Artificial Intelligence

CS13207

Laboratory Manual

Student Copy

FALL 2025

Submitted to:

Dr. Syed M Hamedoon
Assistant Professor

Submitted by:

Name	Aqib Syed
Registration No	70148959
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: Aqib Syed

Registration No.: 70148959

Student Signature: _____

Date: 22-01-2026

Lab Instructor: _____

Laboratory Work Assessment Rubrics

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

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

Lab's Course Learning Outcomes

Course Title : Artificial Intelligence

Course Code : CS13207

Instructor : *Dr. Syed M Hamedoon*

Designation : Assistant Professor

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

Students will be able to:

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

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

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

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

List of Practical Tasks

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

EVALUATION LOG

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

LAB No. 1

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

LAB Description

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

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

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

Lab Objective

The objectives of this lab are:

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

How to use python in VS code?

1. Create environment named 'venv'

python -m venv

2. Activate environment

venv\Scripts\activate

3. Select Environment in VS CodeAfter creating the environment:

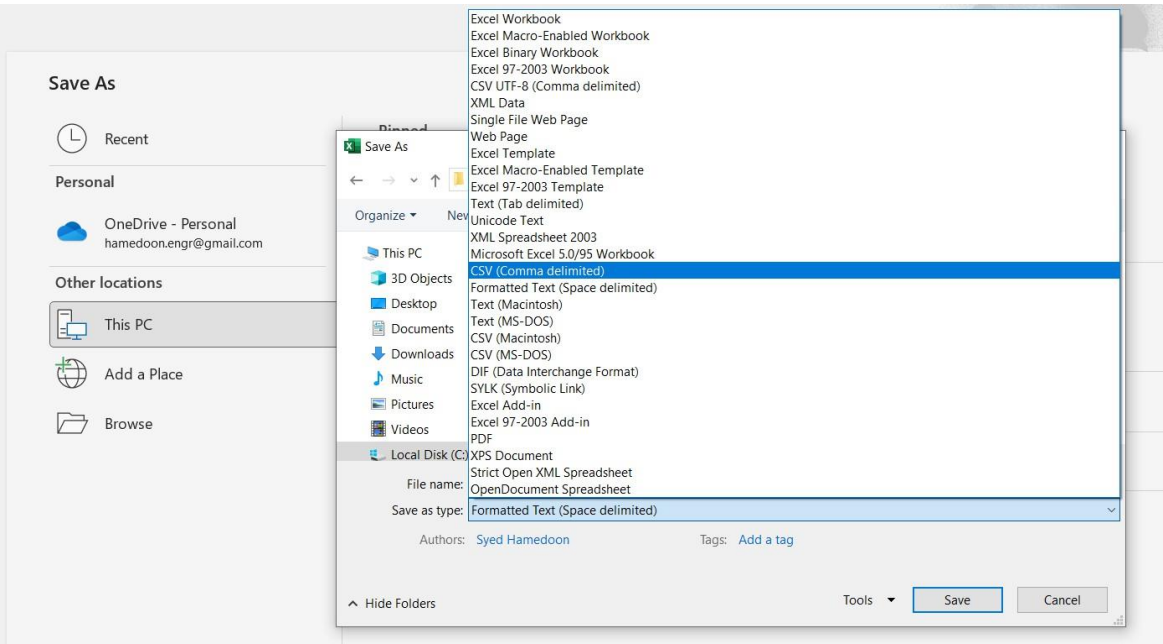
1) Open your project folder in VS Code.
2) Press Ctrl + Shift + P → search for Python:
3) Select Interpreter.
4) Choose your newly created environment (venv or myenv).

4. Install Machine Learning and Deep Learning Libraries

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

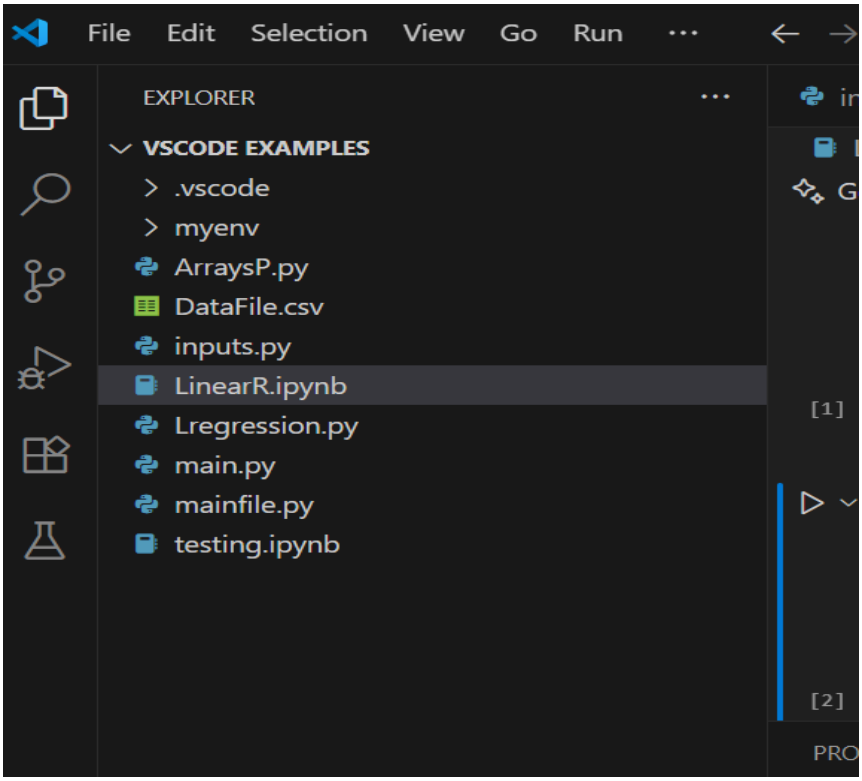
Task 1:

Save Excel file as .csv (Comma delimited)

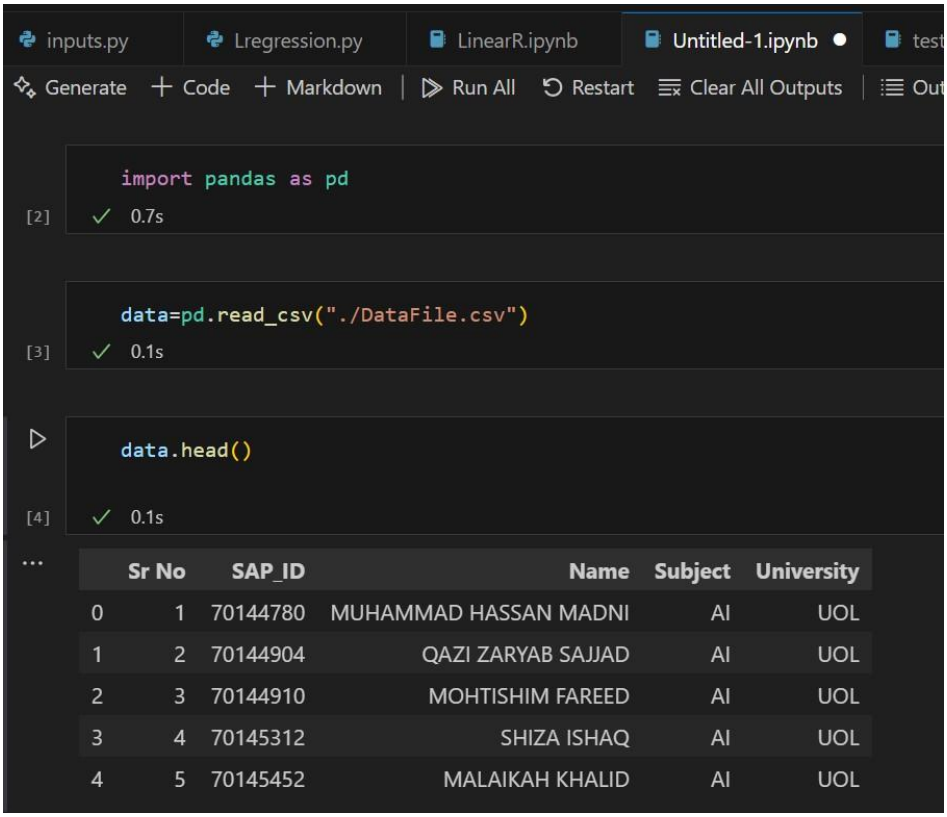


Task 2:

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



Here we can see our file successfully



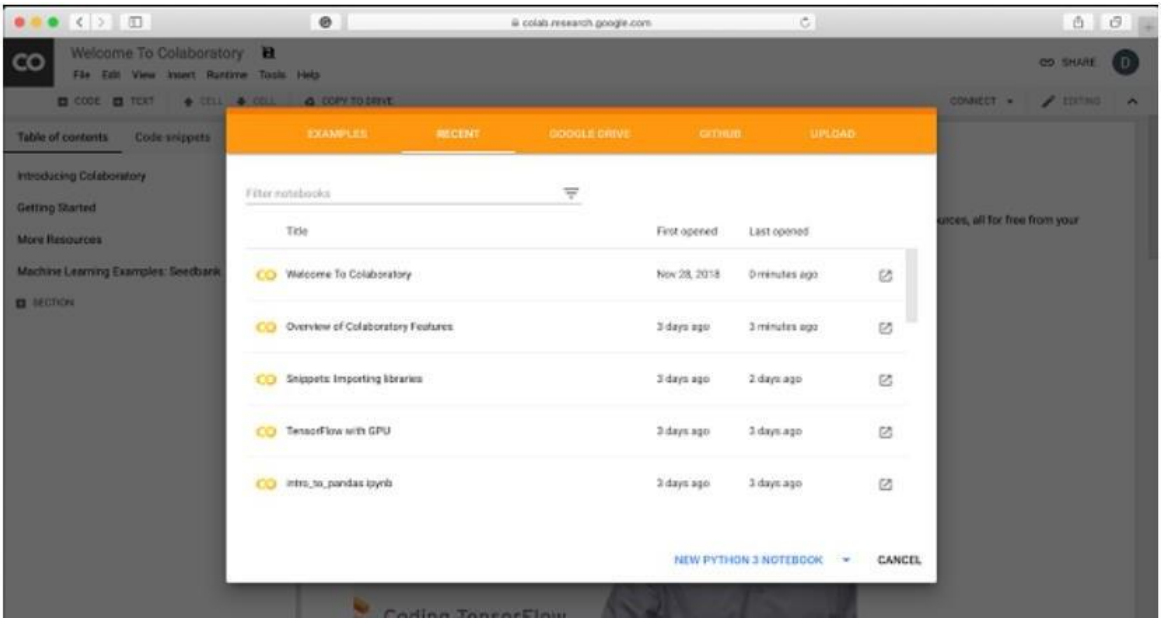
	Sr No	SAP_ID	Name	Subject	University
0	1	70144780	MUHAMMAD HASSAN MADNI	AI	UOL
1	2	70144904	QAZI ZARYAB SAJJAD	AI	UOL
2	3	70144910	MOHTISHIM FAREED	AI	UOL
3	4	70145312	SHIZA ISHAQ	AI	UOL
4	5	70145452	MALAIKAH KHALID	AI	UOL

Introduction of Google Colab

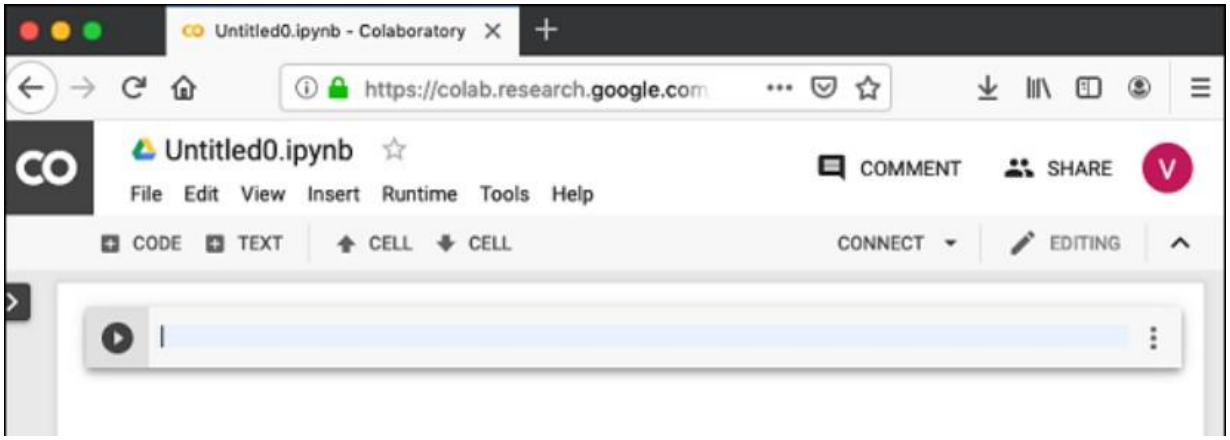
- Colab is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs.
- Colab supports many popular machine learning libraries which can be easily loaded in your notebook.
- Another attractive feature that Google offers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

just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook

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

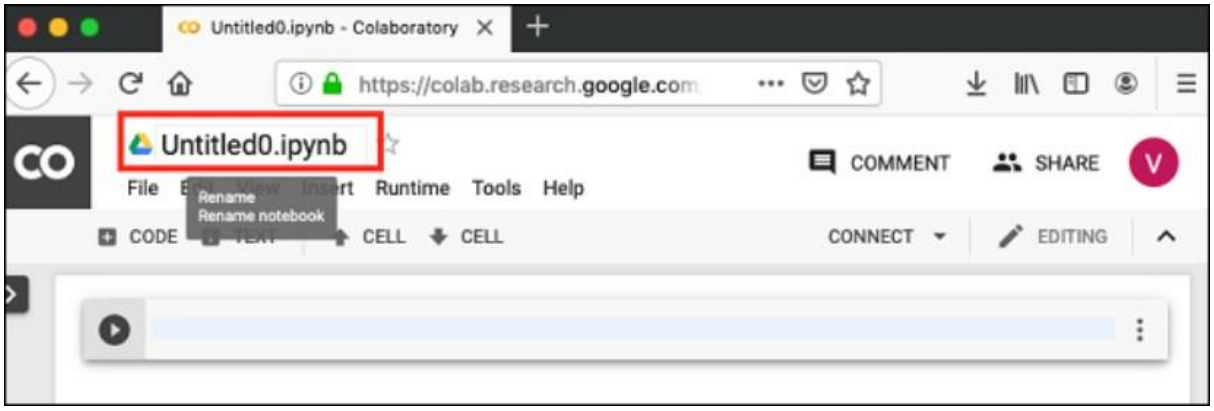


- **Step 2** – Click on the **NEW PYTHON 3 NOTEBOOK** link at the bottom of the screen. A new notebook would open up as shown in the screen below.

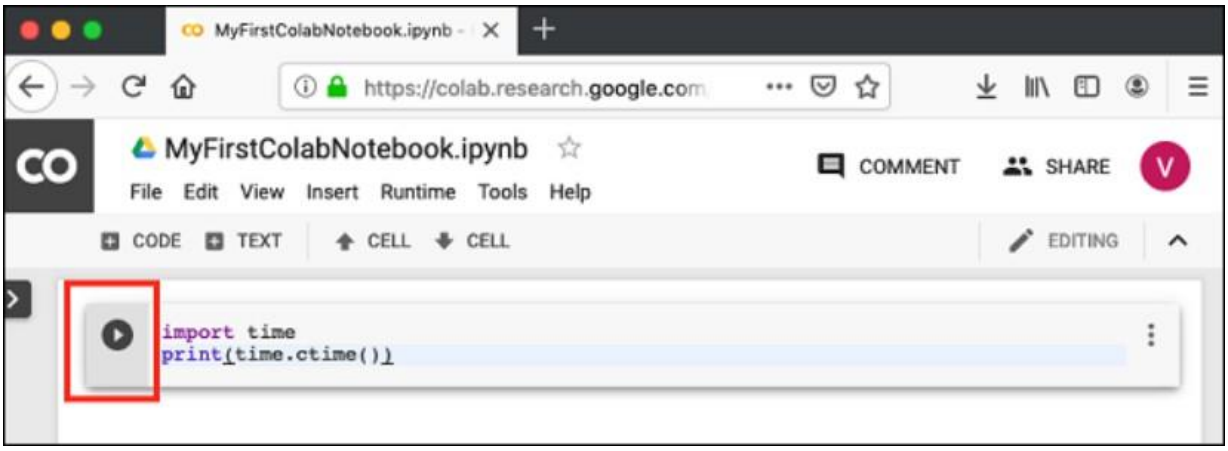


- **Setting Notebook Name**

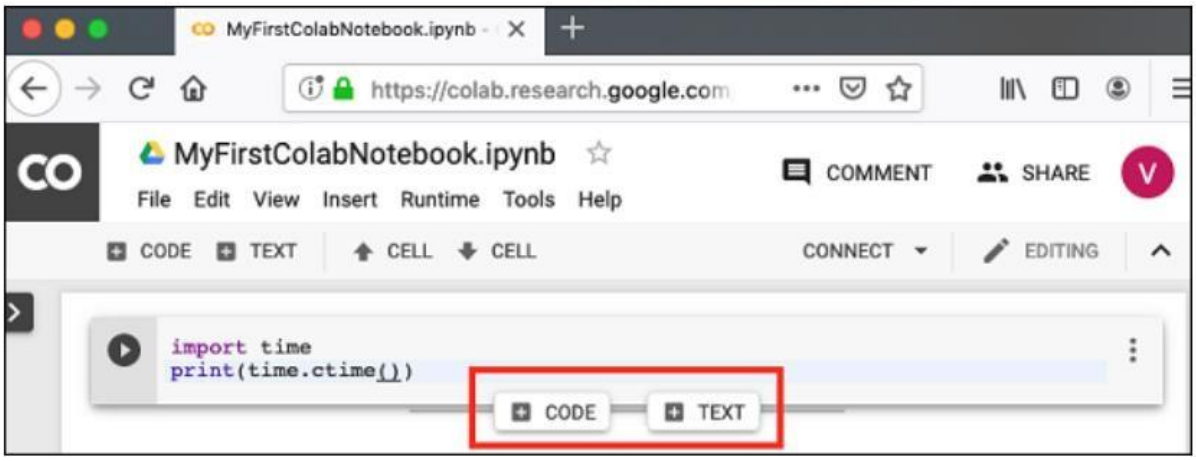
  By default, the notebook uses the naming convention UntitledXX.ipynb. To rename the notebook, click on this name and type in the desired name in the edit box as shown here –



- Entering Code

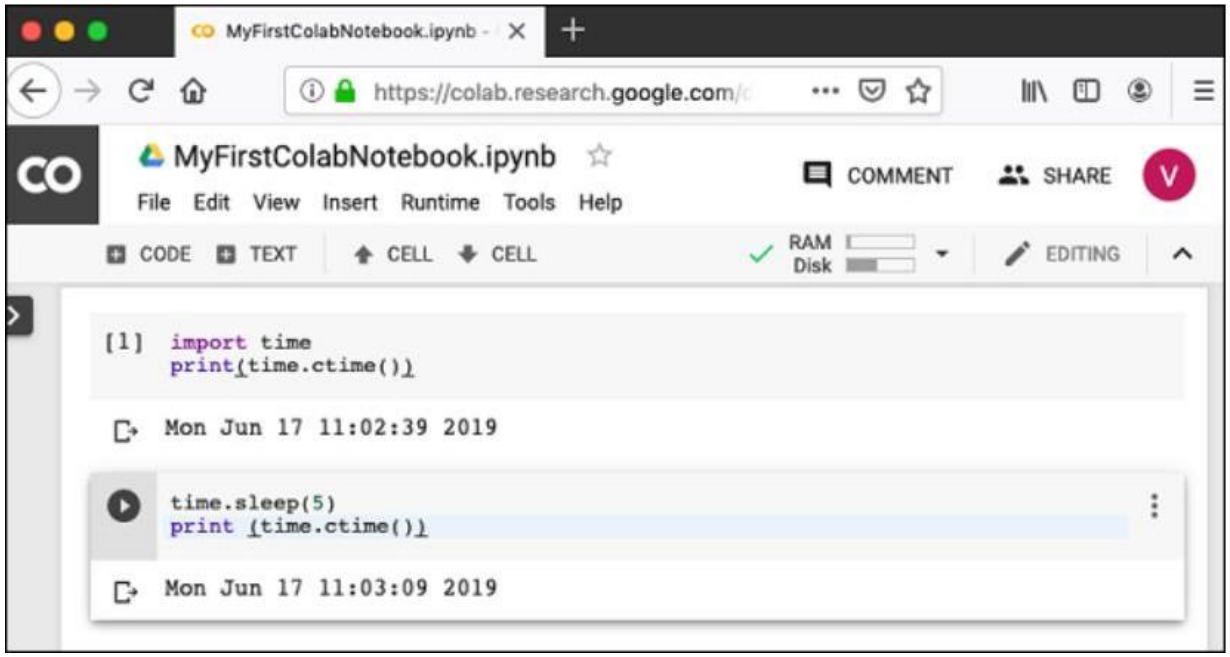


- Adding Code Cells
- To add more code to your notebook, select the following **menu** options –

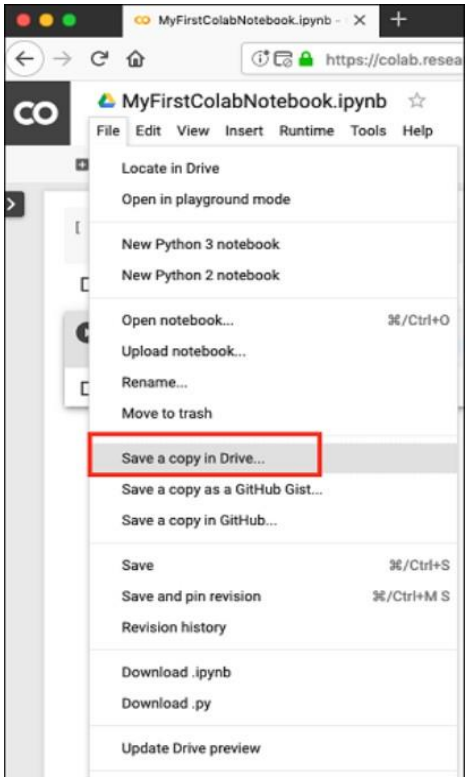


- **Run All**

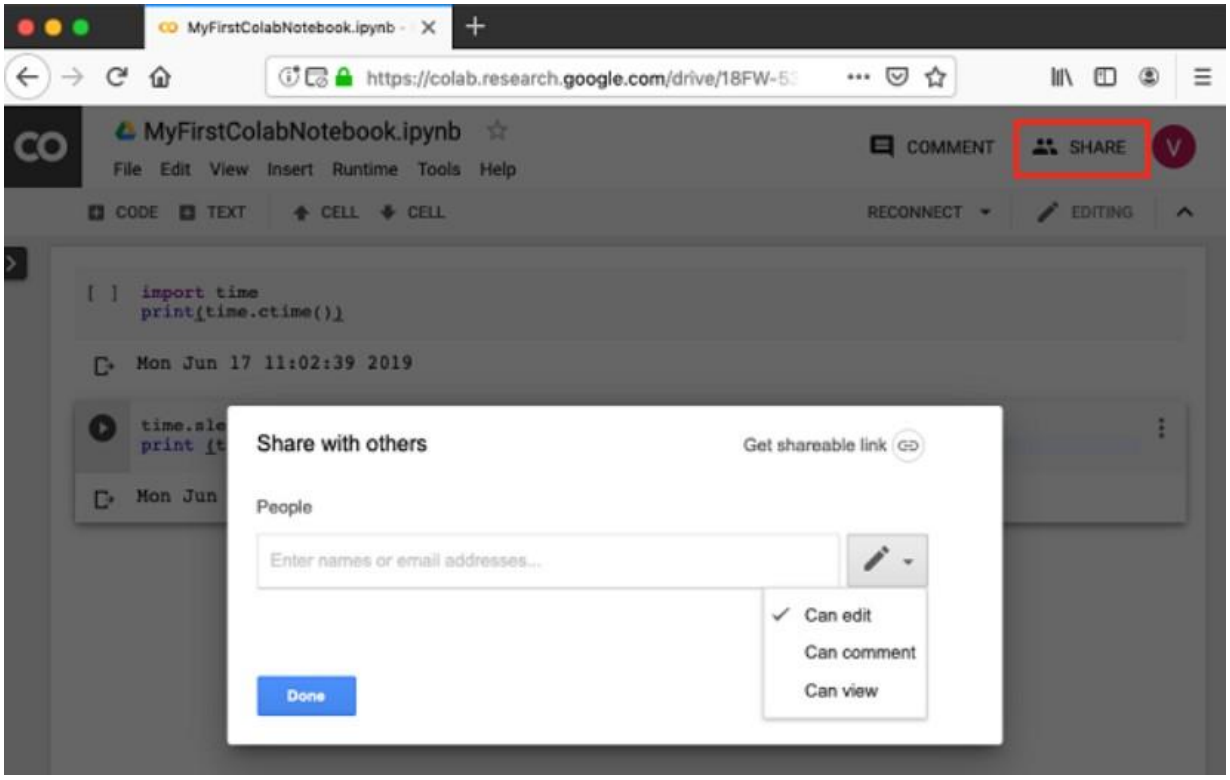
  To run the entire code in your notebook without an interruption, execute the following menu options –



Saving to Google Drive



Google Colab - Sharing Notebook


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

Lab Assignment No. 1

Objective

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

Tasks

🔹 Q1: Create a Dataset Manually

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

```
print("Data Info:\n")
print(data.info())

print("\nData Description:\n")
print(data.describe())

print("\nMean of Math Marks:", data['Marks_Math'].mean())
print("Maximum Science Marks:", data['Marks_Science'].max())
```

```
Data Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Student_ID     20 non-null     int64
 1   Name           20 non-null     object
 2   Age            20 non-null     int64
 3   Marks_Math     20 non-null     int64
 4   Marks_Science  20 non-null     int64
dtypes: int64(4), object(1)
memory usage: 932.0+ bytes
None

Data Description:

       Student_ID        Age  Marks_Math  Marks_Science
count    20.00000  20.000000   20.000000      20.000000
mean     10.50000  20.150000   68.350000      72.550000
std       5.91608   1.663066   16.743498      14.741545
min       1.00000  18.000000   34.000000      40.000000
25%       5.75000  19.000000   57.750000      63.000000
50%      10.50000  20.000000   69.000000      73.000000
75%      15.25000  21.250000   81.250000      83.500000
max      20.00000  23.000000   92.000000      95.000000

Mean of Math Marks: 68.35
Maximum Science Marks: 95
```

Q2: Upload Dataset in Python

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

```
import pandas as pd

data = {
    "Student_ID": list(range(1, 21)),
    "Name": ["Ali", "Sara", "Bilal", "Hina", "Ahmed", "Iqra", "Usman", "Kiran", "Adeel", "Fatima",
             "Zain", "Nida", "Omar", "Laiba", "Hamza", "Anum", "Danish", "Rabia", "Salman", "Maira"],
    "Age": [18, 19, 20, 18, 21, 22, 19, 20, 23, 18,
            21, 19, 22, 20, 23, 21, 18, 19, 20, 22],
    "Marks_Math": [45, 67, 89, 54, 76, 34, 92, 60, 71, 80,
                   50, 66, 72, 85, 91, 48, 59, 63, 77, 88],
    "Marks_Science": [55, 78, 90, 60, 72, 40, 95, 85, 66, 88,
                      52, 81, 74, 69, 92, 58, 64, 70, 79, 83]
}

df = pd.DataFrame(data)
df.to_csv("students.csv", index=False)
df
```

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

```
print("\nStudents with Math > 50:\n", data[data['Marks_Math'] > 50])

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

print("\nCorrelation:\n", data['Marks_Math'].corr(data['Marks_Science']))
```

```
Students with Math > 50:
     Student_ID    Name  Age  Marks_Math  Marks_Science
1             2    Sara   19          67             78
2             3   Bilal   20          89             90
3             4    Hina   18          54             60
4             5   Ahmed   21          76             72
6             7   Usman   19          92             95
7             8   Kiran   20          60             85
8             9   Adeel   23          71             66
9            10  Fatima   18          80             88
11           12    Nida   19          66             81
12           13    Omar   22          72             74
13           14   Laiba   20          85             69
14           15   Hamza   23          91             92
16           17  Danish   18          59             64
17           18   Rabia   19          63             70
18           19  Salman   20          77             79
19           20   Maira   22          88             83

Student with Highest Science Marks:
    Student_ID   Name  Age  Marks_Math  Marks_Science
6            7  Usman   19          92             95

Correlation:
 0.8617120280907411
```

Q4: Perform Some Data Analysis

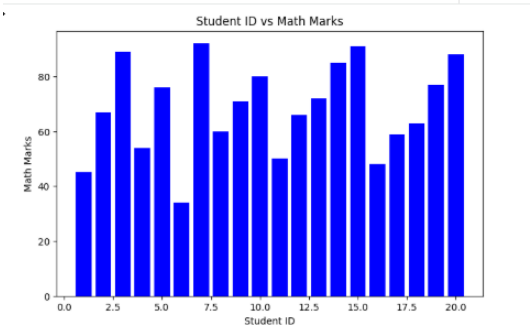
- Find how many students have Marks_Math > 50.
- Find the student with the **highest Science marks**.
- Calculate the **correlation** between Marks_Math and Marks_Science.

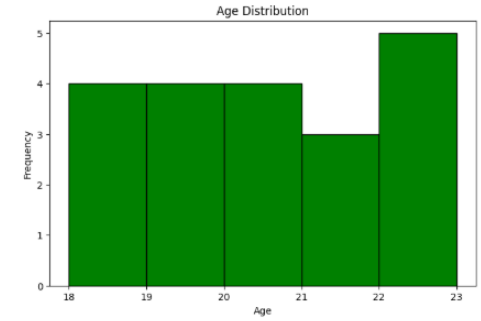
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,5))
plt.bar(data['Student_ID'], data['Marks_Math'], color='blue')
plt.xlabel("Student ID")
plt.ylabel("Math Marks")
plt.title("Student ID vs Math Marks")
plt.show()

plt.figure(figsize=(8,5))
plt.hist(data['Age'], bins=5, color='green', edgecolor='black')
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Age Distribution")
plt.show()

plt.figure(figsize=(8,5))
sns.scatterplot(x=data['Marks_Math'], y=data['Marks_Science'], s=100, color='red')
plt.xlabel("Math Marks")
plt.ylabel("Science Marks")
plt.title("Math vs Science Marks")
plt.show()
```

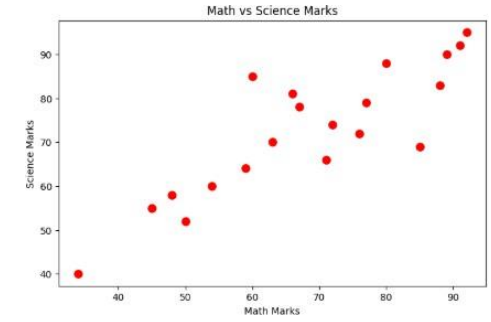
🔹 Q5: Data Visualization

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

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

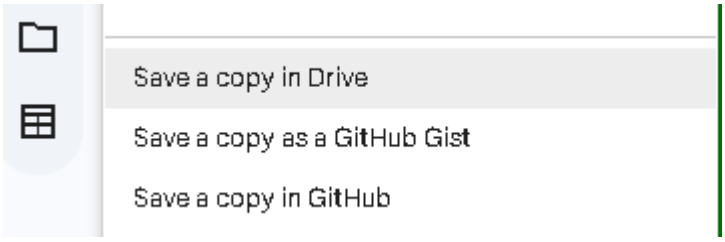






Q6: Save Your Work

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



LAB Assessment

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

Laboratory Work Assessment Rubrics

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

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

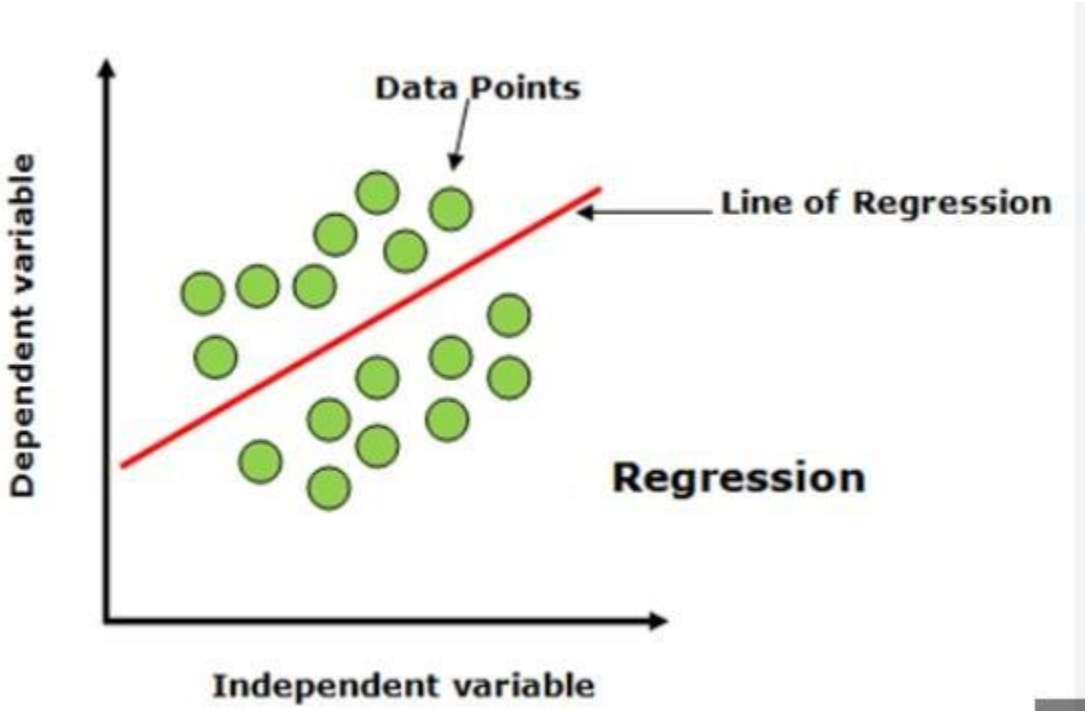
LAB No. 2

Implementation of Regression Analysis using python

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

Introduction

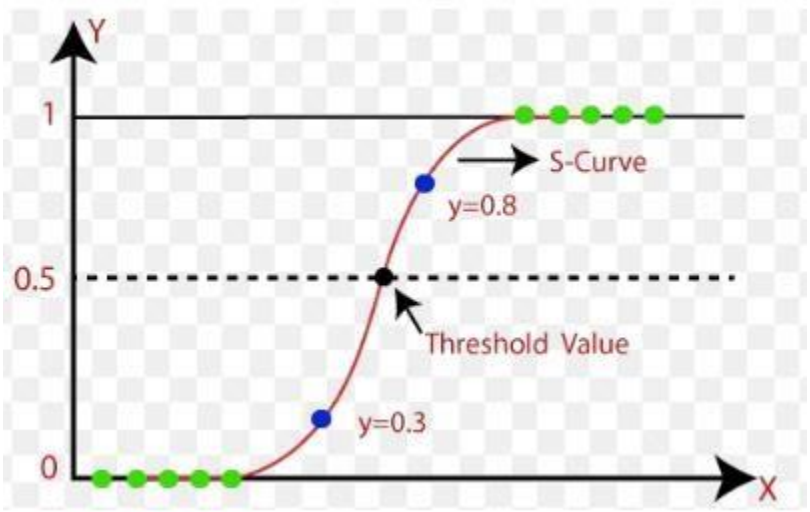
Regression analysis is a fundamental technique in machine learning and data science used to understand the relationship between variables and to make predictions. In this lab, students will explore **Multiple Linear Regression** and **Logistic Regression** using Python.



Multiple Linear Regression is used when a dependent (output) variable is continuous and depends on two or more independent (input) variables. It

helps in modeling and analyzing how changes in multiple features affect a numerical outcome, such as predicting house prices based on area, number of rooms, and location.

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



Solved Examples

Example 1

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

Solution:

```
# Import required libraries
import pandas as pd
```

```
from sklearn.linear_model import LinearRegression

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

df = pd.DataFrame(data)

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

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

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

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

Example 2

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

Solution:

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

# Dataset
data = {
  'Area': [800, 1000, 1200, 1500, 1800],
  'Bedrooms': [1, 2, 2, 3, 3],
```

```
  'Price': [50000, 65000, 75000, 90000, 110000]
}

df = pd.DataFrame(data)

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

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

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

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

Example 3

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

Solution:

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

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

df = pd.DataFrame(data)

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

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

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

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

LAB Assignment No. 2

Lab Practice Questions

Q1. Multiple Linear Regression - House Price Prediction

A dataset contains:

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

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

— Task:

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

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

data = {
    'Size': [1500, 1800, 2400, 3000, 3500],
    'Bedrooms': [3, 4, 3, 5, 4],
    'Age': [10, 15, 20, 5, 8],
    'Price': [300000, 350000, 400000, 500000, 450000]
}

df = pd.DataFrame(data)

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

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

print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

predicted_price = model.predict([[2000, 3, 10]])
print("Predicted Price:", predicted_price[0])
```

```
Intercept: 33318.44795077853
Coefficients: [   67.77810856 45643.54470577  2401.50838543]
Predicted Price: 329820.383050511
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Q2. Multiple Linear Regression – Student Performance

Dataset columns:

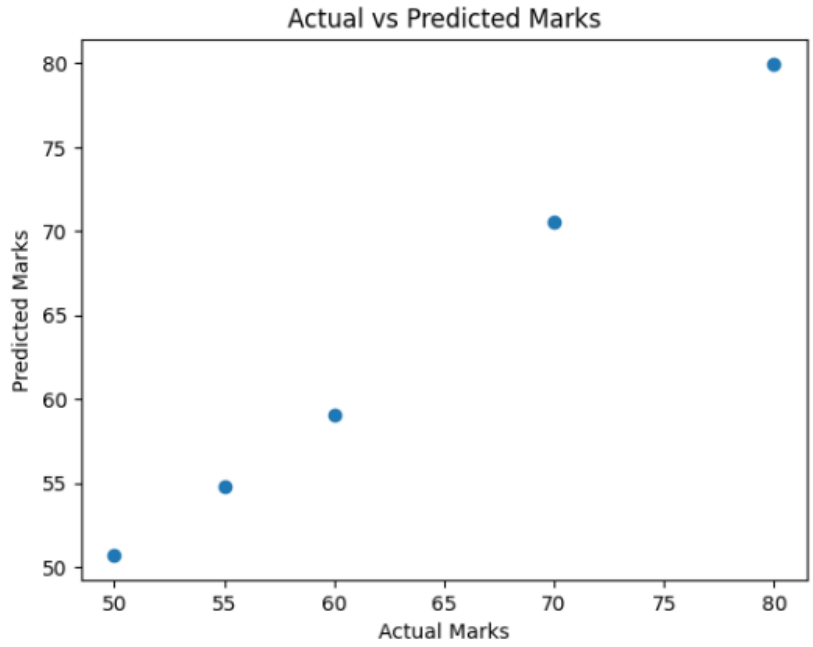
- Hours Study,
- Hours Sleep,
- Attendance (%),
  Target: **Marks in Exam**

— Task:

1. Train a regression model.
2. Plot actual vs predicted marks.
3. Compute **R² score** and **Mean Squared Error (MSE)**.

```
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score, mean_squared_error

data = {
    'Hours_Study': [2, 3, 4, 5, 6],
    'Hours_Sleep': [6, 7, 6, 8, 7],
    'Attendance': [80, 85, 78, 90, 95],
    'Marks': [50, 60, 55, 70, 80]
}

df = pd.DataFrame(data)
X = df[['Hours_Study', 'Hours_Sleep', 'Attendance']]
y = df['Marks']

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

plt.scatter(y, y_pred)
plt.xlabel('Actual Marks')
plt.ylabel('Predicted Marks')
plt.title('Actual vs Predicted Marks')
plt.show()

r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)
print("R² Score:", r2)
print("Mean Squared Error:", mse)
```

..




```
R² Score: 0.9969480950084398
Mean Squared Error: 0.35402097902097973
```

Q3. Logistic Regression – Pass/Fail Classification

Dataset columns:

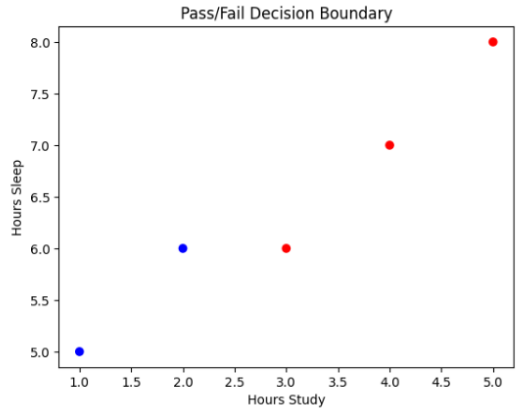
- Hours Study
- Hours Sleep
  Target: Pass (1) / Fail (0)

— Task:

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

```python
import numpy as np
from sklearn.linear_model import LogisticRegression

data = {
    'Hours_Study': [1, 2, 3, 4, 5],
    'Hours_Sleep': [5, 6, 6, 7, 8],
    'Pass': [0, 0, 1, 1, 1]
}

df = pd.DataFrame(data)
X = df[['Hours_Study', 'Hours_Sleep']]
y = df['Pass']

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

prob = model.predict_proba([[30, 6]])[0][1]
print("Probability of passing:", prob)

import matplotlib.pyplot as plt

plt.scatter(df['Hours_Study'], df['Hours_Sleep'], c=df['Pass'], cmap='bwr')
plt.xlabel('Hours Study')
plt.ylabel('Hours Sleep')
plt.title('Pass/Fail Decision Boundary')
plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
Probability of passing: 0.9999999999923157
```



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.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

data = {
    'BMI': [22, 28, 30, 25, 32],
    'Age': [25, 45, 50, 30, 60],
    'Glucose': [100, 150, 180, 120, 200],
    'Diabetic': [0, 1, 1, 0, 1]
}

df = pd.DataFrame(data)
X = df[['BMI', 'Age', 'Glucose']]
y = df['Diabetic']

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

accuracy = accuracy_score(y, y_pred)
precision = precision_score(y, y_pred)
recall = recall_score(y, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)

patient = [[28, 45, 150]]
prediction = model.predict(patient)[0]
print("Is the patient diabetic?", "Yes" if prediction == 1 else "No")
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
Is the patient diabetic? Yes
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

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.

```
#Q5
data = {
    'Hours_Study': [1, 2, 3, 4, 5],
    'Exam_Score': [40, 50, 60, 70, 80],
    'Pass': [0, 0, 1, 1, 1]
}

df = pd.DataFrame(data)
X = df[['Hours_Study']]
y_score = df['Exam_Score']
y_pass = df['Pass']

lin_model = LinearRegression()
lin_model.fit(X, y_score)
y_score_pred = lin_model.predict(X)

log_model = LogisticRegression()
log_model.fit(X, y_pass)
y_pass_pred = log_model.predict(X)

print("Linear Regression predictions:", y_score_pred)
print("Logistic Regression predictions:", y_pass_pred)
```

```
Linear Regression predictions: [40. 50. 60. 70. 80.]
Logistic Regression predictions: [0 0 1 1 1]
```

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

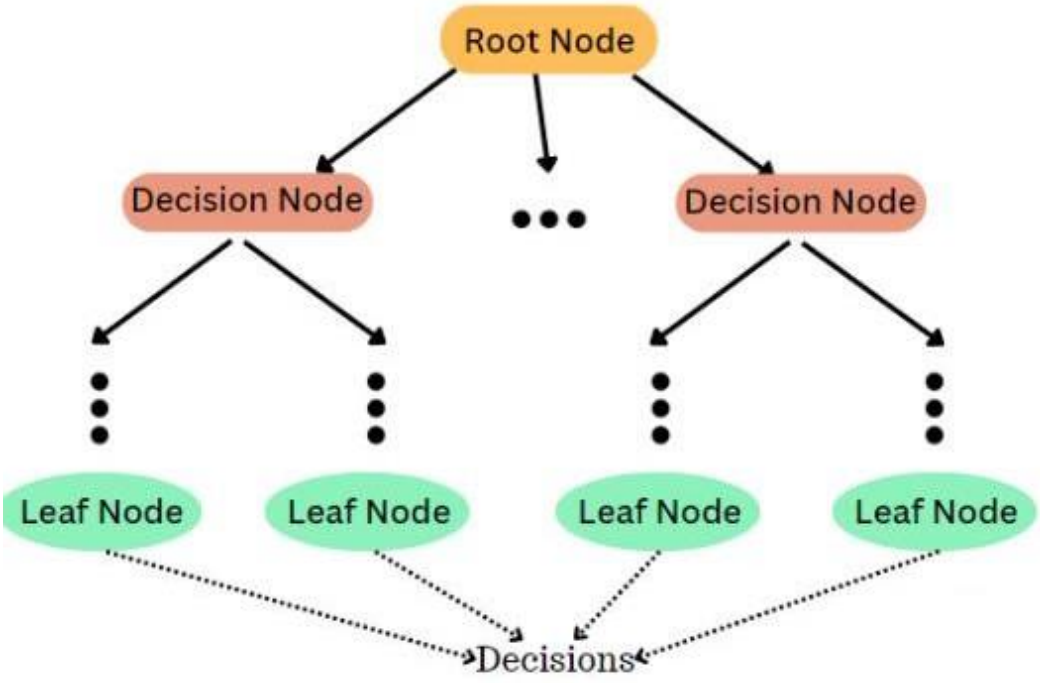
In this lab, students will study and implement the Decision Tree Classifier, a supervised machine learning algorithm used for classification tasks. The lab begins with a manual calculation of entropy and information gain to understand how decision trees choose splitting attributes. Students will then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

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

The construction of a decision tree is based on:

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



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

Solved Examples:

Example 1: Entropy and Information Gain

Dataset

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

1. Entropy of Target Variable (Result)

- Pass = 3
- Fail = 2

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

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

$$Entropy(Result) \approx 0.971$$

2. Information Gain for Study Hours

Study Hours = High

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

Study Hours = Low

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

Weighted Entropy:

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

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

3. Root Node Selection

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

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

Question

Build and visualize a decision tree using entropy.

Solution:

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

# Create dataset
data = {
  'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
```

```
  'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
  'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

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

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

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

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

Example 3: Decision Tree Classifier on Iris Dataset

Objective

Apply decision trees to a real dataset and evaluate accuracy.

Solution

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

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

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

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


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

print("Accuracy:", accuracy)

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

LAB Assignment No. 3

Question 1

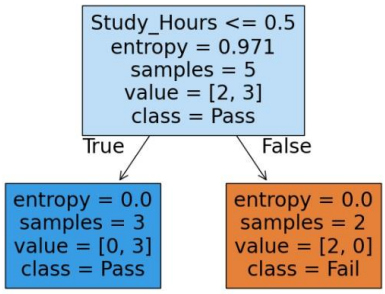
Entropy and Information Gain (Manual Calculation)

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

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

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

```
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', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

le_study = LabelEncoder()
le_attendance = LabelEncoder()
le_result = LabelEncoder()

df['Study_Hours_n'] = le_study.fit_transform(df['Study_Hours'])
df['Attendance_n'] = le_attendance.fit_transform(df['Attendance'])
df['Result_n'] = le_result.fit_transform(df['Result'])

X = df[['Study_Hours_n', 'Attendance_n']]
y = df['Result_n']

clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X, y)

plt.figure(figsize=(8,6))
plot_tree(clf, feature_names=['Study_Hours','Attendance'], class_names=['Fail','Pass'], filled=True)
plt.show()

student = pd.DataFrame({'Study_Hours_n':[le_study.transform(['Low'])[0]],
                        'Attendance_n':[le_attendance.transform(['Good'])[0]]})
prediction = clf.predict(student)[0]
print("Prediction:", le_result.inverse_transform([prediction])[0])
```




```
Prediction: Fail
```

Question No. 2

Implement Decision Tree Classifier on a Small Dataset

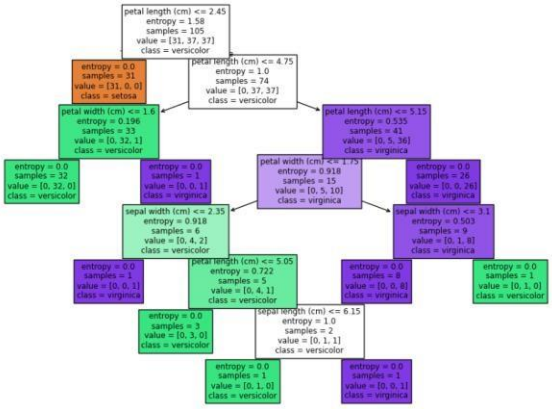
Build and visualize a simple decision tree.

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.

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

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)

clf = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", accuracy)

plt.figure(figsize=(12,8))
plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.show()
```

```
Test Accuracy: 0.9777777777777777
```



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.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
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)

clf = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", accuracy)

plt.figure(figsize=(12,8))
plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.show()
```

```
Test Accuracy: 0.9777777777777777
```

Question 4

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

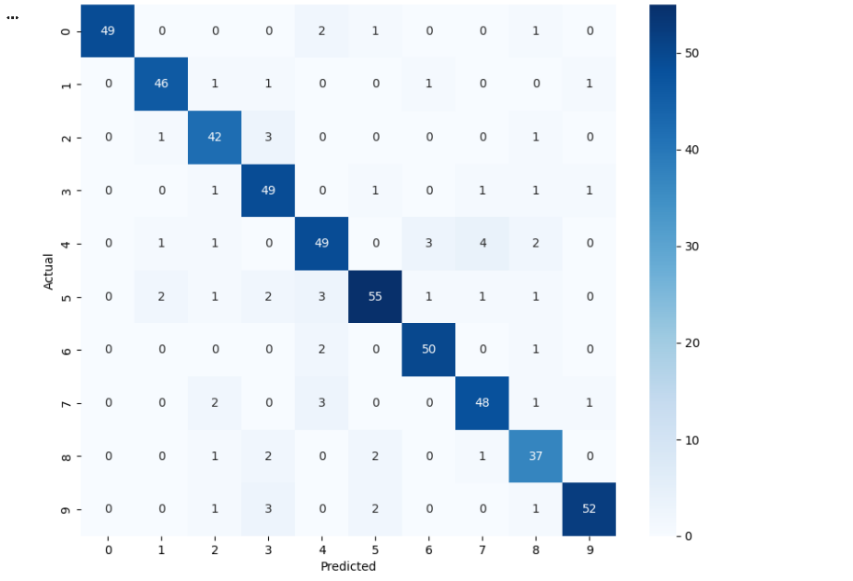
Objectives

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

```python
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns

digits = load_digits()
X = digits.data  # 64 features (8x8 images)
y = digits.target

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

clf = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print("Accuracy:", acc)

# Confusion Matrix
plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Accuracy: 0.8833333333333333
```

Accuracy: 0.8833333333333333



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

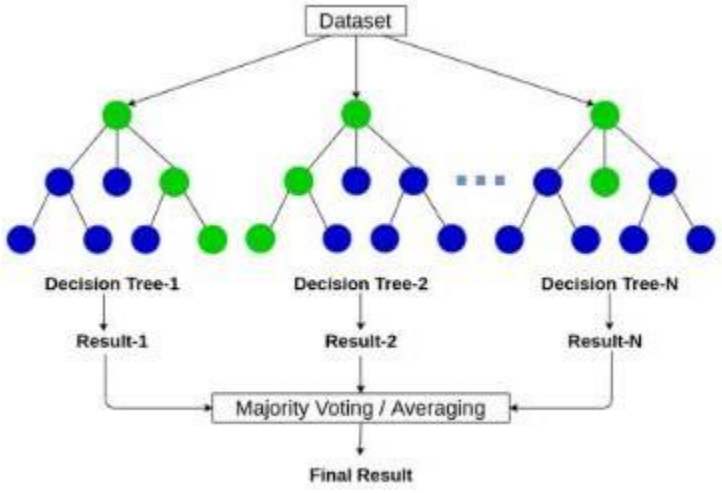
In this lab, students will learn and implement two powerful supervised machine learning algorithms: Random Forest Classifier and Support Vector Machine (SVM) Classifier. The lab focuses on understanding how ensemble learning improves classification accuracy using Random Forests and how SVM constructs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

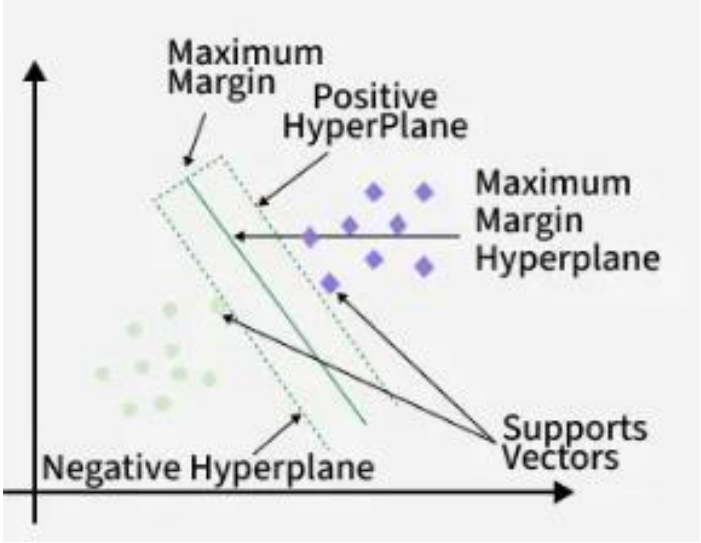
- High accuracy
- Reduces overfitting
- Works well with large datasets

Support Vector Machine (SVM) Classifier

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

Advantages:

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



Solved Examples

Example 1

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

Solution:

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

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

df = pd.DataFrame(data)

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

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

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

Example 2:

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

Solution:

```
from sklearn.svm import SVC

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

df = pd.DataFrame(data)

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

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

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

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

Example 3

Compare Random Forest and SVM classifiers on IRIS dataset.

Solution

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

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

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

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

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

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

Comparison Summary

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

LAB Assignment No 4

Topic: Random Forest and Support Vector Machine Classifier

Question 1

Classify flower species using Random Forest.

Task:

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

```
# Question 1: Random Forest on Iris Dataset

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

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

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

y_pred = rf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forest Accuracy on Iris Dataset: {accuracy:.4f}")
```

```
Random Forest Accuracy on Iris Dataset: 1.0000
```

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**.

```python
# Question 2: SVM on Breast Cancer Dataset

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)

svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print(f"SVM Accuracy on Breast Cancer Dataset: {accuracy:.4f}")
print("Confusion Matrix:")
print(cm)
```

```
SVM Accuracy on Breast Cancer Dataset: 0.9649
Confusion Matrix:
[[ 59   4]
 [  2 106]]
```

Question 3

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

Task:

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

```
# Question 4: SVM on Digits Dataset

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 = SVC(kernel='rbf', gamma='scale', random_state=42)
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)

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

misclassified_idx = np.where(y_test != y_pred)[0]

plt.figure(figsize=(10,4))
for i, idx in enumerate(misclassified_idx[:8]):
    plt.subplot(2, 4, i+1)
    plt.imshow(X_test[idx].reshape(8,8), cmap='gray')
    plt.title(f"True:{y_test[idx]} Pred:{y_pred[idx]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

Question 4

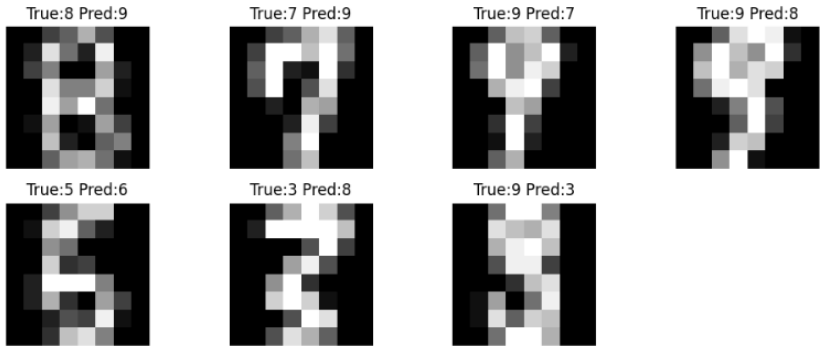
Use SVM on Digits Dataset and to identify the: Handwritten digit recognition.

Task:

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

```
# Question 4: SVM on Digits Dataset

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 = SVC(kernel='rbf', gamma='scale', random_state=42)
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)

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

misclassified_idx = np.where(y_test != y_pred)[0]

plt.figure(figsize=(10,4))
for i, idx in enumerate(misclassified_idx[:8]):
    plt.subplot(2, 4, i+1)
    plt.imshow(X_test[idx].reshape(8,8), cmap='gray')
    plt.title(f"True:{y_test[idx]} Pred:{y_pred[idx]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

```
SVM Accuracy on Digits Dataset: 0.9870
```



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.

```
# Question 5: Compare Random Forest vs SVM on Wine Dataset

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)

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)

svm = SVC(kernel='rbf', gamma='scale', random_state=42)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)

print(f"Random Forest Accuracy on Wine Dataset: {accuracy_rf:.4f}")
print(f"SVM Accuracy on Wine Dataset: {accuracy_svm:.4f}")

if accuracy_rf > accuracy_svm:
    print("Random Forest performs better on Wine Dataset.")
elif accuracy_rf < accuracy_svm:
    print("SVM performs better on Wine Dataset.")
else:
    print("Both models perform equally well on Wine Dataset.")
```

```
Random Forest Accuracy on Wine Dataset: 1.0000
SVM Accuracy on Wine Dataset: 0.7593
Random Forest performs better on Wine Dataset.
```

Lab Assesment

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

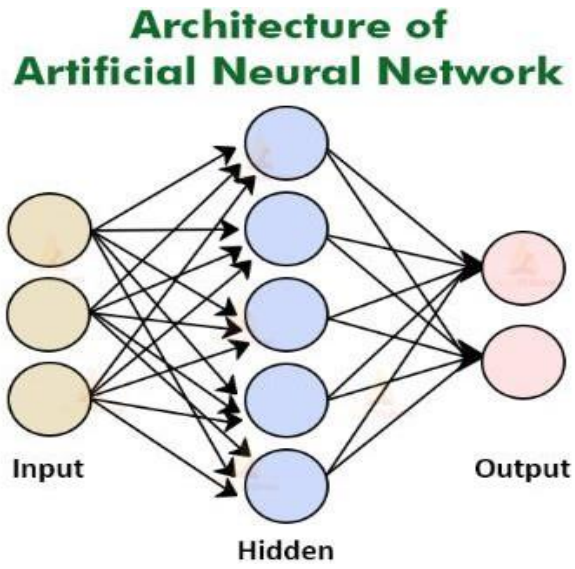
In this lab, students will learn the fundamentals and implementation of an Artificial Neural Network (ANN) for classification and prediction tasks. The lab introduces how neural networks are inspired by the human brain and how they learn patterns from data using layers of interconnected neurons. Students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

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

```python
# Question 1: AND Gate with Neural Network

import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD

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

model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))  # Hidden layer
model.add(Dense(1, activation='sigmoid'))            # Output layer

model.compile(optimizer=SGD(learning_rate=0.1), loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X, y, epochs=500, verbose=0)

loss, accuracy = model.evaluate(X, y, verbose=0)
print(f"Accuracy on AND gate: {accuracy:.4f}")

# Predictions
predictions = model.predict(X)
predictions = (predictions > 0.5).astype(int)
print("Predictions vs Actual:")
for i in range(len(X)):
    print(f"Input: {X[i]}, Predicted: {predictions[i][0]}, Actual: {y[i][0]}")
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Accuracy on AND gate: 1.0000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
Predictions vs Actual:
Input: [0 0], Predicted: 0, Actual: 0
Input: [0 1], Predicted: 0, Actual: 0
Input: [1 0], Predicted: 0, Actual: 0
Input: [1 1], Predicted: 1, Actual: 1
```

Question 2

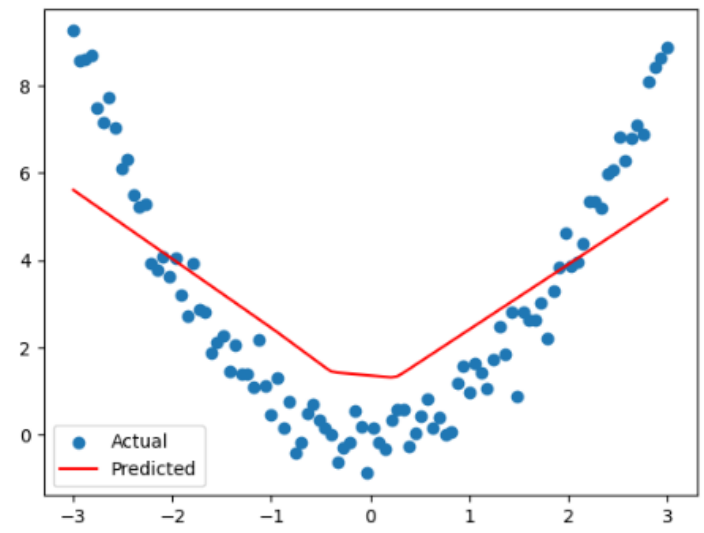
Create a dataset $y = x^2 + \text{noise}$ for x in range [-3,3]. Regression Task with Neural Network

Tasks:

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

```
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)

X = X.reshape(-1,1)
model = Sequential()
model.add(Dense(10, input_dim=1, activation='relu'))
model.add(Dense(1, activation='linear'))

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

model.fit(X, y, epochs=200, verbose=0)

y_pred = model.predict(X)
plt.scatter(X, y, label='Actual')
plt.plot(X, y_pred, color='red', label='Predicted')
plt.legend()
plt.show()
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```



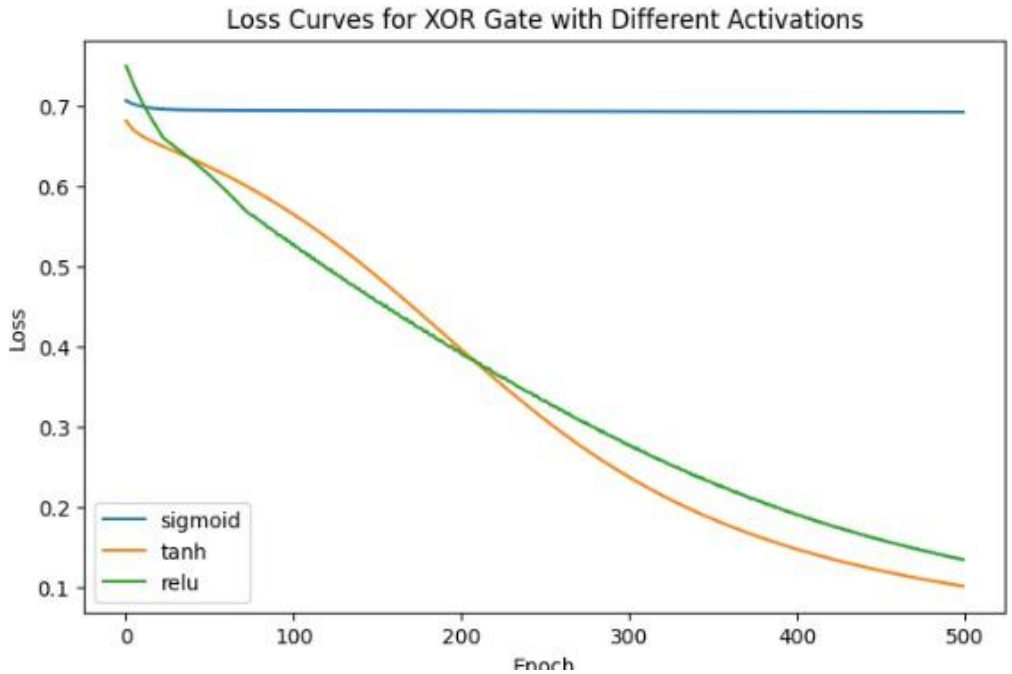
Question 3:

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

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

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

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

activations = ['sigmoid','tanh','relu']
histories = {}

for act in activations:
    model = Sequential()
    model.add(Dense(4, input_dim=2, activation=act))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=SGD(learning_rate=0.1), loss='binary_crossentropy', metrics=['accuracy'])
    history = model.fit(X, y, epochs=500, verbose=0)
    histories[act] = history
    # Evaluate
    loss, acc = model.evaluate(X, y, verbose=0)
    print(f"Activation: {act}, Accuracy: {acc:.4f}")

plt.figure(figsize=(8,5))
for act in activations:
    plt.plot(histories[act].history['loss'], label=f'{act}')
plt.title('Loss Curves for XOR Gate with Different Activations')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Activation: sigmoid, Accuracy: 0.7500
Activation: tanh, Accuracy: 1.0000
Activation: relu, Accuracy: 1.0000
```



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.**

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

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

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

# Build model
model = Sequential()
model.add(Dense(16, input_dim=X.shape[1], activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# Train
model.fit(X_train, y_train, epochs=50, verbose=0)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Breast Cancer Classification Accuracy: {accuracy:.4f}")
```

```
Breast Cancer Classification Accuracy: 0.9532
```

Question 5

Multi-Class Classification on Iris Dataset

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

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

# Load Iris
data = load_iris()
X, y = data.data, data.target
y_cat = to_categorical(y)  # One-hot encoding

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

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

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

# Train
model.fit(X_train, y_train, epochs=50, verbose=0)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Iris Multi-Class Classification Accuracy: {accuracy:.4f}")
```

```
Iris Multi-Class Classification Accuracy: 0.7556
```

Question 6

Regression Problem (House Price Prediction)

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

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import StandardScaler

# Load dataset
data = fetch_california_housing()
X, y = data.data, data.target

# Scale features
scaler = StandardScaler()
X = scaler.fit_transform(X)

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

# Build model
model = Sequential()
model.add(Dense(32, input_dim=X.shape[1], activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='linear'))

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

# Train
model.fit(X_train, y_train, epochs=50, verbose=0)

# Evaluate
loss, mae = model.evaluate(X_test, y_test, verbose=0)
print(f"House Price Prediction MAE: {mae:.4f}")
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
House Price Prediction MAE: 0.3579
```

Question 7

Neural Network with Dropout Regularization

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

```
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

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

# Preprocess
X_train = X_train.reshape(-1, 28*28)/255.0
X_test = X_test.reshape(-1, 28*28)/255.0
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build model with Dropout
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

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

# Train
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1, validation_split=0.1)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"MNIST Test Accuracy with Dropout: {accuracy:.4f}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Epoch 1/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 9s 17ms/step - accuracy: 0.7758 - loss: 0.7008 - val_accuracy: 0.9648 - val_loss: 0.1162
Epoch 2/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 11s 18ms/step - accuracy: 0.9423 - loss: 0.1913 - val_accuracy: 0.9752 - val_loss: 0.0854
Epoch 3/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 8s 20ms/step - accuracy: 0.9546 - loss: 0.1460 - val_accuracy: 0.9757 - val_loss: 0.0809
Epoch 4/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 17ms/step - accuracy: 0.9610 - loss: 0.1259 - val_accuracy: 0.9772 - val_loss: 0.0730
Epoch 5/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 11s 19ms/step - accuracy: 0.9677 - loss: 0.1041 - val_accuracy: 0.9803 - val_loss: 0.0662
Epoch 6/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 8s 19ms/step - accuracy: 0.9716 - loss: 0.0907 - val_accuracy: 0.9807 - val_loss: 0.0688
Epoch 7/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 10s 18ms/step - accuracy: 0.9739 - loss: 0.0881 - val_accuracy: 0.9837 - val_loss: 0.0591
Epoch 8/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 9s 15ms/step - accuracy: 0.9751 - loss: 0.0799 - val_accuracy: 0.9818 - val_loss: 0.0630
Epoch 9/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 8s 18ms/step - accuracy: 0.9767 - loss: 0.0770 - val_accuracy: 0.9843 - val_loss: 0.0588
Epoch 10/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 0.9787 - loss: 0.0684 - val_accuracy: 0.9818 - val_loss: 0.0671
MNIST Test Accuracy with Dropout: 0.9809
```

Lab Assesment

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

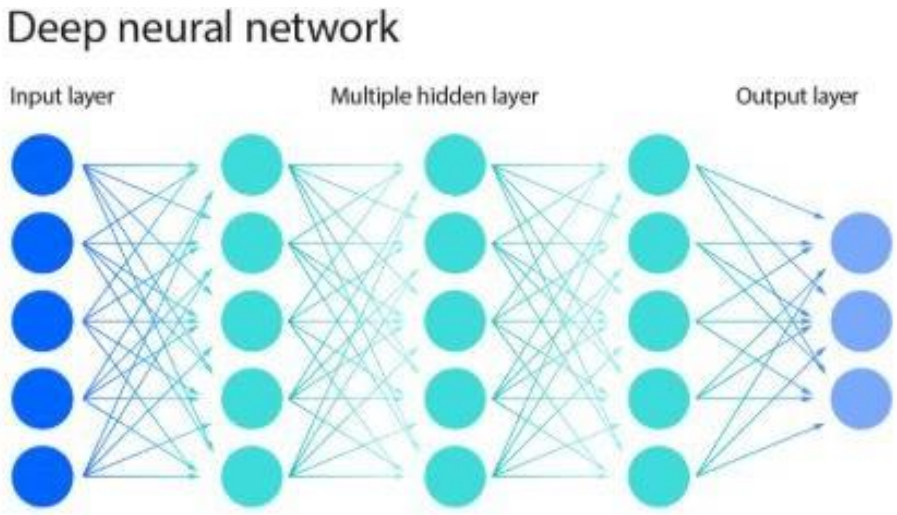
LAB No. 6

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-G).

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. 5

Topic: Artificial Neural Network Model

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

```
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'))
model.add(Dense(1, activation='sigmoid'))

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

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

loss, accuracy = model.evaluate(X, y)
print(f"Accuracy: {accuracy*100:.2f}%")

predictions = (model.predict(X) > 0.5).astype(int)
print("Predictions:", predictions.flatten())
print("Actual:", y)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step - accuracy: 0.5000 - loss: 0.6478
Accuracy: 50.00%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
Predictions: [0 1 0 0]
Actual: [0 0 0 1]
```

Question 2

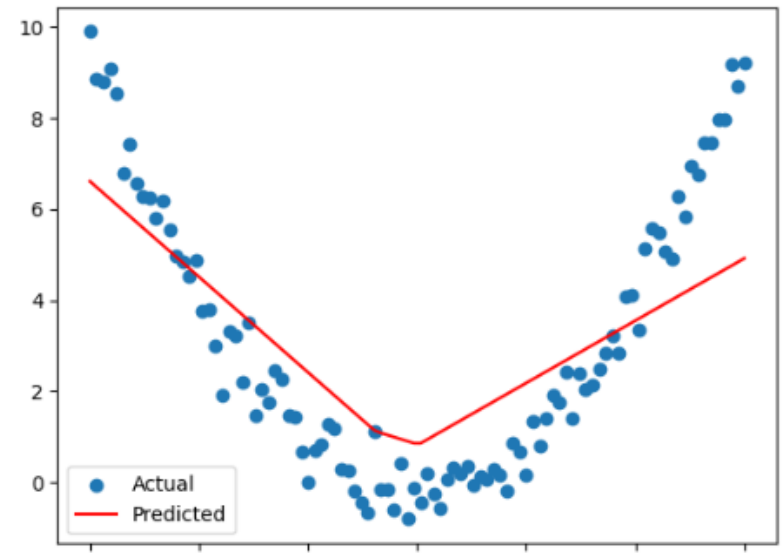
Create a dataset $y = x^2 + \text{noise}$ for x in range [-3,3]. Regression Task with Neural Network

Tasks:

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

```
import matplotlib.pyplot as plt

np.random.seed(0)
X = np.linspace(-3, 3, 100).reshape(-1,1)
y = X**2 + np.random.normal(0, 0.5, X.shape)

model = Sequential()
model.add(Dense(8, input_dim=1, activation='relu'))  # hidden layer
model.add(Dense(1))  # output layer for regression

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

model.fit(X, y, epochs=200, verbose=0)


y_pred = model.predict(X)

plt.scatter(X, y, label='Actual')
plt.plot(X, y_pred, color='red', label='Predicted')
plt.legend()
plt.show()
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
```



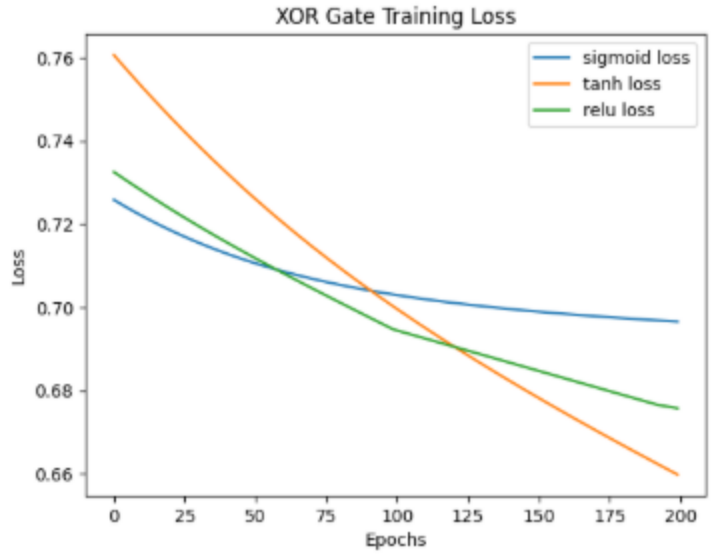
Question 3:

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

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

```
activations = ['sigmoid', 'tanh', 'relu']
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0,1,1,0])

for act in activations:
    model = Sequential()
    model.add(Dense(4, input_dim=2, activation=act))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(X, y, epochs=200, verbose=0)

    loss, acc = model.evaluate(X, y, verbose=0)
    print(f"Activation: {act}, Accuracy: {acc*100:.2f}%")

    plt.plot(history.history['loss'], label=f"{act} loss")

plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("XOR Gate Training Loss")
plt.legend()
plt.show()
```

```
Activation: sigmoid, Accuracy: 50.00%
Activation: tanh, Accuracy: 75.00%
Activation: relu, Accuracy: 75.00%
```



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.**

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


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

model = Sequential()
model.add(Dense(16, input_dim=X.shape[1], activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=50, verbose=0)

loss, acc = model.evaluate(X_test, y_test)
print(f"Breast Cancer Classification Accuracy: {acc*100:.2f}%")
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.9668 - loss: 0.0706
Breast Cancer Classification Accuracy: 95.61%
```

Question 5

Multi-Class Classification on Iris Dataset

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

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

iris = load_iris()
X = iris.data
y = to_categorical(iris.target)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

model = Sequential()
model.add(Dense(8, input_dim=4, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100, verbose=0)

loss, acc = model.evaluate(X_test, y_test)
print(f"Iris Classification Accuracy: {acc*100:.2f}%")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 194ms/step - accuracy: 0.6000 - loss: 0.4772
Iris Classification Accuracy: 60.00%
```

Question 6

Regression Problem (House Price Prediction)

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

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

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(-1,28*28)/255.0
X_test = X_test.reshape(-1,28*28)/255.0
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

model = Sequential()
model.add(Dense(512, activation='relu', input_dim=28*28))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1, validation_split=0.1)

loss, acc = model.evaluate(X_test, y_test)
print(f"MNIST Test Accuracy: {acc*100:.2f}%")
```

Question 7

Neural Network with Dropout Regularization

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

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

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(-1,28*28)/255.0
X_test = X_test.reshape(-1,28*28)/255.0
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

model = Sequential()
model.add(Dense(512, activation='relu', input_dim=28*28))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1, validation_split=0.1)

loss, acc = model.evaluate(X_test, y_test)
print(f"MNIST Test Accuracy: {acc*100:.2f}%")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 8s 15ms/step - accuracy: 0.8482 - loss: 0.5089 - val_accuracy: 0.9723 - val_loss: 0.0941
Epoch 2/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 19s 44ms/step - accuracy: 0.9644 - loss: 0.1164 - val_accuracy: 0.9772 - val_loss: 0.0774
Epoch 3/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 8s 19ms/step - accuracy: 0.9739 - loss: 0.0804 - val_accuracy: 0.9818 - val_loss: 0.0654
Epoch 4/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 17ms/step - accuracy: 0.9801 - loss: 0.0630 - val_accuracy: 0.9775 - val_loss: 0.0695
Epoch 5/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 6s 14ms/step - accuracy: 0.9863 - loss: 0.0451 - val_accuracy: 0.9783 - val_loss: 0.0702
Epoch 6/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 17ms/step - accuracy: 0.9880 - loss: 0.0362 - val_accuracy: 0.9832 - val_loss: 0.0613
Epoch 7/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9882 - loss: 0.0349 - val_accuracy: 0.9820 - val_loss: 0.0709
Epoch 8/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 0.9897 - loss: 0.0302 - val_accuracy: 0.9827 - val_loss: 0.0657
Epoch 9/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 0.9916 - loss: 0.0246 - val_accuracy: 0.9810 - val_loss: 0.0696
Epoch 10/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 6s 15ms/step - accuracy: 0.9913 - loss: 0.0253 - val_accuracy: 0.9805 - val_loss: 0.0740
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9785 - loss: 0.0768
MNIST Test Accuracy: 98.19%
```

Lab Assesment

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

Laboratory Work Assessment Rubrics

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

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

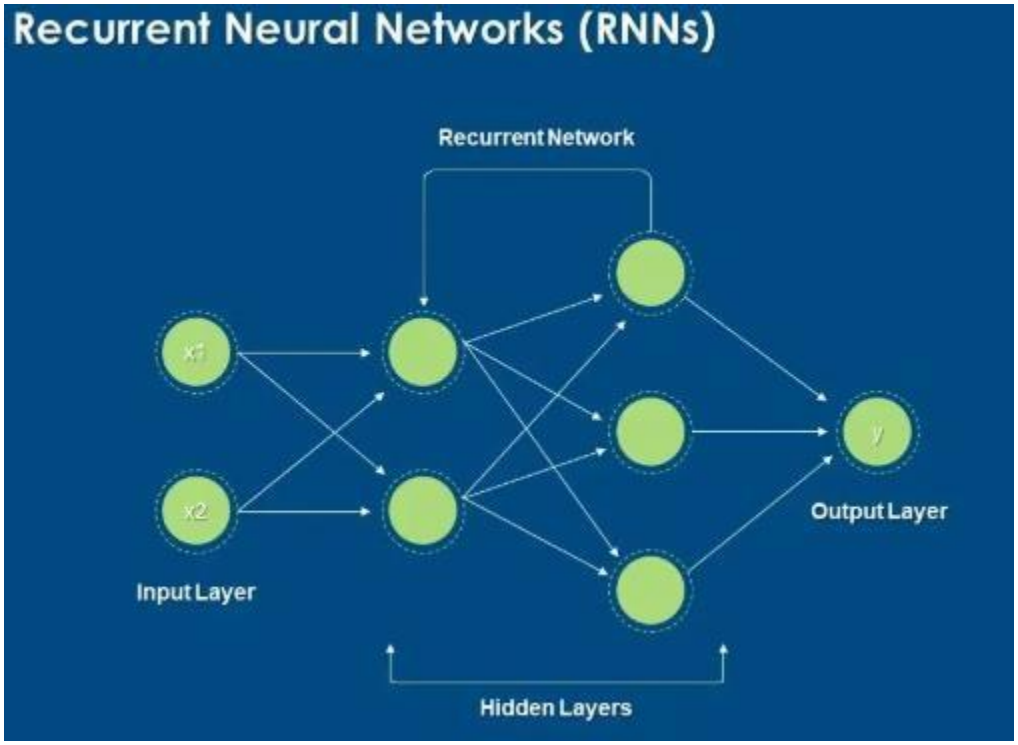
LAB No. 7

Implementation of Recurrent Neural Network (RNN)

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

Introduction

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



Key Concepts:

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

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

Solved Examples:

Example 1:

Simple RNN for Binary Sequence Classification

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

Solution:

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

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

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

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

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

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

Explanation

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

Example 2:

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

Solution:

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

y = np.array([4, 5, 6, 7])
# Build RNN model
```

```
model = Sequential()
model.add(SimpleRNN(10, activation='tanh', input_shape=(3, 1)))
model.add(Dense(1))

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

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

Explanation

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

Example 3:

RNN for Text Classification (Simple Sentiment Analysis)

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

Solution

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

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

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

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

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

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

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

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

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

Limitations of Basic RNN

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

LAB Assignment No 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.

Output:

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

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

text = "The sun is shining. The sun is bright. The moon is glowing."

tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])
total_words = len(tokenizer.word_index) + 1

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

max_seq_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_seq_len, padding='pre'))

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

from tensorflow.keras.utils import to_categorical
y = to_categorical(y, num_classes=total_words)
```

```
model = Sequential()
model.add(Embedding(total_words, 10, input_length=max_seq_len-1))
model.add(SimpleRNN(50))
model.add(Dense(total_words, activation='softmax'))

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

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
simple_rnn (SimpleRNN)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)

```
model.fit(X, y, epochs=200, verbose=0)
```

```
<keras.src.callbacks.history.History at 0x78a3d67345c0>
```

```
def predict_next_word(model, tokenizer, text_seq, max_seq_len):
    token_list = tokenizer.texts_to_sequences([text_seq])[0]
    token_list = pad_sequences([token_list], maxlen=max_seq_len-1, padding='pre')
    predicted = model.predict(token_list, verbose=0)
    predicted_word_index = np.argmax(predicted, axis=1)[0]
    return tokenizer.index_word[predicted_word_index]

print(predict_next_word(model, tokenizer, "The sun is", max_seq_len))
```

```
bright
```

LAB Task 2:

Stock Price Prediction using RNN

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

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

Output:

Line graph showing predicted vs real stock price trend.

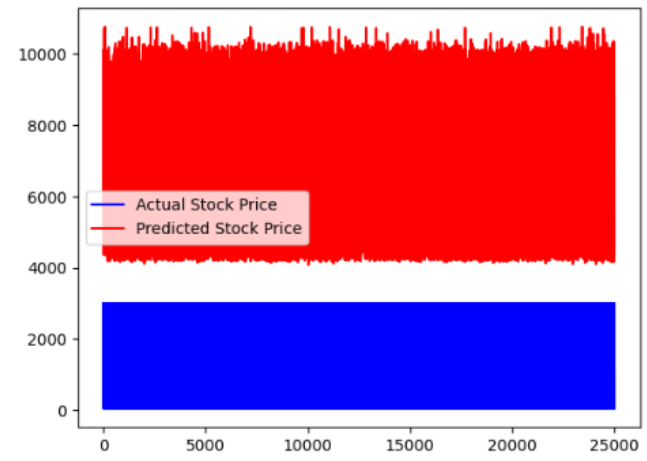
```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

df = pd.read_csv('GOOGL.csv')  # Ensure CSV has 'Date' & 'Close' columns
data = df['Close'].values.reshape(-1,1)

scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

```python
X_train, y_train = [], []
seq_len = 60  # previous 60 days

for i in range(seq_len, len(data_scaled)):
    X_train.append(data_scaled[i-seq_len:i, 0])
    y_train.append(data_scaled[i, 0])

X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

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

model = Sequential()
model.add(SimpleRNN(50, return_sequences=False, input_shape=(X_train.shape[1],1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=50, batch_size=32)
```

```python
predicted_stock = model.predict(X_train)
predicted_stock = scaler.inverse_transform(predicted_stock)
actual_stock = scaler.inverse_transform(y_train.reshape(-1,1))

plt.plot(actual_stock, color='blue', label='Actual Stock Price')
plt.plot(predicted_stock, color='red', label='Predicted Stock Price')
plt.legend()
plt.show()
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step
```



LAB Task 3:

Sentiment Analysis using RNN

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

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

Output:

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

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

vocab_size = 5000
max_len = 100

(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=vocab_size)
X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

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

model = Sequential()
model.add(Embedding(vocab_size, 32, input_length=max_len))
model.add(SimpleRNN(50))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=3, batch_size=64, validation_split=0.2)
```

```
Epoch 1/3
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
313/313 ━━━━━━━━━━━━━━━━━━━━ 13s 35ms/step - accuracy: 0.5588 - loss: 0.6683 - val_accuracy: 0.7922 - val_loss: 0.4530
Epoch 2/3
313/313 ━━━━━━━━━━━━━━━━━━━━ 12s 37ms/step - accuracy: 0.8284 - loss: 0.3890 - val_accuracy: 0.8220 - val_loss: 0.4141
Epoch 3/3
313/313 ━━━━━━━━━━━━━━━━━━━━ 9s 29ms/step - accuracy: 0.8862 - loss: 0.2864 - val_accuracy: 0.7970 - val_loss: 0.4616
<keras.src.callbacks.history.History at 0x78a3d6724470>
```

