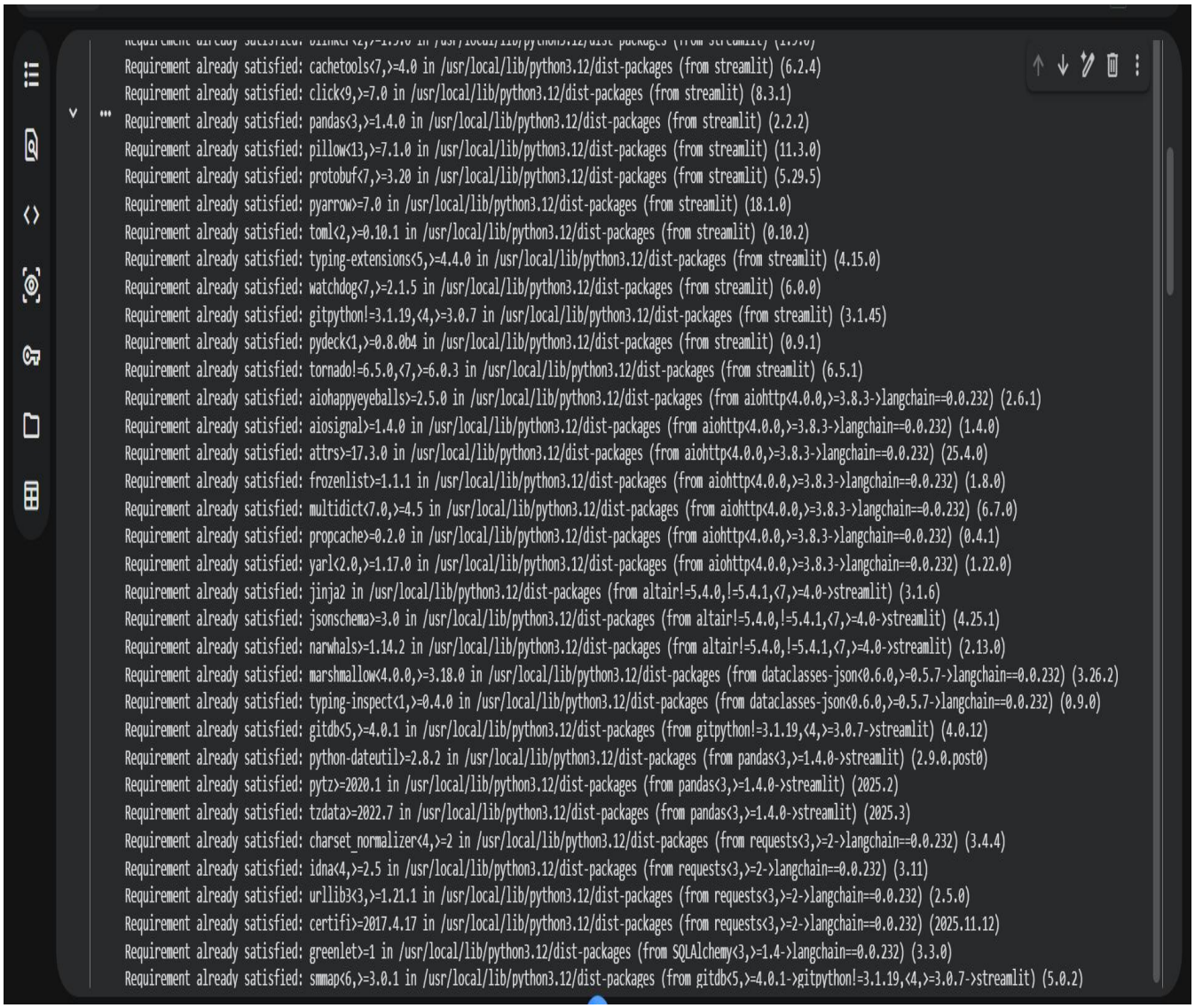
Step 14: Run the Application

```
streamlit run app.py
```

Question:

```
!pip install langchain==0.0.232 faiss-cpu streamlit pyngrok
```

Output:




```
sample_text = """
Python is a programming language that lets you work quickly
and integrate systems more effectively.

LangChain helps build applications with LLMs by chaining
prompts, agents, and memory.
```

```
Ollama provides access to LLAMA2 models locally.
"""
```

