SCHEME: K

Name: _____

Roll No.: _____ Year: 20___ - 20___

Exam Seat No.: _____

LABORATORY MANUAL FOR

ADVANCED ALGORITHM IN AI & ML (316320)

COMPUTER ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous) (ISO21001:2018) (ISO/IEC27001:2013)

VISION

To ensure that the Diploma level Technical Education constantly matches the latest requirements of Technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the challenging technological & environmental challenges.

QUALITY POLICY

We, at MSBTE are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation and monitoring system along with adequate faculty development programmes.

CORE VALUES

MSBTE believes in the following:

- Skill development in line with industry requirements
- Industry readiness and improved employability of Diploma holders
- Synergistic relationship with industry
- Collective and Cooperative development of all stake holders
- Technological interventions in societal development
- Access to uniform quality technical education

A Laboratory Manual

For

Advanced Algorithm in AI & ML (316320)

SEMESTER-VI

"K-SCHEME"

(AI/AN)

Maharashtra State

Board of Technical Education, Mumbai.

(Autonomous) (ISO21001:2018) (ISO/IEC27001:2013)

Maharashtra State Board of Technical Education, Mumbai

(Autonomous) (ISO21001:2018) (ISO/IEC27001:2013)

4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai – 400051.

(Printed on December 2025)

Maharashtra State Board of Technical Education

Certificate

This is to certify that Mr./Ms.…..…… Roll No……. of Sixth Semester of Diploma in …… ……..…….…….…… of Institute, ……...…….…… …… (Instt. Code: ……) has completed the term work satisfactorily in course **Advanced Algorithm in AI & ML (316320)** for the academic year 20…… to 20…… as prescribed in the curriculum**.**

Place: …… Enrollment No: ……..

Date: ……. Exam. Seat No: …….

Course Teacher **Head of the Department** **Principal**

Seal of Institution

PREFACE

The development of the critically important industry-relevant abilities and skills is the main goal of any engineering laboratory or field work in the technical education system. In light of this, MSBTE developed the most recent "K" Scheme curricula for engineering diploma programs, emphasizing outcome-based learning. As a result, a sizable portion of the program is dedicated to practical work. This demonstrates how crucial laboratory work is in helping teachers, instructors, and students understand that every minute of lab time must be used efficiently to create these outcomes rather than wasting it on unnecessary activities. Every practical has thus been created to operate as a "vehicle" to help each student acquire this industry-identified capability in order to ensure the effective implementation of this outcome-based curriculum. The "chalk and duster" practice in the classroom is a challenging way to build practical skills. As a result, the development team of the "K" scheme laboratory manual focused on the intended results when creating the practical, as opposed to the customary approach of performing practical's to "verify the theory".

This lab manual is intended to support all parties involved, particularly the students, instructors, and teachers, in helping the students achieve the pre-established goals. It is required of every student to read through the relevant practical process in its entirety and comprehend the bare minimum of theoretical background related to the practical at least one day in advance of the practical. As a crucial starting point for carrying out the practical, each exercise in this handbook starts with establishing the competency, industry-relevant skills, course outcomes, and practical results. After that, the students will learn about the abilities they will acquire through the process outlined there and the safety measures that must be followed, which will enable them to use in addressing real-world situations in their professional life.

This manual also offers guidance to educators on how to manage resources so that students follow protocols and safety measures methodically and meet learning objectives. This allows teachers and instructors to effectively support student-centered lab activities through each practical exercise.

Today's globalized world has witnessed tremendous technological breakthroughs in surveying equipment and technology. Currently available accurate digital surveying tools are employed because of their speed, precision, and ease of use. The disciplines of civil engineering, mining engineering, environmental engineering, transportation engineering, and marine engineering heavily rely on these tools and applications. Given the importance of remote sensing and Geographic Information Systems (GIS) and their widespread usage in mapping and storing spatial data, it is expected that students will have a basic understanding of these subjects in order to use them in the field. Students who complete this course will have the necessary abilities and competences to perform tasks linked to surveys.

Although best possible care has been taken to check for errors (if any) in this laboratory manual, perfection may elude us as this is the first edition of this manual. Any errors and suggestions for improvement are solicited and highly welcome.

Program outcome (POs) to be achieved through Course:

PO 1. Basic & Discipline specific knowledge: Apply knowledge of basic mathematics, sciences and engineering fundamentals and engineering specialization to solve the engineering problems.

PO 2. Problem Analysis: Identify and analyze well defined engineering problems using codified standard methods.

PO 3. Design /Development Solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO 4. Engineering tools experimentation and testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO 5. Engineering practices for society sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO 6. Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

PO 7. Lifelong learning: Ability to analyze individual needs and engage in updating in context of technological changes.

List of Relevant Skills

On the successful completion of the course the students will acquire the required industry relevant skills and they will be able to:

1. Understanding how to clean and prepare data for analysis.
2. Creating visual representations of data to communicate insights effectively.
3. Learning how to assess the performance of machine learning models.
4. Utilizing cloud-based AI services for efficient model development and deployment.
5. Keeping up with the latest advancements in AI and ML to stay competitive.

Practical Course Outcome Matrix:

CO1 - Apply suitable Machine Learning model for dataset feature extraction.
CO2 - Implement Machine learning algorithms on given problem.
CO3 - Implement Artificial Neural Networks analyzing associated parameters of Deep Learning.
CO4 - Build a Convolutional Neural Network for given context.
CO5 - Classify Sequential and Image Data using Deep Learning.

Pr. No.	Title of the Practical	Mapped Course Outcome				
		CO1	CO2	CO3	CO4	CO5
1	*a. Installation of Tools and Libraries (Jupyter Notebook /Matplotlib/Numpy /Pandas/PyTorch/scikit-learn) b. Use of google colab (https://colab.research.google.com/)	√	--	--	--	--
2	Apply filter and wrapper method on any standard datasets	√	--	--	--	--
3	* Perform following operations: (Assume suitable data/dataset if needed). I. Write program to read dataset (Text, CSV, JSON, XML) II. Which of the attributes are numeric and which are categorical? III. Performing Data Cleaning, Handling Missing Data, Removing Null data IV. Rescaling Data vs. Encoding Data V. Feature Selection	√	--	--	--	--
4	* Write a program to implement SVM for classification using suitable dataset	--	√	--	--	--
5	Write a program to implement unsupervised machine learning algorithm (Clustering – K Medoid) in python on dataset to cluster data. (Assume suitable dataset)	--	√	--	--	--
6	* Write a program to implement Association Rule Learning (Apriori Algorithm) on any dataset	--	√	--	--	--
7	Write a program to implement Association Rule Learning (Eclat Algorithm) on any dataset	--	√	--	--	--
8	* Write a program to implement AND Logic Gate with 2-bit Binary Input using Perceptron algorithm	--	--	√	--	--
9	* Write a program to implement Perceptron Learning in Python using Iris flower dataset	--	--	√	--	--
10	Write a program to implement Gradient Descent in PyTorch	--	--	√	--	--

11	Write a program to implement /Simulate Back propagation/ feed forward neural network	--	--	--	√	--
12	Build a small CNN model consisting of 5 convolution layers	--	--	--	√	--
13	* Write a program to implement CIFAR 10- CNN using PyTorch	--	--	--	√	--
14	Basic Time Series Forecasting with LSTM	--	--	--	--	√
15	* Classification of images using imagenet dataset	--	--	--	--	√

Guidelines to teachers

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain prior concepts to the students before starting of each practical.
3. Involve students in performance of each practical.
4. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
5. Teachers should give opportunity to students for hands on experience after the demonstration.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected the students by the industry.
8. Finally give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions.

Instructions to Students

1. Organize the work in the group and make record all programs.
2. Students shall develop maintenance skill as expected by industries.
3. Students shall attempt to develop related hand-on skills and gain confidence.
4. Students shall develop the habits of evolving more ideas, innovations, skills etc. those included in scope of manual
5. Students shall refer technical magazines.
6. Students should develop habit to submit the practical on date and time.
7. Students should be well prepared for viva while submitting write-up of exercise.
8. Attach /paste separate papers wherever necessary.

Content Page

List of Practical and Formative Assessment sheet.