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

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - accuracy: 0.7928 - loss: 0.4583
Test Accuracy: 0.797760009765625
```

LAB Task 4:

Weather Forecasting using RNN

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

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

Output:

Graph showing predicted vs actual temperature trends.

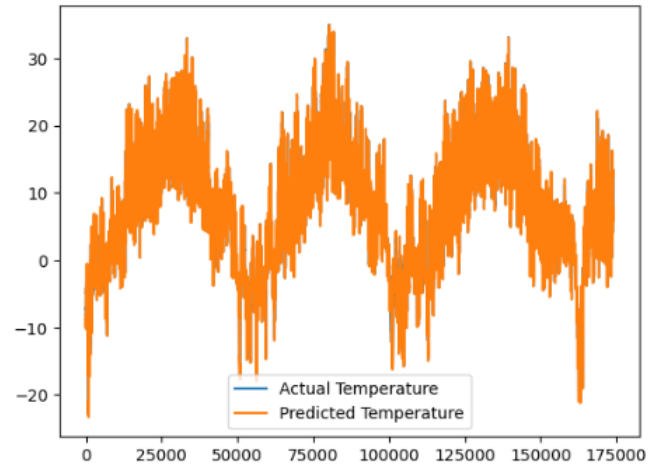
```
df = pd.read_csv('jena_climate_2009_2016.csv')  # example file
temp = df['T (degC)'].values.reshape(-1,1)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
temp_scaled = scaler.fit_transform(temp)
```

```
X, y = [], []
seq_len = 10

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

X, y = np.array(X), np.array(y)
X = X.reshape(X.shape[0], X.shape[1], 1)
```

```
model = Sequential()
model.add(SimpleRNN(50, input_shape=(X.shape[1],1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X, y, epochs=20, batch_size=32)
```

```
pred = model.predict(X)
pred = scaler.inverse_transform(pred)
y_true = scaler.inverse_transform(y.reshape(-1,1))

plt.plot(y_true, label='Actual Temperature')
plt.plot(pred, label='Predicted Temperature')
plt.legend()
plt.show()
```

```
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 10s 2ms/step
```



LAB Task 5:

Music Note Generation using RNN

Objective: Generate new music sequences using RNN.
Dataset: *MIDI music dataset* (short sequences or melodies).
Tasks:

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

Output:

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

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
temp_scaled = scaler.fit_transform(temp)
```

```
X, y = [], []
seq_len = 10

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

X, y = np.array(X), np.array(y)
X = X.reshape(X.shape[0], X.shape[1], 1)
```

```
model = Sequential()
model.add(SimpleRNN(50, input_shape=(X.shape[1],1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X, y, epochs=20, batch_size=32)
```

```
Epoch 1/20
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:1
  super().__init__(**kwargs)
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 26s 4ms/step - loss: 5.3679e-04
Epoch 2/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 2.4912e-05
Epoch 3/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.6479e-05
Epoch 4/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.5800e-05
Epoch 5/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 22s 4ms/step - loss: 1.4955e-05
Epoch 6/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.4889e-05
Epoch 7/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.5090e-05
Epoch 8/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 22s 4ms/step - loss: 1.4941e-05
Epoch 9/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.4847e-05
Epoch 10/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 41s 4ms/step - loss: 1.4656e-05
Epoch 11/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.4513e-05
Epoch 12/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.4415e-05
Epoch 13/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.4547e-05
Epoch 14/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.4277e-05
Epoch 15/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.4700e-05
Epoch 16/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.4559e-05
Epoch 17/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.4510e-05
Epoch 18/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 21s 4ms/step - loss: 1.4404e-05
Epoch 19/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 20s 4ms/step - loss: 1.4222e-05
Epoch 20/20
5437/5437 ━━━━━━━━━━━━━━━━━━━━ 41s 4ms/step - loss: 1.4586e-05
<keras.src.callbacks.history.History at 0x78a3a9f42c60>
```

```
pred = model.predict(X)
pred = scaler.inverse_transform(pred)
y_true = scaler.inverse_transform(y.reshape(-1,1))
```

Lab Assesment

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

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

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

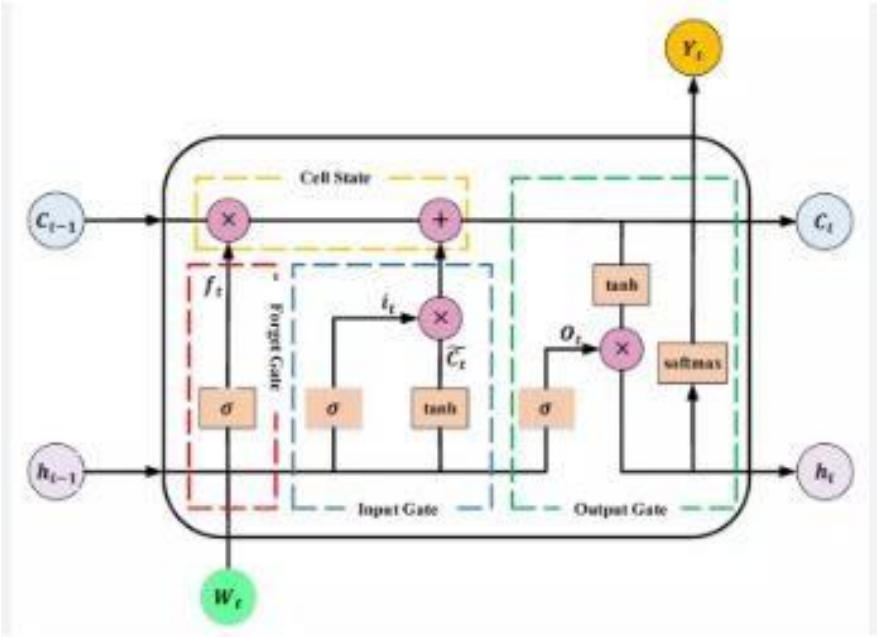
LAB No. 8

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 s 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 No 8

Assignment LSTM 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
```

```
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?

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
"""

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

input_sequences = []
```

```
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(seq) for seq in input_sequences])

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

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

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

vocab_size = len(tokenizer.word_index) + 1

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=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.fit(X, y, epochs=100, verbose=1)

def predict_next_word(model, tokenizer, text, max_len):
    token_list = tokenizer.texts_to_sequences([text])[0]
    token_list = pad_sequences([token_list], maxlen=max_len-1, padding='pre')
    predicted_index = np.argmax(model.predict(token_list, verbose=0))

    for word, index in tokenizer.word_index.items():
        if index == predicted_index:
            return word
    return ""

input_text = "what is the"
next_word = predict_next_word(model, tokenizer, input_text, max_len)
print(f"Input: {input_text}")
```

```
text = input_text
num_words_to_generate = 5

for _ in range(num_words_to_generate):
    next_word = predict_next_word(model, tokenizer, text, max_len)
    text += " " + next_word

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

```
Input: what is the
Predicted next word: total
Generated text: what is the total of of the course
```

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.

```
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

text_data = """
Python is a powerful programming language.
It is widely used for Data Science, Machine Learning, and Web Development.
LSTM networks are useful for sequence prediction tasks.
Deep Learning models learn patterns from large datasets.
"""

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

input_sequences = []
for sentence in text_data.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(seq) for seq in input_sequences])
padded_sequences = pad_sequences(input_sequences, maxlen=max_len, padding='pre')

X = padded_sequences[:, :-1]
y = padded_sequences[:, -1]
y = tf.keras.utils.to_categorical(y, num_classes=len(tokenizer.word_index)+1)

vocab_size = len(tokenizer.word_index)+1

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=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.fit(X, y, epochs=30, verbose=1)
```


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

model.fit(X, y, epochs=30, verbose=1)
```

Question No. 3

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

```
def predict_next_word(model, tokenizer, text, max_len):
    token_list = tokenizer.texts_to_sequences([text])[0]
    token_list = pad_sequences([token_list], maxlen=max_len-1, padding='pre')
    predicted_index = np.argmax(model.predict(token_list, verbose=0))

    for word, index in tokenizer.word_index.items():
        if index == predicted_index:
            return word
    return ""

text = "Python is"
num_words_to_generate = 5

for _ in range(num_words_to_generate):
    next_word = predict_next_word(model, tokenizer, text, max_len)
    text += " " + next_word

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

```
Generated text: Python is is is is for prediction
```

Lab Assesment

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

Laboratory Work Assessment Rubrics

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

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

LAB Assignment No. G

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.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset_dir = "/content/drive/MyDrive/pet"
```

```
all_images = os.listdir(dataset_dir)

cat_images = [os.path.join(dataset_dir, img) for img in all_images if 'cat' in img.lower()]
dog_images = [os.path.join(dataset_dir, img) for img in all_images if 'dog' in img.lower()]

image_paths = cat_images + dog_images
labels = [0]*len(cat_images) + [1]*len(dog_images)  # 0 = Cat, 1 = Dog

print("Total images:", len(image_paths))
print("Cats:", len(cat_images), "Dogs:", len(dog_images))
```

```
Total images: 5
Cats: 3 Dogs: 2
```

```
X = []
for img_path in image_paths:
    img = load_img(img_path, target_size=(150,150))  # resize
    img_array = img_to_array(img)/255.0  # normalize
    X.append(img_array)

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

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

```
Images shape: (5, 150, 150, 3)
Labels shape: (5,)
```

```
from sklearn.model_selection import train_test_split

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

print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

```
Training samples: 4
Testing samples: 1
```

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(512, activation='relu'),
```

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=15,
    batch_size=16
)
```

```
Epoch 1/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - accuracy: 0.2500 - loss: 0.7207 - val_accuracy: 1.0000 - val_loss: 0.5932
Epoch 2/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 770ms/step - accuracy: 0.7500 - loss: 0.5794 - val_accuracy: 1.0000 - val_loss: 0.1071
Epoch 3/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.5000 - loss: 1.3550 - val_accuracy: 0.0000e+00 - val_loss: 5.1380
Epoch 4/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.7500 - loss: 1.5889 - val_accuracy: 0.0000e+00 - val_loss: 2.9704
Epoch 5/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 979ms/step - accuracy: 0.7500 - loss: 0.5635 - val_accuracy: 1.0000 - val_loss: 0.2711
Epoch 6/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 1.0000 - loss: 0.1815 - val_accuracy: 1.0000 - val_loss: 0.1885
Epoch 7/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 925ms/step - accuracy: 1.0000 - loss: 0.2401 - val_accuracy: 1.0000 - val_loss: 0.5851
Epoch 8/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 649ms/step - accuracy: 1.0000 - loss: 0.0775 - val_accuracy: 0.0000e+00 - val_loss: 1.6851
Epoch 9/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 657ms/step - accuracy: 1.0000 - loss: 0.0185 - val_accuracy: 0.0000e+00 - val_loss: 3.0460
Epoch 10/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 670ms/step - accuracy: 1.0000 - loss: 0.1444 - val_accuracy: 0.0000e+00 - val_loss: 1.4353
Epoch 11/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 652ms/step - accuracy: 1.0000 - loss: 0.0060 - val_accuracy: 1.0000 - val_loss: 0.3971
Epoch 12/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 679ms/step - accuracy: 1.0000 - loss: 0.0015 - val_accuracy: 1.0000 - val_loss: 0.0722
Epoch 13/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 1.0000 - loss: 0.0521 - val_accuracy: 1.0000 - val_loss: 0.1490
Epoch 14/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 1.0000 - loss: 0.0015 - val_accuracy: 1.0000 - val_loss: 0.2761
Epoch 15/15
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 668ms/step - accuracy: 1.0000 - loss: 0.0106 - val_accuracy: 1.0000 - val_loss: 0.6427
```

```
plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='train_acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

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

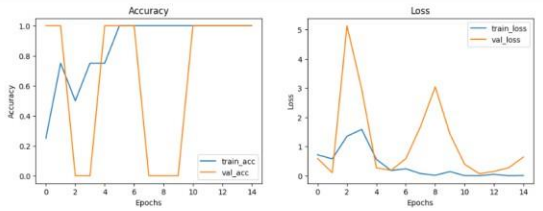
```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten (Flatten)	(None, 36992)	0
dense (Dense)	(None, 512)	18,940,416
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

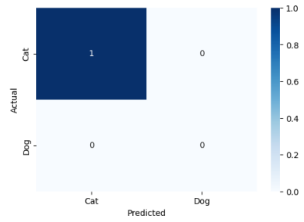
Total params: 19,034,177 (72.61 MB)
Trainable params: 19,034,177 (72.61 MB)
Non-trainable params: 0 (0.00 B)

```
plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='train_acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.title('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='val_loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



```
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

y_pred = (model.predict(X_test) > 0.5).astype(int).flatten()

cm = confusion_matrix(y_test, y_pred, labels=[0,1])

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Cat','Dog'], yticklabels=['Cat','Dog'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print(classification_report(y_test, y_pred, labels=[0,1], target_names=['Cat','Dog'], zero_division=0))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step
```




```
              precision    recall  f1-score   support

         Cat       1.00      1.00      1.00         1
         Dog       0.00      0.00      0.00         0

    accuracy                           1.00         1
   macro avg       0.50      0.50      0.50         1
```

```
test_image_path = "/content/drive/MyDrive/pet/cat1.jpeg"  # update with any image

img = load_img(test_image_path, target_size=(150,150))
img_array = img_to_array(img)/255.0
img_array = np.expand_dims(img_array, axis=0)

prediction = model.predict(img_array)
if prediction < 0.5:
    print("This image is a CAT")
else:
    print("This image is a DOG")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
This image is a CAT
```

Lab Assesment

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

Laboratory Work Assessment Rubrics

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

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

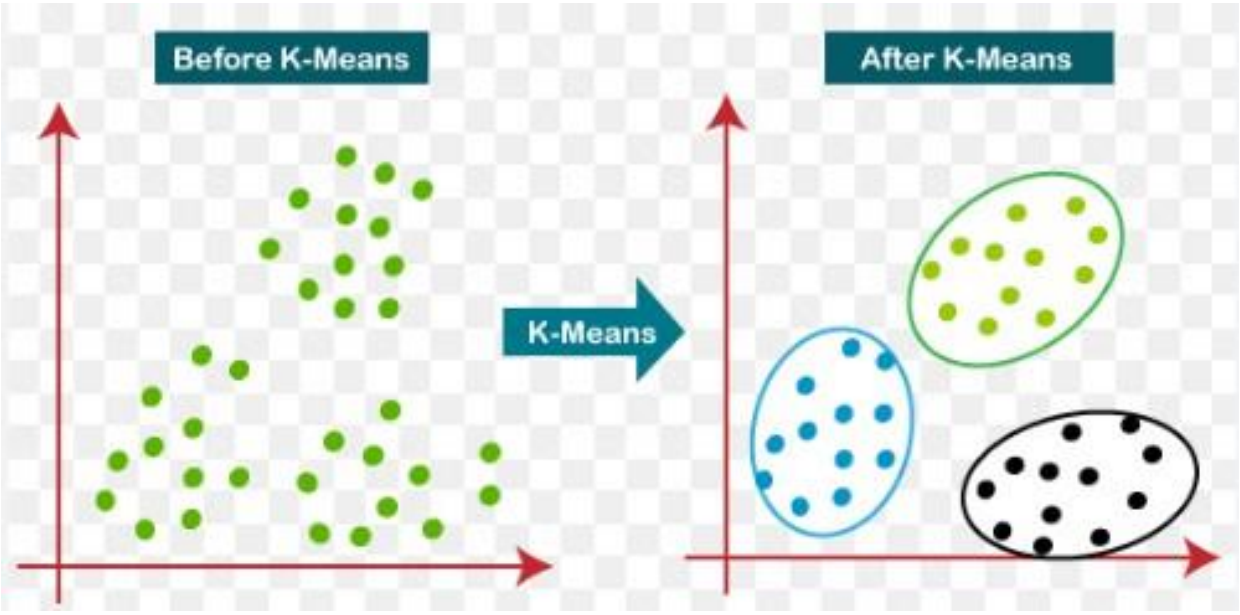
LAB Assignment No 10

K means Clustering Schemes in Machine Learning

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

Introduction

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



Working of K-Means:

1. Select the number of clusters **K**
2. Initialize K centroids randomly

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

Key Concepts:

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

Applications:

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

Solved Examples

Example 1: K-Means Clustering on a Simple Dataset

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

Solution:

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

# Sample data: Marks vs Attendance
X = np.array([
   [40, 60],
   [45, 65],
   [70, 80],
   [75, 85],
```

```
  [90, 95],
  [85, 90]
])
# Apply K-Means
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_

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

Explanation

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

Example 2: Choosing Optimal K Using Elbow Method

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

Solution:

```
inertia = []

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

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

Explanation

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

Example 3: K-Means Clustering on Iris Dataset

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

Solution:

```
from sklearn.datasets import load_iris

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

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

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

```
plt.show()
```

Explanation

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

LAB Assignment No 10

K means Clustering Schemes in Machine Learning

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)

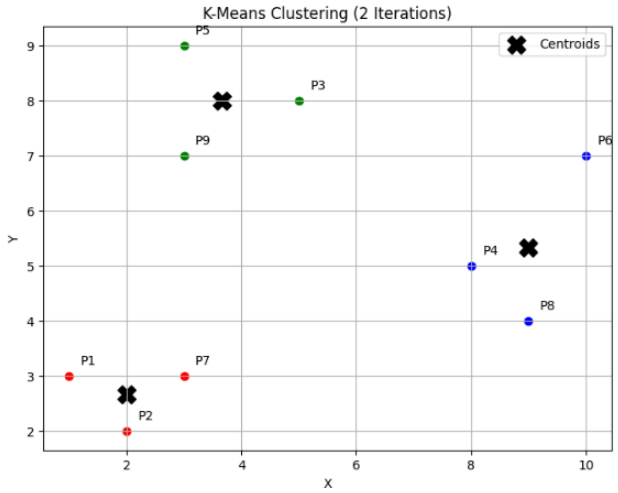
Tasks:

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

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

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

C1 = np.array([3,3])
C2 = np.array([3,7])
C3 = np.array([9,4])
centroids = np.array([C1,C2,C3])

def euclidean_dist(p, c):
    return np.sqrt(np.sum((p-c)**2))

def assign_clusters(points, centroids):
    clusters = []
    for p in points:
        distances = [euclidean_dist(p, c) for c in centroids]
        cluster = np.argmin(distances)
        clusters.append(cluster)
    return np.array(clusters)

def update_centroids(points, clusters, k):
    new_centroids = []
    for i in range(k):
        cluster_points = points[clusters == i]
        if len(cluster_points) > 0:
            new_centroids.append(cluster_points.mean(axis=0))
        else:
            new_centroids.append(centroids[i])
    return np.array(new_centroids)

for iteration in range(2):
    clusters = assign_clusters(points, centroids)
    centroids = update_centroids(points, clusters, 3)
    print(f"Iteration {iteration+1}:")
    print("Clusters:", clusters)
    print("Centroids:\n", centroids)

colors = ['r','g','b']
plt.figure(figsize=(8,6))
for i, point in enumerate(points):
    plt.scatter(point[0], point[1], color=colors[clusters[i]])
    plt.text(point[0]+0.2, point[1]+0.2, labels[i])
```

```python
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.grid(True)
plt.show()
```

```
Iteration 1:
Clusters: [0 0 1 2 1 2 0 2 1]
Centroids:
 [[2.         2.66666667]
 [3.66666667 8.        ]
 [9.         5.33333333]]
Iteration 2:
Clusters: [0 0 1 2 1 2 0 2 1]
Centroids:
 [[2.         2.66666667]
 [3.66666667 8.        ]
 [9.         5.33333333]]
```



Question No. 2

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

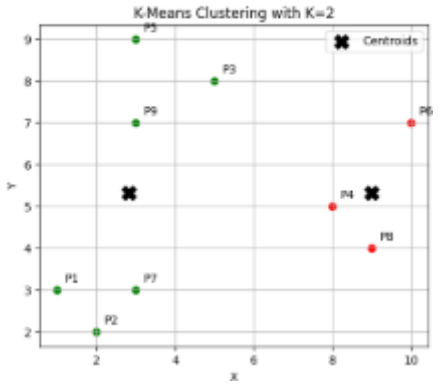
P1(1,3), P2(2,2), P3(5,8), P4(8,5), P5(3,9), P6(10,7), P7(3,3), P8(9,4), P9(3,7)

Tasks:

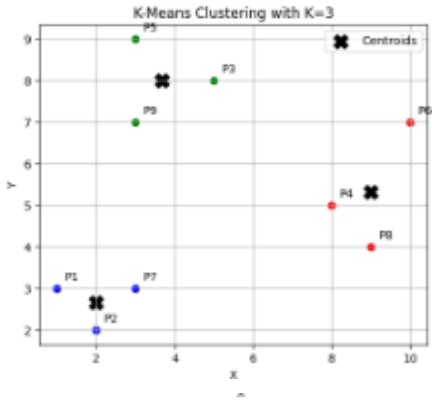
1. Use K = 2, 3, and 4
2. Plot the clustering result for each K
3. Compare:
    - Number of points in each cluster
    - Final centroid locations
4. Draw the 3 graphs in your lab copy and explain how the shapes change with K.