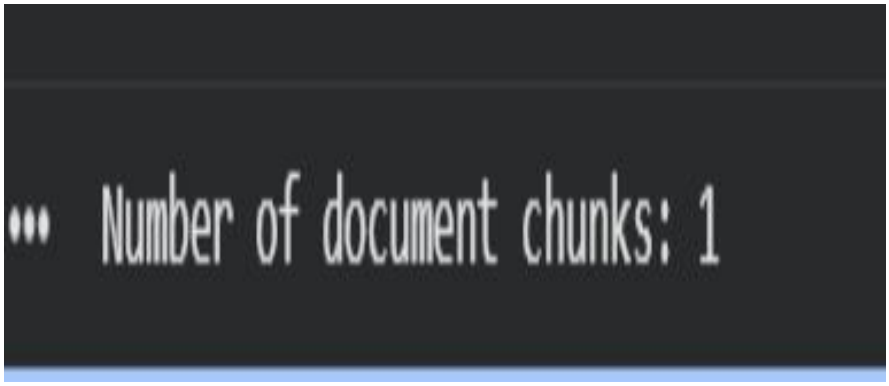
```
with open("sample_docs.txt", "w") as f:
    f.write(sample_text)
```

```
from langchain.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Load document
loader = TextLoader("sample_docs.txt")
documents = loader.load()
```

```
# Split into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=50)
docs = text_splitter.split_documents(documents)
```

```
print("Number of document chunks:", len(docs))
```

Output:




```
from langchain.embeddings.base import Embeddings
from langchain.vectorstores import FAISS
import numpy as np
```

```
# Mock embeddings (random vectors)
class MockEmbeddings(Embeddings):
    def embed_documents(self, texts):
        return [np.random.rand(1536).tolist() for _ in texts]
    def embed_query(self, text):
        return np.random.rand(1536).tolist()
```

```
embeddings = MockEmbeddings()
vectorstore = FAISS.from_documents(docs, embeddings)
retriever = vectorstore.as_retriever()# Simple function to simulate RAG retrieval and
response
def rag_chatbot(query):
    # Retrieve docs (top 2 chunks)
    results = retriever.get_relevant_documents(query)[:2]
    context_text = "\n".join([doc.page_content for doc in results])

    # Simulated answer
    answer = f"Answer based on context:\n{context_text}\n\nYour question: {query}"
    return answer
question = "What is LangChain?"
response = rag_chatbot(question)
print(response)
```

Output:

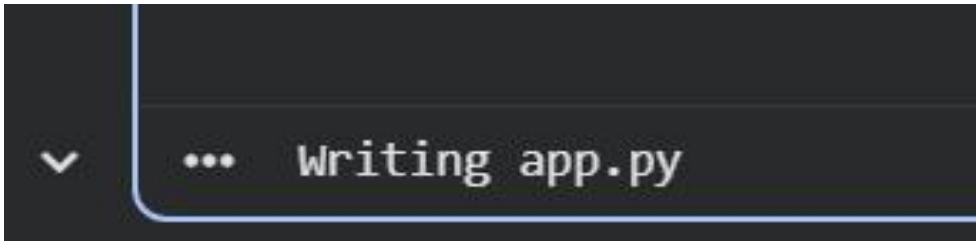
```
Answer based on context:
Python is a programming language that lets you work quickly
and integrate systems more effectively.

LangChain helps build applications with LLMs by chaining
prompts, agents, and memory.

Ollama provides access to LLAMA2 models locally.

Your question: What is LangChain?
```

```
%%writefile app.py

import streamlit as st


st.title("RAG Chatbot Demo (Colab)")
```

```
question = st.text_input("Ask a question:")
```

```
if question:

    # Simple mock RAG answer

    answer = f"Answer based on context:\nPython is a programming language...\nLangChain helps build applications...\nYour question: {question}"

    st.write(answer)
```

Output:




```
# Ask multiple questions

while True:

    question = input("Ask a question (or type 'exit' to stop): ")

    if question.lower() == "exit":

        break

    print(rag_chatbot(question))
```

Output:

```
Ask a question (or type 'exit' to stop): what is python?
Answer based on context:
Python is a programming language that lets you work quickly
and integrate systems more effectively.

LangChain helps build applications with LLMs by chaining
prompts, agents, and memory.

Ollama provides access to LLAMA2 models locally.

Your question: what is python?
Ask a question (or type 'exit' to stop): exit
```

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines