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:

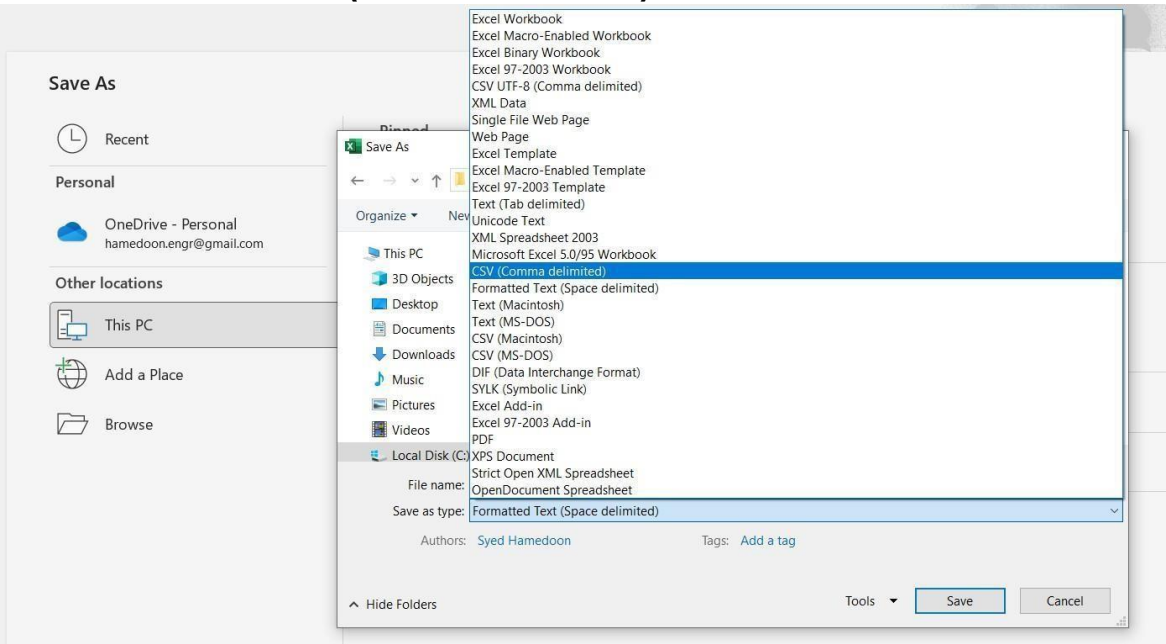
1) Open your project folder in VS Code.
2) Press Ctrl + Shift + P → search for Python:
3) Select Interpreter.
4) Choose your newly created environment (venv or myenv).

4. Install Machine Learning and Deep Learning Libraries

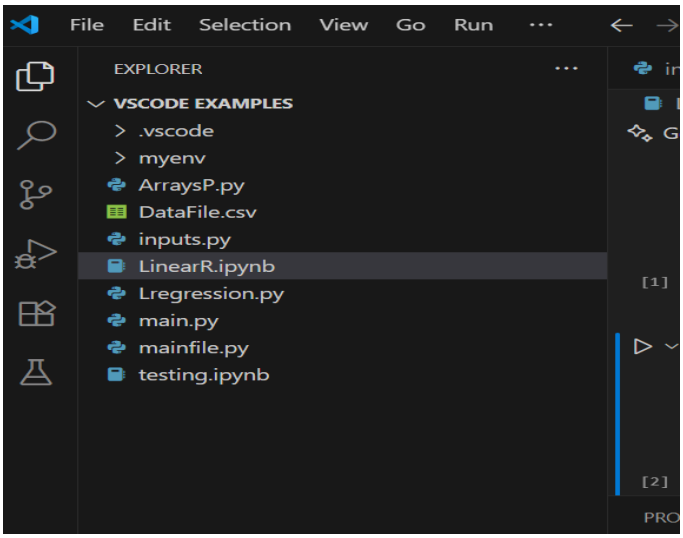
- pip install –upgrade pip
- pip install pandas
- pip install matplotlib
- pip install seaborn
- pip install scikit-learn

- pip install scipy
- pip install numpy
- pip install xgboost
- pip install lightgbm
- pip install catboost
- pip install tensorflow
- pip install keras
- pip install torch

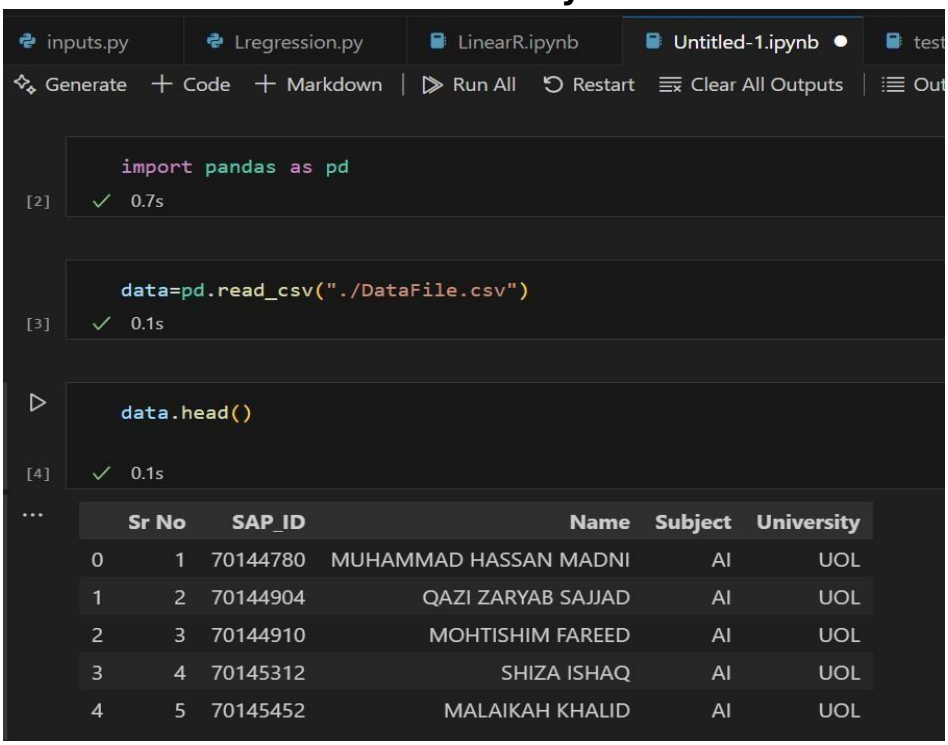
Task 1:
Save Excel file as .csv (Comma delimited)



Task 2:
Save .csv file in directory and folder where we created a python environment



Here we can see our file successfully

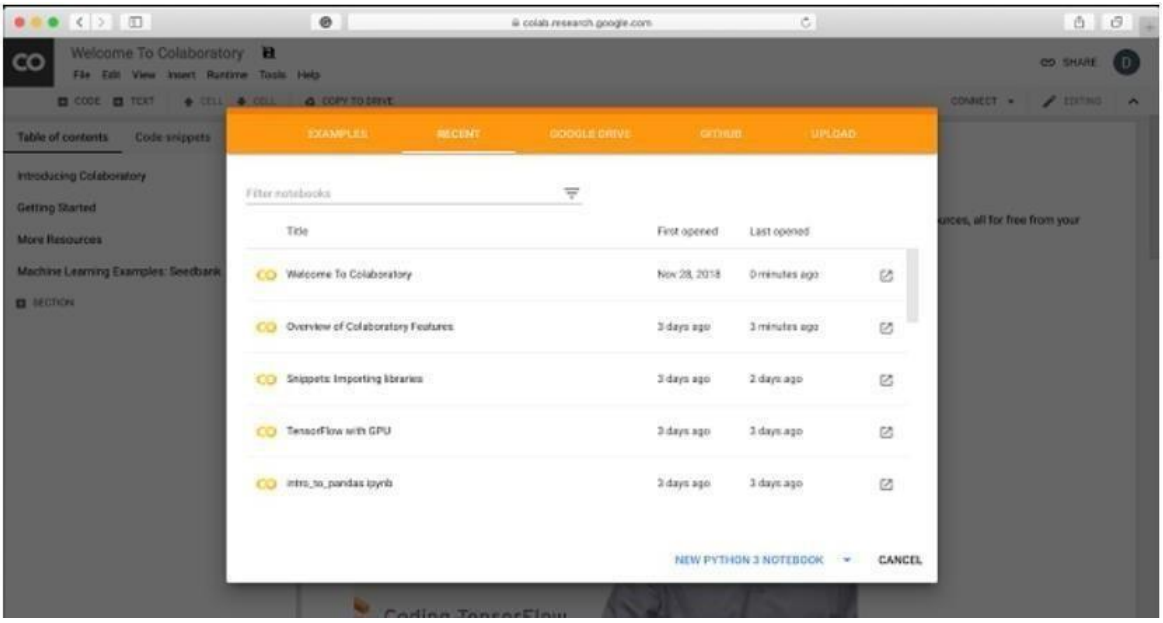


	Sr No	SAP_ID	Name	Subject	University
0	1	70144780	MUHAMMAD HASSAN MADNI	AI	UOL
1	2	70144904	QAZI ZARYAB SAJJAD	AI	UOL
2	3	70144910	MOHTISHIM FAREED	AI	UOL
3	4	70145312	SHIZA ISHAQ	AI	UOL
4	5	70145452	MALAIKAH KHALID	AI	UOL

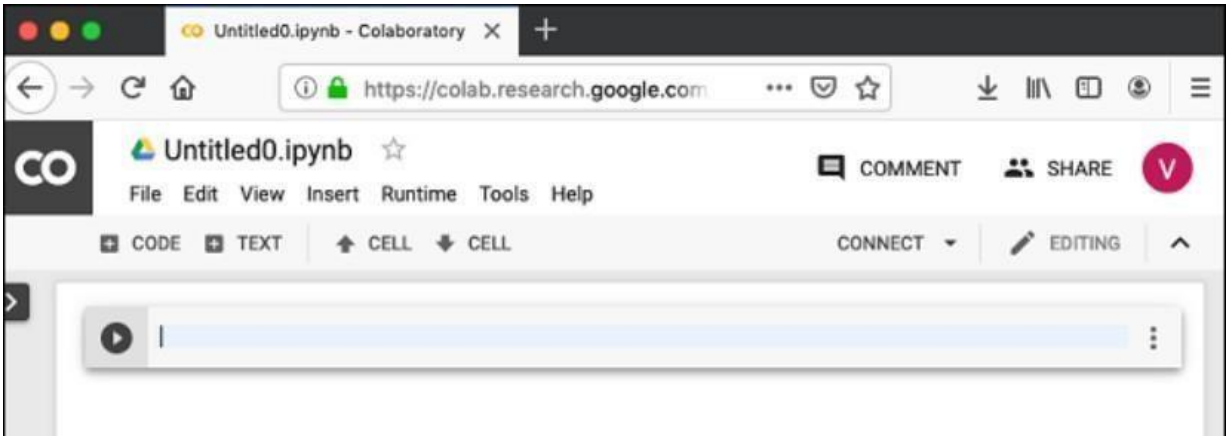
Introduction of Google Colab

- 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) –

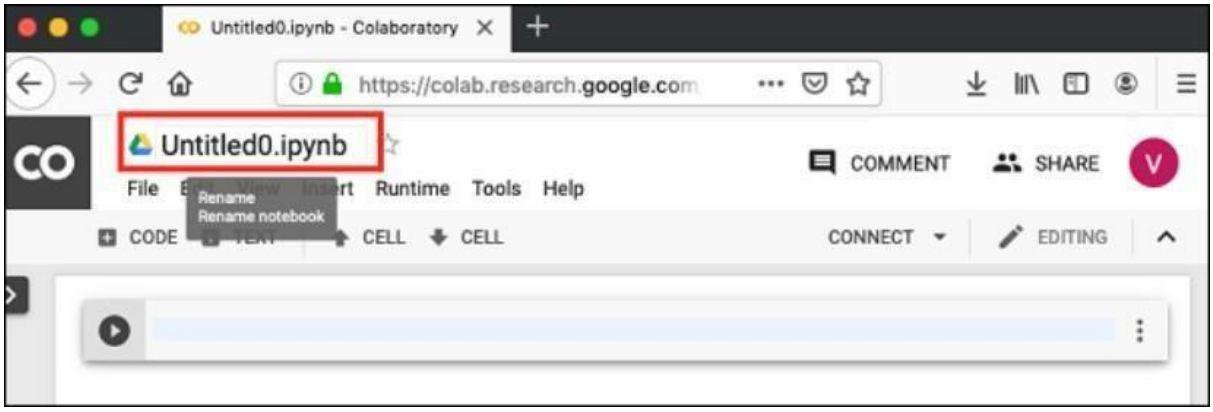


- **Step 2** – Click on the **NEW PYTHON 3 NOTEBOOK** link at the bottom of the screen. A new notebook would open up as shown in the screen below.

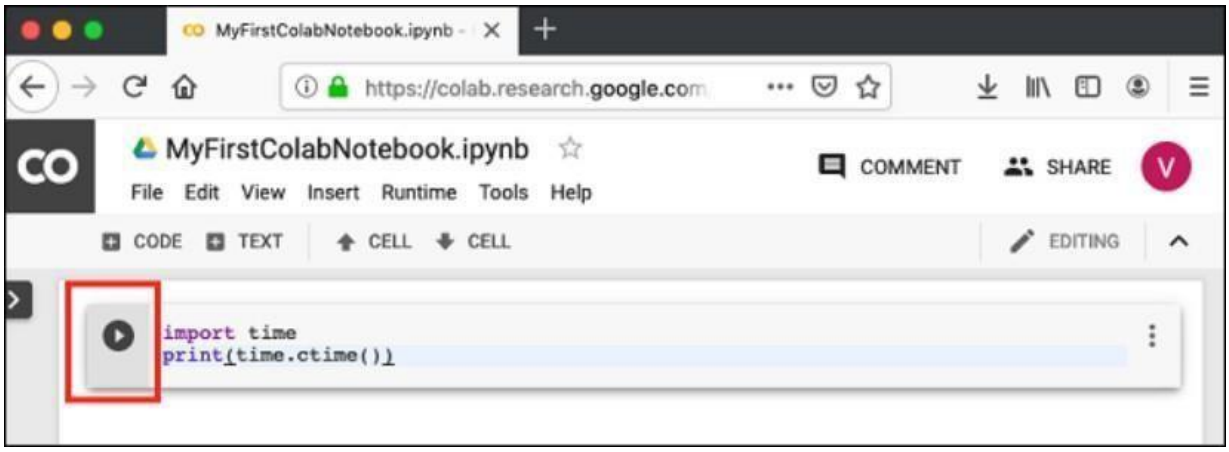


- **Setting Notebook Name**

  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 –

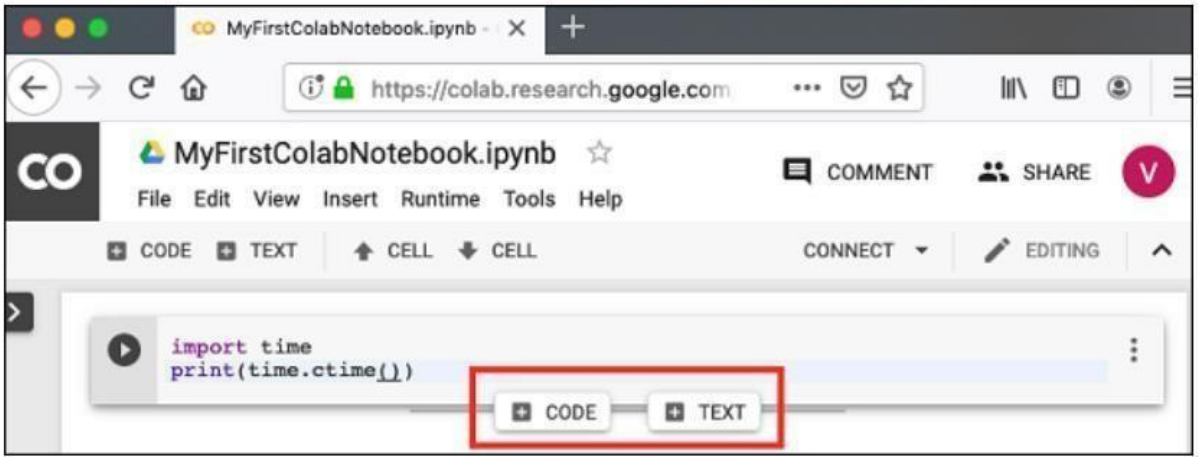


- Entering Code



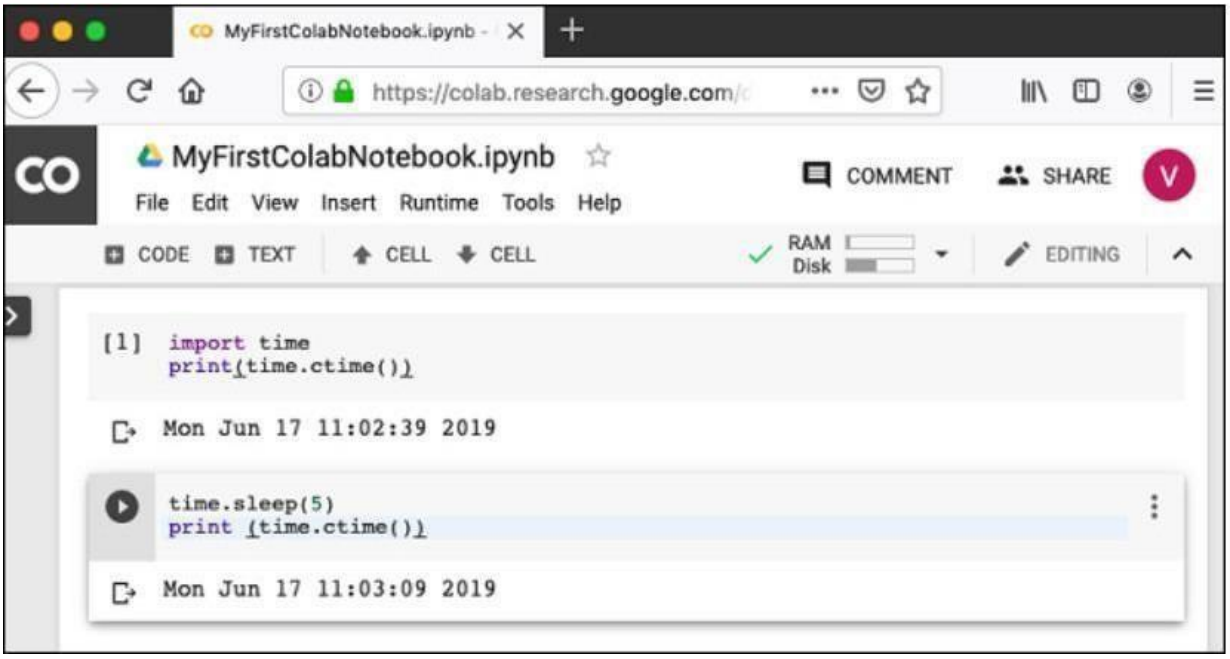
- Adding Code Cells

- To add more code to your notebook, select the following **menu** options –

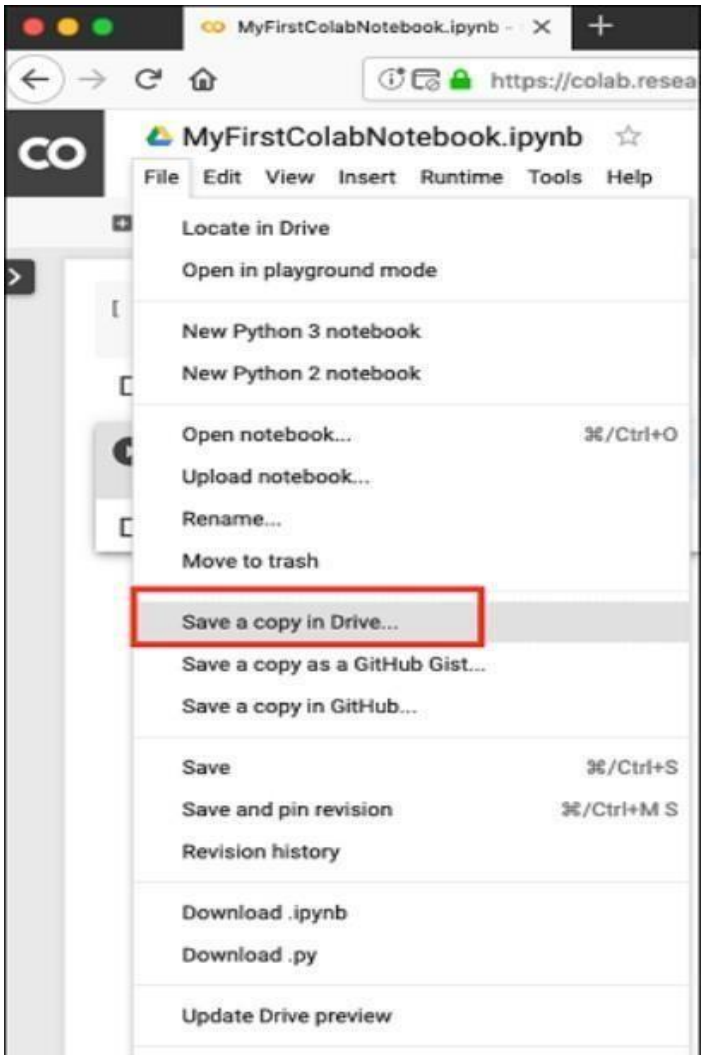


- **Run All**

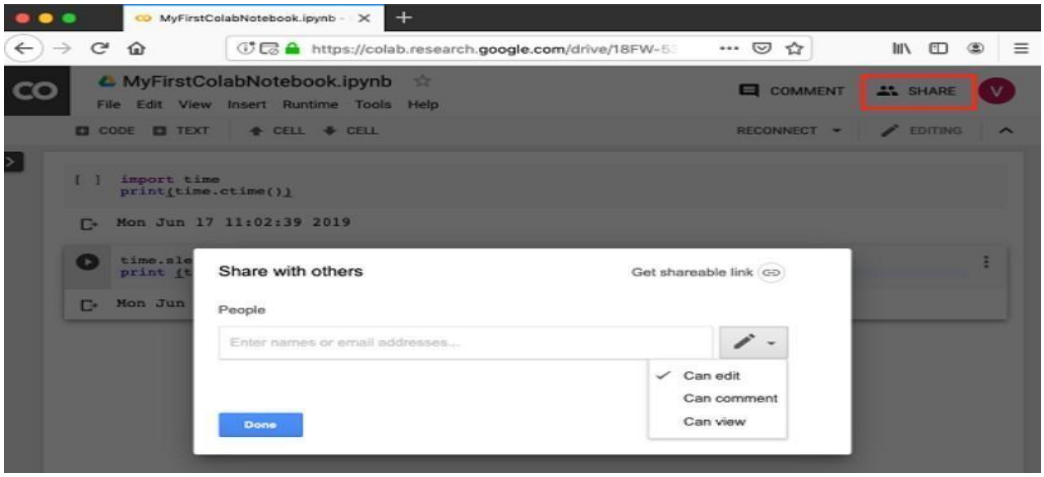
To run the entire code in your notebook without an interruption, execute the following menu options –



Saving to Google Drive



Google Colab - Sharing Notebook



LAB Assignment No. 1

Lab Assignment – Dataset Creation s Analysis

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.

Code:

```
import pandas as pd
data = {
  'Student_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
  'Name': ['Arooba', 'Azariya', 'salar', 'rohan', 'anaya', 'Bashir', 'Sana', 'saima', 'rida',
'saif'],
  'Age': [23, 78, 29, 15, 12, 14, 15, 19, 30, 28],
  'Marks_Math': [85, 25, 50, 58, 85, 90, 95, 75, 85, 75],
  'Marks_Science': [60, 80, 95, 50, 80, 75, 90, 70, 85, 90],
  'CGPA': [3.4, 3.8, 2.8, 3.5, 2.6, 3.5, 3.6, 2.4, 3.7, 3.4]
}
df = pd.DataFrame(data)
print(df)
```

Output:

	Student_ID	Name	Age	Marks_Math	Marks_Science	CGPA
0	1	Arooba	23	85	60	3.4
1	2	Azariya	78	25	80	3.8
2	3	salar	29	50	95	2.8
3	4	rohan	15	58	50	3.5
4	5	anaya	12	85	80	2.6
5	6	Bashir	14	90	75	3.5
6	7	Sana	15	95	90	3.6
7	8	saima	19	75	70	2.4
8	9	rida	30	85	85	3.7
9	10	saif	28	75	90	3.4

Q2: Upload Dataset in Python

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

Code:

```
print(df.info())
print(df.head())
print(df['Marks_Math'].mean())
print(df['Marks_Science'].max())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Student_ID     10 non-null     int64
 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
 5   CGPA           10 non-null     float64
dtypes: float64(1), int64(4), object(1)
memory usage: 612.0+ bytes
None
   Student_ID     Name  Age  Marks_Math  Marks_Science  CGPA
0           1   Arooba   23          85             60   3.4
1           2  Azariya   78          25             80   3.8
2           3    salar   29          50             95   2.8
3           4    rohan   15          58             50   3.5
4           5    anaya   12          85             80   2.6
72.3
95
```

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

Code:

```
print(df[df['Marks_Math'] > 50])
print(df.loc[df['Marks_Science'].idxmax()])
print(df['Marks_Math'].corr(df['Marks_Science']))
```

output:

```
   Student_ID    Name  Age  Marks_Math  Marks_Science  CGPA
0           1  Arooba   23          85             60   3.4
3           4   rohan   15          58             50   3.5
4           5   anaya   12          85             80   2.6
5           6  Bashir   14          90             75   3.5
6           7    Sana   15          95             90   3.6
7           8   saima   19          75             70   2.4
8           9    rida   30          85             85   3.7
9          10    saif   28          75             90   3.4
Student_ID           3
Name             salar
Age                 29
Marks_Math          50
Marks_Science       95
CGPA               2.8
Name: 2, dtype: object
0.015283907715545006
```

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.

Code:

```
print(df[df['Marks_Math'] > 50])
print(df.loc[df['Marks_Science'].idxmax()])
print(df['Marks_Math'].corr(df['Marks_Science']))
```

Output:

```
   Student_ID    Name  Age  Marks_Math  Marks_Science  CGPA
0           1  Arooba   23          85             60   3.4
3           4   rohan   15          58             50   3.5
4           5   anaya   12          85             80   2.6
5           6  Bashir   14          90             75   3.5
6           7    Sana   15          95             90   3.6
7           8   saima   19          75             70   2.4
8           9    rida   30          85             85   3.7
9          10    saif   28          75             90   3.4
Student_ID           3
Name             salar
Age                 29
Marks_Math          50
Marks_Science       95
CGPA               2.8
Name: 2, dtype: object
0.015283907715545006
```

🔷 Q5: Data Visualization

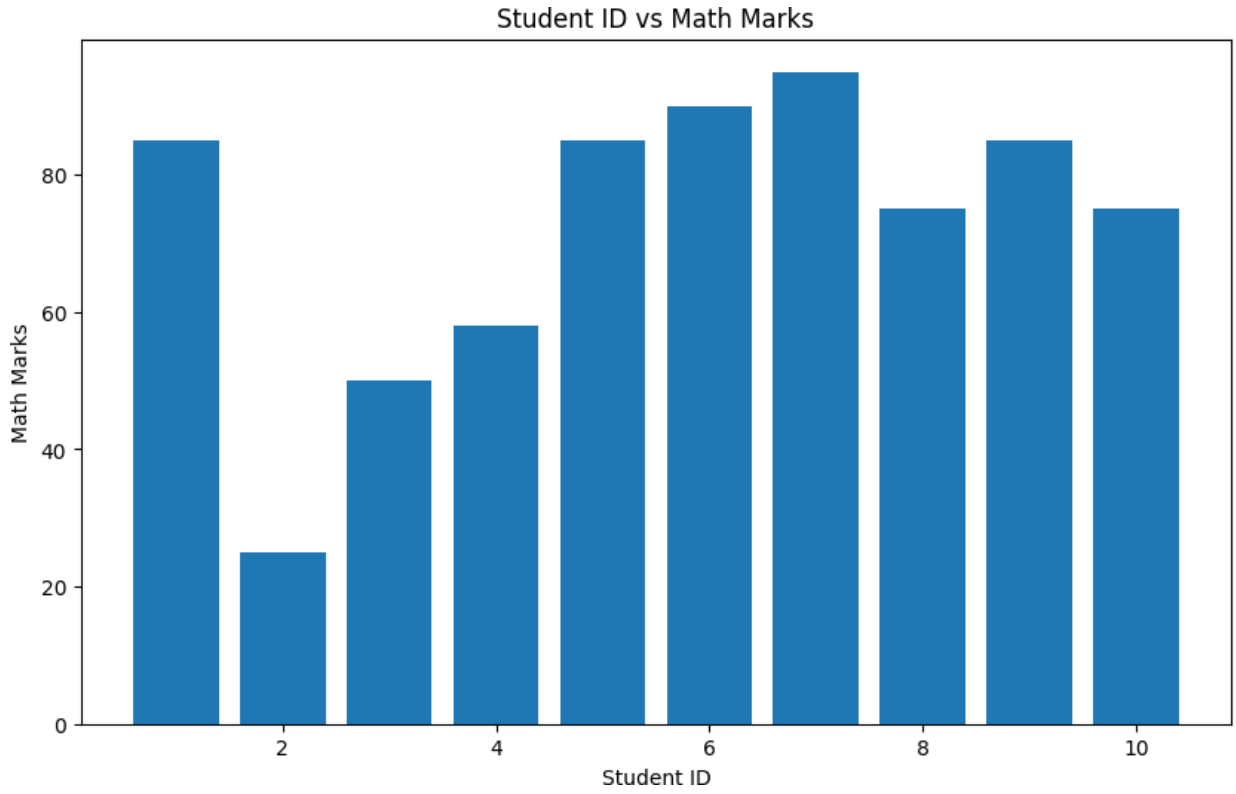
Use **Matplotlib/Seaborn** to create graphs:

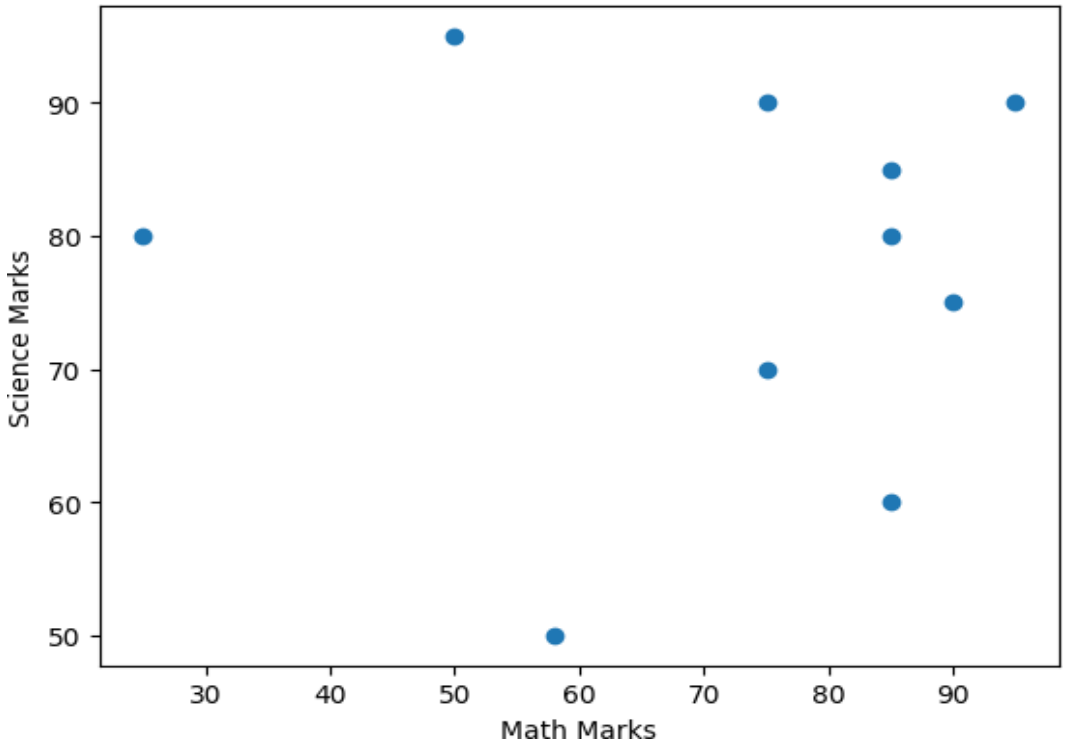
1. A bar chart of Student_ID vs Marks_Math.
2. A histogram of Age.
3. A scatter plot of Marks_Math vs Marks_Science.

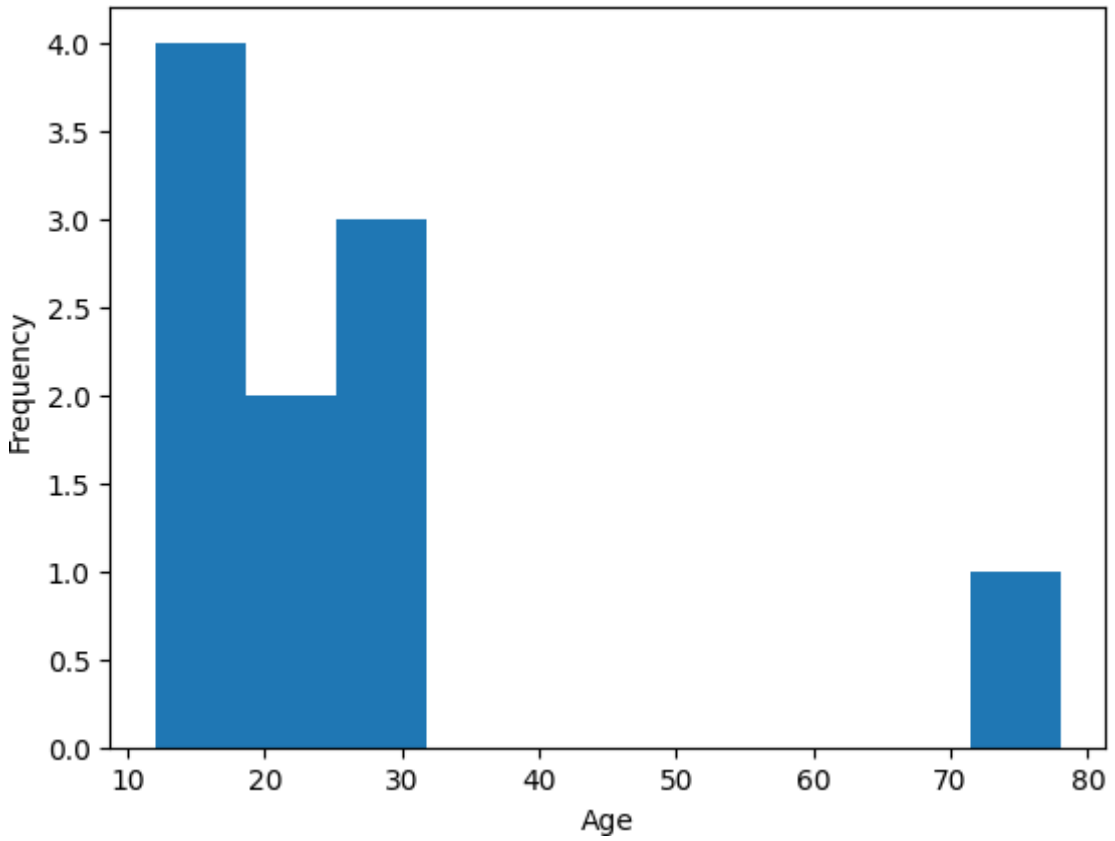
Code:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
plt.bar(df['Student_ID'], df['Marks_Math'])
plt.xlabel('Student ID')
plt.ylabel('Math Marks')
plt.title('Student ID vs Math Marks')
plt.show()
import matplotlib.pyplot as plt
plt.scatter(df['Marks_Math'], df['Marks_Science'])
plt.xlabel('Math Marks')
plt.ylabel('Science Marks')
plt.show()
plt.hist(df['Age'])
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

Output:


Student ID vs Math Marks
Math Marks
Student ID
0
20
40
60
80
2
4
6
8
10


Science Marks
Math Marks
50
60
70
80
90
30
40
50
60
70
80
90



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.

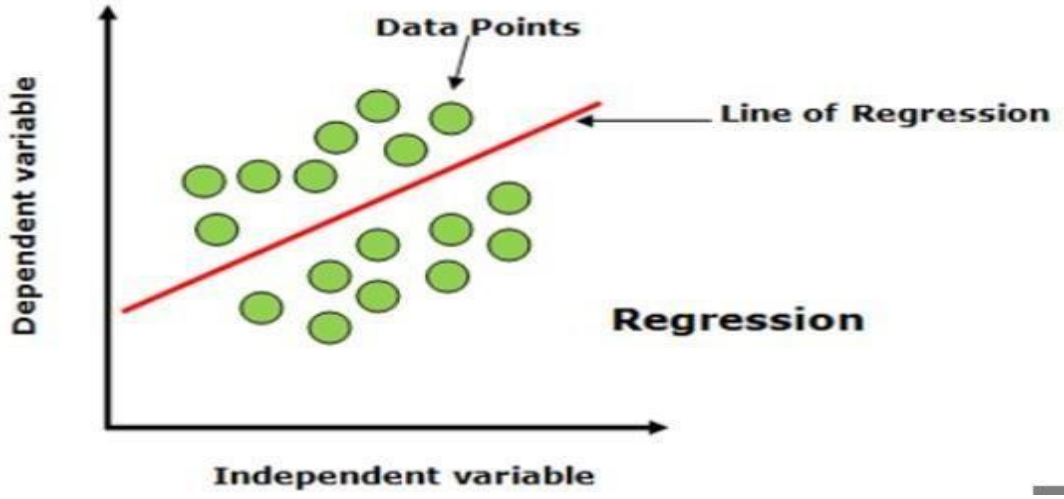
6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design 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.

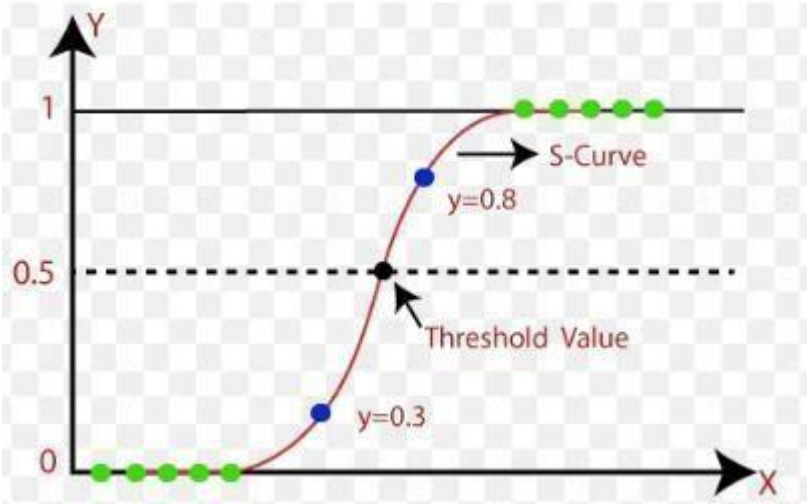
Introduction

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

Topic: Multiple and Logistic Linear Regression

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.

Code:

```
import pandas as pd

from sklearn.linear_model import LinearRegression
import numpy as np
data = pd.read_csv('house_features_data.csv')
data['Price'] = 50 + (data['Size_sqft'] * 0.15) + (data['Bedrooms'] * 10) - (data['Age'] * 1.2) +
np.random.randint(-20, 20, len(data))

X = data[['Size_sqft', 'Bedrooms', 'Age']]
y = data['Price']

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

```
new_house = np.array([[2000, 3, 10]])
predicted_price = model.predict(new_house)[0]

print(f"Intercept: {model.intercept_:.2f}")
print(f"  Size: {model.coef_[0]:.4f}")
print(f"  Bedrooms: {model.coef_[1]:.4f}")
print(f"  Age: {model.coef_[2]:.4f}")
print(f"\nPredicted price for a 2000 sqft, 3-bed, 10-year-old house: ${predicted_price *
1000:.2f}")
```

output:

```
Intercept: 57.73
  Size: 0.1503
  Bedrooms: 8.0593
  Age: -1.4136
```

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)**.

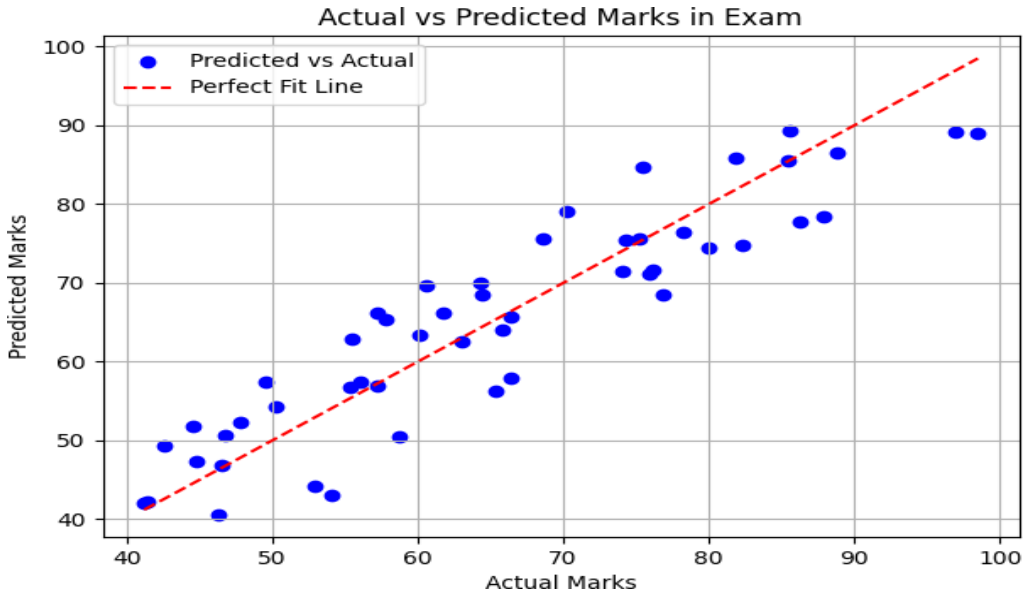
Code:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt
data = pd.read_csv("student_performance_data.csv")
X = data[["Hours_Study", "Hours_Sleep", "Attendance_%"]]
```

```
y = data["Marks_in_Exam"]

# Step 3: Train model
model = LinearRegression()
model.fit(X, y)

# Step 4: Predict
y_pred = model.predict(X)

# Step 5: Model evaluation
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print("📊 Model Evaluation Results:")
print(f"R² Score: {r2:.4f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Intercept: {model.intercept_:.2f}")
print(f"Coefficients:")
print(f" Hours Study: {model.coef_[0]:.4f}")
print(f" Hours Sleep: {model.coef_[1]:.4f}")
print(f" Attendance: {model.coef_[2]:.4f}")

# Step 6: Plot actual vs predicted marks
plt.figure(figsize=(7,5))
plt.scatter(y, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', label='Perfect Fit
Line')
plt.xlabel('Actual Marks')
plt.ylabel('Predicted Marks')
plt.title('Actual vs Predicted Marks in Exam')
plt.legend()
plt.grid(True) plt.show(
```

output

```
📊 Model Evaluation Results:
R² Score: 0.8406
Mean Squared Error (MSE): 36.16
Intercept: -0.95
Coefficients:
  Hours Study: 5.2922
  Hours Sleep: 2.5849
  Attendance:  0.2272
```



Q3. Logistic Regression – Pass/Fail Classification

Dataset columns:

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

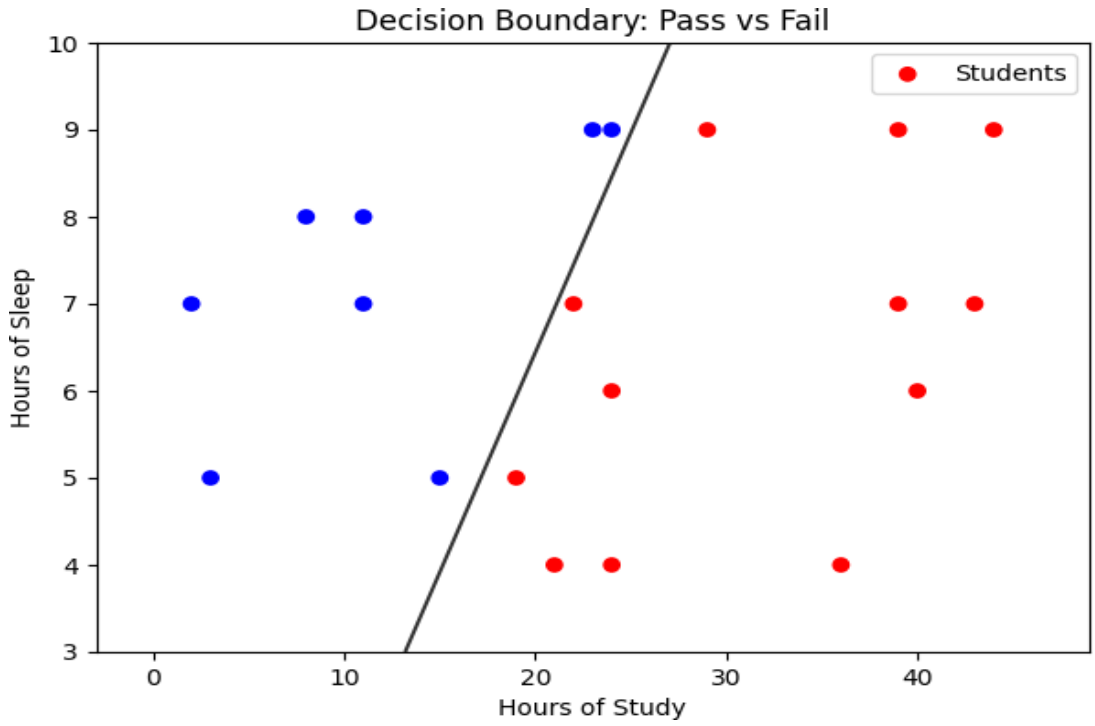
S Task:

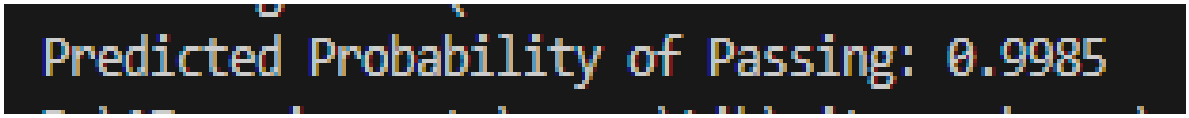
1. Fit a **logistic regression classifier**.
2. Predict the probability of passing if a student studies 30 hours and sleeps 6 hours.
3. Plot the **decision boundary** (pass vs fail)

Code:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
data = pd.read_excel("student_pass_fail_20.xlsx")
X = data[["Hours_Study", "Hours_Sleep"]]
y = data["Pass"]
model = LogisticRegression()
model.fit(X, y)
new_student = np.array([[30, 6]])
prob_pass = model.predict_proba(new_student)[0][1]
print(f"Predicted Probability of Passing: {prob_pass:.4f}")
plt.figure(figsize=(7,5))
plt.scatter(data["Hours_Study"], data["Hours_Sleep"], c=data["Pass"], cmap="bwr",
label="Students")
x_min, x_max = X["Hours_Study"].min()-5, X["Hours_Study"].max()+5
y_min, y_max = X["Hours_Sleep"].min()-1, X["Hours_Sleep"].max()+1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))
 grid = np.c_[xx.ravel(), yy.ravel()]
probs = model.predict_proba(grid)[:, 1].reshape(xx.shape)
# Contour plot  (boundary at probability = 0.5)
plt.contour(xx, yy, probs, levels=[0.5], cmap="Greys", vmin=0, vmax=0.6)
plt.xlabel("Hours of Study")
plt.ylabel("Hours of Sleep")
plt.title("Decision Boundary: Pass vs Fail")
plt.legend()
plt.show()
```

output:





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.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
data = pd.read_excel("student_pass_fail_20.xlsx")
X = data[["Hours_Study", "Hours_Sleep"]]
y = data["Pass"]
model = LogisticRegression()
model.fit(X, y)
new_student = np.array([[30, 6]])
prob_pass = model.predict_proba(new_student)[0][1]
print(f"Predicted Probability of Passing: {prob_pass:.4f}")
plt.figure(figsize=(7,5))
plt.scatter(data["Hours_Study"], data["Hours_Sleep"], c=data["Pass"], cmap="bwr",
label="Students")
x_min, x_max = X["Hours_Study"].min()-5, X["Hours_Study"].max()+5
y_min, y_max = X["Hours_Sleep"].min()-1, X["Hours_Sleep"].max()+1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))
 grid = np.c_[xx.ravel(), yy.ravel()]
probs = model.predict_proba(grid)[:, 1].reshape(xx.shape)
# Contour plot (boundary at probability = 0.5)
plt.contour(xx, yy, probs, levels=[0.5], cmap="Greys", vmin=0, vmax=0.6)
plt.xlabel("Hours of Study")
plt.ylabel("Hours of Sleep")
plt.title("Decision Boundary: Pass vs Fail")
plt.legend()
plt.show()
```

output:

Model Evaluation Results:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
Patient Info → BMI=28, Age=45,
Glucose=150 Predicted: Diabetic
Probability of being diabetic: 0.GGG8

Q5. Comparison – Linear vs Logistic Regression

Dataset columns:

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

Task:

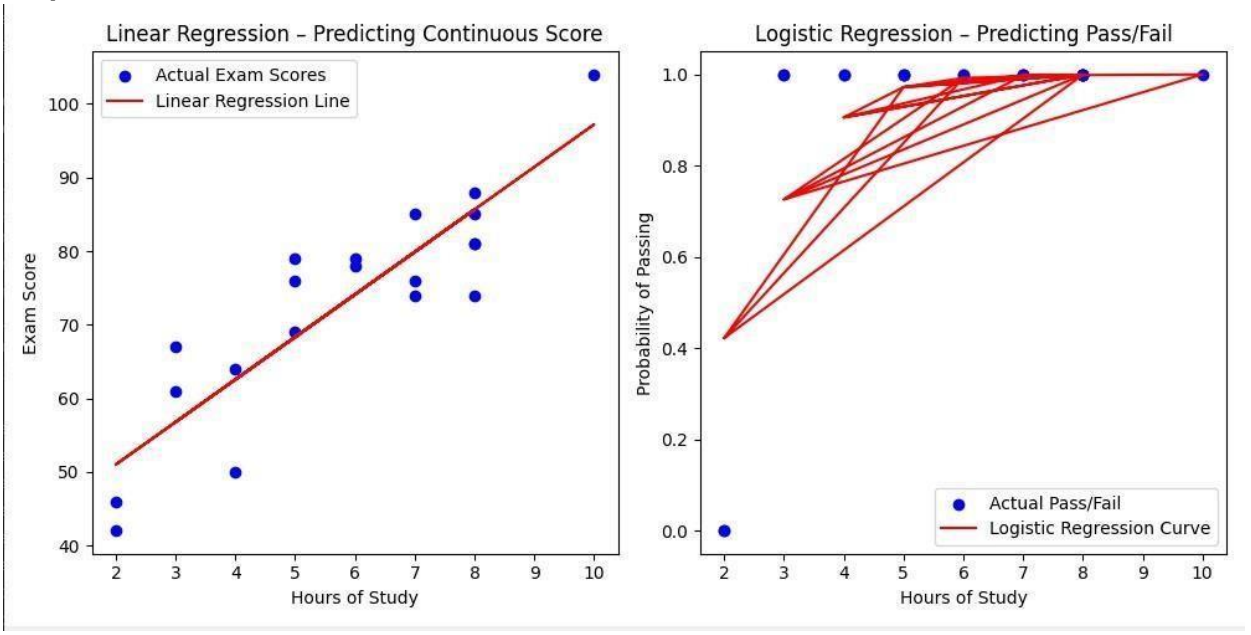
1. Use **Linear Regression** to predict exam scores.
2. Use **Logistic Regression** to predict pass/fail.
3. Compare results — explain why linear regression is unsuitable for classification.

Code:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, LogisticRegression
import matplotlib.pyplot as plt
data = pd.read_excel("linear_vs_logistic.xlsx")
X = data[["Hours_Study"]]
y_score = data["Exam_Score"]
y_pass = data["Pass"]
lin_model = LinearRegression()
lin_model.fit(X, y_score)
score_pred = lin_model.predict(X)
log_model = LogisticRegression()
log_model.fit(X, y_pass)
pass_pred_prob = log_model.predict_proba(X)[:, 1]
pass_pred = log_model.predict(X)
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.scatter(X, y_score, color='blue', label='Actual Exam Scores')
plt.plot(X, score_pred, color='red', label='Linear Regression Line')
plt.xlabel("Hours of Study")
plt.ylabel("Exam Score")
plt.title("Linear Regression – Predicting Continuous Score")
plt.legend()
```

```
plt.subplot(1,2,2)
plt.scatter(X, y_pass, color='blue', label='Actual Pass/Fail')
plt.plot(X, pass_pred_prob, color='red', label='Logistic Regression Curve')
plt.xlabel("Hours of Study")
plt.ylabel("Probability of Passing")
plt.title("Logistic Regression – Predicting Pass/Fail")
plt.legend()
plt.tight_layout()
plt.show()
print(" Comparison Results:")
print(f"Linear Regression predicts continuous scores (e.g., {score_pred[:5].round(2)})")
print(f"Logistic Regression predicts probabilities (e.g., {pass_pred_prob[:5].round(2)})")
print("\n+ Linear regression is unsuitable for classification because:")
print(" - It can predict values below 0 or above 1, which are invalid probabilities.")
print(" - It assumes a linear relationship, while classification requires an S-shaped (sigmoid) curve.")
print("  - Logistic regression outputs probabilities that can be thresholded (e.g., >0.5 → Pass).")
```

output:



') 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.

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 3

Decision Tree Classifier

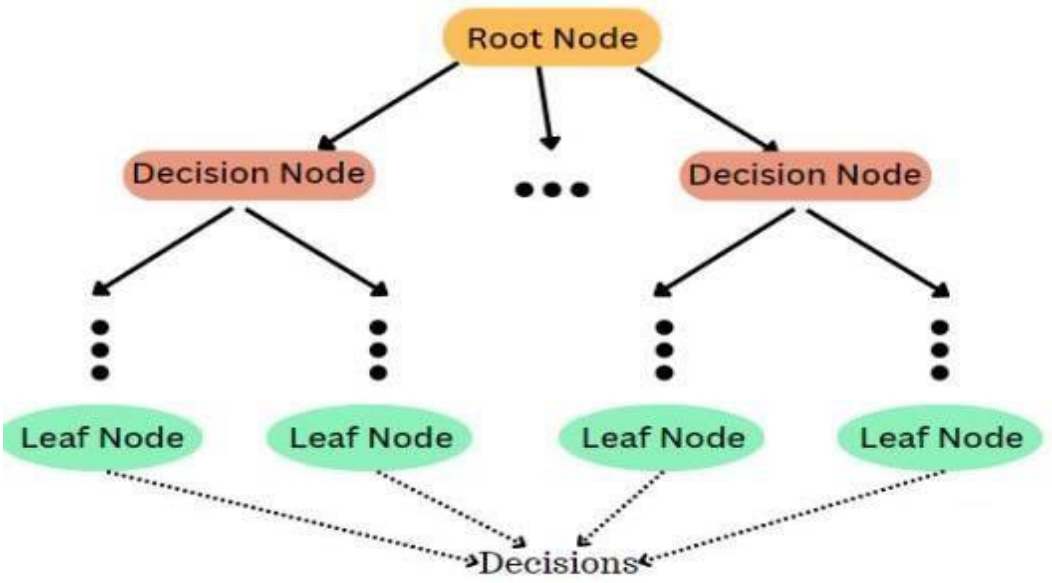
In this lab, students will study and implement the Decision Tree Classifier, a supervised machine learning algorithm used for classification tasks. The lab begins with a manual calculation of entropy and information gain to understand how decision trees choose splitting attributes. Students will then 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 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)

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)

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)
```

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:

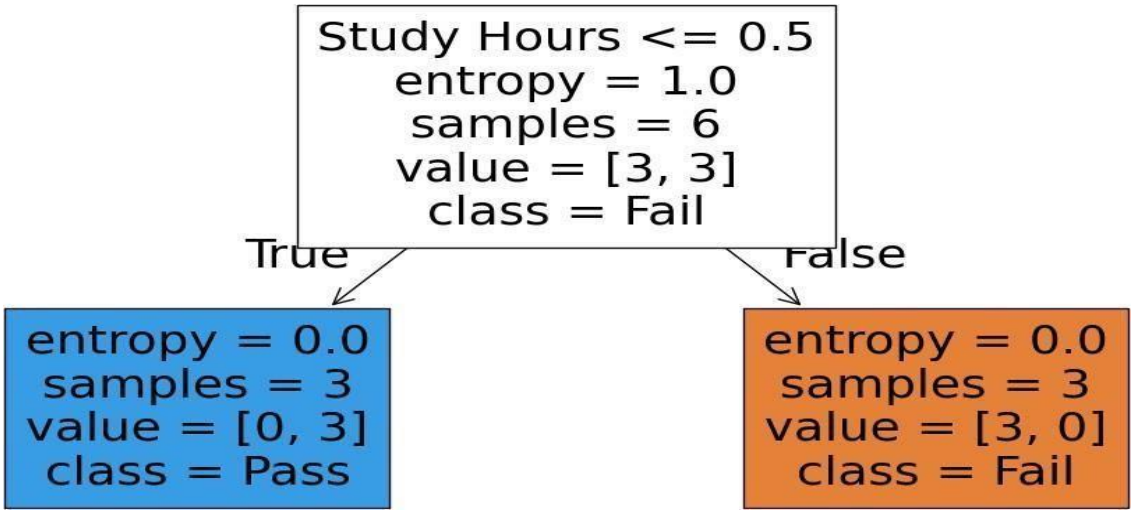
1. Use pandas to create a DataFrame.
2. Convert categorical values into numerical using LabelEncoder.
3. Train a **DecisionTreeClassifier** using **criterion='entropy'**.
4. Visualize the decision tree using plot_tree() from sklearn.tree.

Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree import matplotlib.pyplot as plt
data = {
'Study Hours': ['Low', 'Low', 'High', 'High', 'Low', 'High'],
'Attendance': ['Good', 'Poor', 'Good', 'Poor', 'Good', 'Poor'],
'Result': ['Fail', 'Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}
df = pd.DataFrame(data) le = LabelEncoder()
df['Study Hours'] = le.fit_transform(df['Study Hours']) df['Attendance'] =
le.fit_transform(df['Attendance']) df['Result'] = le.fit_transform(df['Result']) X =
df[['Study Hours', 'Attendance']] y = df['Result']
model = DecisionTreeClassifier(criterion='entropy') model.fit(X, y) plt.figure(figsize=(10, 6))
plot_tree(model, feature_names=['Study Hours', 'Attendance'], class_names=['Fail', 'Pass'],
filled=True)
plt.show()
prediction = model.predict([[0, 0]])
result = 'Pass' if prediction[0] == 1 else 'Fail'
print("Prediction for Study Hours=Low and Attendance=Good:", result)
```

Output:

Prediction for Study Hours=Low and Attendance=Good: Pass



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.

Code:

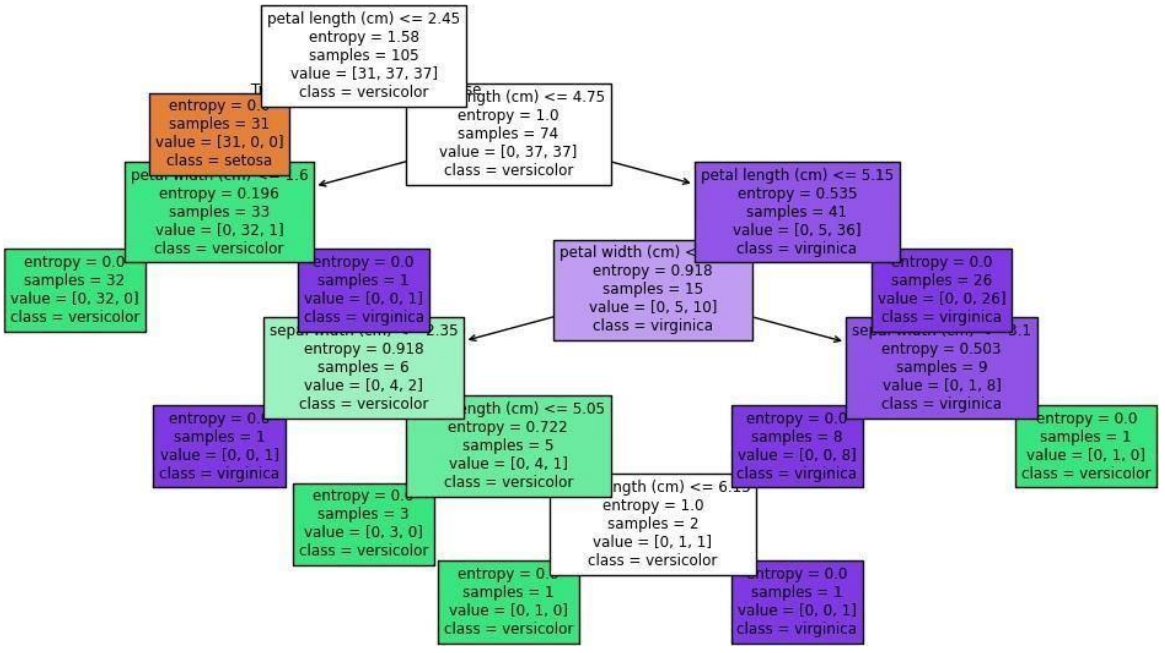
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,
plot_tree from sklearn.metrics
 import accuracy_score
import matplotlib.pyplot as plt iris =
load_iris() X = iris.data y = iris.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42) model = DecisionTreeClassifier(criterion='entropy', random_state=42)
model.fit(X_train, y_train) y_pred = model.predict(X_test)
print("✓ Accuracy on Test Set:",
accuracy_score(y_test, y_pred))
plot_tree( model,
feature_names=iris.feature_names,
class_names=iris.target_names, filled=True) plt.show()
sample = [[5.1, 3.5, 1.4, 0.2]] # example flower measurements
 predicted_class = model.predict(sample)
print(f"Prediction for sample {sample}: {iris.target_names[predicted_class][0]}")
```

Output:

✓ Accuracy on Test Set: 0.G777777777777777

Prediction for sample [[5.1, 3.5, 1.4, 0.2]]: setosa



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

Code:

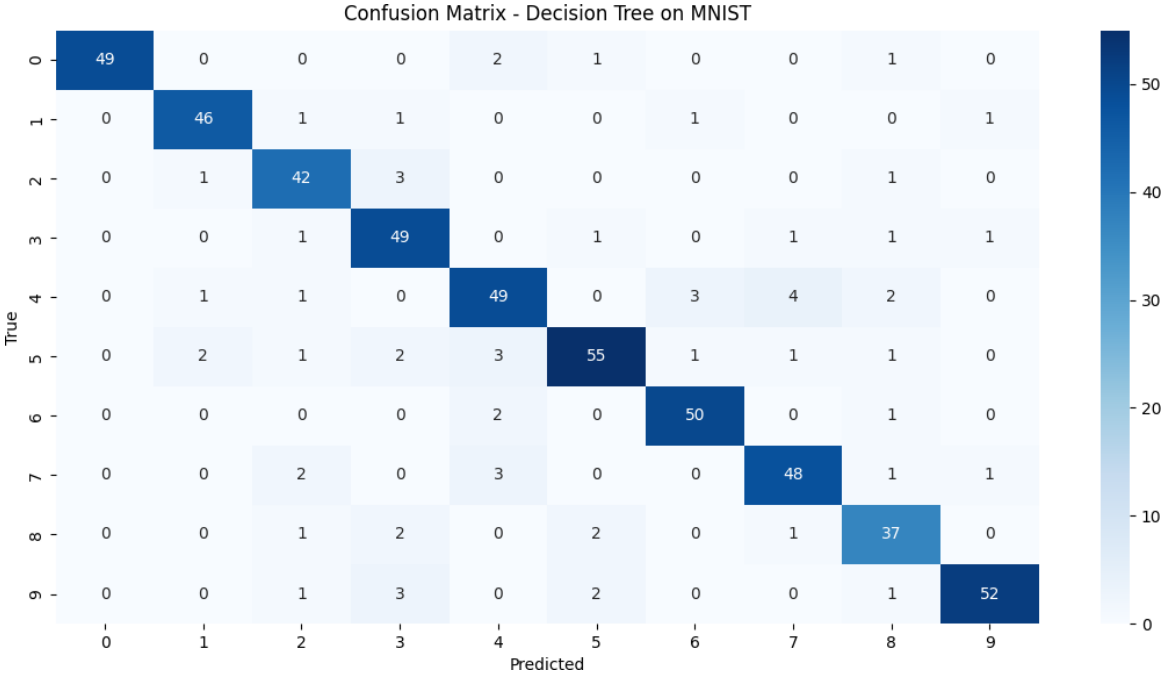
```
from sklearn.datasets import load_digits

from sklearn.model_selection import train_test_split from sklearn.tree
import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns

import matplotlib.pyplot as plt digits = load_digits()

X = digits.data# Flattened 8x8 images (64 features per sample) y = digits.target # Labels

0–9 X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=42)

model
= DecisionTreeClassifier(criterion='entropy', random_state=42)

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

acc = accuracy_score(y_test, y_pred) print("✓ Accuracy on Test Set:", acc)

print("\n # ]¡ Classification Report:\n", classification_report(y_test, y_pred)) cm =
confusion_matrix(y_test, y_pred)

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

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues') plt.title("Confusion Matrix -
Decision Tree on MNIST") plt.xlabel("Predicted")

plt.ylabel("True") plt.show()

plt.figure(figsize=(10,2))

for index, (image, label) in enumerate(zip(digits.images[:5], digits.target[:5])):
plt.subplot(1, 5, index + 1)

plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest') plt.title(f'True:
```

Output:

Accuracy on Test Set: 0.8833333333333333

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.G2	0.G6	53
1	0.G2	0.G2	0.G2	50
2	0.84	0.8G	0.87	47
3	0.82	0.G1	0.86	54
4	0.83	0.82	0.82	60
5	0.G0	0.83	0.87	66
6	0.G1	0.G4	0.G3	53
7	0.87	0.87	0.87	55
8	0.80	0.86	0.83	43
G	0.G5	0.88	0.G1	5G
accuracy			0.88	540
macro avg	0.88	0.8G	0.88	540
weighted avg	0.8G	0.88	0.88	540



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. 4

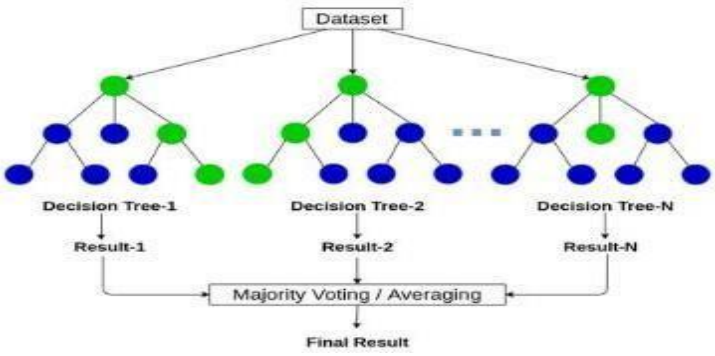
Random Forest and Support Vector Machine Classifier

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 s 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:

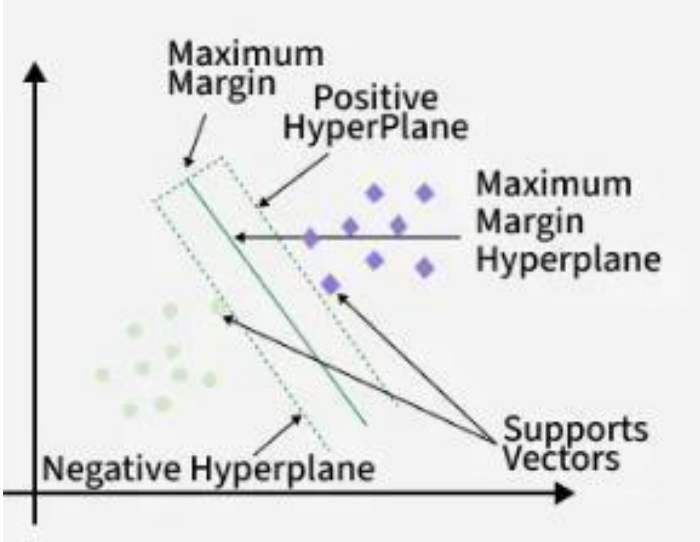
- High accuracy
- Reduces overfitting
- Works well with large dataset

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.

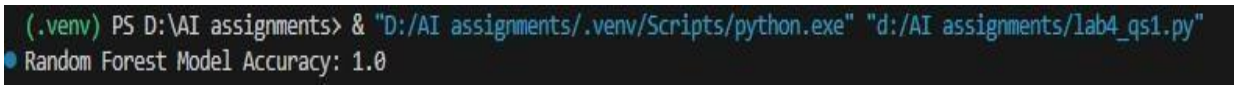
Task:

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**.

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
iris =
load_iris
() X =
iris.data
y
= iris.target
# 3. Split into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) #
4. Train a Random Forest Classifier
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

# 5. Predict flower species on the test set y_pred
= rf.predict(X_test)
```

Output:



Question 2

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

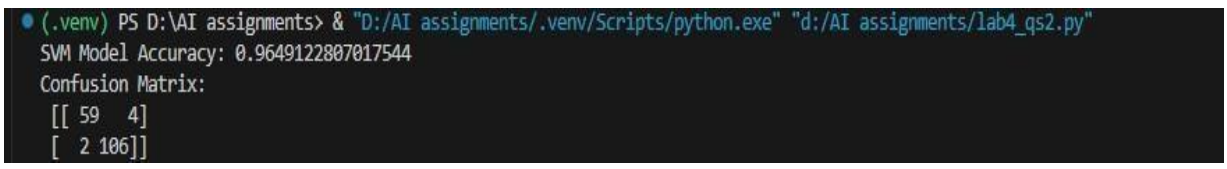
Task:

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**.

Code:

```
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 cancer
= load_breast_cancer()
X = cancer.data
y = cancer.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 4. Train SVM classifier with linear kernel svm_model
= SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# 5. Make predictions on the test set
y_pred = svm_model.predict(X_test)

# 6. Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# 7. Print results
print("SVM Model Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
```

output:



Question 3

Use Random Forest on CSV Dataset (Custom)

: Predict student pass/fail based on study hours and scores.

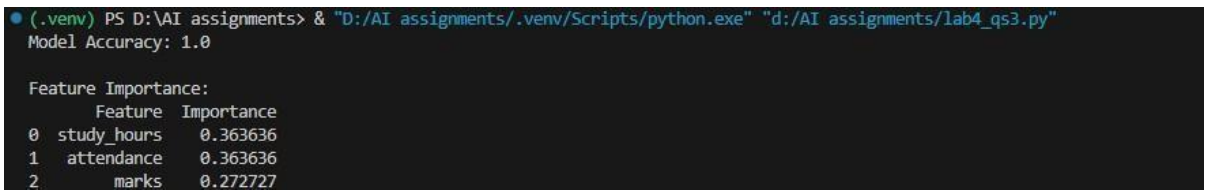
Task:

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**.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import
RandomForestClassifier from sklearn.metrics import
accuracy_score
data = pd.read_csv("students.csv")
X = data[['study_hours', 'attendance', 'marks']] y
= data['result']
y = y.map({'Fail': 0, 'Pass': 1})
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred =
rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
feature_importance = pd.DataFrame({
   'Feature': X.columns,
   'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:")
print(feature_importance)
```

output:



Question 4

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

Task:

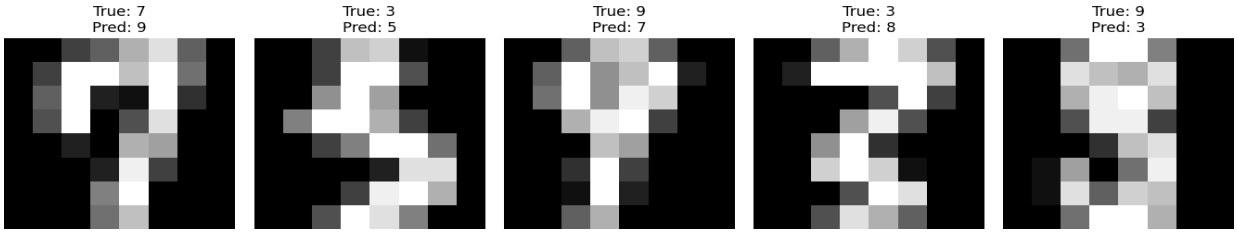
1. Load the *Digits dataset* from sklearn.datasets.load_digits.
2. Train an **SVM classifier** with an RBF kernel.
3. Test on unseen data.
4. Print **accuracy** and visualize some **misclassified samples**.

Code:

```
# Import required libraries
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
import numpy as np digits
= load_digits() X
= digits.data y
= digits.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) svm_model
= SVC(kernel='rbf', gamma=0.001, C=10) # RBF kernel
svm_model.fit(X_train, y_train) y_pred
= svm_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred) print("SVM
Model Accuracy:", accuracy)
misclassified_indices = np.where(y_test != y_pred)[0]
print(f"\nNumber of misclassified samples: {len(misclassified_indices)}")

plt.figure(figsize=(10, 4))
  for i, index in enumerate(misclassified_indices[:5]):
     plt.subplot(1, 5, i + 1)
     plt.imshow(X_test[index].reshape(8, 8), cmap='gray')
     plt.title(f"True:                    {y_test[index]}\nPred:
     {y_pred[index]}") plt.axis('off') plt.tight_layout()

plt.show()
```

output:

```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs4.py"
SVM Model Accuracy: 0.9907407407407407
```



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.

Code:

```
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 wine
= load_wine() X
= wine.data y
= wine.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 3. Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
```

```
# 4. Train SVM with RBF kernel
svm_model = SVC(kernel='rbf', gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test) svm_accuracy
= accuracy_score(y_test, svm_pred)

# 5. Print accuracy of both models
print("Random Forest Accuracy:", rf_accuracy)
print("SVM (RBF) Accuracy:", svm_accuracy)

# 6. Conclude which model performs better if
rf_accuracy > svm_accuracy:
  print("\n✅ Random Forest performs better on the Wine dataset.") elif
svm_accuracy > rf_accuracy:
  print("\n✅ SVM (RBF) performs better on the Wine dataset.") else:
    print("\n      Both models perform equally well on the Wine dataset.")
```

output:

```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs5.py"
Random Forest Accuracy: 1.0
SVM (RBF) 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

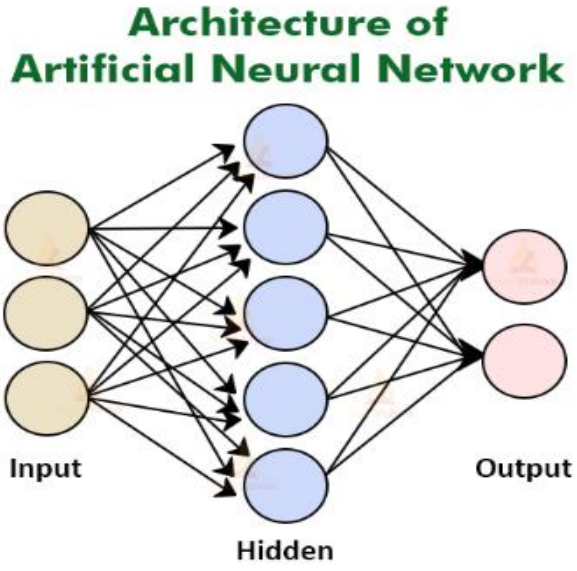
In this lab, students will learn the fundamentals and implementation of an Artificial Neural Network (ANN) for classification and prediction tasks. The lab introduces how neural networks are inspired by the human brain and how they learn patterns from data using layers of interconnected neurons. Students 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

LAB Assignment No. 5

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:

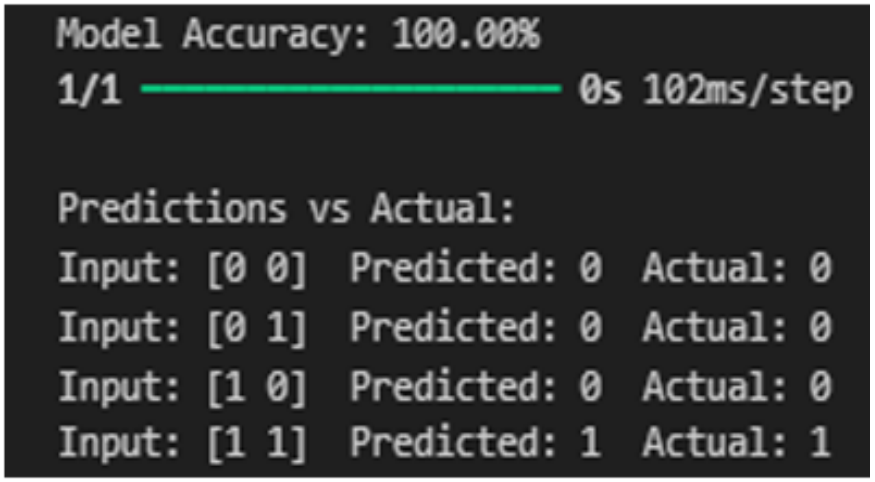
1. Create dataset using NumPy or pandas.
2. Build a neural network with one hidden layer using TensorFlow/Keras or PyTorch.
3. Train it and show accuracy.
4. Compare model predictions with actual outputs

Code:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0], [0], [0], [1]])
model = Sequential()
model.add(Dense(4, input_dim=2, activation='relu'))  neurons
model.add(Dense(1, activation='sigmoid'
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=500, verbose=0)
```

```
loss, accuracy = model.evaluate(X, y, verbose=0)

print(f"Model Accuracy: {accuracy*100:.2f}%")

predictions = model.predict(X)

predicted_classes = (predictions > 0.5).astype(int)

print("\nPredictions vs Actual:")

for i in range(len(X)):

    print(f"Input: {X[i]}  Predicted: {predicted_classes[i][0]}  Actual: {y[i][0]}")
```

Output:



Question 2

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

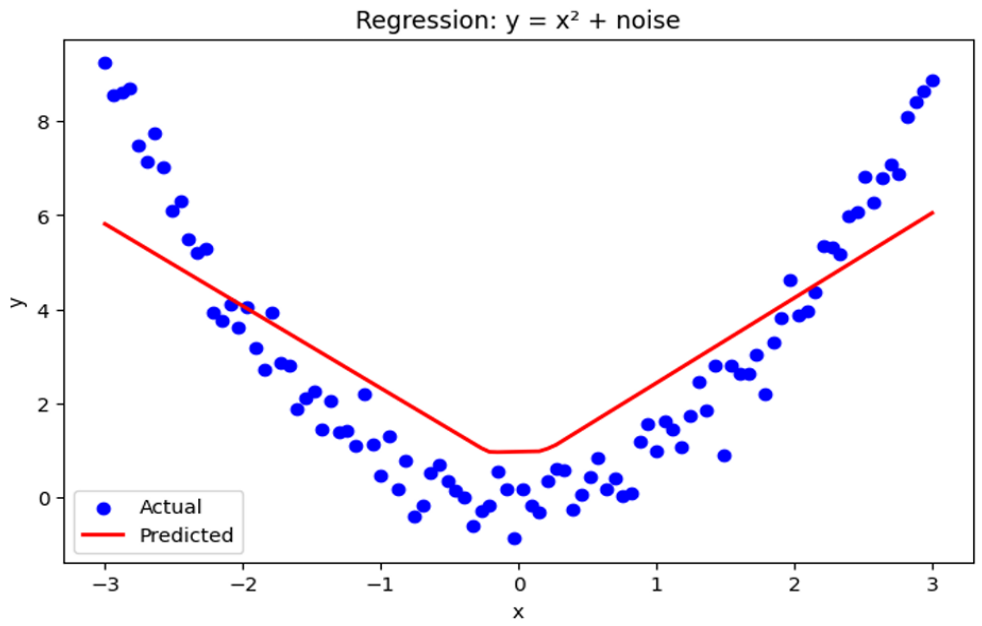
Tasks:

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.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
np.random.seed(42)
X = np.linspace(-3, 3, 100)
y = X**2 + np.random.normal(0, 0.5, X.shape)   # Add Gaussian noise
# Reshape for Keras (expects 2D inputs)
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)
model = Sequential()
model.add(Dense(10, input_dim=1, activation='relu'))   # Hidden layer (10 neurons)
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200, verbose=0)
y_pred = model.predict(X)
plt.figure(figsize=(8,5))
plt.scatter(X, y, label='Actual', color='blue')
plt.plot(X, y_pred, label='Predicted', color='red', linewidth=2)
plt.title("Regression: y = x^2 + noise")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```

Output:



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.

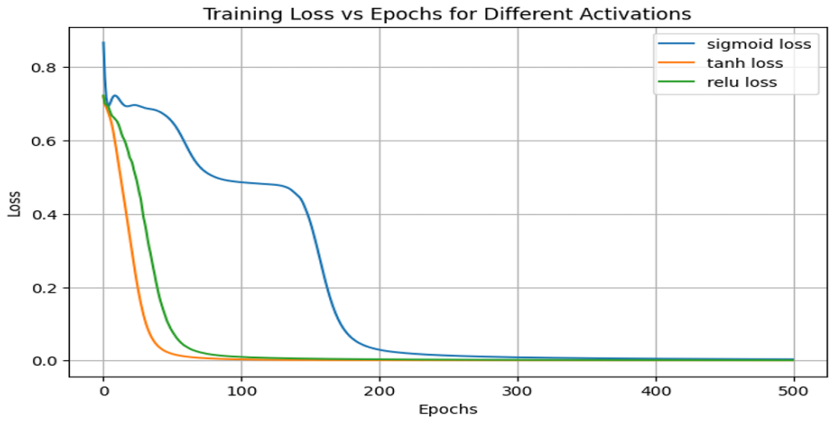
Code:

```
import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

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

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

def train_xor_model(activation, epochs=500):  model = Sequential([Dense(4, input_dim=2, activation=activation),  Dense(1, activation='sigmoid') ])
```

```
 return model, history, loss, acc
activations = ['sigmoid', 'tanh', 'relu']
results = {}
for act in activations:model, history, loss, acc = train_xor_model(act)
results[act] = {'loss': loss, 'acc': acc, 'history': history}
   print(f"{act.upper()} → Accuracy: {acc*100:.2f}%, Final Loss: {loss:.4f}")
plt.figure(figsize=(8,5))
for act in activations:plt.plot(results[act]['history'].history['loss'], label=f'{act} loss')
plt.title("Training Loss vs Epochs for Different Activations")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
SIGMOID → Accuracy: 100.00%, Final Loss: 0.0031
TANH → Accuracy: 100.00%, Final Loss: 0.0002
RELU → Accuracy: 100.00%, Final Loss: 0.0007
```



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**.

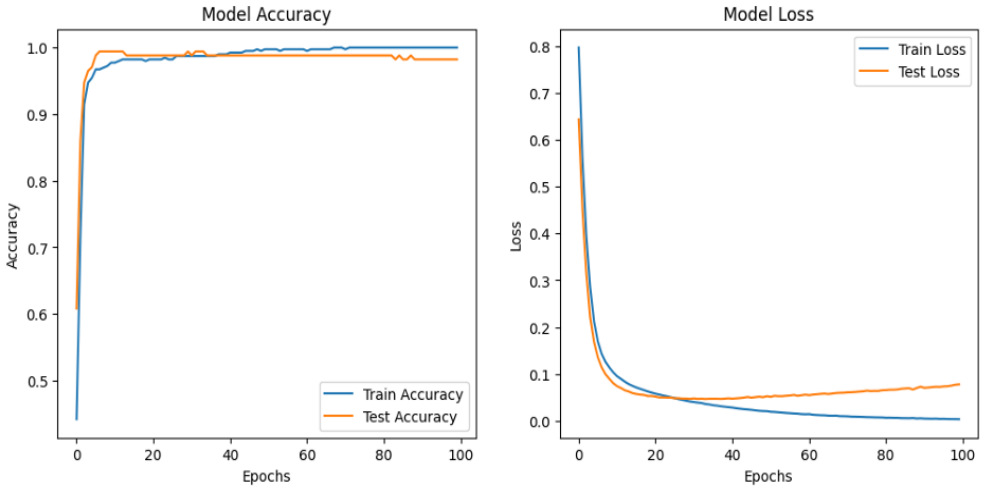
Code:

```
import numpy as np

from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

data = load_breast_cancer()

X = data.data

y = data.target  # 0 = malignant, 1 = benign

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

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

model = Sequential([Dense(16, input_dim=X.shape[1],

activation='relu'),   layer 1Dense(8, activation='relu'),

Dense(1, activation='sigmoid')])model.compile(optimizer='adam',

 loss='binary_crossentropy', metrics=['accuracy'])
```

```
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"✅ Test Accuracy: {accuracy*100:.2f}%")
print(f"Loss: {loss:.4f}")
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Test Accuracy')
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
sample = X_test[0].reshape(1, -1)
prediction = model.predict(sample)
print(f"Predicted Class: {'Benign' if prediction[0][0] > 0.5 else 'Malignant'}")
```

Output:

```
✅ Test Accuracy: 98.25%
Loss: 0.0782
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
Predicted Class: Benign
```

Question 5

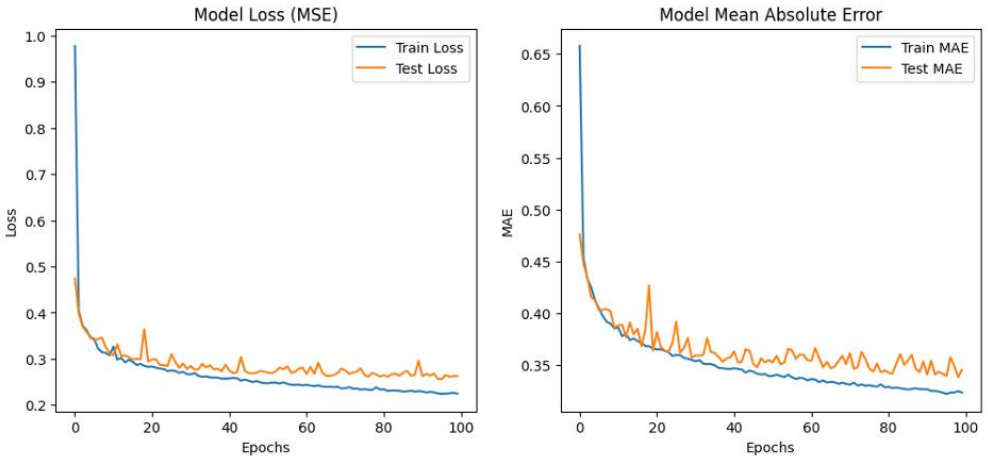
Multi-Class Classification on Iris Dataset

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

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
iris = load_iris()
X = iris.data
y = iris.target
```

```
y_encoded = to_categorical(y)

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

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

model = Sequential([Dense(8, input_dim=4, activation='relu'),   # hidden layer 1

Dense(6, activation='relu'),Dense(3, activation='softmax') neurons = 3 classes)])

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

history = model.fit(X_train, y_train, validation_data=(X_test, y_test),epochs=100, batch_size=8, verbose=0)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val_accuracy'], label='Test Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.ylabel('Loss')

plt.legend()

plt.show()

sample = np.array([[5.1, 3.5, 1.4, 0.2]])  # Example: Setosa

sample_scaled = scaler.transform(sample)

prediction = model.predict(sample_scaled)

predicted_class = np.argmax(prediction)

print(f"Predicted Species: {iris.target_names[predicted_class]}")
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
Predicted Price: $50448.07
```

Question 6

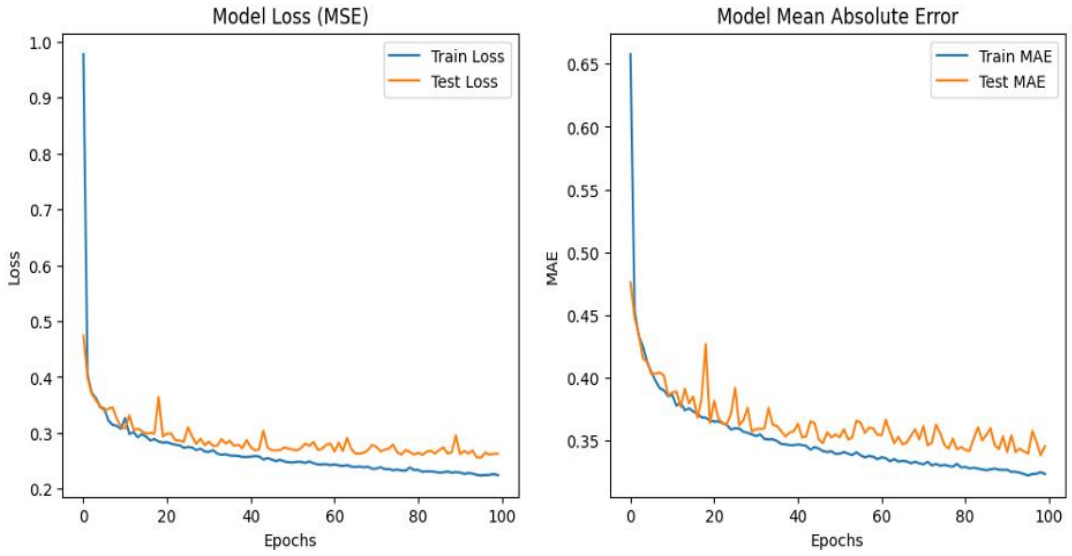
Regression Problem (House Price Prediction)

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

Code:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import fetch_california_housing

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

data = fetch_california_housing()

X = data.data

y = data.target   # Median house value (in 100,000s)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
model = Sequential([Dense(64, input_dim=X.shape[1],activation='relu'),  Dense(32, activation='relu'), Dense(1)]

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

istory = model.fit(X_train, y_train, validation_data=(X_test, y_test),

epochs=100, batch_size=32, verbose=0)loss, mae = model.evaluate(X_test, y_test, verbose=0)

print(f"✅ Test MAE (Mean Absolute Error): {mae:.3f}")

print(f"Test MSE: {loss:.3f}")

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.title('Model Loss (MSE)')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.subplot(1,2,2)

plt.plot(history.history['mae'], label='Train MAE')

plt.plot(history.history['val_mae'], label='Test MAE')

plt.title('Model Mean Absolute Error')

plt.xlabel('Epochs')

plt.ylabel('MAE')

plt.legend()

plt.show()

sample = X_test[0].reshape(1, -1)

predicted_price = model.predict(sample)[0][0]

print(f"Predicted Price: ${predicted_price * 100000:.2f}")
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
Predicted Price: $50448.07
```

Question 7

Neural Network with Dropout Regularization

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

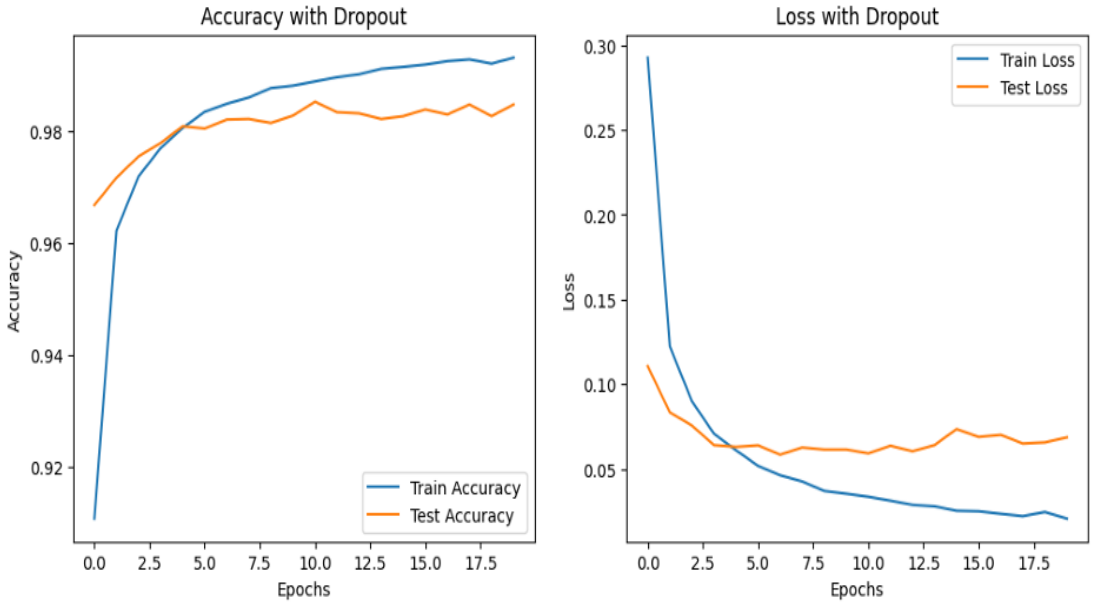
Code:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.utils import to_categorical
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

```
X_train = X_train.reshape(-1, 28*28)

X_test = X_test.reshape(-1, 28*28)

y_train = to_categorical(y_train, 10)

y_test = to_categorical(y_test, 10)

model = Sequential([Dense(512, input_dim=784, activation='relu'),

Dropout(0.3),  Denes(256, activation='relu'),  Dropout(0.3),
Dense(10,activation='softmax') ])

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

loss, accuracy = model.evaluate(X_test, y_test, verbose=0)

print(f"✅ Test Accuracy: {accuracy*100:.2f}%")

print(f"Test Loss: {loss:.4f}")

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.plot(history.history['val_accuracy'], label='Test Accuracy')

plt.title('Accuracy with Dropout')

plt.legend()

plt.subplot(1,2,2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.title('Loss with Dropout')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()
```

Output:

```
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 14s 22ms/step - accuracy: 0.8350 - loss: 0.5230 - val_accuracy: 0.9668 - val_loss: 0.1107
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9608 - loss: 0.1280 - val_accuracy: 0.9717 - val_loss: 0.0834
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9720 - loss: 0.0910 - val_accuracy: 0.9755 - val_loss: 0.0757
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9777 - loss: 0.0695 - val_accuracy: 0.9779 - val_loss: 0.0641
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.9820 - loss: 0.0575 - val_accuracy: 0.9809 - val_loss: 0.0631
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 8s 18ms/step - accuracy: 0.9843 - loss: 0.0492 - val_accuracy: 0.9805 - val_loss: 0.0640
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.9861 - loss: 0.0426 - val_accuracy: 0.9821 - val_loss: 0.0584
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9866 - loss: 0.0411 - val_accuracy: 0.9822 - val_loss: 0.0627
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9878 - loss: 0.0372 - val_accuracy: 0.9815 - val_loss: 0.0615
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 19ms/step - accuracy: 0.9892 - loss: 0.0329 - val_accuracy: 0.9828 - val_loss: 0.0615
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 21ms/step - accuracy: 0.9901 - loss: 0.0309 - val_accuracy: 0.9853 - val_loss: 0.0592
Epoch 12/20
...
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 21ms/step - accuracy: 0.9938 - loss: 0.0192 - val_accuracy: 0.9848 - val_loss: 0.0687
✅ Test Accuracy: 98.48%
Test Loss: 0.0687
```



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

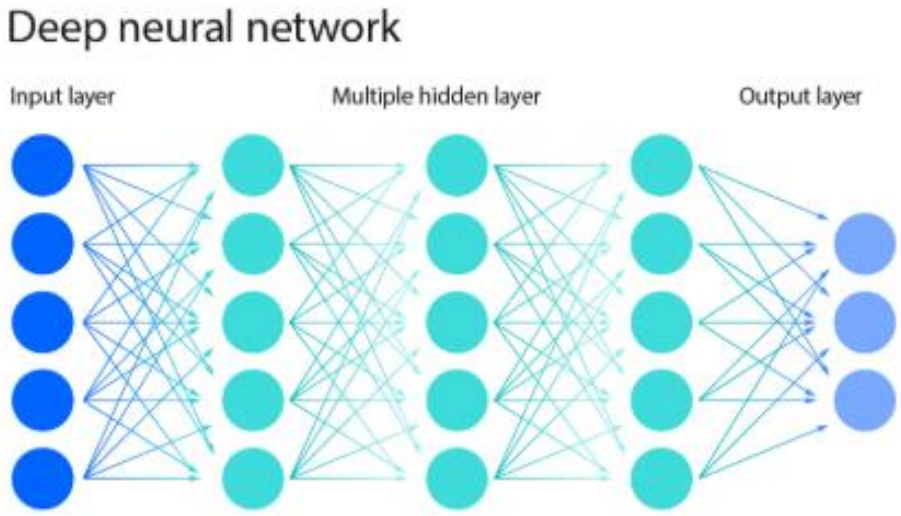
LAB No. 6

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*.

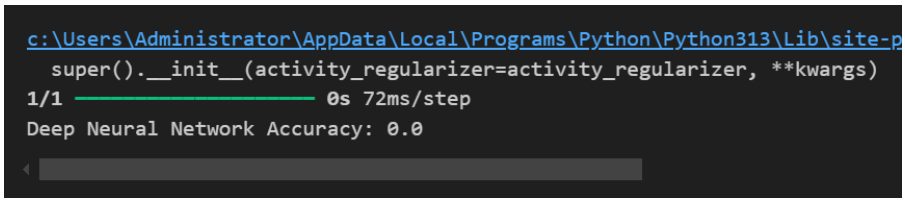
- Use **at least three hidden layers**
- Experiment with different numbers of neurons
- Compare the accuracy with a shallow ANN (one hidden layer)

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score


# Sample dataset
X = np.array([
   [2, 60], [4, 70], [6, 80], [8, 90],
   [1, 50], [3, 65], [7, 85], [9, 95]
])
y = np.array([0, 0, 1, 1, 0, 0, 1, 1])


# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

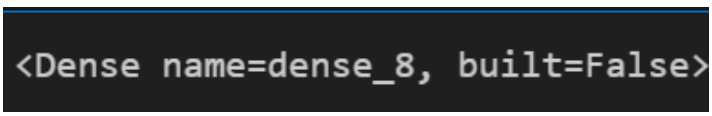
```
# Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Shallow ANN
shallow_model = Sequential([
   Dense(8, activation='relu', input_shape=(2,)),
   Dense(1, activation='sigmoid')
])

shallow_model.compile(optimizer='adam',
                loss='binary_crossentropy',
                metrics=['accuracy'])

shallow_model.fit(X_train, y_train, epochs=50, verbose=0)
# Prediction
y_pred = (shallow_model.predict(X_test) > 0.5).astype(int)
shallow_accuracy = accuracy_score(y_test, y_pred)
print("Shallow ANN Accuracy:", shallow_accuracy)
```

```
c:\Users\Administrator\AppData\Local\Programs\Python\Python313\Lib\site-
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
Shallow ANN Accuracy: 0.0
```

```
# Deep Neural Network
deep_model = Sequential([
   Dense(16, activation='relu', input_shape=(2,)),
   Dense(12, activation='relu'),
   Dense(8, activation='relu'),
   Dense(1, activation='sigmoid')
])
deep_model.compile(optimizer='adam',
                   loss='binary_crossentropy',
                   metrics=['accuracy'])
deep_model.fit(X_train, y_train, epochs=50, verbose=0)
# Prediction
y_pred_deep = (deep_model.predict(X_test) > 0.5).astype(int)
deep_accuracy = accuracy_score(y_test, y_pred_deep)
print("Deep Neural Network Accuracy:", deep_accuracy)
```




```
Dense(32, activation='relu')
Dense(16, activation='relu')
Dense(8, activation='relu')
```



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.

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


# Load data
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=0.25, random_state=42
)


# Feature scaling
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model_relu = Sequential([
    Dense(16, activation='relu', input_shape=(4,)),
    Dense(12, activation='relu'),
    Dense(8, activation='relu'),
    Dense(3, activation='softmax')
])
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model_relu = Sequential([
    Dense(16, activation='relu', input_shape=(4,)),
    Dense(12, activation='relu'),
    Dense(8, activation='relu'),
    Dense(3, activation='softmax')
])
```

```
ReLU Model Accuracy: 0.8684210777282715
```

```
model_tanh = Sequential([
   Dense(16, activation='tanh', input_shape=(4,)),
   Dense(12, activation='tanh'),
   Dense(8, activation='tanh'),
   Dense(3, activation='softmax')
])

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

history_tanh = model_tanh.fit(
   X_train, y_train,
   epochs=50,
   validation_split=0.2,
   verbose=0
)

tanh_accuracy = model_tanh.evaluate(X_test, y_test, verbose=0)[1]
print("tanh Model Accuracy:", tanh_accuracy)
```

```
tanh Model Accuracy: 0.8947368264198303
```

Observations

- **ReLU outperformed tanh** in terms of both accuracy and training speed.
- ReLU enabled **faster learning and better gradient flow** in deep layers.
- tanh worked well but required **more epochs** to reach similar accuracy.
- For deeper neural networks, **ReLU is generally preferred**.

Practice Question 3:

Hyperparameter Tuning in DNN

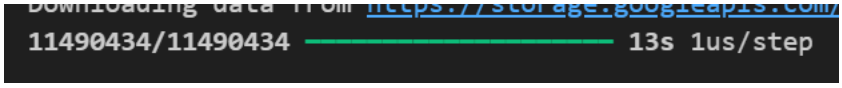
Train a DNN on the **MNIST dataset** by changing:

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

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

Common Preprocessing

```
from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.utils import to_categorical

import time

# Load dataset

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

# Flatten images

X_train = X_train.reshape(-1, 784) / 255.0

X_test = X_test.reshape(-1, 784) / 255.0

# One-hot encoding

y_train = to_categorical(y_train)

y_test = to_categorical(y_test)
```



Base Model Function

```
def build_model(layers, neurons):
  model = Sequential()
  model.add(Dense(neurons, activation='relu', input_shape=(784,)))

  for _ in range(layers - 1):
    model.add(Dense(neurons, activation='relu'))

  model.add(Dense(10, activation='softmax'))

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

Experiment 1: Varying Number of Hidden Layers

```
for layers in [1, 2, 3]:
  model = build_model(layers, 128)
  start = time.time()
  model.fit(X_train, y_train, epochs=5, batch_size=32, verbose=0)
  duration = time.time() - start
  acc = model.evaluate(X_test, y_test, verbose=0)[1]
  print(layers, "layers → Accuracy:", acc, "Time:", duration)
```

```
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1 layers → Accuracy: 0.9768999814987183 Time: 17.460973262786865
2 layers → Accuracy: 0.9706000089645386 Time: 16.407267808914185
3 layers → Accuracy: 0.9733999967575073 Time: 17.572629928588867
```

Experiment 2: Varying Number of Neurons per Layer

```
for neurons in [64, 128, 256]:
    model = build_model(2, neurons)
    start = time.time()
    model.fit(X_train, y_train, epochs=5, batch_size=32, verbose=0)
    duration = time.time() - start
    acc = model.evaluate(X_test, y_test, verbose=0)[1]
    print(neurons, "neurons → Accuracy:", acc, "Time:", duration)
```

```
64 neurons → Accuracy: 0.9733999967575073 Time: 12.973463296890259
128 neurons → Accuracy: 0.9768000245094299 Time: 19.133976936340332
256 neurons → Accuracy: 0.9778000116348267 Time: 24.602784633636475
```

Generate + Code +

Experiment 3: Varying Batch Size

```
for batch in [16, 32, 64]:
    model = build_model(2, 128)
    start = time.time()
    model.fit(X_train, y_train, epochs=5, batch_size=batch, verbose=0)
    duration = time.time() - start
    acc = model.evaluate(X_test, y_test, verbose=0)[1]
    print("Batch size", batch, "→ Accuracy:", acc, "Time:", duration)
```

```
Batch size 16 → Accuracy: 0.9789999723434448 Time: 29.874204397201538
Batch size 32 → Accuracy: 0.9754999876022339 Time: 16.57361674308777
Batch size 64 → Accuracy: 0.9778000116348267 Time: 10.232374906539917
```

Observations

1. Effect of Hidden Layers

- Increasing layers **improves accuracy** slightly.
- More layers increase **training time**.
- Very deep networks offer **diminishing returns** on MNIST.

2. Effect of Neurons per Layer

- More neurons improve model capacity and accuracy.
- Large layers significantly increase computation cost.
- Risk of overfitting increases with very large layers.

3. Effect of Batch Size

- Smaller batch sizes give **better generalization**.
- Larger batch sizes train faster but may reduce accuracy.
- Batch size = 32 gives a good balance.

Practice Question 4:

Overfitting and Regularization

Build a deep neural network on any classification dataset and:

```
from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.utils import to_categorical


# Load dataset

data = load_breast_cancer()

X = data.data

y = data.target
```

```
# Flatten images
X_train = X_train.reshape(-1, 784) / 255.0
X_test = X_test.reshape(-1, 784) / 255.0


# One-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Model Without Regularization (Overfitting Case)

Model Architecture

- Hidden layers: 3
- Neurons: 64 → 64 → 32
- Activation: ReLU

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


model_no_reg = Sequential([
    Dense(64, activation='relu', input_shape=(30,)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])


model_no_reg.compile(
    optimizer='adam',
```

```
    loss='binary_crossentropy',
    metrics=['accuracy']
)
history_no_reg = model_no_reg.fit(
    X_train, y_train,
    epochs=50,
    validation_split=0.2,
    verbose=0
)
```

Model With Dropout Regularization

Regularized Architecture

- Dropout rate: 0.5
- Same network depth

```
from tensorflow.keras.layers import Dropout


model_dropout = Sequential([
    Dense(64, activation='relu', input_shape=(30,)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

```
model_dropout.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

history_dropout = model_dropout.fit(
    X_train, y_train,
    epochs=50,
    validation_split=0.2,
    verbose=0
)
```

Performance Comparison

```
acc_no_reg = model_no_reg.evaluate(X_test, y_test, verbose=0)[1]
acc_dropout = model_dropout.evaluate(X_test, y_test, verbose=0)[1]

print("Accuracy without regularization:", acc_no_reg)
print("Accuracy with Dropout:", acc_dropout)
```

```
Accuracy without regularization: 0.9720279574394226
Accuracy with Dropout: 0.9720279574394226
```

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.

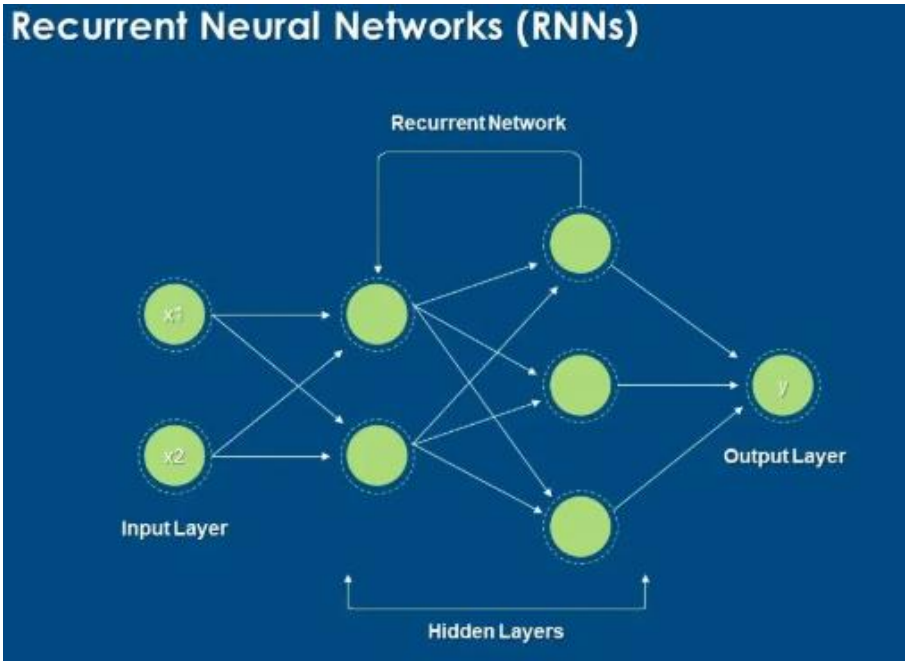
6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design 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 = 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 7

Recurrent Neural Network (RNN)

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:

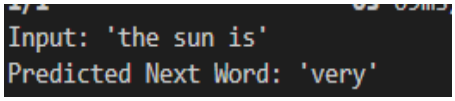
1. Load and clean the text data.
2. Tokenize and convert text into sequences.
3. Build a simple **RNN model** using keras.layers.SimpleRNN.
4. Train it to predict the next word given previous 3–5 words.
5. Test by entering a custom text prompt and predict the next word.

```
import numpy as np
import tensorflow as tf
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, SimpleRNN, Dense
text = """
The sun is shining bright
The sun is hot today
The weather is sunny
The sun is very bright
"""
```

```
for line in text.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_seq_len = max(len(seq) for seq in
input_sequences) input_sequences =
pad_sequences(input_sequences,
                        maxlen=max_seq
                         _len,
                         padding='pre')
X =
input_sequence
s[:, :-1] y =
input_sequence
s[:, -1]
y = tf.keras.utils.to_categorical(y,
num_classes=total_words) model = Sequential()
model.add(Embedding(total_words, 50, input_length=max_seq_len-1))
model.add(SimpleRNN(100))
model.add(Dense(total_words, activation='softmax'))
model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
model.summary()

model.fit(X, y, epochs=200, verbose=1)

def predict_next_word(model, tokenizer, text,
    max_seq_len): text = text.lower()
    sequence =
    tokenizer.texts_to_sequences([text])[0]
```

```
    prediction = model.predict(sequence, verbose=0)
    predicted_index = np.argmax(prediction)
    for word, index in tokenizer.word_index.items():
        if index == predicted_index:
            return word
input_text = "the sun
is"
predicted_word = predict_next_word(model, tokenizer, input_text, max_seq_len)
print(f"Input: '{input_text}'")
print(f"Predicted Next Word: '{predicted_word}'")
```

Output:



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).

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

from sklearn.preprocessing import
MinMaxScaler from
tensorflow.keras.models import
Sequential
from tensorflow.keras.layers import
SimpleRNN, Dense data = {
  "Date": [

    "2023-01-02","2023-01-03","2023-01-04","2023-01-05","2023-01-06",

    "2023-01-09","2023-01-10","2023-01-11","2023-01-12","2023-01-13",

    "2023-01-16","2023-01-17","2023-01-18","2023-01-19","2023-01-20",

    "2023-01-23","2023-01-24","2023-01-25","2023-01-26","2023-01-27",

    "2023-01-30","2023-01-31","2023-02-01","2023-02-02","2023-02-03",

    "2023-02-06","2023-02-07","2023-02-08","2023-02-09","2023-02-10",

    "2023-02-13","2023-02-14","2023-02-15","2023-02-16","2023-02-17",

    "2023-02-20","2023-02-21","2023-02-22","2023-02-23","2023-02-24",

    "2023-02-27","2023-02-28","2023-03-01","2023-03-02","2023-03-03",

    "2023-03-06","2023-03-07","2023-03-08","2023-03-09","2023-03-10"

  ],"Close": [
    2685.5,2690.2,2688.75,2701.
    3,812.4,2820.9,2828.6,2835.2,
    28
```

```
    2786.9,2795.4,2802.1,2810.3,2818.7,

    2812.4,2820.9,2828.6,2835.2,2842.9,

    2838.5,2846.1,2852.8,2860.4,2868.9,

    2865.3,2872.7,2879.5,2886.2,2893.8,

    2890.1,2898.6,2905.2,2912.8,2920.4,

    2916.9,2925.5,2932.1,2938.7,2945.3

  ]

}

dataset                    =
pd.DataFrame(data)
prices                     =
dataset[['Close']].valu
es
scaler                                 =
MinMaxScaler(feature_range=(0,
1))          prices_scaled          =
scaler.fit_transform(prices) X = []
y = []

for i in range(10, len(prices_scaled)):

  X.append(prices_scaled[i-10:i,  0])

  y.append(prices_s
caled[i, 0]) X =
np.array(X)
y = np.array(y)

X = np.reshape(X, (X.shape[0],
X.shape[1], 1)) model = Sequential()
model.add(SimpleRNN(50, activation='tanh', input_shape=(10, 1)))
model.add(Dense(1))
model.compile(optimizer='adam',
loss='mean_squared_error') model.summary()
= scaler.inverse_transform(y.reshape(-1, 1))
```

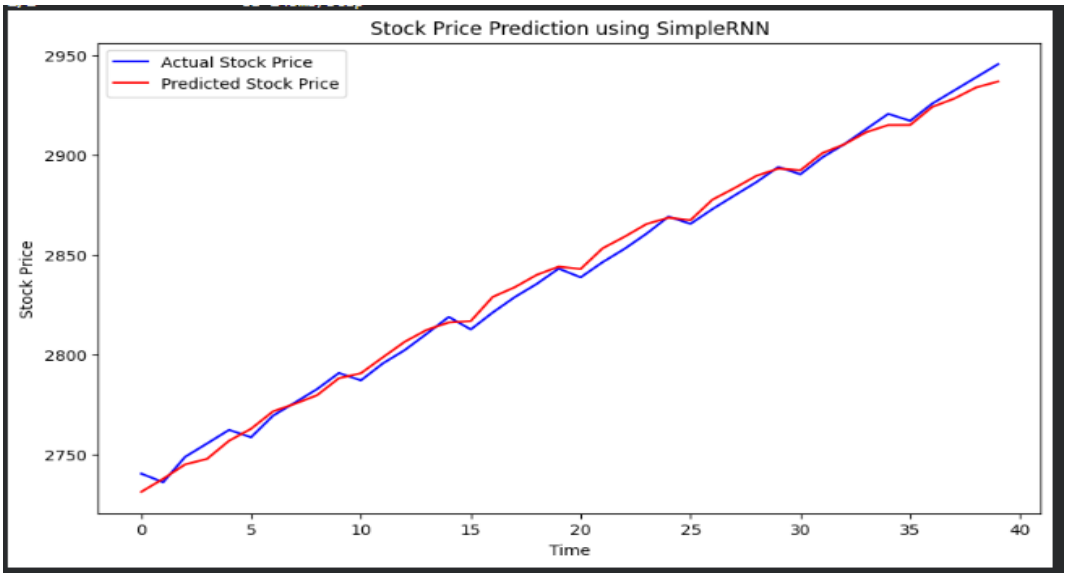
```
plt.figure(figsize=(10,  6))
plt.plot(real_prices, color='blue', label='Actual Stock Price')
plt.plot(predicted_prices, color='red', label='Predicted Stock Price')
plt.title('Stock Price Prediction using SimpleRNN')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

Output:

Model: "sequential"

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 50)	2,600
dense (Dense)	(None, 1)	51

Total params: 2,651 (10.36 KB)
Trainable params: 2,651 (10.36 KB)
Non-trainable params: 0 (0.00 B)



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:

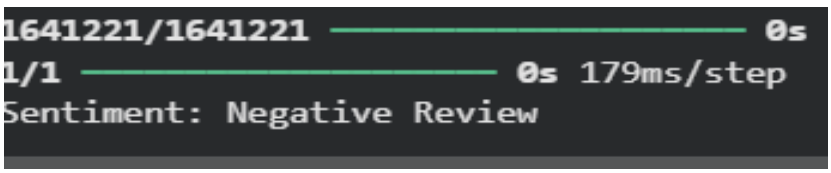
1. Load dataset and preprocess text (tokenize and pad sequences).
2. Build RNN with Embedding + SimpleRNN layers.
3. Train for binary classification (positive/negative).
4. Evaluate accuracy on test data.

Code:

```
import numpy as np

from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences
vocab_size = 10000
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)
max_len = 200
x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)
model = Sequential([
  Embedding(input_dim=vocab_size, output_dim=128, input_length=max_len),
  SimpleRNN(64, activation='tanh'),
  Dense(1, activation='sigmoid')])
model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy'])
model.fit(
  x_train,
```

```
  epochs=5,
  batch_size=64,
  validation_split=0.2)
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
word_index = imdb.get_word_index()
def encode_review(text):
  encoded = []
  for word in text.lower().split():
    index = word_index.get(word)
    if index is not None and index < vocab_size:
      encoded.append(index + 3)
  return pad_sequences([encoded], maxlen=max_len)
custom_text = "This movie was amazing and full of emotions"
encoded_review = encode_review(custom_text)
prediction =
model.predict(encoded_review) if
prediction[0][0] > 0.5:
  print("Sentiment: Positive
Review") else:
```

Output:



LAB Task 4:

Weather Forecasting using RNN

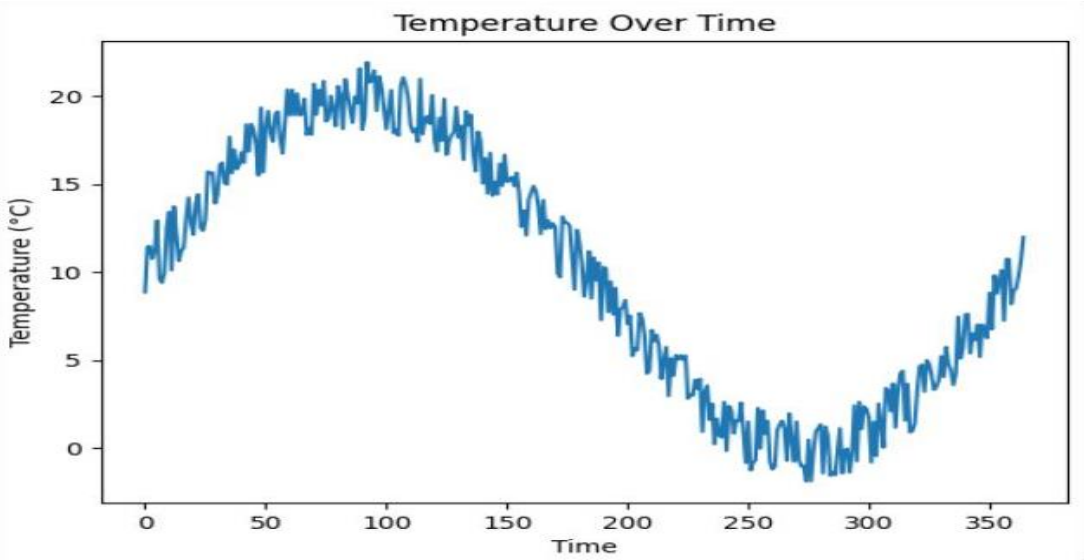
Objective: Predict future temperature based on previous days' readings.
Dataset: *Daily temperature dataset* (e.g., "Jena Climate Dataset" from TensorFlow).
Tasks:

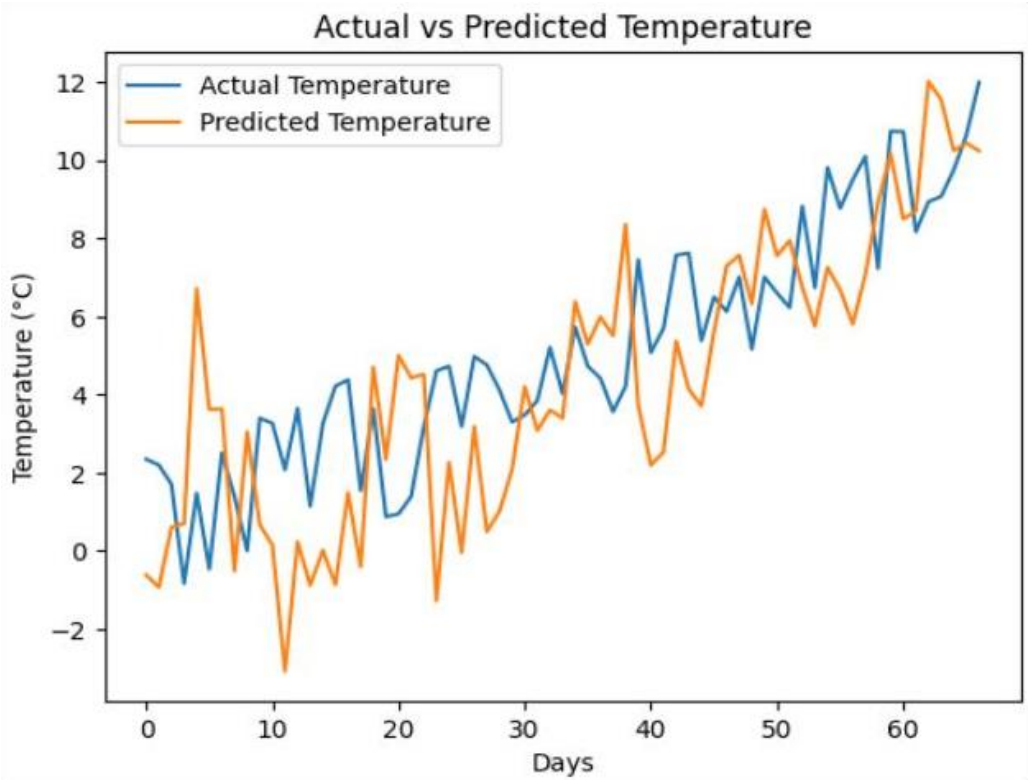
1. Load and visualize temperature over time.
2. Prepare input-output sequences for time series prediction.
3. Build an RNN to predict next day's temperature.
4. Plot actual vs predicted temperature.

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

from sklearn.preprocessing import
MinMaxScaler from tensorflow.keras.models
import Sequential
from tensorflow.keras.layers import SimpleRNN,
Dense data =
pd.read_csv("daily_temperature_dataset.csv")
temperature = data["Temperature (degC)"].values
plt.figure()
plt.plot(temperature[:2000])
plt.title("Temperature Over
Time") plt.xlabel("Time")
plt.ylabel("Temperature (°C)")
plt.show()
scaler = MinMaxScaler()

temperature_scaled =
scaler.fit_transform(temperature.reshape(-1, 1)) 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)
sequence_length = 30
X, y = create_sequences(temperature_scaled,
sequence_length) split = int(0.8 * len(X))
X_train, X_test =
X[:split], X[split:]
y_train, y_test =
y[:split], y[split:] model
= Sequential([
  SimpleRNN(50, activation='tanh', input_shape=(sequence_length,
  1)), Dense(1)])
model.compile(  optimizer='adam',
loss='mse') model.fit(  X_train,
y_train,
 epochs=10,

  batch_size=32,
  validation_split=0.
  2)
predicted = model.predict(X_test)

predicted_temp =
scaler.inverse_transform(predicted)
actual_temp =
scaler.inverse_transform(y_test) plt.figure()
plt.plot(actual_temp, label="Actual

Temperature") plt.plot(predicted_temp,

label="Predicted Temperature") plt.title("Actual

vs Predicted Temperature")

plt.xlabel("Days")

plt.ylabel("Temperatu

re (°C)") plt.legend()
```

Output:


Temperature Over Time
Temperature (°C)
Time
0
5
10
15
20
0
50
100
150
200
250
300
350


Actual vs Predicted Temperature
Actual Temperature
Predicted Temperature
Temperature (°C)
Days
−2
0
2
4
6
8
10
12
0
10
20
30
40
50
60

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).

```
import numpy as np

from music21 import note, chord, stream

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import
SimpleRNN, Dense notes = [
  "C4","D4","E4","F4","G4","A4","B4","C5",

  "C5","B4","A4","G4","F4","E4","D4","C4",

  "C4.E4.G4","D4.F4.A4","E4.G4.B4","F4.A4.C5"]

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)}
sequence_length = 4
X, y = [], []

for i in range(len(notes)-
  sequence_length):seq_in=
  notes[i:i+sequence_lengt
  h] seq_out =
  notes[i+sequence_length]
  X.append([note_to_int[n] for n in seq_in])
  y.append(note_to_int[seq_out])
```

```
X = np.reshape(X, (len(X),
sequence_length, 1)) X = X /
float(len(unique_notes))
y = np.array(y)model = Sequential()

model.add(SimpleRNN(128, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(len(unique_notes), activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
optimizer='adam') model.fit(X, y, epochs=100, batch_size=1,
verbose=0)
start = np.random.randint(0,
len(X)-1) pattern = X[start]
generated
_notes = []
for _ in
range(10:
   prediction = model.predict(pattern.reshape(1, sequence_length, 1),
   verbose=0) index = np.argmax(prediction)
   result = int_to_note[index]

   generated_notes.append(result)

   next_input = index /
   float(len(unique_notes)) pattern =
   np.append(pattern, [[next_input]], axis=0)
   pattern = pattern[1:]
print("Generated Notes:",
generated_notes) output_notes = []
offset = 0

for pattern in generated_notes:

if '.' in pattern: chord_notes = pattern.split('.')

      chord_objects = [note.Note(n) for n in

chord_notes] new_chord =

chord.Chord(chord_objects)

      new_chord.offset = offset
```

```
 output_notes.append(new_chord
   ) else:
      new_note =
      note.Note(pattern)
      new_note.offset = offset
      output_notes.append(new_
   note) offset += 0.5
midi_stream = stream.Stream(output_notes)
midi_stream.write('midi',  fp='generated_music_demo.mid')
```

Output:

```
Generated Notes: ['F4', 'E4', 'D4', 'C4', 'C4.E4.G4', 'D4.F4.A4', 'E4.G4.B4', 'F4.A4.C5', 'G4', 'A4']
 MIDI file generated: generated_music_demo.mid
```

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

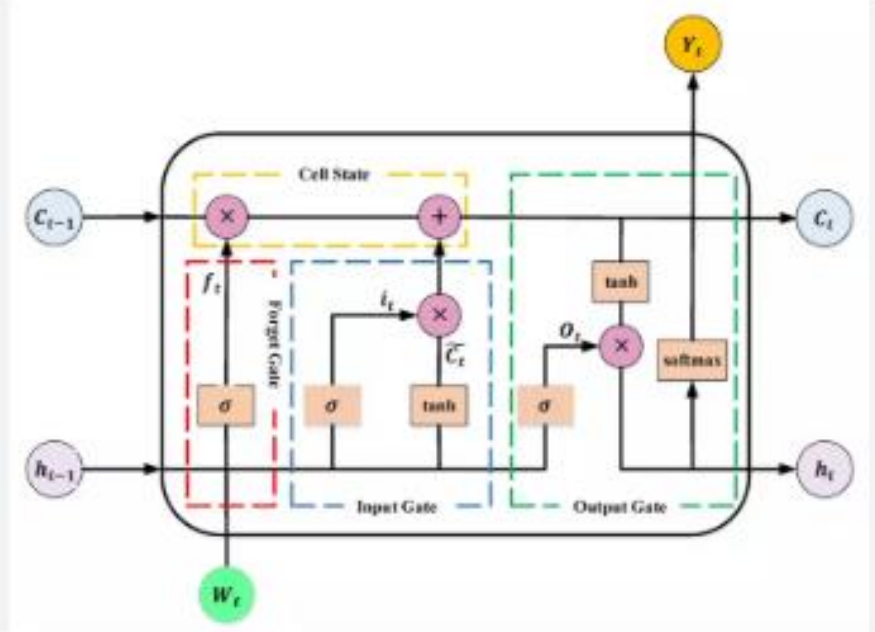
LAB No. 8

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.

Code:

```
import numpy as np

import tensorflow as tf

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
```

```
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?... (use full text from your description) ...Discussion on Job hunting strategies"""

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])

padded_input_sequences = pad_sequences(input_sequences, maxlen=max_len, padding='pre')

X = padded_input_sequences[:, :-1]

y = padded_input_sequences[:, -1]

y = tf.keras.utils.to_categorical(y, num_classes=len(tokenizer.word_index)+1)

vocab_size = len(tokenizer.word_index) + 1

print("Vocabulary size:", vocab_size)

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'])

model.build(input_shape=(None, max_len-1))
```

```
history = model.fit(X, y, epochs=100, verbose=1)
def predict_next_word(model, tokenizer, text_seq, max_len=max_len):
    tokenized = tokenizer.texts_to_sequences([text_seq])[0]
    padded = pad_sequences([tokenized], maxlen=max_len-1, padding='pre')
    predicted_index = np.argmax(model.predict(padded, verbose=0))
    for word, index in tokenizer.word_index.items():
        if index == predicted_index: return word
seed_text = "what is the fee"
next_word = predict_next_word(model, tokenizer, seed_text)
print(f"Next word prediction: '{next_word}'")
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 17, 100)	4,200
lstm_2 (LSTM)	(None, 17, 150)	150,600
lstm_3 (LSTM)	(None, 150)	180,600
dense_1 (Dense)	(None, 42)	6,342

```
Input: 'hello how' → Next Word Prediction: 'are'
```

Question No. 3

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

```
seed_text = "what is the fee"

next_word = predict_next_word(model, tokenizer, seed_text)

print(f"Next word prediction: '{next_word}'")

def generate_words(model, tokenizer, seed_text, n_words=5,
max_len=max_len):

  text = seed_text

  for _ in range(n_words):

    word = predict_next_word(model, tokenizer, text, max_len)

    text += " " + word return text

generated_text = generate_words(model, tokenizer, seed_text, n_words=5)

print(f"\nGenerated sequence (5 words): {generated_text}")
```

```
Generated Text: hello how are you doing today about
```

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 C 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.

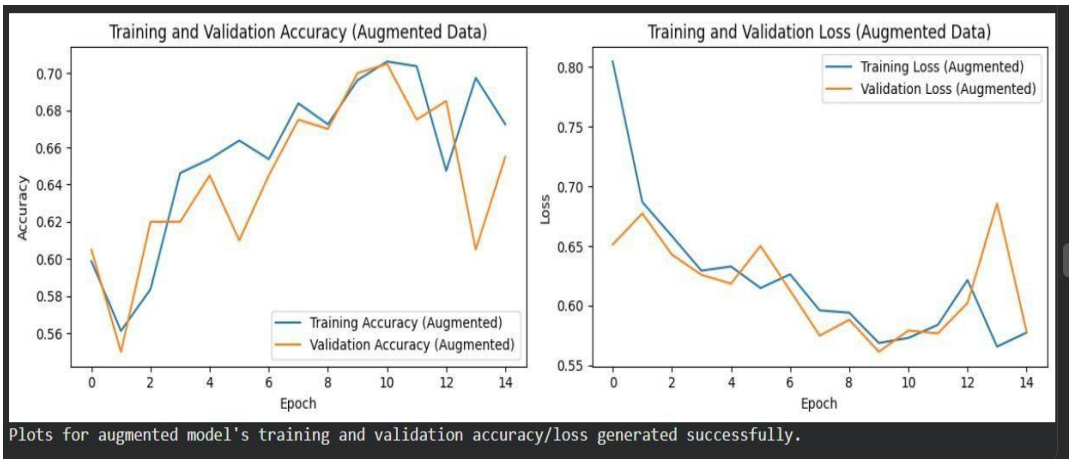
Code:

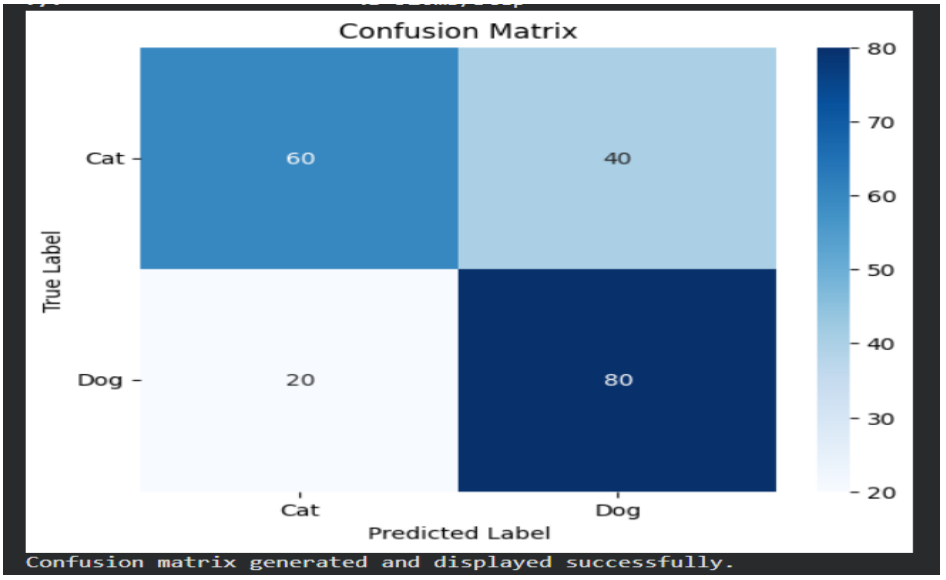
```
import numpy as np
import matplotlib.pyplot as plt
import kagglehub
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay from
tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical
path = kagglehub.dataset_download("karakaggle/kaggle-cat-vs-dog-dataset") print("Path
to dataset files:", path)
DATASET_PATH = None
for root, dirs, files in os.walk(path):
    if "PetImages" in dirs:
        DATASET_PATH = os.path.join(root, "PetImages") break
if DATASET_PATH is None:
    raise Exception("PetImages folder not found!")
print("Final Dataset Path:", DATASET_PATH)
print("Classes:", os.listdir(DATASET_PATH))
IMG_SIZE = 150
EPOCHS = 15
data = []
labels = []
categories = ["Cat", "Dog"]
for label, category in enumerate(categories):
    folder_path = os.path.join(DATASET_PATH, category) for
    img_name in os.listdir(folder_path):
img_path = os.path.join(folder_path, img_name
```

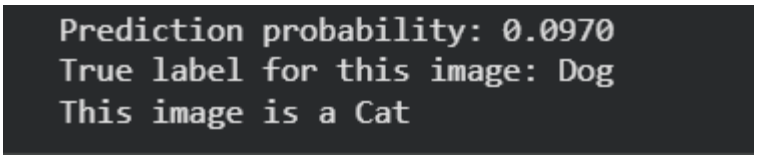
```
Img=cv2.imread(img_pat)
if img is None:  continue
    img = cv2.resize(img, (IMG_SIZE,
    IMG_SIZE)) img = img / 255.0
    data.append(img)
    labels.append(label)
data = np.array(data)
labels = to_categorical(labels,
num_classes=2) X_train, X_test, y_train, y_test
= train_test_split( data, labels, test_size=0.2,
random_state=42)
model=Sequential([Conv2D(32, (3,3), activation='relu',input_shape=(IMG_SIZE,
IMG_SIZE, 3)),
  MaxPooling2D(2,2),
  Conv2D(64, (3,3), activation='relu'),
  MaxPooling2D(2,2),
  Conv2D(128, (3,3), activation='relu'),
  MaxPooling2D(2,2),
  Flatten(),
  Dense(128,
  activation='relu'),
  Dropout(0.5),
  Dense(2,
activation='softmax')])
model.compile(
  optimizer='adam',
  loss='categorical_crossentrop
  y', metrics=['accuracy'])
model.summary(
    ) history =
    model.fit(
```

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title("Accuracy")
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss') plt.legend()
plt.title("Loss")  plt.show()
y_pred =  model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1) y_true
= np.argmax(y_test, axis=1)
cm  =  confusion_matrix(y_true,  y_pred_classes)
disp = ConfusionMatrixDisplay(cm, display_labels=["Cat", "Dog"])
disp.plot(cmap=plt.cm.Blues) plt.show()
def predict_image(img_path): img =
   cv2.imread(img_path)
   img = cv2.resize(img, (IMG_SIZE, IMG_SIZE)) img
   = img / 255.0
   img = img.reshape(1, IMG_SIZE, IMG_SIZE, 3) prediction
   = model.predict(img)
   if np.argmax(prediction) == 0: print("This image
      is a CAT  else:
predict_image(os.path.join(DATASET_PATH,  "Cat",
os.listdir(os.path.join(DATASET_PATH,   "Cat"))[0]))
```

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 inappropriatel y
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

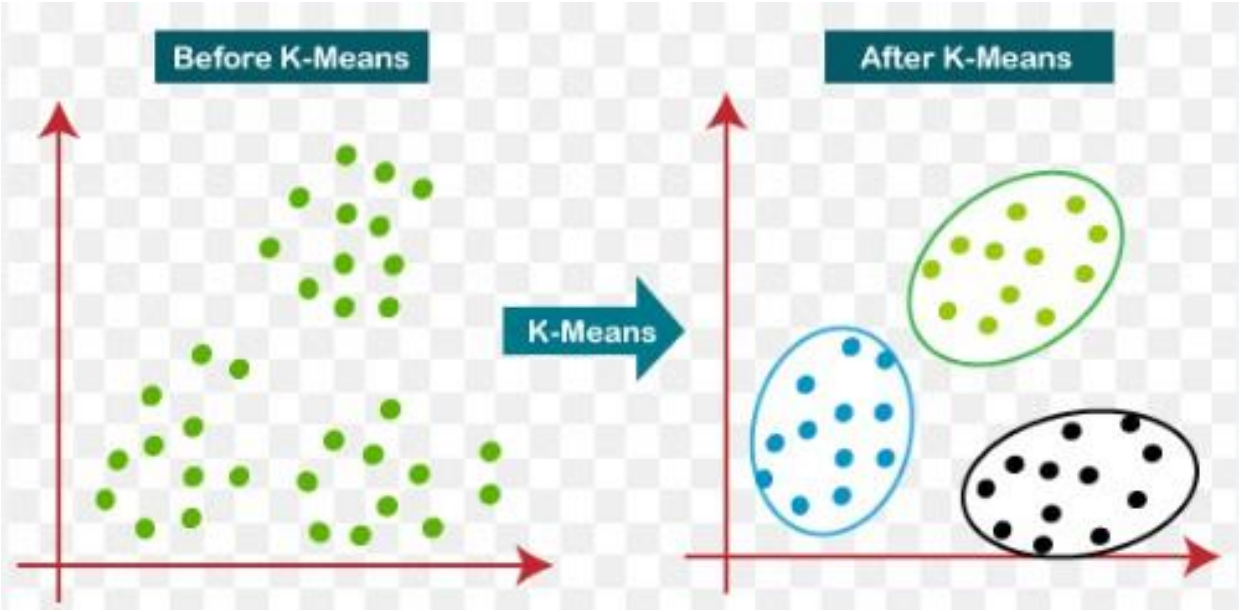
LAB Assignment No 10

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

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…PG)**
6. Sketch the clusters using matplot library in python

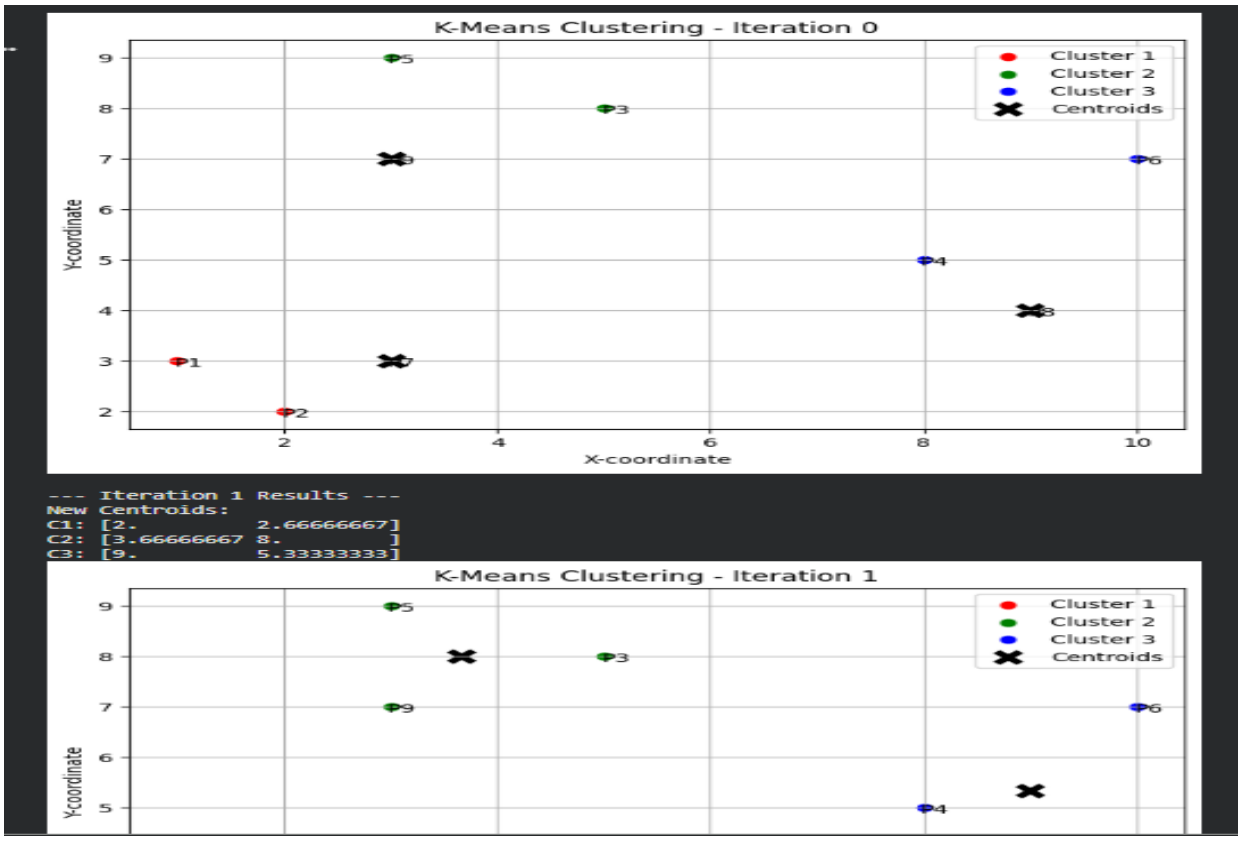
Code:

```
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]])
point_labels = [f'P{i+1}' for i in range(len(points))]K = 3 centroids = np.array([
points[6], points[8], points[7] ])
def assign_clusters(data, centroids):
```

```
def euclidean_distance(a, b):
"""Computes the Euclidean distance between two points."""
return np.sqrt(np.sum((a - b)**2))
assignments = []
distances = []
for point in data:
dist_to_centroids = [euclidean_distance(point, c)
for c in centroids]
assignments.append(np.argmin(dist_to_centroids))
distances.append(dist_to_centroids)
 return np.array(assignments),
np.array(distances)
def update_centroids(data, assignments, K):
new_centroids = []
 for k in range(K):
cluster_points = data[assignments == k]
if len(cluster_points) > 0:
new_centroids.append(cluster_points.mean(axis=0))
else: new_centroids.append(centroids[k])
return np.array(new_centroids)
def plot_clusters(data, assignments, centroids, iteration):
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b', 'y', 'c', 'm'] for
i in range(K):
cluster_points = data[assignments == i]
```

```
plt.scatter(cluster_points[:, 0], cluster_points[:, 1],
c=colors[i], label=f'Cluster {i+1}')
plt.scatter(centroids[:, 0],
centroids[:, 1],
marker='X', s=200, c='k',
label='Centroids', edgecolor='w')
for i, txt in enumerate(point_labels):
plt.annotate(txt, (data[i, 0], data[i, 1]),
textcoords="offset points", xytext=(5,-5), ha='center')
plt.title(f'K-Means Clustering - Iteration {iteration}') plt.xlabel('X-coordinate')
plt.ylabel('Y-coordinate')
plt.legend()
plt.grid(True) plt.show()
print("--- Initial State ---")
assignments_0, _ = assign_clusters(points, centroids)
plot_clusters(points, assignments_0, centroids, 0)
assignments_1, _ = assign_clusters(points, centroids) centroids_1 = update_centroids(points, assignments_1, K) print(f"\n--- Iteration 1 Results ---")
print(f"New Centroids:\nC1: {centroids_1[0]}\nC2: {centroids_1[1]}\n
C3: {centroids_1[2]}") centroids = centroids_1 # Update centroids for the next iteration
plot_clusters(points, assignments_1, centroids_1, 1)
assignments_2, _ = assign_clusters(points, centroids)
centroids_2 = update_centroids(points, assignments_2, K)
print(f"\n--- Iteration 2 Results ---")
print(f"New Centroids:\nC1: {centroids_2[0]}\n
C2: {centroids_2[1]}\nC3: {centroids_2[2]}")
centroids = centroids_2 # Final centroids
plot_clusters(points, assignments_2, centroids_2, 2)
```

output:



Question No. 2

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

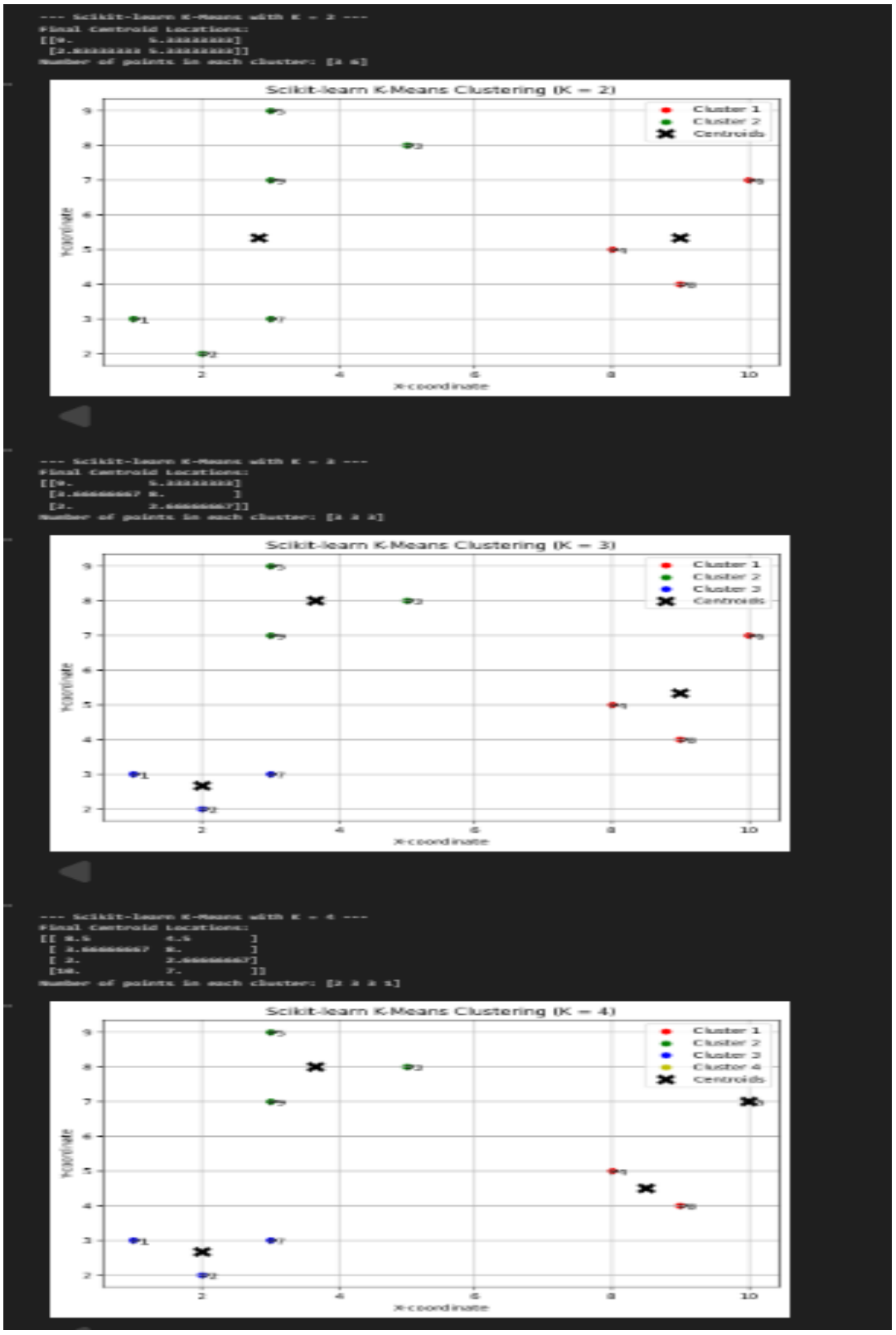
P1(1,3), P2(2,2), P3(5,8), P4(8,5), P5(3,9), P6(10,7), P7(3,3), P8(9,4), P9(3,7)

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.

```
Def run_sk_learn _kmeans(data, k_val, point_labels):
"""Runs Scikit-learn KMeans for a given K and plots the result.""" #
 n_init='auto' and random_state for reproducibility
kmeans = KMeans(n_clusters=k_val,
random_state=42, n_init='auto')
kmeans.fit(data) labels = kmeans.labels_ centroids = kmeans.cluster_centers_
  cluster_counts = np.bincount(labels)
print(f"\n--- Scikit-learn K-Means with K = {k_val} ---")
print(f"Final Centroid Locations:\n{centroids}")
print(f"Number of points in each cluster: {cluster_counts}")
plt.figure(figsize=(8, 6)) colors = ['r', 'g', 'b', 'y']
for i in range(k_val):
cluster_points = data[labels == i]
plt.scatter(cluster_points[:, 0], cluster_points[:, 1],
c=colors[i], label=f'Cluster {i+1}') plt.scatter(centroids[:, 0], centroids[:, 1],
marker='X', s=200, c='k', label='Centroids',
edgecolor='w', zorder=10) for i, txt in enumerate(point_labels):
plt.annotate(txt, (data[i, 0], data[i, 1]),
textcoords="offset points",
xytext=(5,-5), ha='center')
plt.title(f'Scikit-learn K-Means Clustering (K = {k_val})')
plt.xlabel('X-coordinate')
plt.ylabel('Y-coordinate')
plt.legend()
plt.grid(True)
plt.show()
run_sklearn_kmeans(POINTS_9, 2, POINT_LABELS_9)
run_sklearn_kmeans(POINTS_9, 3, POINT_LABELS_9)
run_sklearn_kmeans(POINTS_9, 4, POINT_LABELS_9)
```

Output:


Scikit-learn K-Means Clustering (K = 2)
Cluster 1
Cluster 2
Centroids
Y-coordinate
X-coordinate
Scikit-learn K-Means Clustering (K = 3)
Cluster 1
Cluster 2
Cluster 3
Centroids
Y-coordinate
X-coordinate
Scikit-learn K-Means Clustering (K = 4)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroids
Y-coordinate
X-coordinate

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

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

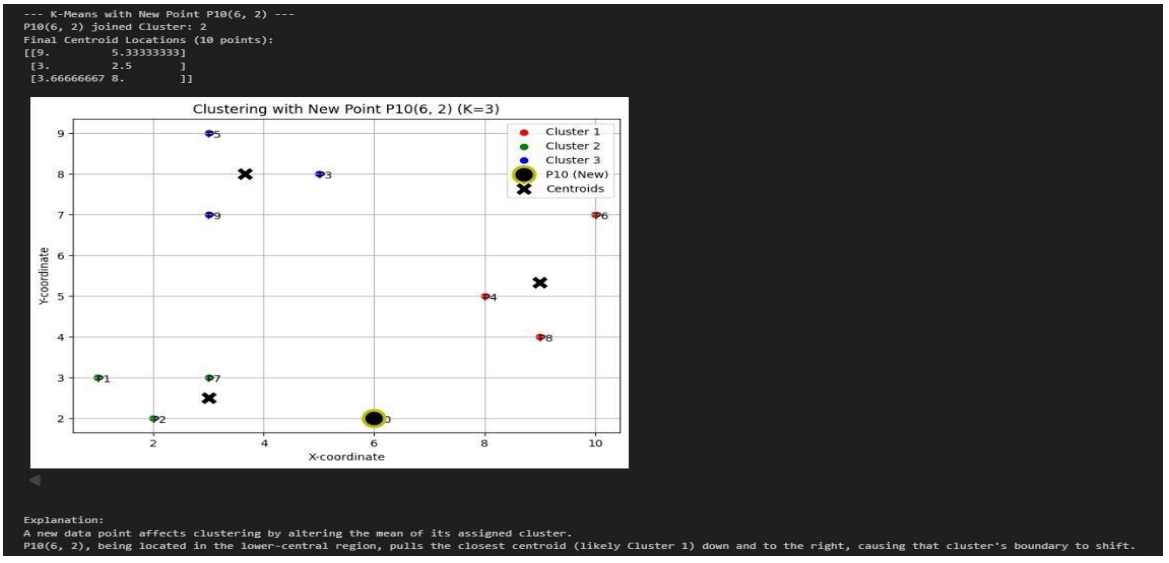
Tasks:

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.

```
P10 = np.array([6, 2])

points_10 = np.vstack([POINTS_9, P10]) point_labels_10 = POINT_LABELS_9 + ['P10']

kmeans_q3 = KMeans(n_clusters=K_VALUE, random_state=42, n_init='auto')
kmeans_q3.fit(points_10)

labels_q3 = kmeans_q3.labels_ centroids_q3 = kmeans_q3.cluster_centers_
P10_cluster_index = labels_q3[-1]

print(f"--- K-Means with New Point P10(6, 2) ---")

print(f"P10(6, 2) joined Cluster: {P10_cluster_index + 1}")

print(f"Final Centroid Locations (10 points):\n{centroids_q3}") plt.figure(figsize=(8, 6))

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

for i in range(K_VALUE):

cluster_points = points_10[labels_q3 == i]

plt.scatter(cluster_points[:, 0], cluster_points[:, 1], c=colors[i],

label=f'Cluster {i+1}')
```

```
plt.scatter(P10[0], P10[1],
marker='o', s=300, c='k', edgecolor='y', linewidth=3,
label='P10 (New)', zorder=10)
plt.scatter(centroids_q3[:, 0], centroids_q3[:, 1], marker='X', s=200, c='k',
label='Centroids', edgecolor='w', zorder=10)
for i, txt in enumerate(point_labels_10):
plt.annotate(txt, (points_10[i, 0], points_10[i, 1]),
textcoords="offset points", xytext=(5,-5), ha='center')
plt.title('Clustering with New Point P10(6, 2) (K=3)')
plt.xlabel('X-coordinate')
plt.ylabel('Y-coordinate') plt.legend()
plt.grid(True) plt.show()
print("\nExplanation:")
print("A new data point affects clustering by altering the mean of its assigned cluster.")
print("P10(6, 2), being located in the lower-central region, pulls the closest centroid (likely Cluster 1) down and to the right, causing that cluster's boundary to shift.")
```

Output:



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:

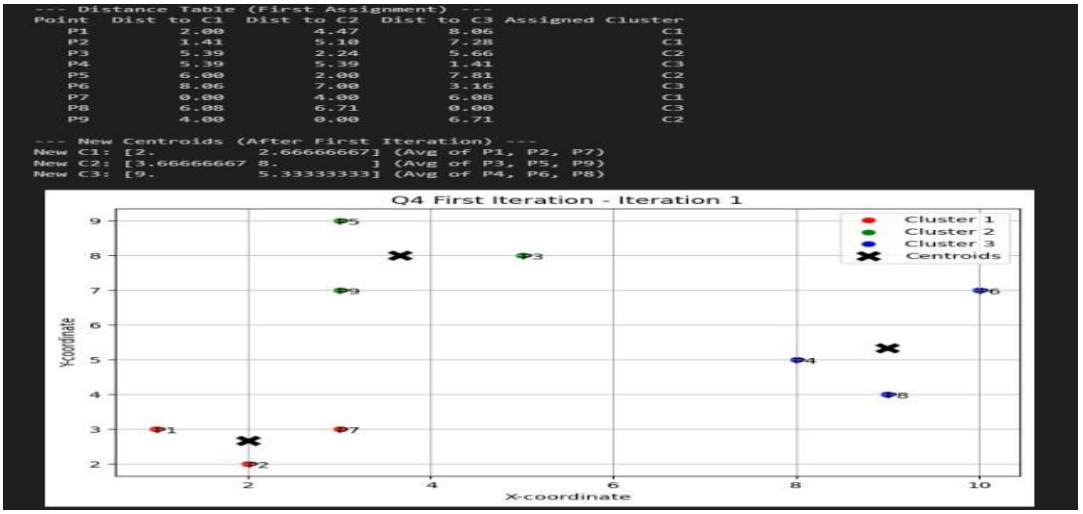
Point Dist to C1 Dist to C2 Dist to C3 Assigned Cluster

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.

```
C1_Q4 = np.array([3, 3])
C2_Q4 = np.array([3, 7])
C3_Q4 = np.array([9, 4])
centroids_Q4 = np.array([C1_Q4, C2_Q4, C3_Q4]
) distances_Q4 = []
assignments_Q4 = []
for i,
 point in enumerate(POINTS_9):
dist_c1 = euclidean_distance(point, C1_Q4) dist_c2 = euclidean_distance(point, C2_Q4)
dist_c3 = euclidean_distance(point, C3_Q4)
 dists = [dist_c1, dist_c2, dist_c3]
assigned_cluster_index = np.argmin(dists)
 distances_Q4.append(dists)
assignments_Q4.append(assigned_cluster_index)
```

```
df_distances['Point'] = POINT_LABELS_9

df_distances['Assigned Cluster'] = [f'C{i+1}'

for i in assignments_Q4]

df_distances = df_distances[['Point', 'Dist to C1', 'Dist to C2', 'Dist to C3', 'Assigned Cluster']]

def highlight_min(s):is_min = s == s.min() return

['background-color: yellow' if v else ''

for v in is_min] print("--- Distance Table (First Assignment) ---")

print(df_distances.to_string(index=False, float_format="%.2f")) assignments_Q4 = np.array(assignments_Q4)

new_centroids_Q4 = update_centroids(POINTS_9, assignments_Q4, K_VALUE) print("\n--- New Centroids (After First Iteration) ---")

print(f"New C1: {new_centroids_Q4[0]} (Avg of P1, P2, P7)") print(f"New C2: {new_centroids_Q4[1]} (Avg of P3, P5, P9)") print(f"New C3: {new_centroids_Q4[2]} (Avg of P4, P6, P8)")

plot_clusters(POINTS_9, assignments_Q4, new_centroids_Q4, 1, title_prefix="Q4 First Iteration")
```

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.

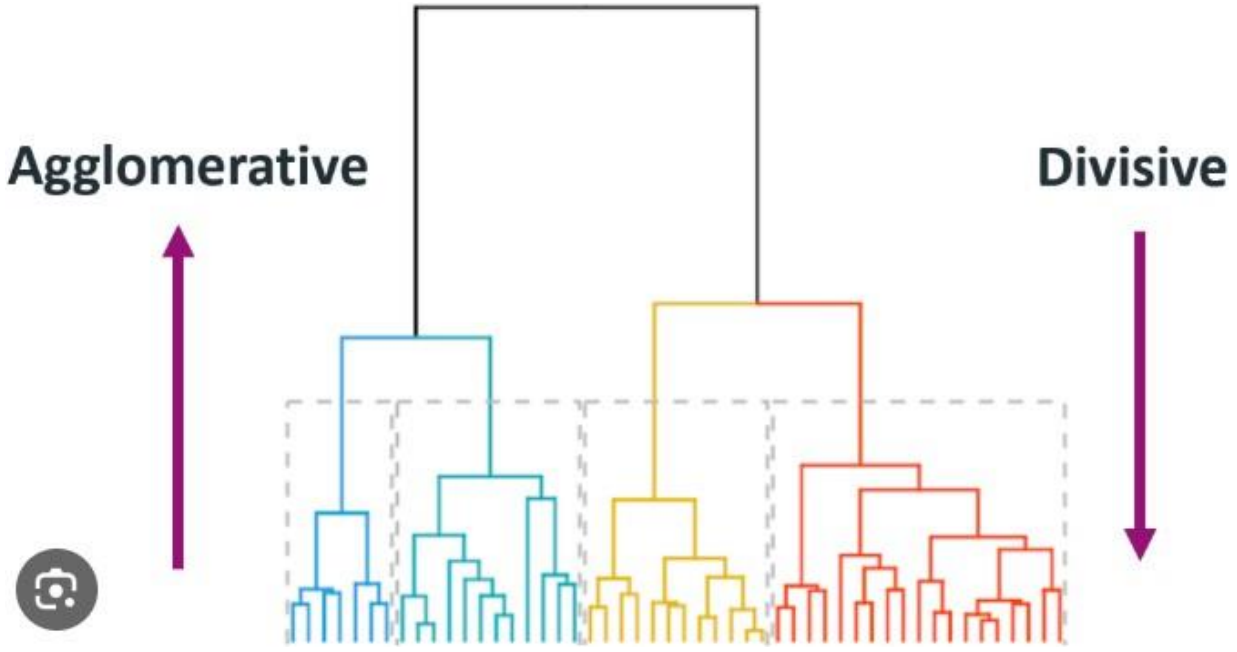
Linkage Methods:

- **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
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:

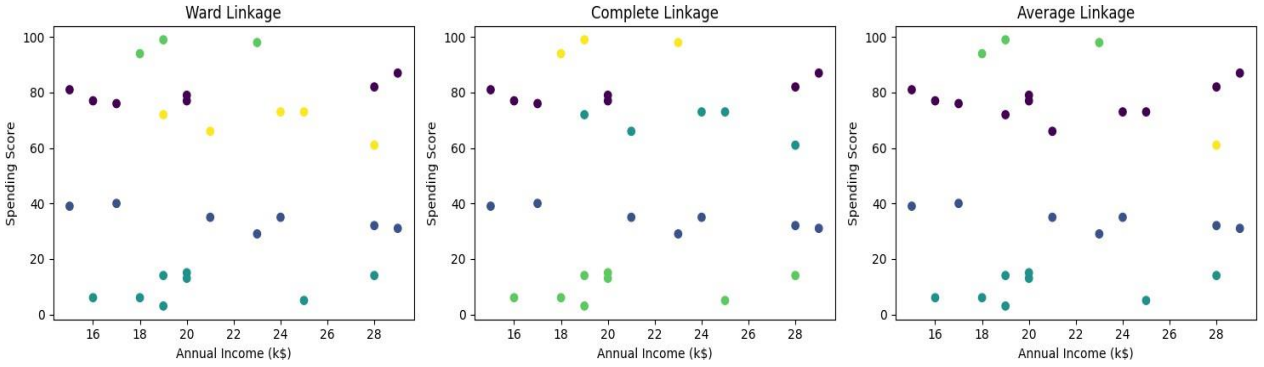
1. Perform clustering using AgglomerativeClustering.
2. Plot the clusters using matplotlib.
3. Compare how the cluster structure changes with each linkage method.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
data = pd.read_csv("shopping-data.csv")
X = data[['Annual Income (k$)', 'Spending Score (1-100)']]


ward = AgglomerativeClustering(n_clusters=5, linkage='ward')
complete = AgglomerativeClustering(n_clusters=5, linkage='complete')
average = AgglomerativeClustering(n_clusters=5, linkage='average')


labels_ward = ward.fit_predict(X)
labels_complete = complete.fit_predict(X)
labels_average = average.fit_predict(X)
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
plt.scatter(X.iloc[:,0], X.iloc[:,1], c=labels_ward)
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score")
plt.title("Ward Linkage")
plt.subplot(1,3,2)
plt.scatter(X.iloc[:,0], X.iloc[:,1], c=labels_complete)
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score")
plt.title("Complete Linkage")
```

Output:



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:

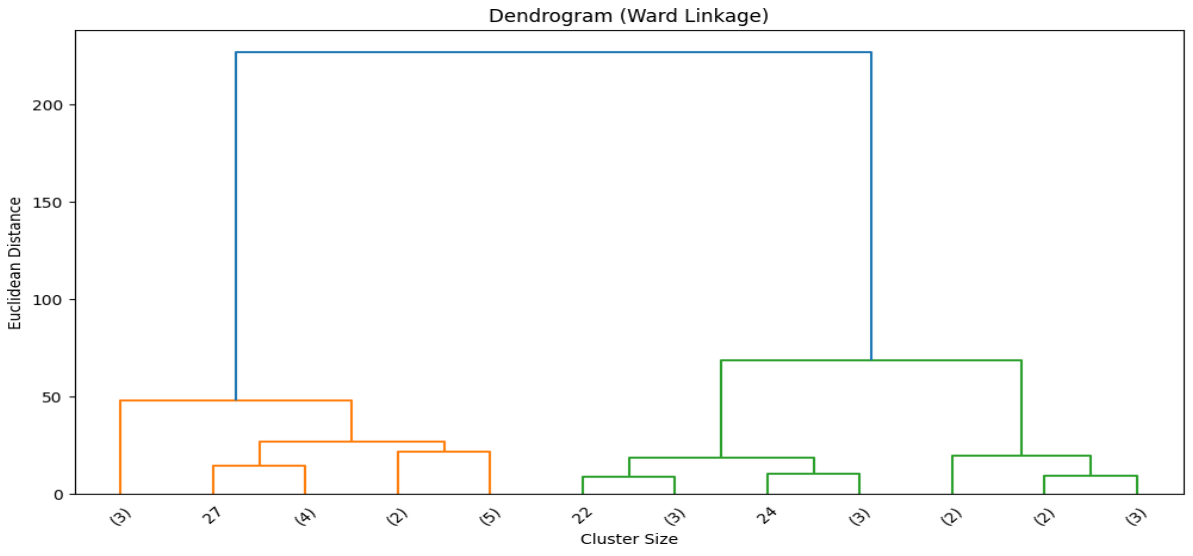
```
from scipy.cluster.hierarchy import dendrogram, linkage
```

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

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
data = pd.read_csv("shopping-data.csv")
X = data[['Annual Income (k$)', 'Spending Score (1-100)']]
linked = linkage(X, method='ward')
plt.figure(figsize=(12,6))
dendrogram(
  linked,
  truncate_mode='lastp',
  p=12,
  leaf_rotation=45,
  leaf_font_size=10
)
plt.title("Dendrogram (Ward Linkage)")
plt.xlabel("Cluster Size")
plt.ylabel("Euclidean Distance")
plt.show()
```

Output:



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)

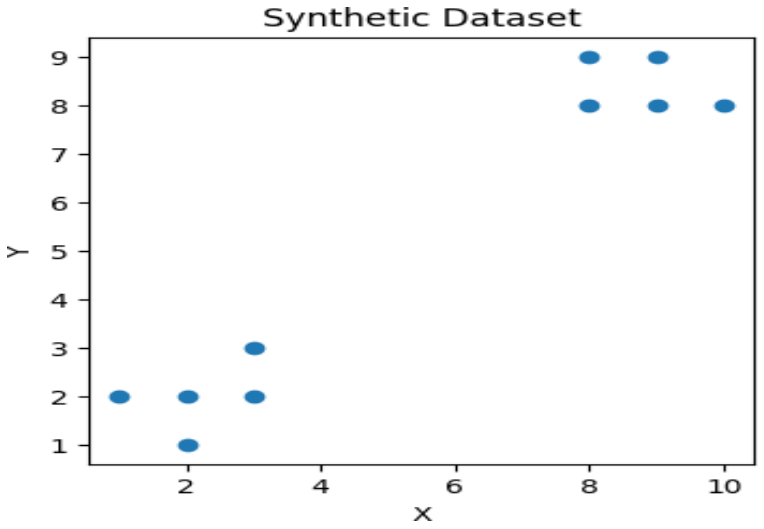
Instructions:

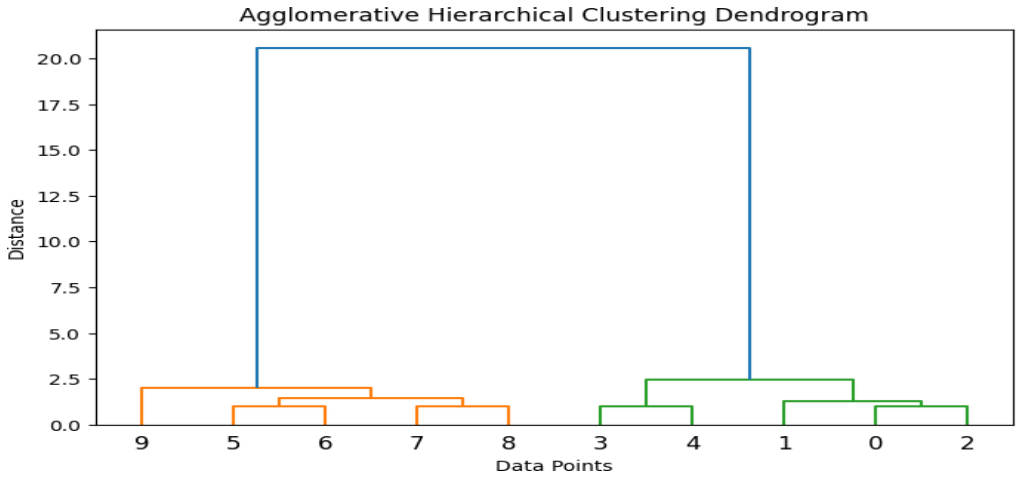
1. Plot dendrograms for both methods.
2. Compare the merge/split patterns.
3. Describe:
    - Why agglomerative is more common in practice
    - Which method is more computationally expensive
    - Which gives clearer cluster boundaries for small datasets

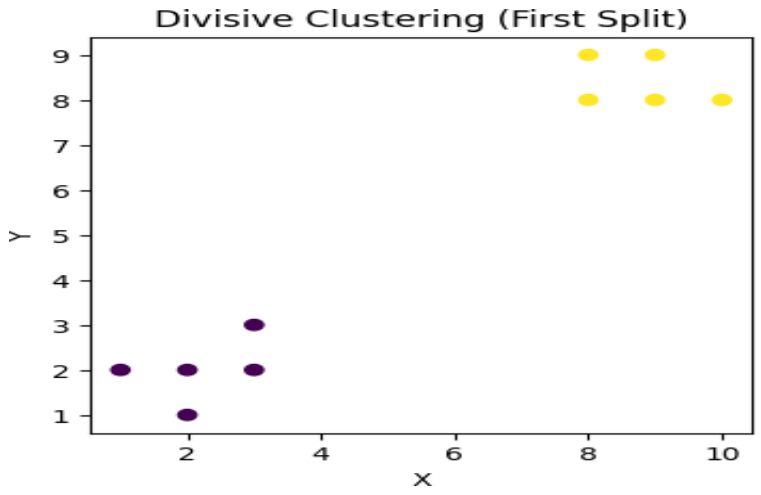
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
X = np.array([
   [1,2],[2,1],[2,2],[3,2],[3,3],
   [8,8],[9,8],[8,9],[9,9],[10,8]])
plt.figure(figsize=(4,4))
plt.scatter(X[:,0], X[:,1])
plt.title("Synthetic Dataset")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
Z_agg = linkage(X, method='ward')
plt.figure(figsize=(8,5))
dendrogram(Z_agg)
plt.title("Agglomerative Hierarchical Clustering Dendrogram")
```

```
plt.xlabel("Data Points")
plt.ylabel("Distance")
plt.show()
kmeans = KMeans(n_clusters=2, random_state=42)
labels = kmeans.fit_predict(X)
plt.figure(figsize=(4,4))
plt.scatter(X[:,0], X[:,1], c=labels)
plt.title("Divisive Clustering (First Split)")
plt.xlabel("X")
plt.ylabel("Y")
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 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

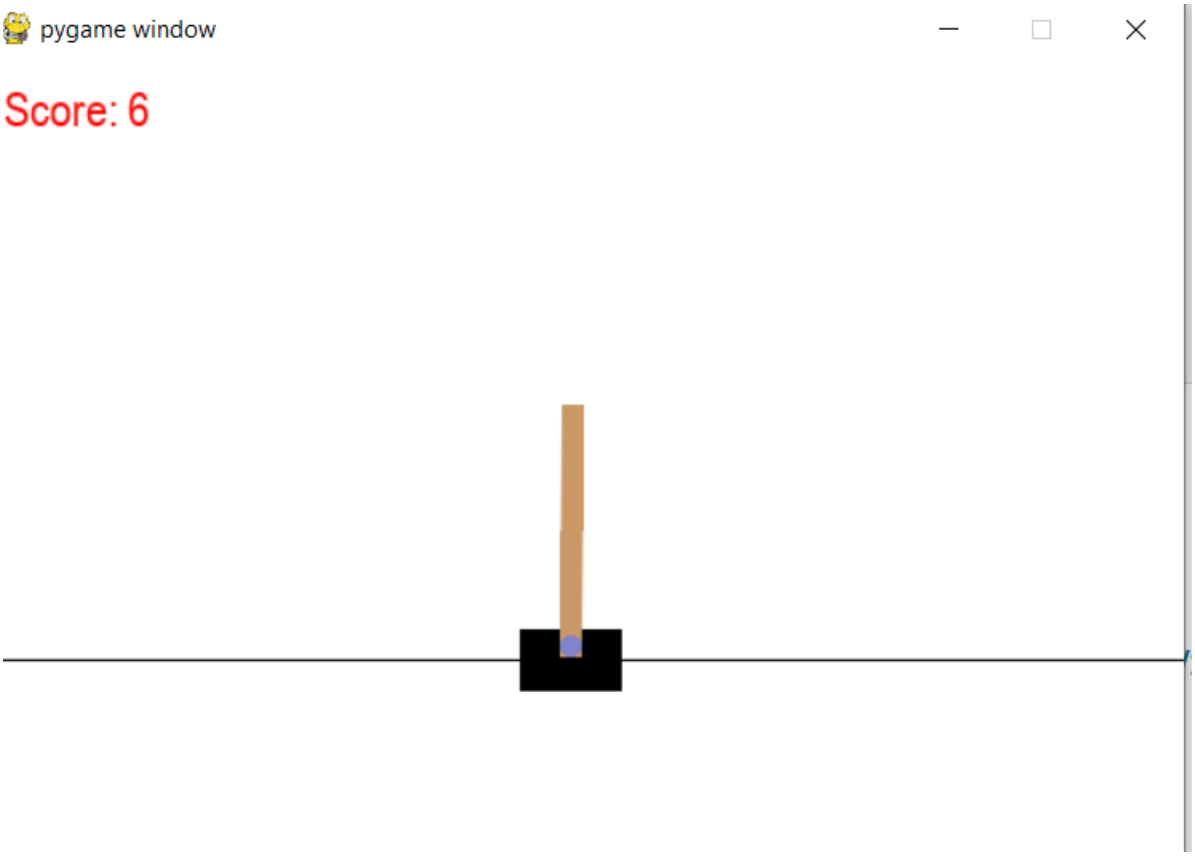
🎯 Lab Objectives

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

Q1.

What is **Reinforcement Learning**? Identify the **agent**, **environment**, **state**, **action**, and **reward** in the given code.

Answer:

RL is a type of machine learning where an agent learns by **taking actions in an environment** and getting **rewards**. The goal is to maximize total rewards over time.

In the CartPole code:

Component	Description in the Code
Agent	The part of the code that chooses actions: action = env.action_space.sample()
Environment	The CartPole simulation created with: env = gym.make("CartPole-v1", render_mode="human")
State	The current situation of the system returned by env.reset() or env.step(action) It includes: [cart_position, cart_velocity, pole_angle, pole_angular_velocity]
Action	What the agent does at each step: 0 = push cart left, 1 = push cart right
Reward	The feedback the agent gets: reward = 1 for every step the pole stays upright

---

Q2.

Explain the purpose of the following line:

env = gym.make("CartPole-v1", render_mode="human")

Answer:

This line **sets up the environment** and makes it **visible to the user** so you can watch the agent act in real time.

Q3.

What does env.reset() return? Why are two values returned?

Answer:

What it does:

- env.reset() **resets the environment** to the starting state for a new episode.
- It returns **two values:**
    1. **state** → The initial observation of the environment (CartPole variables: cart position, cart velocity, pole angle, pole angular velocity)
    2. **info** → Additional information from the environment (usually empty or extra metadata; not used in basic experiments)

Why two values:

- Gymnasium separates **the important observation (state)** from **optional metadata (info)**.
- This allows the code to use the state for decision-making while still having access to extra info if needed.

Q4.

Explain the difference between: Terminated and truncated

Answer:

Term	**Meaning in Gymnasium / CartPole**
terminated	The episode ended because the **goal was reached** or **failure occurred**. In CartPole: the pole fell too far or cart moved out of bounds.
truncated	The episode ended because it **reached the maximum allowed steps**. This is not due to failure, just a **time limit**.

Q5.

What is the role of the variable score? How is it calculated?

Answer:

Role:

- score keeps track of the **total reward** the agent receives during an episode.
- It measures **how well the agent is performing**—higher score means the pole stayed upright longer.

How it is calculated:

score += reward

- At each step, the environment gives a **reward** (in CartPole, reward = 1 per step).
- The code **adds this reward to score** until the episode ends.
- At the end of the episode, score represents the **total steps the pole stayed balanced**.

Q6.

Why is action = env.action_space.sample() used?
Is this an intelligent agent? Justify your answer.

Answer:

action = env.action_space.sample() is used to **choose a random action** at each step.

This is not an intelligent agent because it **does not learn** from rewards or past experience; it acts **randomly**.

Q7.

Explain how **Pygame** is used to display the score on the screen.

Answer:

Pygame is used to **draw the score on the simulation window**.

Steps in the code:

1. Initialize Pygame font: pygame.font.SysFont("Arial", 24)
2. Create a surface (the window) using pygame.display.get_surface()
3. Render the score as text: font.render(f"Score: {int(score)}", True, (255,0,0))
4. Draw it on the window at a position: surface.blit(text, (10, 10))
5. Update the display: pygame.display.update()

Q8.

What happens if the pygame.display.update() line is removed?

Answer:

- If pygame.display.update() is removed, the **score will not appear or refresh** on the screen.
- The **Pygame** window **won't show changes**, so the score text won't be visible while the simulation runs.

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

Answer:

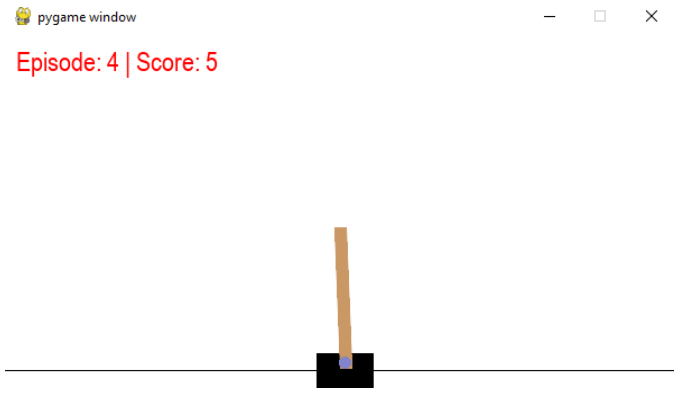
- The score **changes a lot from episode to episode**.
- Some episodes have **low scores** (around 9–15).
- Some episodes have **higher scores** (around 40–58).
- There is **no fixed pattern** in the scores

Task 2: Display Episode Number on Screen

Modify the code to show:

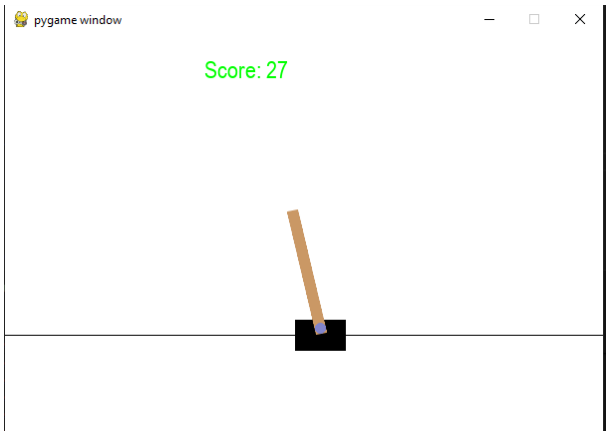
Episode: X | Score: Y

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**

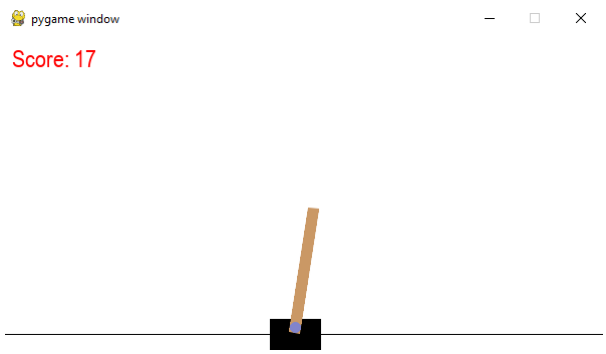
```
Maximum Score: 59.0
(venv) PS D:\AI>
```

Task 5: Slow Down the Environment

Insert a small delay using:

pygame.time.delay(20)

Observe the effect on visualization.

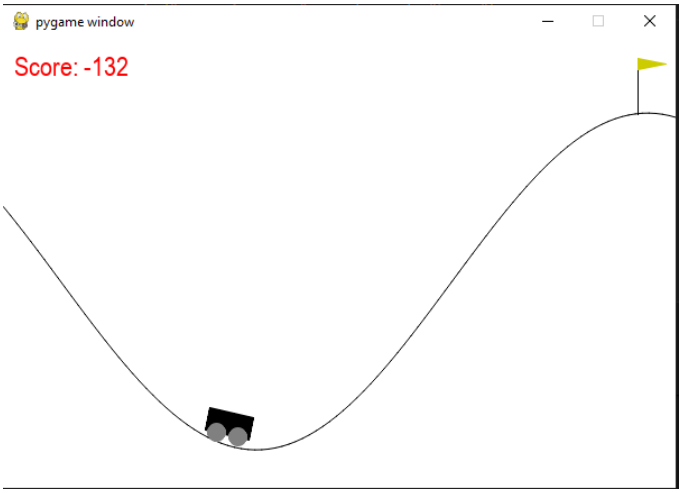


Task 6: Replace CartPole with MountainCar

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:

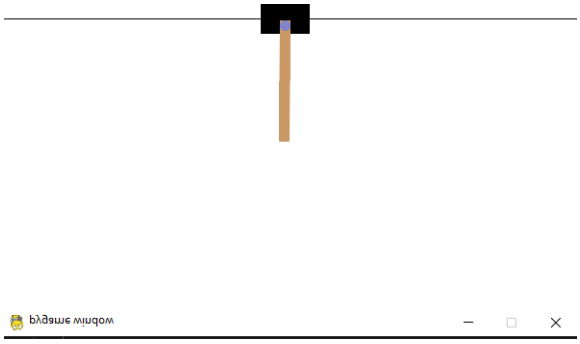
- How many state variables are there?
- What does each variable represent?

```
Episode 19 Score: -200.0
```

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		

Experiment: MountainCar Environment using Gymnasium s

Pygame

Lab Objectives

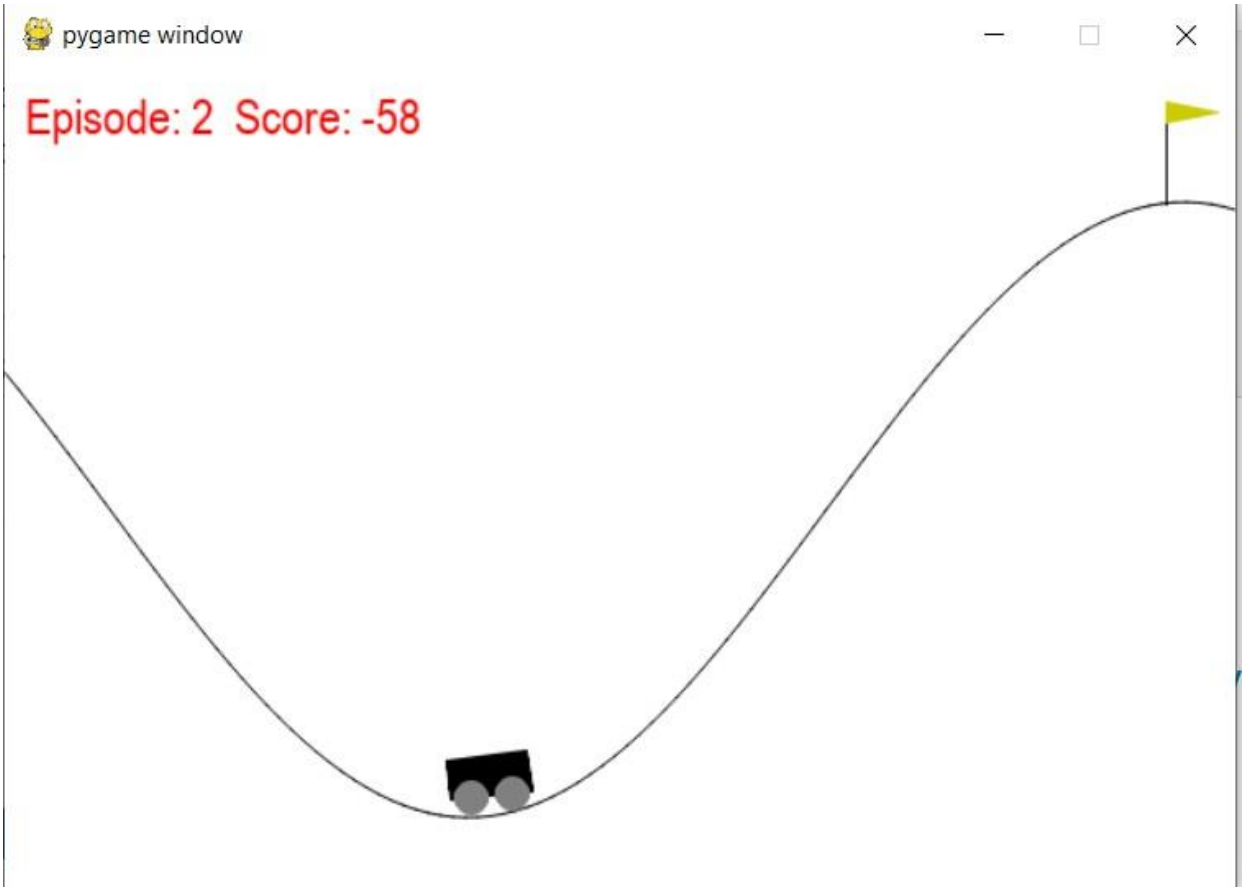
After completing this lab, students will be able to:

- 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

Q1.

What is **Reinforcement Learning**? Identify the **agent**, **environment**, **state**, **action**, and **reward** in the MountainCar code.

Answer:

Reinforcement Learning (RL) is a learning method where an agent learns by interacting with an environment and receiving rewards.

In MountainCar code:

- **Agent:** The car controller (our program)
- **Environment:** MountainCar-v0
- **State:** [position, velocity]
- **Action:** Push left (0), no push (1), push right (2)
- **Reward:** -1 at every step until goal is reached

Q2.

Explain the purpose of the following statement:

env = gym.make("MountainCar-v0", render_mode="human")

Answer:

- Creates the MountainCar environment
- render_mode="human" displays the environment visually on the screen

---

Q3.

What are the **state variables** in MountainCar-v0? What does each state represent?

Answer:

- **Position (state[0])**
  → Horizontal position of the car on the hill

- **Velocity (state[1])**
  → Speed and direction of the car's movement

Q4.

Describe the **action space** of MountainCar-v0. How many actions are available and what do they mean?

Answer:

Action	Meaning
0	Push car left
1	No push
2	Push car right

Q5.

Explain the reward mechanism in MountainCar-v0.
Why does the agent receive a **negative reward** at each step?

Answer:

- The agent receives **-1 reward at every step**
- The goal is to **reach the hilltop in fewer steps**

Reason for negative reward:
To encourage the agent to reach the goal as quickly as possible.

Q6.

What is the difference between:

Terminated and truncated in this environment?

Answer:

- **terminated:** Episode ends because the goal is reached
- **truncated:** Episode ends because maximum step limit is reached

---

Q7.

Why does the agent fail to reach the goal when using action_space.sample()?

Answer:

- Random actions do not build momentum
- The car cannot climb the hill without coordinated left-right movement
- Therefore, the agent fails to reach the goal

Q8.

Explain the role of **momentum** in solving the MountainCar problem.

Answer:

- The car must first move **away from the goal** to gain speed
- Momentum helps the car climb the steep hill
- Without momentum, the car cannot reach the top

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:

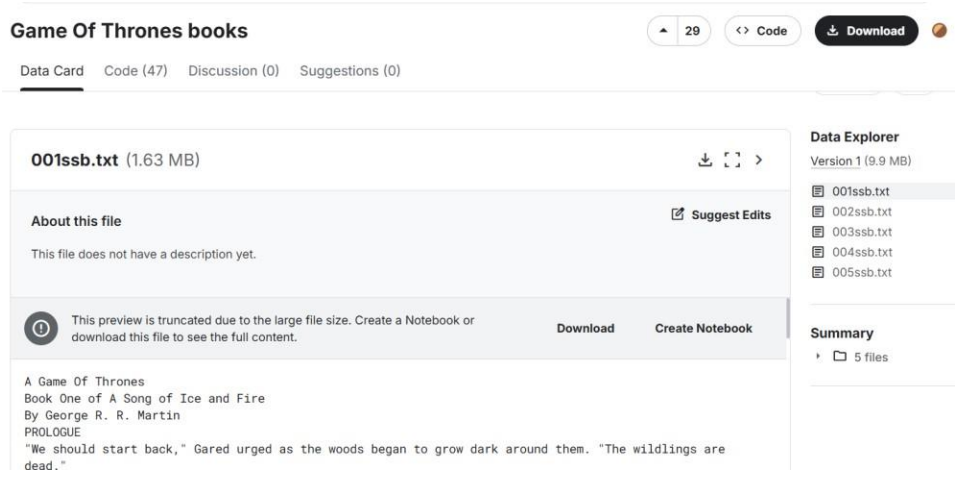
- Understand **distributional semantics**
- Learn how **Word2Vec** converts words into dense vectors
- Apply **Word2Vec (Skip-Gram / CBOW)** on real textual data (Game of Thrones)
- Explore **word similarity, analogy, and visualization**

game-of-thrones-word2vec

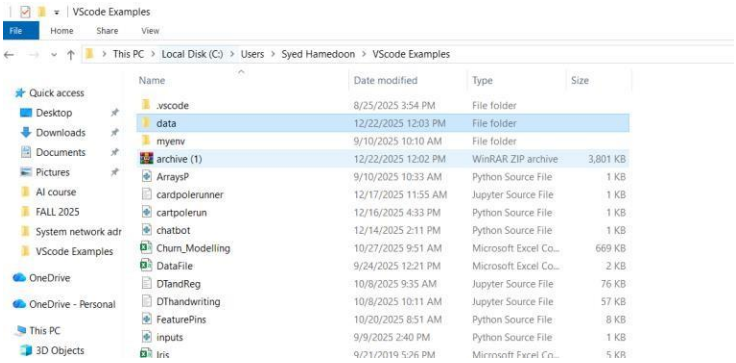
word2vec applied on game of thrones data

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
```

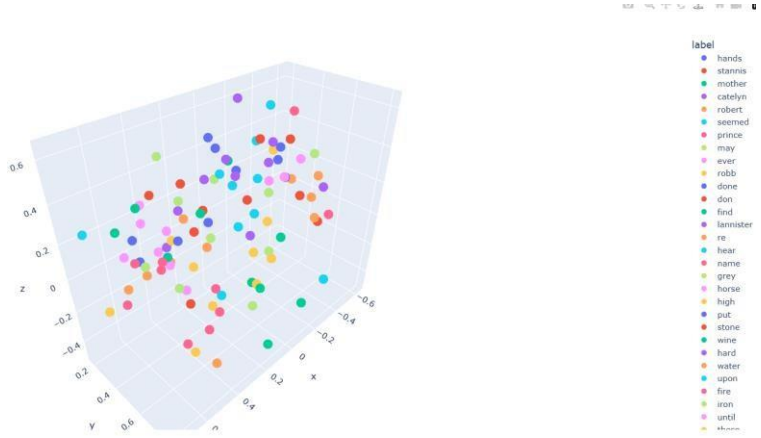
```
pca = PCA(n_components=3)
```

```
X = pca.fit_transform(model.wv.get_normed_vectors())
```

```
!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?

Answer:

The core idea behind **Word2Vec** is that **words appearing in similar contexts tend to have similar meanings**.
Word2Vec learns **dense numerical vectors (embeddings)** for words by analyzing surrounding words in large text data.
These vectors capture **semantic and syntactic relationships** between words.

Example:
king and queen appear in similar contexts → their vectors become close in space

2. Difference between CBOW and Skip-Gram?

Answer:

Feature	**CBOW**	**Skip-Gram**
Prediction	Predicts target word from context	Predicts context from target word
Speed	Faster	Slower
Best for	Large datasets	Small datasets
Rare words	Poor performance	Better performance

3. Why is one-hot encoding inefficient?

Answer:

One-hot encoding is inefficient because:

- Vectors are **very large** (equal to vocabulary size)
- Mostly **zeros**, wasting memory
- Cannot capture **semantic similarity**
- No relationship between words

Example:
king and queen have completely different vectors although meanings are related.

---

4. Why do character names appear close in vector space?

Answer:

Character names appear close because:

- They occur in **similar contexts**
- Appear together in scenes, families, or storylines
- Share roles (e.g., Stark family members)

Example:
arya, sansa, bran → frequently mentioned together → similar embeddings

5. How does window size affect semantic learning?

Answer:

- Small window size (2–5):
    - Learns syntactic relationships
    - Focuses on nearby words
- Large window size (8–15):
    - Learns semantic relationships
    - Captures broader context

6. Why might rare characters have poor embeddings?

Answer:

Rare characters have poor embeddings because:

- They appear **few times** in the corpus
- Model cannot learn enough context
- Limited co-occurrence with other words

---

7. Which model performed better: CBOW or Skip-Gram? Why?

Answer:

Skip-Gram performed better for this dataset because:

- Game of Thrones has many **rare character names**
- Skip-Gram handles rare words more effectively
- Produces more meaningful embeddings for names

8. What happens if vector size is too small or too large?

Answer:

Too small vector size:

- Cannot capture enough semantic information
- Poor similarity results

Too large vector size:

- Overfitting
- Slower training
- Higher memory usage

9. Can Word2Vec understand word meaning without labels? Explain.

Answer:

Yes, **Word2Vec learns word meaning without labels** because:

- It uses **unsupervised learning**
- Learns from **word co-occurrence patterns**
- Discovers semantic relationships automatically

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.