```python
#Q2
from sklearn.cluster import KMeans

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

for k in [2,3,4]:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    clusters = kmeans.fit_predict(points)
    centroids = kmeans.cluster_centers_
    print(f"\nK={k}")
    print("Cluster Assignments:", clusters)
    print("Centroids:\n", centroids)

    # Plot
    colors = ['r','g','b','y']
    plt.figure(figsize=(6,5))
    for i, point in enumerate(points):
        plt.scatter(point[0], point[1], color=colors[clusters[i]])
        plt.text(point[0]+0.2, point[1]+0.2, labels[i])
    plt.scatter(centroids[:,0], centroids[:,1], color='k', marker='X', s=150, label='Centroids')
    plt.title(f"K-Means Clustering with K={k}")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.legend()
    plt.grid(True)
    plt.show()
```

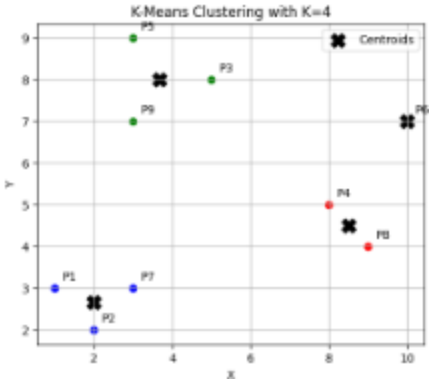
```
K=2
Cluster Assignments: [1 1 1 0 1 0 1 0 1]
Centroids:
 [[9.         5.33333333]
 [2.83333333 5.33333333]]
```



K-3
Cluster Assignments: [2 2 1 0 1 0 2 0 1]
Centroids:
[[9. 5.33333333]
[3.66666667 8.]
[2. 2.66666667]]



K-4
Cluster Assignments: [2 2 1 0 1 3 2 0 1]
Centroids:
[[8.5 4.5]
[3.66666667 8.]
[2. 2.66666667]
[10. 7.]]



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

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

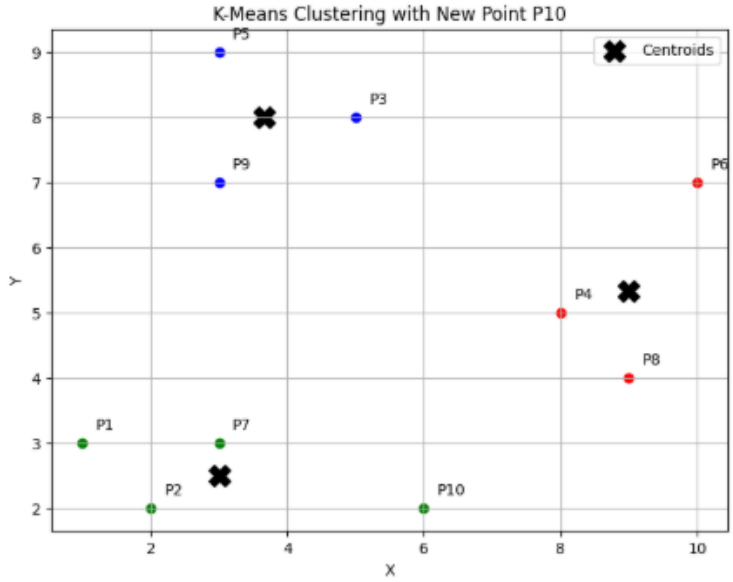
Tasks:

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.

```
points_new = np.vstack([points, [6,2]])
labels_new = labels + ['P10']

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters_new = kmeans.fit_predict(points_new)
centroids_new = kmeans.cluster_centers_

print("New Clusters:", clusters_new)
print("New Centroids:\n", centroids_new)

# Plot
colors = ['r','g','b']
plt.figure(figsize=(8,6))
for i, point in enumerate(points_new):
    plt.scatter(point[0], point[1], color=colors[clusters_new[i]])
    plt.text(point[0]+0.2, point[1]+0.2, labels_new[i])

plt.scatter(centroids_new[:,0], centroids_new[:,1], color='k', marker='X', s=200, label='Centroids')
plt.title("K-Means Clustering with New Point P10")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.grid(True)
plt.show()
```

```
New Clusters: [1 1 2 0 2 0 1 0 2 1]
New Centroids:
 [[9.         5.33333333]
 [3.         2.5       ]
 [3.66666667 8.        ]]
```



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:

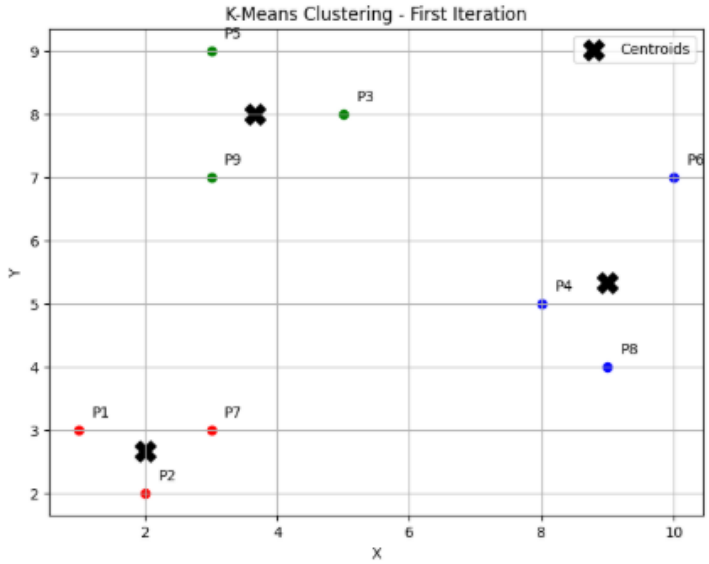
1. Compute **Euclidean distance** of each point to each centroid (manually or in python)
2. Create a distance table:

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.

```
#Q4
C1 = np.array([3,3])
C2 = np.array([3,7])
C3 = np.array([9,4])
centroids = np.array([C1,C2,C3])

distance_table = []
for i, point in enumerate(points):
    dist_C1 = euclidean_dist(point, C1)
    dist_C2 = euclidean_dist(point, C2)
    dist_C3 = euclidean_dist(point, C3)
    assigned = np.argmin([dist_C1, dist_C2, dist_C3])
    distance_table.append([labels[i], dist_C1, dist_C2, dist_C3, assigned])

import pandas as pd
df = pd.DataFrame(distance_table, columns=['Point','Dist to C1','Dist to C2','Dist to C3','Assigned Cluster'])
print(df)

clusters = df['Assigned Cluster'].values
new_centroids = update_centroids(points, clusters, 3)
print("\nNew Centroids after 1st iteration:\n", new_centroids)

colors = ['r','g','b']
plt.figure(figsize=(8,6))
for i, point in enumerate(points):
    plt.scatter(point[0], point[1], color=colors[clusters[i]])
    plt.text(point[0]+0.2, point[1]+0.2, labels[i])
plt.scatter(new_centroids[:,0], new_centroids[:,1], color='k', marker='X', s=200, label='Centroids')
plt.title("K-Means Clustering - First Iteration")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.grid(True)
plt.show()
```

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

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



Lab Assesment

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

Laboratory Work Assessment Rubrics

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

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

LAB No 11

Agglomerative Hierarchical Clustering

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

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

Objectives:

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

Theory

1. Introduction to Hierarchical Clustering

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

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

2. Agglomerative Hierarchical Clustering

- Start with each data point as a separate cluster.

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

Distance Metrics:

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

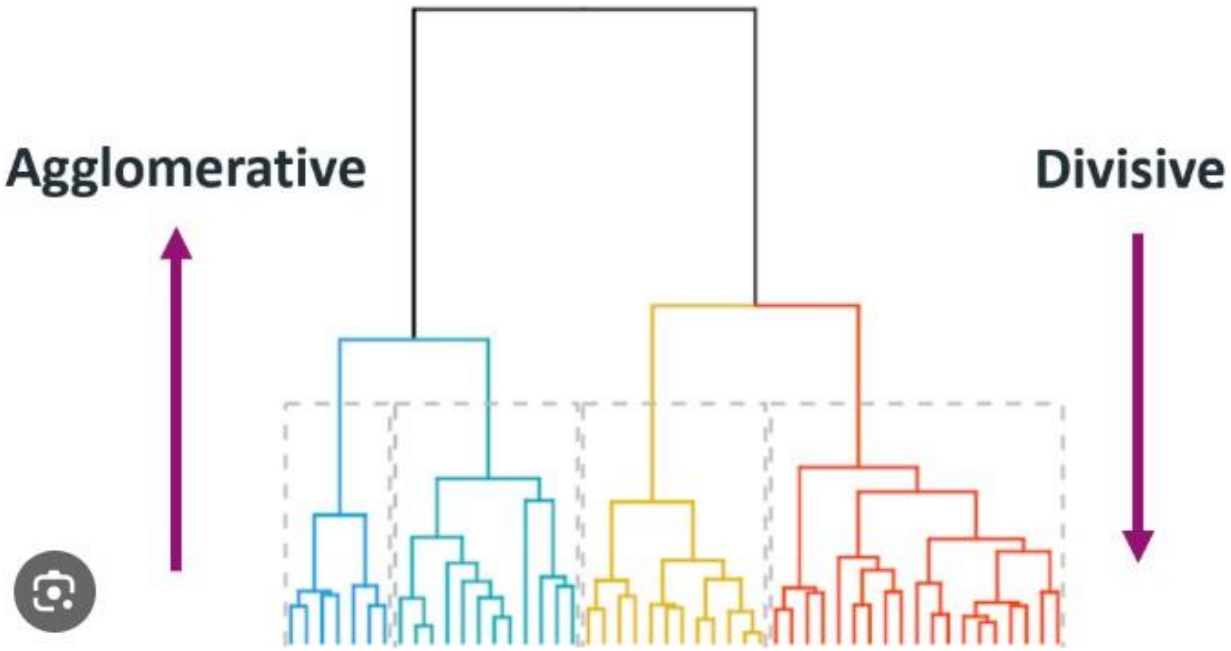
Linkage Methods:

- **Single Linkage:** Distance between closest points of two clusters.
- **Complete Linkage:** Distance between farthest points of two clusters.
- **Average Linkage:** Average distance between all points in two clusters.
- **Ward's Method:** Minimizes variance within clusters.

3. Dendrogram

A **dendrogram** is a tree-like diagram showing the order of cluster merges. It helps to:

- Visualize the hierarchy of clusters.
- Decide the optimal number of clusters by cutting the dendrogram.



4. Applications

- Customer segmentation in marketing.
- Document clustering in NLP.
- Gene expression analysis in bioinformatics.
- Image segmentation.

Python Libraries Required

```
import numpy as np
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

Solved Examples

Example 1: Clustering Simple 2D Points

Dataset:

```
data = np.array([[1, 2], [2, 3], [5, 8], [6, 9], [10, 12]])
```

Solution

```
# Step 1: Import libraries
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt

# Step 2: Linkage matrix
Z = linkage(data, method='ward')  # Using Ward's method

# Step 3: Plot dendrogram
plt.figure(figsize=(6,4))
dendrogram(Z)
plt.title("Dendrogram - Example 1")
plt.show()

# Step 4: Form clusters (choose 2 clusters)
clusters = fcluster(Z, t=2, criterion='maxclust')
print("Cluster assignments:", clusters)
```

Output:

less Copy code

```
Cluster assignments: [1 1 2 2 2]
```

Explanation: The first two points are grouped together; the last three points form the second cluster.

Example 2: Agglomerative Clustering on Random Dataset

Solution

```
from sklearn.datasets import make_blobs
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

# Generate random data
X, _ = make_blobs(n_samples=8, centers=3, random_state=42)

# Linkage
Z = linkage(X, method='complete')

# Dendrogram
plt.figure(figsize=(6,4))
dendrogram(Z)
plt.title("Dendrogram - Example 2")
plt.show()

# Form clusters
clusters = fcluster(Z, t=3, criterion='maxclust')
print("Cluster assignments:", clusters)
```

Explanation: The dendrogram shows three distinct clusters; cluster labels indicate the group each point belongs to.

Example 3: Agglomerative Clustering on Iris Dataset (subset)

Solution

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt

# Load Iris dataset
iris = load_iris()
X = iris.data[:, :2] # Use only sepal length and width
X = StandardScaler().fit_transform(X)

# Linkage
Z = linkage(X, method='average', metric='euclidean')

# Plot dendrogram
plt.figure(figsize=(8,5))
dendrogram(Z)
```

```
plt.title("Dendrogram - Iris Example")
plt.show()

# Form clusters (3 clusters)
clusters = fcluster(Z, t=3, criterion='maxclust')
print("Cluster assignments:", clusters)
```

Explanation:

- Standardization is important to normalize features.
- The dendrogram helps to visualize clusters of similar iris species.
- fcluster assigns each data point to a cluster.

LAB Assignment No. 11

Question 1:

Perform Agglomerative Clustering with Different Linkages.

Task:

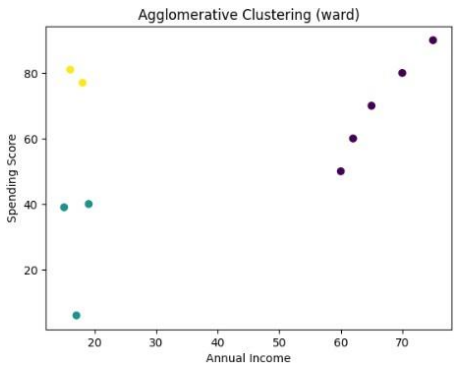
Load the "shopping-data.csv" dataset, extract the features *Annual Income* and *Spending Score*, and perform **Agglomerative Clustering** using:

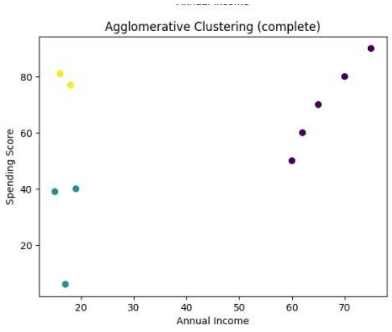
- linkage = "ward"
- linkage = "complete"
- linkage = "average"

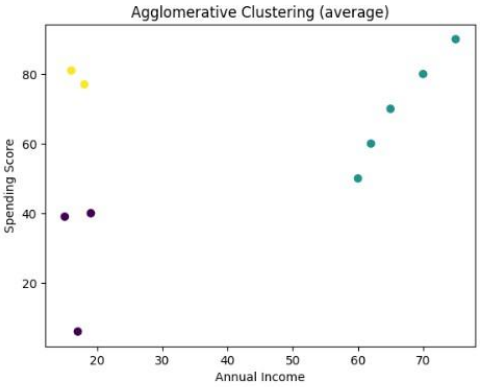
Instructions:

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

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering

# Sample dataset (replace with CSV if needed)
data = {
    "Annual Income": [15, 16, 17, 18, 19, 60, 62, 65, 70, 75],
    "Spending Score": [39, 81, 6, 77, 40, 50, 60, 70, 80, 90]
}

df = pd.DataFrame(data)
X = df.values

linkages = ["ward", "complete", "average"]

for link in linkages:
    model = AgglomerativeClustering(n_clusters=3, linkage=link)
    labels = model.fit_predict(X)

    plt.figure()
    plt.scatter(X[:,0], X[:,1], c=labels)
    plt.title(f"Agglomerative Clustering ({link})")
    plt.xlabel("Annual Income")
    plt.ylabel("Spending Score")
    plt.show()
```







Question 2:

Draw a Dendrogram and Identify the Optimal Number of Clusters

Task:

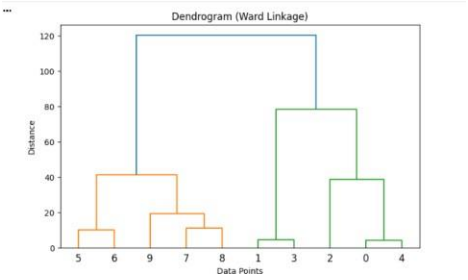
Using the same dataset or any synthetic dataset, draw a **dendrogram** using:

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

Instructions:

1. Fit the data using linkage(method='ward').
2. Plot a dendrogram.
3. From the dendrogram, visually determine:
    - The optimal number of clusters
    - The height at which clusters merge
4. Explain why hierarchical clustering may be preferred over K-Means.

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

Z = linkage(X, method="ward")

plt.figure(figsize=(8,5))
dendrogram(Z)
plt.title("Dendrogram (Ward Linkage)")
plt.xlabel("Data Points")
plt.ylabel("Distance")
plt.show()
```



Question 3: Compare Agglomerative vs Divisive Hierarchical Clustering

Task:

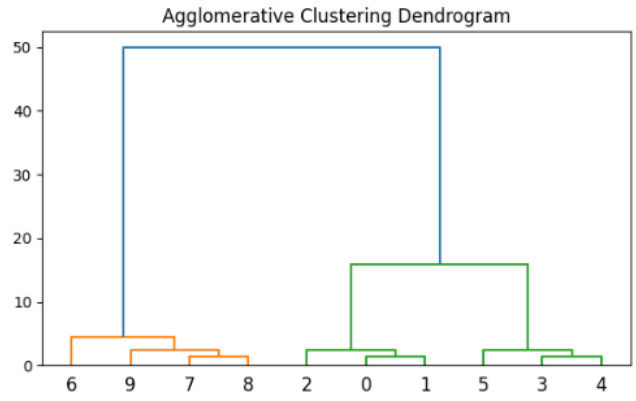
Using a small synthetic dataset (e.g., 10–12 points), perform:

- 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

```python
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage

X = np.array([
    [1,2],[2,3],[3,4],[8,8],[9,9],[10,10],
    [20,20],[21,22],[22,23],[23,24]
])

Z = linkage(X, method="ward")

plt.figure(figsize=(7,4))
dendrogram(Z)
plt.title("Agglomerative Clustering Dendrogram")
plt.show()
```



LAB Assessment

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

Laboratory Work Assessment Rubrics

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

Experiment: CartPole Environment using Gymnasium s Pygame

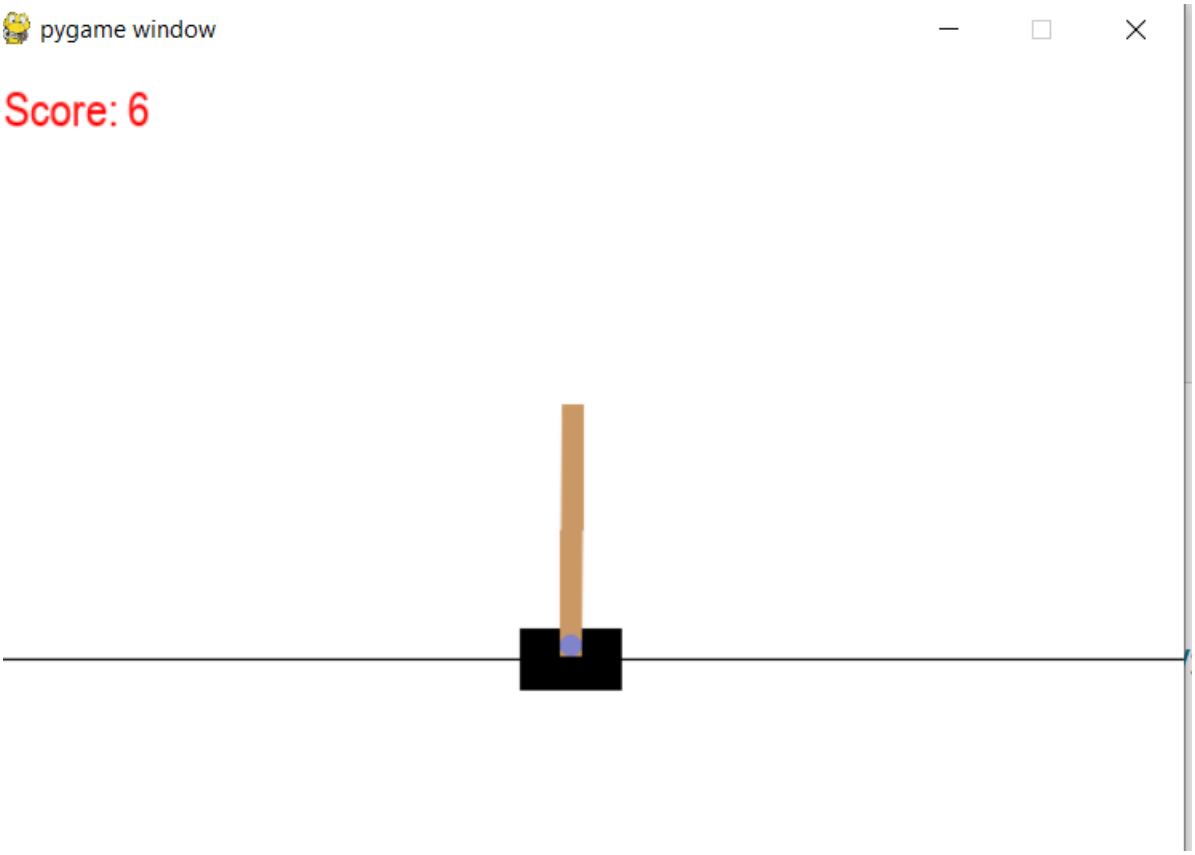
Lab Objectives

After completing this lab, students will be able to:

- Understand the **Reinforcement Learning interaction loop**
- Use **Gymnasium environments**
- Visualize agent behavior using **Pygame**
- Interpret **states, actions, rewards, and episodes**
- Modify and analyze RL environment parameters

```
import gymnasium as gym
import pygame

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

font = None

for episode in range(1, 20):
    score = 0
    state, info = env.reset()
    done = False

    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward

        if font is None:
            pygame.font.init()
            font = pygame.font.SysFont("Arial", 24)

        surface = pygame.display.get_surface()
```

```
        text = font.render(f"Score: {int(score)}", True, (255, 0, 0))
        surface.blit(text, (10, 10))
        pygame.display.update()

    print(f"Episode {episode} Score: {score}")

env.close()
pygame.quit()
```



Lab Questions (Conceptual Understanding)

Q1. What is Reinforcement Learning? Identify the agent, environment, state, action, and reward in the given code.

Reinforcement Learning is a learning paradigm where an agent learns to make decisions by interacting with an environment, taking actions, and receiving rewards, with the goal of maximizing cumulative reward. The program/code controlling the cart.

```
State shape: (4,)
```

4 continuous values describing the system.

Q2. Explain the purpose of the following line:

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

- Creates an instance of the CartPole-v1 environment
- `render mode="human"` enables visual rendering of the environment in a window
- Required to visually see the cart and pole during execution

Q3. What does env.reset() return? Why are two values returned?

Returns:

1. observation → initial state of the environment
2. info → additional diagnostic information (metadata)

Q4. Explain the difference between: Terminated s truncated?

terminated	Episode ended due to task success or failure (e.g., pole fell)
truncated	Episode ended due to external limit (e.g., max time steps reached)

Q5. What is the role of the variable score? How is it calculated?

- ---
  Tracks total reward accumulated in one episode
- Used as a performance measure

Q6. Why is action = env.action_space.sample() used? Is this an intelligent agent? Justify your answer.

No,

- No learning
- No policy
- No memory of past rewards
- Actions are purely random .This is a **random agent,** not an RL-trained agent.

Q7. Explain how Pygame is used to display the score on the screen?

---

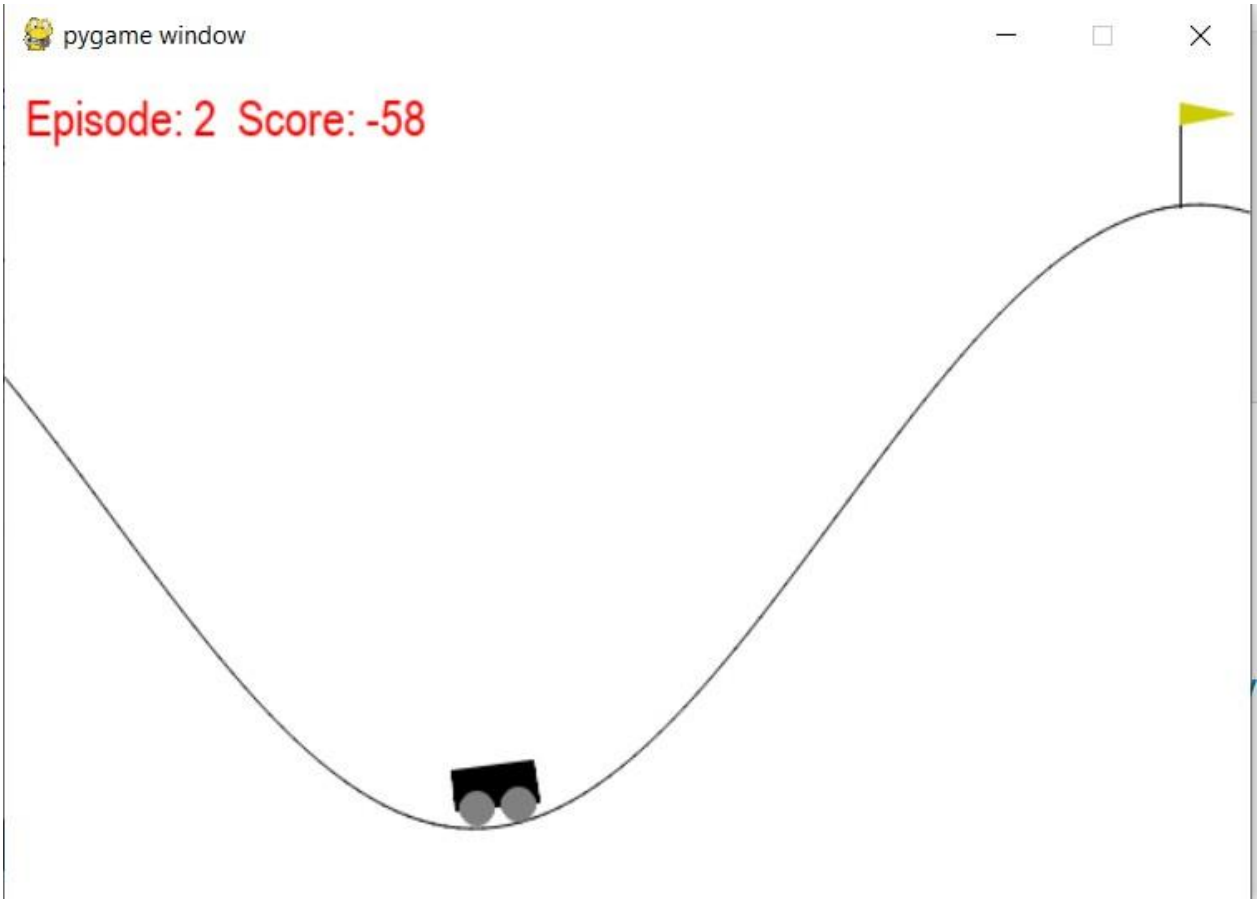
font.render(f"Score: {score}", True, color);

pygame.display.update()

Q8. What happens if the pygame.display.update() line is removed?

- The score will not appear or update on the screen
- Rendering happens in memory only
- The display remains frozen or unchanged

➡ `pygame.display.update()` is required to **refresh the screen**.


pygame window
Episode: 2 Score: -58

✂ Lab Tasks (Hands-on Practice)

🔹 Task 1: Increase Episodes

Modify the code to run **50 episodes** instead of 20.
Observe the score trend across episodes.

```
import gymnasium as gym

env = gym.make("CartPole-v1")

for episode in range(1, 51):
    state, info = env.reset()
    done = False
    score = 0

    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward

    print(f"Episode {episode} Score: {score}")

env.close()
```

```
Episode 1 Score: 28.0
Episode 2 Score: 18.0
Episode 3 Score: 17.0
Episode 4 Score: 40.0
Episode 5 Score: 16.0
Episode 6 Score: 15.0
Episode 7 Score: 25.0
Episode 8 Score: 63.0
Episode 9 Score: 18.0
Episode 10 Score: 12.0
Episode 11 Score: 28.0
Episode 12 Score: 16.0
Episode 13 Score: 18.0
Episode 14 Score: 18.0
Episode 15 Score: 26.0
Episode 16 Score: 12.0
Episode 17 Score: 15.0
Episode 18 Score: 14.0
Episode 19 Score: 44.0
Episode 20 Score: 13.0
Episode 21 Score: 13.0
Episode 22 Score: 39.0
Episode 23 Score: 13.0
Episode 24 Score: 37.0
Episode 25 Score: 17.0
Episode 26 Score: 21.0
Episode 27 Score: 27.0
Episode 28 Score: 15.0
Episode 29 Score: 20.0
Episode 30 Score: 14.0
Episode 31 Score: 14.0
Episode 32 Score: 26.0
Episode 33 Score: 15.0
Episode 34 Score: 17.0
Episode 35 Score: 19.0
Episode 36 Score: 12.0
Episode 37 Score: 14.0
Episode 38 Score: 12.0
Episode 39 Score: 25.0
Episode 40 Score: 19.0
Episode 41 Score: 11.0
Episode 42 Score: 17.0
Episode 43 Score: 17.0
Episode 44 Score: 35.0
Episode 45 Score: 13.0
Episode 46 Score: 10.0
Episode 47 Score: 24.0
Episode 48 Score: 11.0
Episode 49 Score: 12.0
Episode 50 Score: 15.0
```

🔹 Task 2: Display State Values

Print the state array in each step and identify:

- Car position
- Car velocity

```python
import gymnasium as gym

env = gym.make("CartPole-v1")

for episode in range(1, 6):
    state, info = env.reset()
    done = False
    score = 0

    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward

    print(f"Episode: {episode} | Score: {score}")

env.close()
```

```
Episode: 1 | Score: 27.0
Episode: 2 | Score: 16.0
Episode: 3 | Score: 80.0
Episode: 4 | Score: 20.0
Episode: 5 | Score: 33.0
```

🔹 Task 3: Change Text Color s Location

- Change score color from **red to blue**
- Display text at position **(200, 20)**

🔹 Task 4: Track Best Performance

- Store the score of each episode
- Print the **best (highest) score** at the end

```python
import gymnasium as gym

env = gym.make("CartPole-v1")
scores = []

for episode in range(1, 21):
    state, info = env.reset()
    done = False
    score = 0

    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward

    scores.append(score)

env.close()
print("Maximum Score Achieved:", max(scores))
```

```
Maximum Score Achieved: 54.0
```

🔹 Task 5: Slow Down Visualization

Add the following line inside the loop:

pygame.time.delay(20)

Observe the change in animation speed.

```
import gymnasium as gym
import time

env = gym.make("CartPole-v1")

state, info = env.reset()
done = False
score = 0

while not done:
    action = env.action_space.sample()
    state, reward, terminated, truncated, info = env.step(action)
    done = terminated or truncated
    score += reward
    time.sleep(0.02)

print("Final Score:", score)
env.close()
```

```
Final Score: 13.0
```

🔹 Task 6: Compare with CartPole

Replace the environment with:

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

Compare:

- Reward behavior
- Episode termination
- Learning difficulty

```python
import gymnasium as gym

env = gym.make("MountainCar-v0")

for episode in range(1, 6):
    state, info = env.reset()
    done = False
    score = 0

    while not done:
        action = env.action_space.sample()
        state, reward, terminated, truncated, info = env.step(action)
        done = terminated or truncated
        score += reward

    print(f"Episode {episode} Score: {score}")

env.close()
```

```
Episode 1 Score: -200.0
Episode 2 Score: -200.0
Episode 3 Score: -200.0
Episode 4 Score: -200.0
Episode 5 Score: -200.0
```

🔷 Task 7: Simple Rule-Based Policy (Intermediate)

Replace random actions with:

if state[1] > 0:

  action = 2

else:

  action = 0

Observe whether the agent reaches the hilltop

```python
import gymnasium as gym

env = gym.make("CartPole-v1")
state, info = env.reset()

print("State Vector:", state)
print("Number of State Variables:", len(state))
print("Variables represent: cart position, cart velocity, pole angle, pole angular velocity")

env.close()
```

```
State Vector: [0.03961765 0.03290732 0.03363164 0.03891516]
Number of State Variables: 4
Variables represent: cart position, cart velocity, pole angle, pole angular velocity
```

🔷 Task 8 (Advanced): Episode Length Analysis

Print the **number of steps per episode** and analyze:

- Why some episodes last longer
- Relation between steps and score

📊 Observation Table

Episode	Steps	Score	Goal Reached (Yes/No)
1			
2			
...			
20			

```
import gymnasium as gym

env = gym.make("CartPole-v1")

state, info = env.reset()
done = False
score = 0

while not done:
    if state[2] > 0:
        action = 1
    else:
        action = 0

    state, reward, terminated, truncated, info = env.step(action)
    done = terminated or truncated
    score += reward

print("Final Score (Rule-Based):", score)
env.close()
```

```
Final Score (Rule-Based): 49.0
```

Lab Assessment

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

Laboratory Work Assessment Rubrics

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

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

Lab No. 13

Natural Language Processing (NLP)

This laboratory session introduces students to Word2Vec, a popular technique in Natural Language Processing (NLP) used to represent words as meaningful dense vectors. Using textual data from the *Game of Thrones* series, students will learn how word embeddings capture semantic relationships between words based on their context. Through preprocessing, model training, and similarity analysis, this lab helps students understand how machines learn word meanings and relationships from large text corpora in an unsupervised manner.

LAB Objectives:

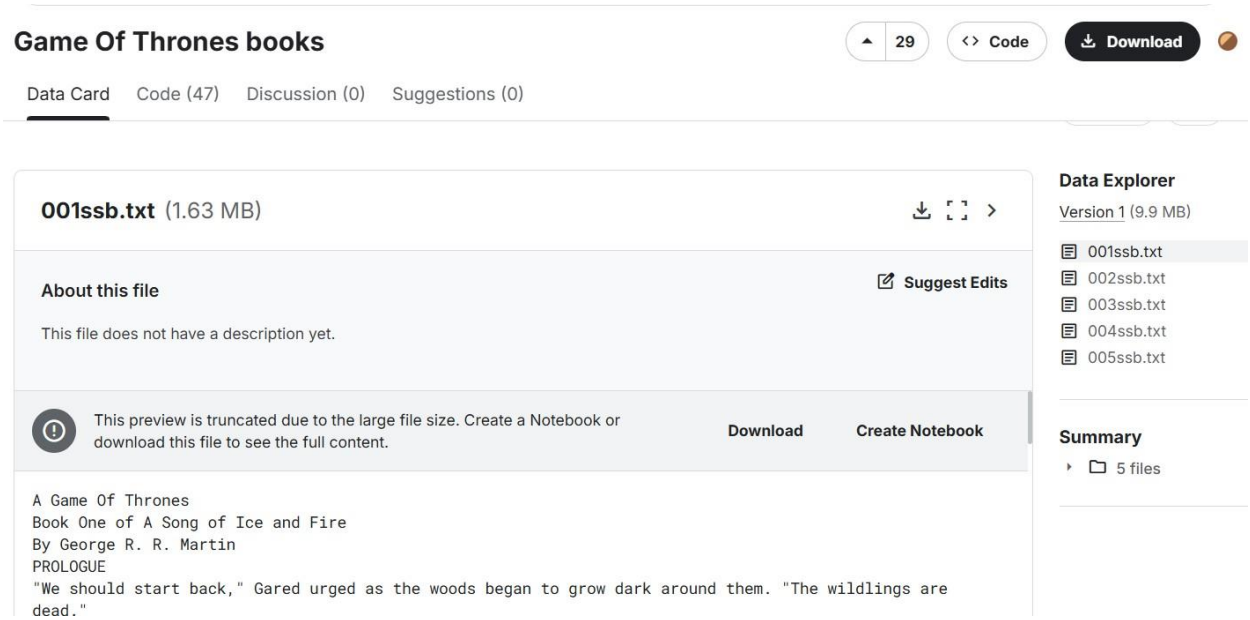
- Understand **distributional semantics**
- Learn how **Word2Vec** converts words into dense vectors
- Apply **Word2Vec (Skip-Gram / CBOW)** on real textual data (Game of Thrones)
- Explore **word similarity, analogy, and visualization**

game-of-thrones-word2vec

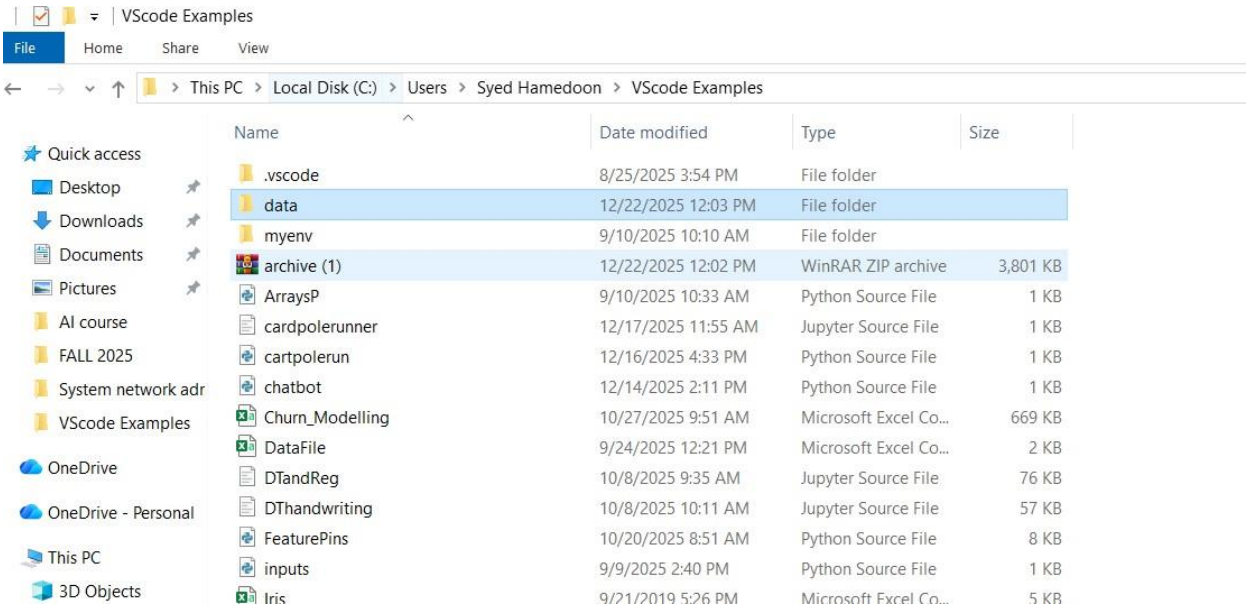
word2vec applied on game of thrones data

Dataset Link: https://www.kaggle.com/khulasasndh/game-of-thrones-books

Download the data set from Kaggle



Add the dataset in folder data where VS code directory present



Code in jupter VS code:

```
import numpy as np
import pandas as pd
```

```
!pip install gensim
```

```
import gensim
import os
```

```
!pip install nltk
```

```
data = "C:/Users/Syed Hamedoon/VScode Examples/data"
```

```
import nltk

nltk.download('punkt')
nltk.download('punkt_tab')
```

```
import os
from nltk import sent_tokenize
from gensim.utils import simple_preprocess

DATA_PATH = r"C:\Users\Syed Hamedoon\VScode Examples\data"

story = []

for filename in os.listdir(DATA_PATH):
  if filename.endswith(".txt"):
    file_path = os.path.join(DATA_PATH, filename)

    try:
      with open(file_path, "r", encoding="utf-8") as f:
        corpus = f.read()
    except UnicodeDecodeError:
      with open(file_path, "r", encoding="cp1252") as f:
        corpus = f.read()

    for sent in sent_tokenize(corpus):
```

```
        story.append(simple_preprocess(sent))
```

```
print(len(story))
print(story[:2])
```

```
model = gensim.models.Word2Vec(
   window=10,
   min_count=2
)
```

```
model.build_vocab(story)
```

```
model.train(story, total_examples=model.corpus_count, epochs=model.epochs)
```

```
model.wv.most_similar('daenerys')
```

```
model.wv.doesnt_match(['jon','rikon','robb','arya','sansa','bran'])
```

```
model.wv.doesnt_match(['cersei', 'jaime', 'bronn', 'tyrion'])
```

```
model.wv['king']
```

```
model.wv.similarity('arya','sansa')
```

```
model.wv.similarity('tywin','sansa')
```

```
model.wv.get_normed_vectors()
```

```
y = model.wv.index_to_key
```

```
len(y)
```

```
y
```

```
from sklearn.decomposition import PCA
```

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

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

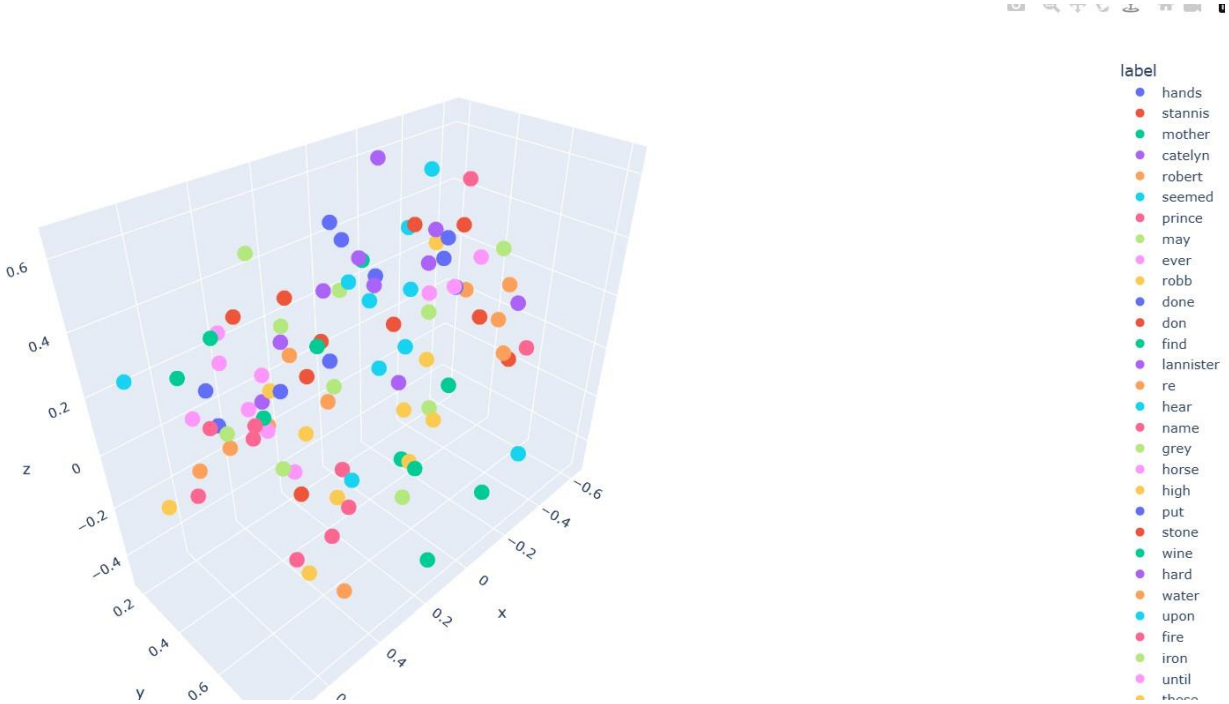
```
X.shape
```