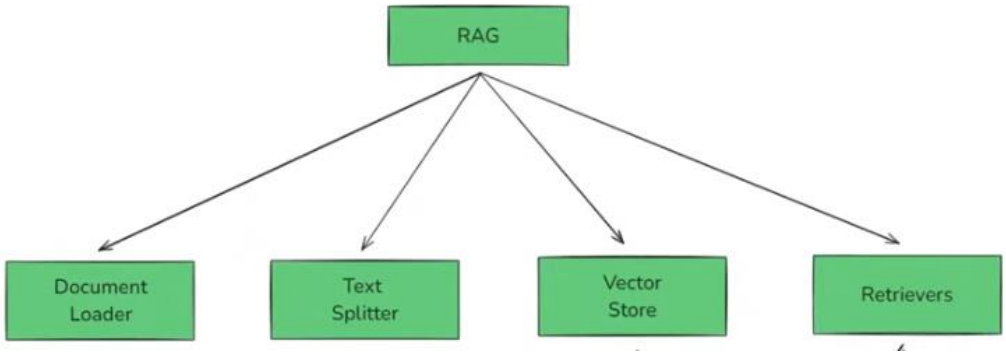
LAB Objectives

- 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

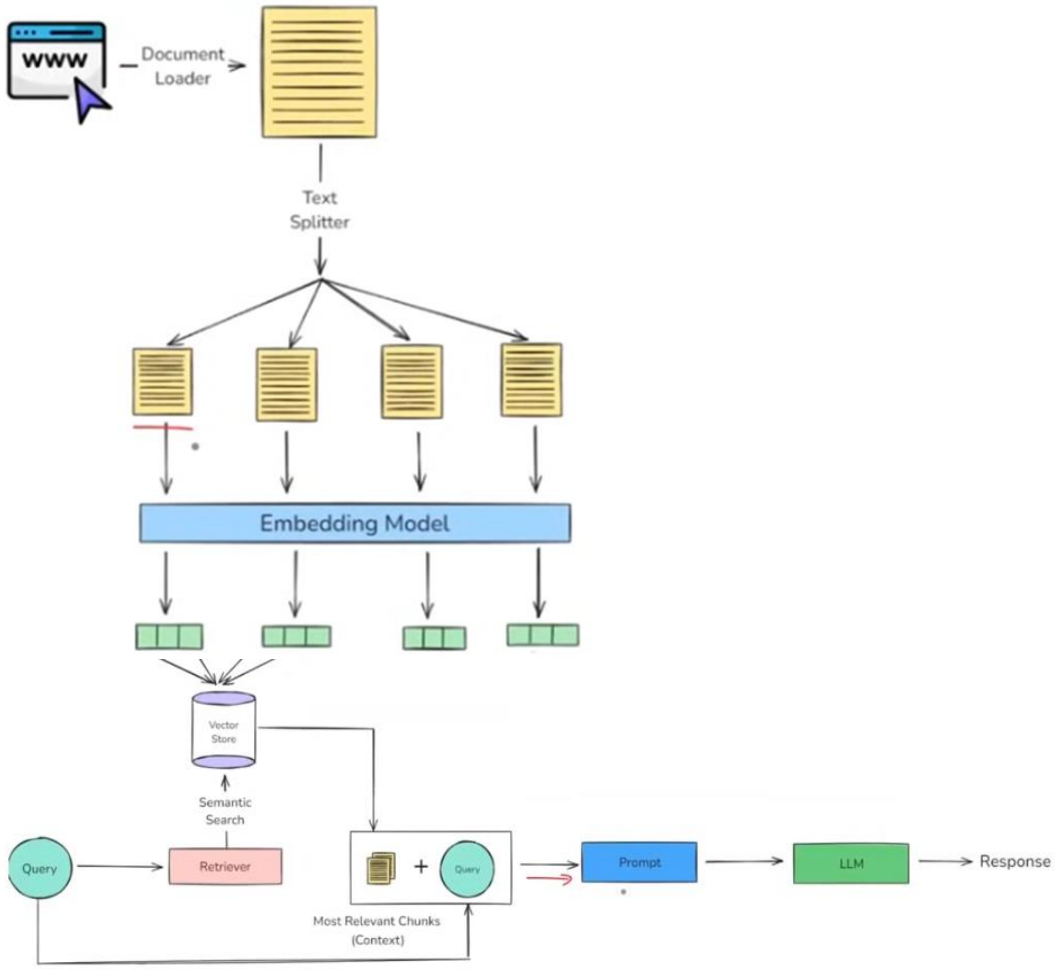
Tools & Libraries

- Python 3.9+
- Required libraries:
    - langchain
    - langchain-community
    - pypdf
    - unstructured

Flow Methodology of RAG



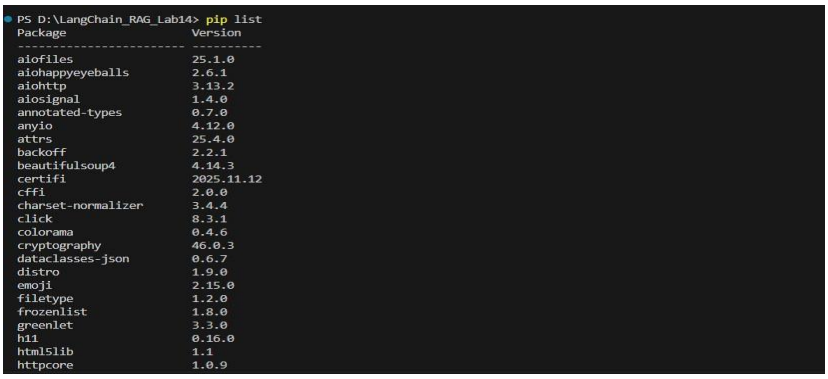
Complete Flow diagram of RAG



Lab Tasks (Practice Steps)

1: Environment Setup

- Create a virtual environment
- Install required LangChain libraries
- Verify installation



```
PS D:\LangChain_RAG_Lab14> pip list
Package                  Version
------------------------ ---------
aiofiles                 25.1.0
aiohappyeyeballs         2.6.1
aiohttp                  3.13.2
aiosignal                1.4.0
annotated-types          0.7.0
anyio                    4.12.0
attrs                    25.4.0
backoff                  2.2.1
beautifulsoup4           4.14.3
certifi                  2025.11.12
cffi                     2.0.0
charset-normalizer       3.4.4
click                    8.3.1
colorama                 0.4.6
cryptography             46.0.3
dataclasses-json         0.6.7
distro                   1.9.0
emoji                    2.15.0
filetype                 1.2.0
frozenlist               1.8.0
greenlet                 3.3.0
h11                      0.16.0
html5lib                 1.1
httpcore                 1.0.9
```

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

Document loaders in LangChain are used to load data from different sources such as text files, PDF documents, CSV files, and web pages. In this lab, different loaders like TextLoader, PyPDFLoader, CSVLoader, WebBaseLoader, and DirectoryLoader are used. These loaders convert raw data into LangChain Document objects. Each Document object contains page_content, which stores the extracted text, and metadata, which stores information such as file name, page number, or URL. In Retrieval-Augmented Generation (RAG), document loaders prepare external knowledge so that it can be retrieved and supplied to large language models for generating accurate and context-aware response

Task 3: Load PDF Data (PyPDFLoader)

- Load lecture_notes.pdf
- Count total pages
- Display content of first page
- Attach code with output screenshot

```
from langchain_community.document_loaders import PyPDFLoader

loader = PyPDFLoader('dl-curriculum.pdf')

docs = loader.load()

print(len(docs))

print(docs[0].page_content)

print(docs[1].metadata)
```

```
  from pydantic.v1.fields import FieldInfo as FieldInfoV1
23
CampusXDeepLearningCurriculum
A.ArtificialNeuralNetworkandhowtoimprovethem
1.BiologicalInspiration
● Understandingtheneuronstructure● Synapsesandsignaltransmission● Howbiologicalconceptstranslatetoartificialneur
2.HistoryofNeuralNetworks
● Earlymodels(Perceptron)● BackpropagationandMLPs● The"AIWinter"andresurgenceofneuralnetworks● Emergenceofdeeplearning
3.PerceptronandMultilayerPerceptrons(MLP)
● Single-layerperceptronlimitations● XORproblemandtheneedforhiddenlayers● MLParchitecture
4. LayersandTheirFunctions
● InputLayero Acceptinginputdata● HiddenLayerso Featureextraction● OutputLayero Producingfinalpredictions
5.ActivationFunctions
{'producer': 'Skia/PDF m131 Google Docs Renderer', 'creator': 'PyPDF', 'creationdate': '', 'title': 'Deep Learning Curriculum', 'sourc
e': 'dl-curriculum.pdf', 'total_pages': 23, 'page': 1, 'page_label': '2'}
```

Task : 4 Structured Data (CSVLoader)

- Load students.csv
- Inspect how rows are converted into documents
- Print one document sample
- Attach code with output screenshot

```
from langchain_community.document_loaders import CSVLoader

loader = CSVLoader(file_path='Social_Network_Ads.csv')

docs = loader.load()


print(len(docs))

print(docs[1])
```

```
400
page_content='User ID: 15810944
Gender: Male
Age: 35
EstimatedSalary: 20000
Purchased: 0' metadata={'source': 'Social_Network_Ads.csv', 'row': 1}
```

Task 6: Compare All Loaders

Students must compare:

- Content format
- Metadata fields
- Attach code with output screenshot

```
from langchain_community.document_loaders import PyPDFLoader, CSVLoader, WebBaseLoader, TextLoader

loaders = {
    "PDF": PyPDFLoader("dl-curriculum.pdf"),
    "CSV": CSVLoader("Social_Network_Ads.csv"),
    "WEB": WebBaseLoader("https://www.langchain.com"),
    "TEXT": TextLoader("cricket.txt")
}

for name, loader in loaders.items():
    docs = loader.load()
    print(f"\n===== {name} Document Loader =====")
    print("Sample text:", docs[0].page_content[:200])
    print("Metadata:", docs[0].metadata)
```

```
===== CSV Document Loader =====
Sample text: User ID: 15624510
Gender: Male
Age: 19
EstimatedSalary: 19000
Purchased: 0
Metadata: {'source': 'Social_Network_Ads.csv', 'row': 0}

===== WEB Document Loader =====
Sample text: LangChain
```

COMPARISON TABLE:

Loader	Content Format	Metadata
PyPDFLoader	Page-wise text	page number, source
WebBaseLoader	Clean web text	URL
CSVLoader	Row-wise text	file path, row index

Lab Questions

1. What is the role of document loaders in RAG?

Answer:

They load external data and convert it into **Document objects** so LLMs can retrieve and use knowledge

2. Why is metadata important in LangChain documents?

Answer:

Metadata helps track **source, page number, URL**, improving accuracy and traceability.

3. Difference between TextLoader and PyPDFLoader?

Answer:

TextLoader: Loads plain text files

PyPDFLoader: Loads PDF files page by page

4. What happens if a PDF has scanned images instead of text?

Answer:

Text cannot be extracted without **OCR**, so content will be empty or unreadable.

5. Why is directory-based loading useful in real applications?

Answer:

It allows loading **multiple files automatically**, useful for large document collections.

6. How does document quality affect RAG performance?

Answer:

Clean, accurate documents give **better retrieval and more correct answers**.

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 N0. 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

```
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
```

Open browser at:

```
arduino

http://localhost:8501
```

```
from dotenv import load_dotenv
import os

load_dotenv()  # Reads the .env file

# Get API keys safely
openai_api_key = os.getenv("OPENAI_API_KEY")
if openai_api_key is None:
    raise ValueError("OPENAI_API_KEY not found! Please check your .env file.")
os.environ["OPENAI_API_KEY"] = openai_api_key

langchain_api_key = os.getenv("LANGCHAIN_API_KEY")
if langchain_api_key:
    os.environ["LANGCHAIN_API_KEY"] = langchain_api_key

# Optional: enable Langchain tracing
os.environ["LANGCHAIN_TRACING_V2"] = "true"

from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
```

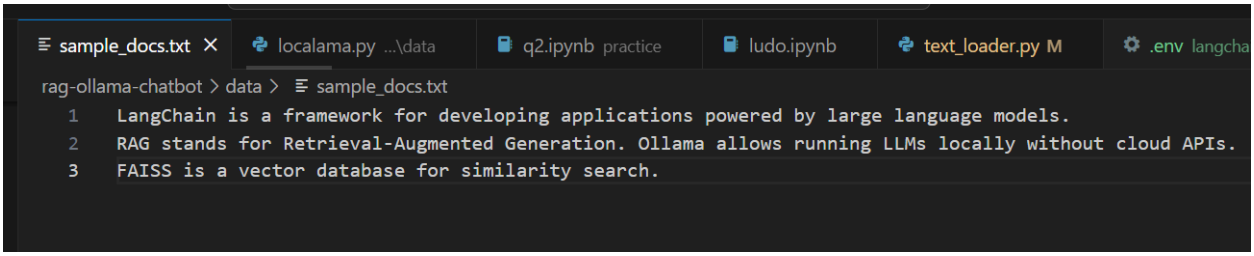
```
import streamlit as st
import os
from dotenv import load_dotenv


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}))
```

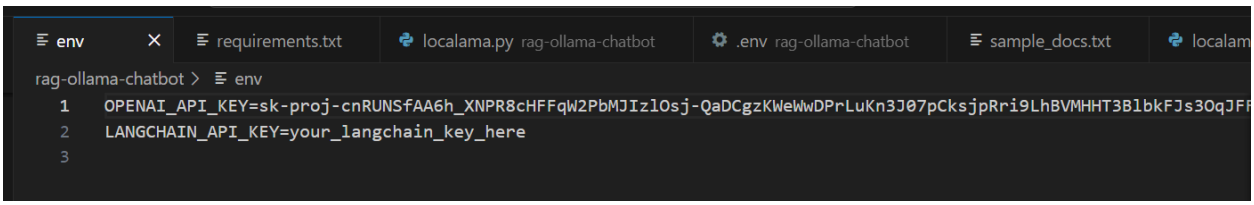
Requirements.txt:

```
langchain
langchain-community
langchain-core
langchain-openai
streamlit
faiss-cpu
python-dotenv
ollama
```

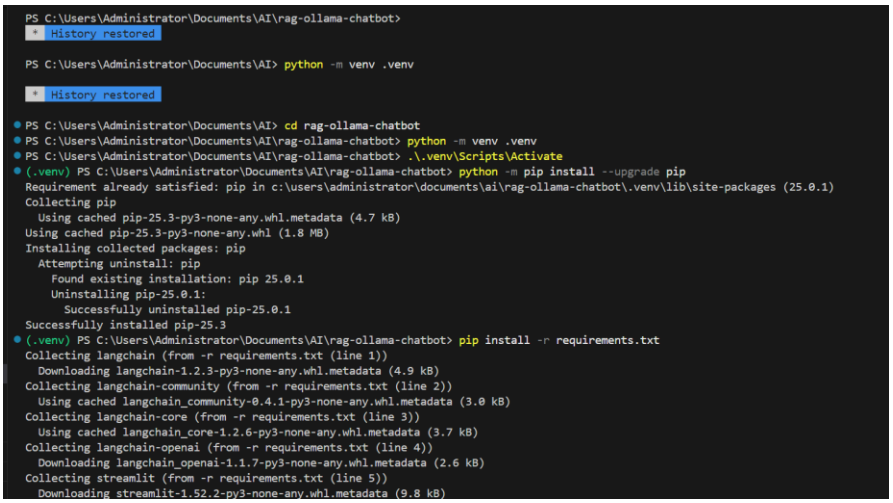
Sample.txt

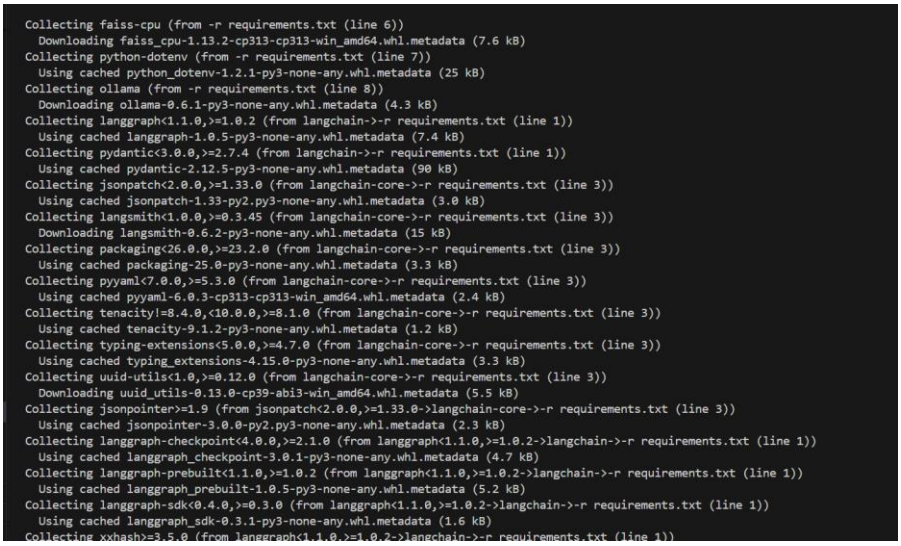


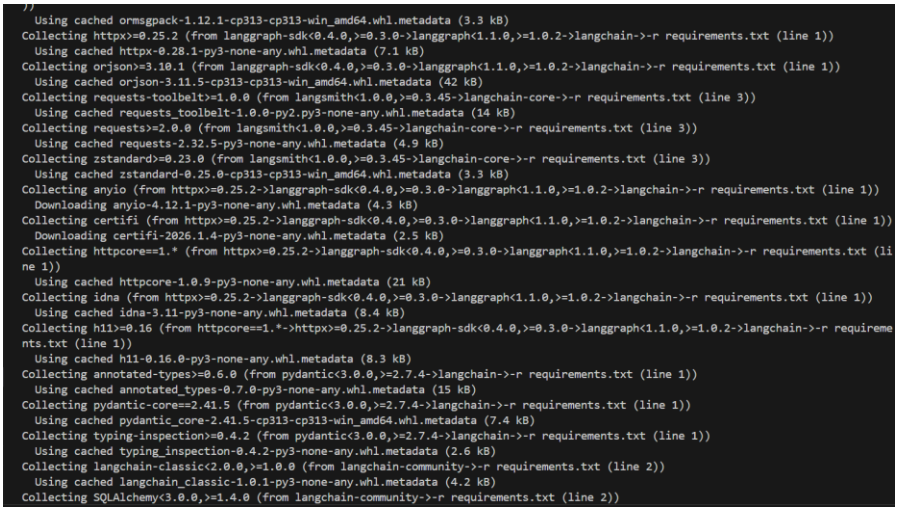
.env



Output:

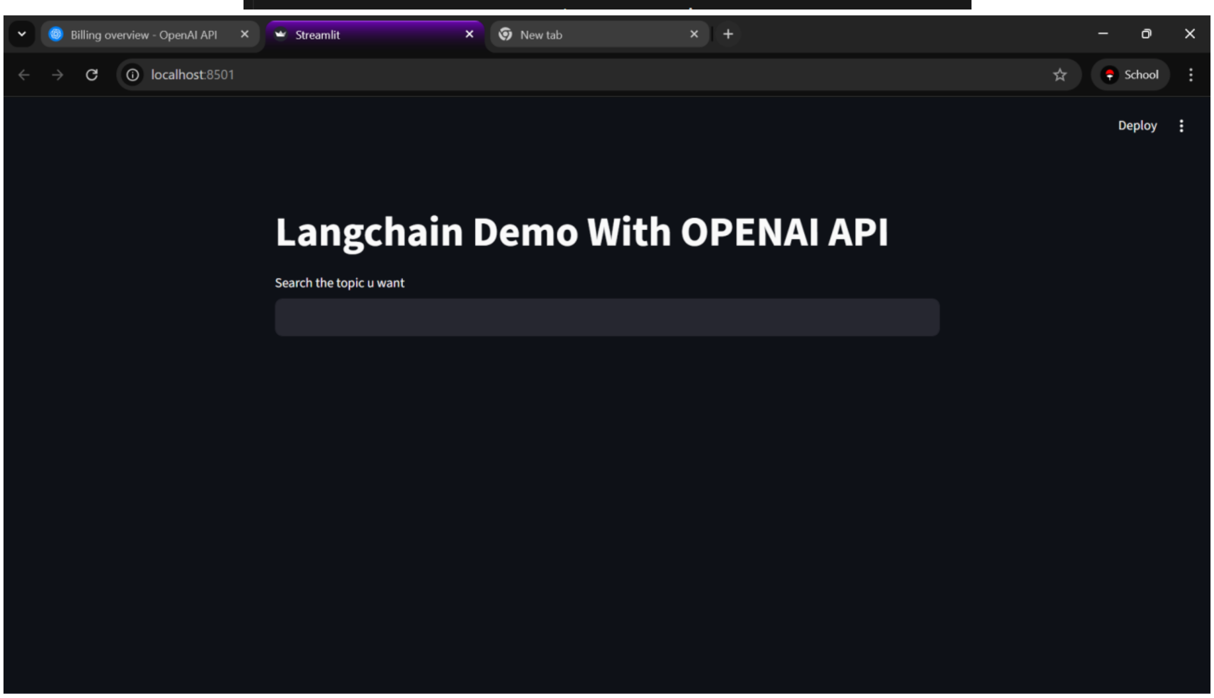

PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot>
History restored
PS C:\Users\Administrator\Documents\AI> python -m venv .venv
History restored
PS C:\Users\Administrator\Documents\AI> cd rag-ollama-chatbot
PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> python -m venv .venv
PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> .\.venv\Scripts\Activate
(.venv) PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\administrator\documents\ai\rag-ollama-chatbot\.venv\lib\site-packages (25.0.1)
Collecting pip
Using cached pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Using cached pip-25.3-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
Attempting uninstall: pip
Found existing installation: pip 25.0.1
Uninstalling pip-25.0.1:
Successfully uninstalled pip-25.0.1
Successfully installed pip-25.3
(.venv) PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> pip install -r requirements.txt
Collecting langchain (from -r requirements.txt (line 1))
Downloading langchain-1.2.3-py3-none-any.whl.metadata (4.9 kB)
Collecting langchain-community (from -r requirements.txt (line 2))
Using cached langchain_community-0.4.1-py3-none-any.whl.metadata (3.0 kB)
Collecting langchain-core (from -r requirements.txt (line 3))
Using cached langchain_core-1.2.6-py3-none-any.whl.metadata (3.7 kB)
Collecting langchain-openai (from -r requirements.txt (line 4))
Downloading langchain_openai-1.1.7-py3-none-any.whl.metadata (2.6 kB)
Collecting streamlit (from -r requirements.txt (line 5))
Downloading streamlit-1.52.2-py3-none-any.whl.metadata (9.8 kB)


Collecting faiss-cpu (from -r requirements.txt (line 6))
Downloading faiss_cpu-1.13.2-cp313-cp313-win_amd64.whl.metadata (7.6 kB)
Collecting python-dotenv (from -r requirements.txt (line 7))
Using cached python_dotenv-1.2.1-py3-none-any.whl.metadata (25 kB)
Collecting ollama (from -r requirements.txt (line 8))
Downloading ollama-0.6.1-py3-none-any.whl.metadata (4.3 kB)
Collecting langgraph<1.1.0,>=1.0.2 (from langchain->-r requirements.txt (line 1))
Using cached langgraph-1.0.5-py3-none-any.whl.metadata (7.4 kB)
Collecting pydantic<3.0.0,>=2.7.4 (from langchain->-r requirements.txt (line 1))
Using cached pydantic-2.12.5-py3-none-any.whl.metadata (90 kB)
Collecting jsonpatch<2.0.0,>=1.33.0 (from langchain-core->-r requirements.txt (line 3))
Using cached jsonpatch-1.33-py2.py3-none-any.whl.metadata (3.0 kB)
Collecting langsmith<1.0.0,>=0.3.45 (from langchain-core->-r requirements.txt (line 3))
Downloading langsmith-0.6.2-py3-none-any.whl.metadata (15 kB)
Collecting packaging<26.0.0,>=23.2.0 (from langchain-core->-r requirements.txt (line 3))
Using cached packaging-25.0-py3-none-any.whl.metadata (3.3 kB)
Collecting pyyaml<7.0.0,>=5.3.0 (from langchain-core->-r requirements.txt (line 3))
Using cached pyyaml-6.0.3-cp313-cp313-win_amd64.whl.metadata (2.4 kB)
Collecting tenacity!=8.4.0,<10.0.0,>=8.1.0 (from langchain-core->-r requirements.txt (line 3))
Using cached tenacity-9.1.2-py3-none-any.whl.metadata (1.2 kB)
Collecting typing-extensions<5.0.0,>=4.7.0 (from langchain-core->-r requirements.txt (line 3))
Using cached typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Collecting uuid-utils<1.0,>=0.12.0 (from langchain-core->-r requirements.txt (line 3))
Downloading uuid_utils-0.13.0-cp39-abi3-win_amd64.whl.metadata (5.5 kB)
Collecting jsonpointer>=1.9 (from jsonpatch<2.0.0,>=1.33.0->langchain-core->-r requirements.txt (line 3))
Using cached jsonpointer-3.0.0-py2.py3-none-any.whl.metadata (2.3 kB)
Collecting langgraph-checkpoint<4.0.0,>=2.1.0 (from langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Using cached langgraph_checkpoint-3.0.1-py3-none-any.whl.metadata (4.7 kB)
Collecting langgraph-prebuilt<1.1.0,>=1.0.2 (from langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Using cached langgraph_prebuilt-1.0.5-py3-none-any.whl.metadata (5.2 kB)
Collecting langgraph-sdk<0.4.0,>=0.3.0 (from langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Using cached langgraph_sdk-0.3.1-py3-none-any.whl.metadata (1.6 kB)
Collecting xxhash>=3.5.0 (from langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))


)
Using cached ormsgpack-1.12.1-cp313-cp313-win_amd64.whl.metadata (3.3 kB)
Collecting httpx>=0.25.2 (from langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Using cached httpx-0.28.1-py3-none-any.whl.metadata (7.1 kB)
Collecting orjson>=3.10.1 (from langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Using cached orjson-3.11.5-cp313-cp313-win_amd64.whl.metadata (42 kB)
Collecting requests-toolbelt>=1.0.0 (from langsmith<1.0.0,>=0.3.45->langchain-core->-r requirements.txt (line 3))
Using cached requests_toolbelt-1.0.0-py2.py3-none-any.whl.metadata (14 kB)
Collecting requests>=2.0.0 (from langsmith<1.0.0,>=0.3.45->langchain-core->-r requirements.txt (line 3))
Using cached requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting zstandard>=0.23.0 (from langsmith<1.0.0,>=0.3.45->langchain-core->-r requirements.txt (line 3))
Using cached zstandard-0.25.0-cp313-cp313-win_amd64.whl.metadata (3.3 kB)
Collecting anyio (from httpx>=0.25.2->langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Downloading anyio-4.12.1-py3-none-any.whl.metadata (4.3 kB)
Collecting certifi (from httpx>=0.25.2->langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Downloading certifi-2026.1.4-py3-none-any.whl.metadata (2.5 kB)
Collecting httpcore==1.* (from httpx>=0.25.2->langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (li
ne 1))
Using cached httpcore-1.0.9-py3-none-any.whl.metadata (21 kB)
Collecting idna (from httpx>=0.25.2->langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requirements.txt (line 1))
Using cached idna-3.11-py3-none-any.whl.metadata (8.4 kB)
Collecting h11>=0.16 (from httpcore==1.*->httpx>=0.25.2->langgraph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain->-r requireme
nts.txt (line 1))
Using cached h11-0.16.0-py3-none-any.whl.metadata (8.3 kB)
Collecting annotated-types>=0.6.0 (from pydantic<3.0.0,>=2.7.4->langchain->-r requirements.txt (line 1))
Using cached annotated_types-0.7.0-py3-none-any.whl.metadata (15 kB)
Collecting pydantic-core==2.41.5 (from pydantic<3.0.0,>=2.7.4->langchain->-r requirements.txt (line 1))
Using cached pydantic_core-2.41.5-cp313-cp313-win_amd64.whl.metadata (7.4 kB)
Collecting typing-inspection>=0.4.2 (from pydantic<3.0.0,>=2.7.4->langchain->-r requirements.txt (line 1))
Using cached typing_inspection-0.4.2-py3-none-any.whl.metadata (2.6 kB)
Collecting langchain-classic<2.0.0,>=1.0.0 (from langchain-community->-r requirements.txt (line 2))
Using cached langchain_classic-1.0.1-py3-none-any.whl.metadata (4.2 kB)
Collecting SQLAlchemy<3.0.0,>=1.4.0 (from langchain-community->-r requirements.txt (line 2))

```
zdata-2025.3 urllib3-2.6.3 uuid-utils-0.13.0 watchdog-6.0.0 xxhash-3.6.0 yarl-1.22.0 zstandard-0.25.0
(.venv) PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> ollama pull llama2
pulling manifest
pulling 8934d96d3f08: 100%  3.8 GB
pulling 8c17c2ebb0ea: 100%  7.0 KB
pulling 7c23fb36d801: 100%  4.8 KB
pulling 2e0493f67d0c: 100%    59 B
pulling fa304d675061: 100%    91 B
pulling 42ba7f8a01dd: 100%   557 B
verifying sha256 digest
writing manifest
success
(.venv) PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> ollama --version
ollama version is 0.13.5
(.venv) PS C:\Users\Administrator\Documents\AI\rag-ollama-chatbot> streamlit run app.py

      Welcome to Streamlit!

      If you'd like to receive helpful onboarding emails, news, offers, promotions,
      and the occasional swag, please enter your email address below. Otherwise,
      leave this field blank.

      Email:

  You can find our privacy policy at https://streamlit.io/privacy-policy

  Summary:
  - This open source library collects usage statistics.
  - We cannot see and do not store information contained inside Streamlit apps,
    such as text, charts, images, etc.
  - Telemetry data is stored in servers in the United States.
  - If you'd like to opt out, add the following to %userprofile%/.streamlit/config.toml,
    creating that file if necessary:
```

```
    [browser]
    gatherUsageStats = false


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.113:8501
```



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