```
!pip install --upgrade nbformat
```

```
import pandas as pd
```

```
import plotly.express as px
import plotly.io as pio

pio.renderers.default = "browser"  # ← IMPORTANT

df = pd.DataFrame(X[200:300], columns=["x", "y", "z"])
df["label"] = y[200:300]

fig = px.scatter_3d(
   df,
   x="x",
   y="y",
   z="z",
   color="label"
)

fig.show()
```

Output:


label
hands
stannis
mother
catelyn
robert
seemed
prince
may
ever
robb
done
don
find
lannister
re
hear
name
grey
horse
high
put
stone
wine
hard
water
upon
fire
iron
until
0.6
0.4
0.2
z
0
−0.2
−0.4
0.2
0.4
0.6
y
−0.6
−0.4
−0.2
0
0.2
0.4
x

LAB Questions

1. **What is the core idea behind Word2Vec?**

   Word2Vec learns dense vector representations (embeddings) of words such that words appearing in similar contexts have similar vectors.
   It is based on the distributional hypothesis: *"You shall know a word by the company it keeps."*
   The model is trained using a shallow neural network to predict context words from a target word or vice versa, without any labeled data.

2. **Difference between CBOW and Skip-Gram?**

Aspect	CBOW (Continuous Bag of Words)	Skip-Gram
Predicts	Target word	Context words
Input	Context words	Target word
Output	Target word	Surrounding words
Speed	Faster	Slower
Rare words	Poor performance	Better performance
Best for	Large datasets	Small / sparse datasets

3. **Why is one-hot encoding inefficient?**

- ---
  Very high dimensional (size = vocabulary size)
- Sparse vectors (mostly zeros)
- No notion of semantic similarity (all words equally distant)
- High memory and computation cost
  Word2Vec solves this by learning dense, low-dimensional vectors.

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

- Appear in **similar contexts**
- Co-occur with similar actions, places, or dialogue
- Share narrative roles

5. **How does window size affect semantic learning?**

❖ **Small window size**
   - ➢ Captures **syntactic** relationships (grammar, function words)

- **Large window size**
  - Captures **semantic** relationships (topic, meaning)

6. **Why might rare characters have poor embeddings?**

- Few training examples
- Insufficient context exposure
- Weights are updated less often

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

input: one-hot vector of size **V**
Output: probability distribution over **V** words

$V \rightarrow D \rightarrow V$

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

Too small

- Cannot capture enough semantic information
- Words become indistinguishable

Too large

- Overfitting
- Increased computation
- Redundant dimensions

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

Word2Vec is **unsupervised**:

- No labeled data required
- Meaning emerges from **context patterns**
- Semantic relationships are learned implicitly

```
import numpy as np

import pandas as pd
```

```
!pip install gensim
```

```
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (8.4 kB)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (2.0.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.16.3)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.5.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim) (2.0.1)
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (27.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 79.1 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

+ Code + Text

```
import gensim

import os
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
```

```
DATA_PATH = "/content/game-of-thrones-books.txt"
```

```
import nltk
nltk.download('punkt')

nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True
```

+ Code + Text

```
import os
from nltk import sent_tokenize
from gensim.utils import simple_preprocess

file_path = "/content/game-of-thrones-books.txt"
story = []

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[12])
```

```
27244
[['game', 'of', 'thrones', 'book', 'one', 'of', 'song', 'of', 'ice', 'and', 'fire', 'by', 'george', 'martin', 'prologue', 'we', 'should', 'start', 'back', 'gared', 'urged', 'as', 'the', 'woods', 'began', 'to', 'grow', 'dark', 'around', 'them'], ['t
```

```
print(len(story))

print(story[12])
```

```
27244
[['game', 'of', 'thrones', 'book', 'one', 'of', 'song', 'of', 'ice', 'and', 'fire', 'by', 'george', 'martin', 'prologue', 'we', 'should', 'start', 'back', 'gared', 'urged', 'as', 'the', 'woods', 'began', 'to', 'grow', 'dark', 'around', 'them'], ['t
```

```
model = gensim.models.Word2Vec(
window=10,
min_count=2

)
```

```
model.build_vocab(story)
```

```
model.train(story, total_examples=model.corpus_count, epochs=model.epochs)
```

```
(1058991, 1423500)
```

```
model.wv.most_similar('daenerys')
```

```
[('eunuch', 0.9972313046455383),
 ('giant', 0.9969602823257446),
 ('flowers', 0.9958428144454956),
 ('squires', 0.9956861138343811),
 ('met', 0.995603084564209),
 ('size', 0.9955247640609741),
 ('cooks', 0.9955064058303833),
 ('nine', 0.9954379796981812),
 ('noye', 0.9952808618545532),
 ('rich', 0.9951012134552002)]
```

```
model.wv.doesnt_match(['jon','rikon','robb','arya','sansa','bran'])
```

```
WARNING:gensim.models.keyedvectors:vectors for words {'rikon'} are not present in the model, ignoring these words
'robb'
```

```
model.wv.doesnt_match(['cersei', 'jaime', 'bronn', 'tyrion'])
```

```
'jaime'
```

```
array([-0.32364684,  0.14699282, -0.2014781 ,  0.411841  , -0.42814672,
       -0.6645096 ,  0.8767231 ,  1.2312515 , -0.7817989 , -0.26366857,
       -0.1288221 ,  0.38670003, -0.32529774,  0.02383385, -0.27395973,
       -0.8687137 ,  0.39080772, -1.2682753 , -0.39742675, -0.14850628,
        0.5177883 ,  0.14793532, -0.5513239 , -0.31911954, -0.21880631,
       -0.60318357,  0.3608716 , -0.4638677 , -0.831075  , -0.36981553,
        1.2066488 ,  0.39672247, -0.18338498,  0.03865973,  0.18228538,
        0.7217221 , -0.28702644, -0.32573026,  0.07801952, -1.4524268 ,
       -0.56677324,  0.04206057, -0.17786478,  0.2015173 ,  0.08465559,
       -0.5794629 , -0.09701911,  0.0581483 ,  1.1127372 , -0.20436169,
        0.16156015, -0.32228398, -0.5621291 , -0.6148126 ,  0.4278058 ,
        0.92534274,  0.02756782,  0.3747875 , -0.4596324 , -0.61628896,
        0.49131042,  0.16848524,  0.04391758, -0.43641034, -1.0697795 ,
        0.3366395 , -0.3251882 ,  0.70725197, -1.2791224 ,  0.5583585 ,
       -0.5434768 ,  0.30506063,  0.74065155,  0.21314897,  1.0477426 ,
       -0.59085435,  0.5275562 ,  0.01475037, -0.42453963,  0.01001824,
        0.08467942,  0.70829916, -1.0051441 ,  1.4805706 , -0.7734855 ,
       -0.6483895 ,  0.1382422 ,  0.46904463,  0.67336994,  0.2627122 ,
        0.5806621 ,  0.10132672,  0.5788254 ,  0.21193454,  0.12871997,
        0.60438794, -0.05561303,  0.81188494,  0.06673696, -0.30300218],
      dtype=float32)
```

```
model.wv.similarity('arya','sansa')
```

```
np.float32(0.9643241)
```

```
model.wv.similarity('tywin','sansa')
```

```
np.float32(0.5510077)
```

```
model.wv.get_normed_vectors()
```

```
array([[-0.1248031 ,  0.01005179, -0.03457814, ..., -0.12131795,
         0.01348632,  0.17582259],
       [-0.06992937,  0.05892245,  0.01990201, ..., -0.12860942,
         0.08207524,  0.09966443],
       [-0.14605236,  0.01376512,  0.0734483 , ..., -0.05151271,
         0.0974825 , -0.13760775],
       ...,
       [ 0.05540429, -0.1417207 ,  0.0541888 , ..., -0.13647927,
```

```
y = model.wv.index_to_key
```

```
len(y)
```

```
7432
```

+ Code + Text

```
y
```

```
'to',
'of',
'he',
'his',
'was',
'her',
'you',
'in',
'it',
'she',
'had',
'as',
'him',
'with',
'that',
'said',
'not',
'for',
'at',
'on',
'they',
'but',
'is',
'my',
'lord',
'no',
'from',
'them',
'have',
'would',
'were',
```

```
from sklearn.decomposition import PCA
```

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

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

```
X.shape
```

```
(7432, 3)
```

```
!pip install --upgrade nbformat
```

```
Requirement already satisfied: nbformat in /usr/local/lib/python3.12/dist-packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.12/dist-packages (from nbformat) (2.21.2)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.12/dist-packages (from nbformat) (4.25.1)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /usr/local/lib/python3.12/dist-packages (from nbformat) (5.9.1)
Requirement already satisfied: traitlets>=5.1 in /usr/local/lib/python3.12/dist-packages (from nbformat) (5.7.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbformat) (25.4.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbformat) (2025.9.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbformat) (0.37.0)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbformat) (0.30.0)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.12/dist-packages (from jupyter-core!=5.0.*,>=4.12->nbformat) (4.5.1)
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.12/dist-packages (from referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.15.0)
```

```
import pandas as pd
import plotly.express as px
import plotly.io as pio

pio.renderers.default = "colab"
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",
```

Lab Assesment

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

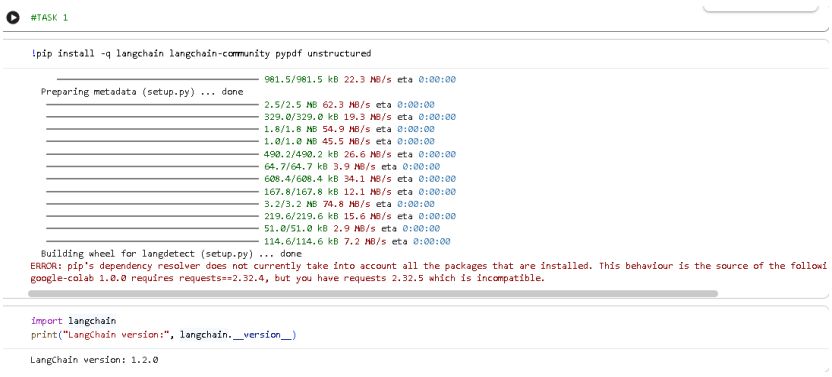
Tools & Libraries

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

Lab Tasks (Practice Steps)

Task 1: Environment Setup

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



Task 2: Understand the Main Concept – Document Loaders

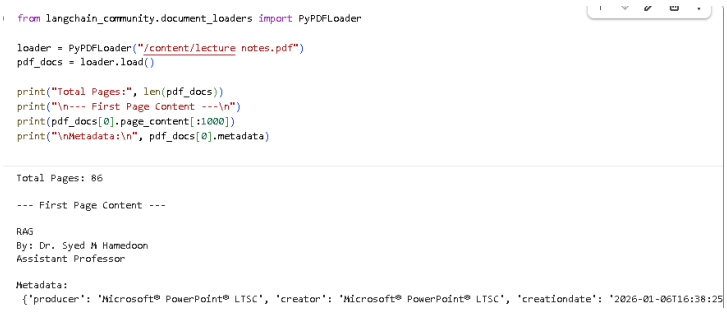
- Study the role of **Document Loaders** in RAG
- Explain how loaders convert raw data into LangChain Document objects

```python
from langchain_core.documents import Document

doc = Document(
    page_content="This is sample content for understanding document loaders.",
    metadata={"source": "example.txt"}
)

print("Content:")
print(doc.page_content)

print("\nMetadata:")
print(doc.metadata)
```

```
Content:
This is sample content for understanding document loaders.

Metadata:
{'source': 'example.txt'}
```

Task 3: Load PDF Data (PyPDFLoader)

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



Task 4: Load Web Data (WebBaseLoader)

- Use **WebBaseLoader** to load a webpage
- Extract main textual content
- Observe metadata (URL source)
- Attach code with output screenshot

```
from langchain_community.document_loaders import WebBaseLoader

url = "https://en.wikipedia.org/wiki/Retrieval-augmented_generation"

loader = WebBaseLoader(url)
web_docs = loader.load()

print("Total Web Documents:", len(web_docs))
print("\n--- Web Page Content (First 1000 characters) ---\n")
print(web_docs[0].page_content[:1000])
print("\nMetadata:\n", web_docs[0].metadata)
```

```
move to sidebar
hide
…

(Top)

1
RAG and LLM limitations

2
Process

Toggle Process subsection

2.1
RAG key stages
```

Metadata: {'source': 'https://en.wikipedia.org/wiki/Retrieval-augmented_generation', 'title': 'Retrieval-augmented generation - Wikipedia', 'language': 'en'}

Task 5: Load Structured Data (CSVLoader)

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

```python
import pandas as pd

data = {
    "id": [1, 2, 3],
    "name": ["Ali", "Ayesha", "Hamza"],
    "department": ["CS", "SE", "IT"],
    "marks": [85, 90, 88]
}

df = pd.DataFrame(data)
df.to_csv("/content/students.csv", index=False)
df
```

	id	name	department	marks
0	1	Ali	CS	85
1	2	Ayesha	SE	90
2	3	Hamza	IT	88

```python
from langchain_community.document_loaders import CSVLoader

loader = CSVLoader("/content/students.csv")
csv_docs = loader.load()

print("Total Rows Loaded:", len(csv_docs))
print("\n--- Sample Document ---\n")
print(csv_docs[0].page_content)
print("\nMetadata:\n", csv_docs[0].metadata)
```

```
Total Rows Loaded: 3

--- Sample Document ---

id: 1
name: Ali
department: CS
marks: 85

Metadata:
 {'source': '/content/students.csv', 'row': 0}
```

Task 6: Compare All Loaders

Students must compare:

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

```python
print("===== PDF LOADER =====")
print("Content Type:", type(pdf_docs[0].page_content))
print("Metadata:", pdf_docs[0].metadata)

print("\n===== WEB LOADER =====")
print("Content Type:", type(web_docs[0].page_content))
print("Metadata:", web_docs[0].metadata)

print("\n===== CSV LOADER =====")
print("Content Type:", type(csv_docs[0].page_content))
print("Metadata:", csv_docs[0].metadata)
```

```
===== PDF LOADER =====
Content Type: <class 'str'>
Metadata: {'producer': 'Microsoft® PowerPoint® LTSC', 'creator': 'Microsoft® PowerPoint® LTSC', 'creationdate': '2026-01-0

===== WEB LOADER =====
Content Type: <class 'str'>
Metadata: {'source': 'https://en.wikipedia.org/wiki/Retrieval-augmented_generation', 'title': 'Retrieval-augmented generat

===== CSV LOADER =====
Content Type: <class 'str'>
Metadata: {'source': '/content/students.csv', 'row': 0}
```

Lab Questions

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

   Document loaders convert raw external data into structured Document objects that can be indexed, embedded, and retrieved to provide context-aware responses in RAG systems.

2. **Why is metadata important in LangChain documents?**

   Metadata helps identify the source, page number, URL, or row index, enabling better traceability, filtering, and accurate retrieval.

3. **Difference between TextLoader and PyPDFLoader?**

- TextLoader :loads plain `.txt` files as a single document
- PyPDFLoader :extracts text page-by-page from PDFs with page metadata.

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

   PyPDFLoader cannot extract text from images. OCR tools (e.g., Tesseract) are required before loading.

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

   It allows bulk loading of large document collections automatically, making it scalable for enterprise RAG systems.

6. **How does document quality affect RAG performance?**

   Poor-quality or noisy documents lead to inaccurate embeddings and weak retrieval, reducing response accuracy.

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 s Technologies

- Python 3.9+
- VS Code
- Ollama (LLAMA2)
- LangChain
- Streamlit
- FAISS (Vector Store)
- Dotenv

File code 1

Localama.py

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_community.llms import Ollama
import streamlit as st
import os
from dotenv import load_dotenv
```

```
load_dotenv()

os.environ["LANGCHAIN_TRACING_V2"]="true"
os.environ["LANGCHAIN_API_KEY"]=os.getenv("LANGCHAIN_API_KEY")

## Prompt Template

prompt=ChatPromptTemplate.from_messages(
  [
    ("system","You are a helpful assistant. Please response to the user queries"),
    ("user","Question:{question}")
  ]
)
## streamlit framework

st.title('Langchain Demo With LLAMA2 API')
input_text=st.text_input("Search the topic u want")

# ollama LLAma2 LLm
llm=Ollama(model="llama2")
output_parser=StrOutputParser()
chain=prompt|llm|output_parser

if input_text:
  st.write(chain.invoke({"question":input_text}))
```

Code File 2

App.py

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

import streamlit as st
import os
from dotenv import load_dotenv

os.environ["OPENAI_API_KEY"]=os.getenv("OPENAI_API_KEY")
## Langmith tracking
os.environ["LANGCHAIN_TRACING_V2"]="true"
os.environ["LANGCHAIN_API_KEY"]=os.getenv("LANGCHAIN_API_KEY")
```

```
## Prompt Template

prompt=ChatPromptTemplate.from_messages(
  [
    ("system","You are a helpful assistant. Please response to the user queries"),
    ("user","Question:{question}")
  ]
)

## streamlit framework

st.title('Langchain Demo With OPENAI API')
input_text=st.text_input("Search the topic u want")

# openAI LLm
llm=ChatOpenAI(model="gpt-3.5-turbo")
output_parser=StrOutputParser()
chain=prompt|llm|output_parser

if input_text:
  st.write(chain.invoke({'question':input_text}))
```

Step 1: Create Project Structure

Open **VS Code** and create the following folder structure:

```
rag-ollama-chatbot/
│
├── app.py
├── requirements.txt
├── .env
├── data/
│   └── sample_docs.txt
```

Step 2: Install and Verify Ollama

2.1 Install Ollama

Download and install Ollama from:

```
https://ollama.com
```

2.2 Pull LLAMA2 Model

Open terminal and run:

```bash
ollama pull llama2
```

2.3 Verify Ollama

```bash
ollama run llama2
```

If the model responds, Ollama is working correctly.

Step 3: Create Virtual Environment

```bash
python -m venv myenv
myenv\Scripts\activate   # Windows
```

Step 4: Install Required Libraries

Create **requirements.txt**:

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

Install dependencies:

```bash
pip install -r requirements.txt
```

Step 5: Add Sample Knowledge Base

Create **data/sample_docs.txt** and add:

```pgsql
LangChain is a framework for developing applications powered by large language mo
RAG stands for Retrieval-Augmented Generation.
Ollama allows running LLMs locally without cloud APIs.
FAISS is a vector database for similarity search.
```

Step 6: Environment Configuration

Create **.env** file:

```env
LANGCHAIN_API_KEY=your_langchain_api_key
```

> Note: Even with Ollama, LangChain tracing may require this key.

Step 7: Understand the Base Chatbot Code (Given Code)

The provided code:

- Uses **Ollama LLAMA2**
- Accepts user input via Streamlit
- Sends query directly to the LLM
- ❌ Does NOT use retrieval (no RAG)

We will **extend this code** to add:

- Document loading
- Text splitting
- Embeddings
- Vector database
- Retriever

Step 8: Add RAG Components

8.1 Import Additional Modules

Update **app.py**:

python

```python
from langchain_community.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.embeddings import OllamaEmbeddings
from langchain_community.vectorstores import FAISS
from langchain_core.runnables import RunnablePassthrough
```

Step 9: Load and Process Documents

Add below `.env` loading:

python

```
# Load documents
loader = TextLoader("data/sample_docs.txt")
documents = loader.load()

# Split documents
text_splitter = RecursiveCharacterTextSplitter(
  chunk_size=500,
  chunk_overlap=50
)
docs = text_splitter.split_documents(documents)
```

Step 10: Create Embeddings and Vector Store

```
# Create embeddings
embeddings = OllamaEmbeddings(model="llama2")
```

```
# Create FAISS vector store
vectorstore = FAISS.from_documents(docs, embeddings)

# Create retriever
retriever = vectorstore.as_retriever()
```

Step 11: Modify Prompt for RAG

Replace your prompt template with:

```
prompt = ChatPromptTemplate.from_messages(
  [
    ("system", "Answer the question using the provided context only."),
    ("user", "Context:\n{context}\n\nQuestion:\n{question}")
  ]
)
```

Step 12: Build the RAG Chain

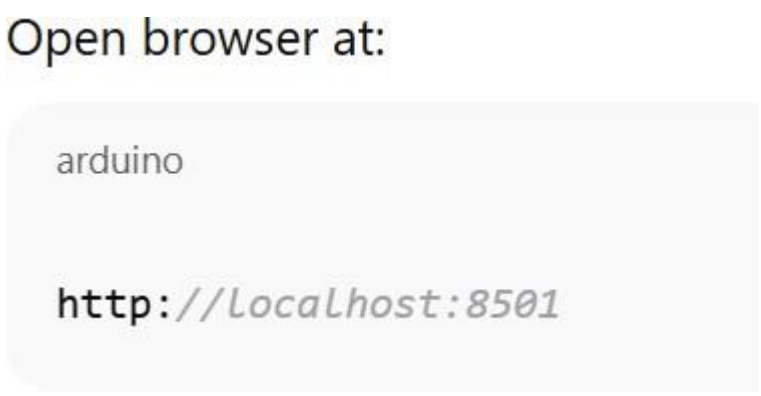
```
llm = Ollama(model="llama2")
output_parser = StrOutputParser()

rag_chain = (
  {
    "context": retriever,
    "question": RunnablePassthrough()
  }
  | prompt
  | llm
  | output_parser
)
```

Step 13: Update Streamlit UI

```
st.title("RAG Chatbot with Ollama C 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
```



LAB Assessment

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