Sr. No	Title of the Practical	Date of performance	Date of Submission	Assessment Marks (25)	Dated sign of teacher	Remarks (if any)
1	*a. Installation of Tools and Libraries (Jupyter Notebook /Matplotlib/Numpy / Pandas / PyTorch/ scikit-learn) b. Use of google colab (https://colab.research.google.com/)					
2	Apply filter and wrapper method on any standard datasets					
3	* Perform following operations: (Assume suitable data/dataset if needed). I. Write program to read dataset (Text, CSV, JSON, XML) II. Which of the attributes are numeric and which are categorical? III. Performing Data Cleaning, Handling Missing Data, Removing Null data IV. Rescaling Data vs. Encoding Data V. Feature Selection					
4	* Write a program to implement SVM for classification using suitable dataset					
5	Write a program to implement unsupervised machine learning algorithm (Clustering – K Medoid) in python on dataset to cluster data. (Assume suitable dataset)					

6	* Write a program to implement Association Rule Learning (Apriori Algorithm) on any dataset					
7	Write a program to implement Association Rule Learning (Eclat Algorithm) on any dataset					
8	* Write a program to implement AND Logic Gate with 2-bit Binary Input using Perceptron algorithm					
9	* Write a program to implement Perceptron Learning in Python using Iris flower dataset					
10	Write a program to implement Gradient Descent in PyTorch					
11	Write a program to implement /Simulate Back propagation/ feed forward neural network					
12	Build a small CNN model consisting of 5 convolution layers					
13	* Write a program to implement CIFAR 10- CNN using PyTorch					
14	Basic Time Series Forecasting with LSTM					
15	* Classification of images using imagenet dataset					
	Total marks :					

These marks are to be transferred in pro-forma published by MSBTE.

- '*' Marked Practical (LLOs) are mandatory.
- Minimum 80% of above list of lab experiment are to be performed.
- Judicial mix of LLOs are to be performed to achieve desired outcomes.

Practical No. 1

***a. Installation of Tools and Libraries (Jupyter Notebook/Matplotlib/Numpy/Pandas/ PyTorch/scikit-learn)**
b. Use of google colab (https://colab.research.google.com/)

I. Practical Significance:

Python acts as the bridge between AI/ML algorithms and real-world applications. Its libraries and frameworks allow developers to implement, test, and deploy AI and ML solutions efficiently, making it the most popular programming language in the AI/ML field.
Jupyter Notebook is an open-source tool used to write and share documents that combine code, visualizations, equations, and explanations, making it ideal for interactive data analysis. Matplotlib is a Python library used to create various types of visual charts and graphs for data visualization. NumPy provides fast multi-dimensional arrays and mathematical functions for numerical computing. Pandas offers powerful data structures like DataFrames to clean, organize, and analyze structured data efficiently. PyTorch is a deep learning library used to build and train neural networks, especially for computer vision and NLP tasks. Scikit-learn is a machine learning library that provides algorithms and tools for tasks such as classification, regression, clustering, and model evaluation.

II. Industry/Employer expected outcome(s):

- Install Jupyter Notebook and appropriate Python libraries for various AI and ML tasks.

III. Course Level Learning Outcome (COs):

CO1 – Apply suitable Machine Learning Model for dataset feature extraction.

IV. Laboratory Learning Outcome (LLO):

LLO 1.1 - Install required platform to use Python, PyTorch, Scikit-learn library.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

A. Installation of Tools and Libraries:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. Jupyter has support for over 40 different programming languages and Python is one of them.

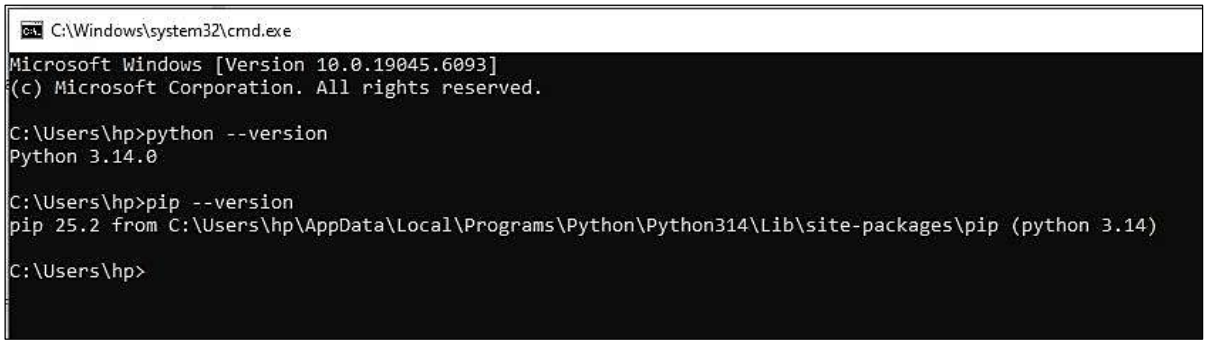
- **Installing Jupyter Notebook:**

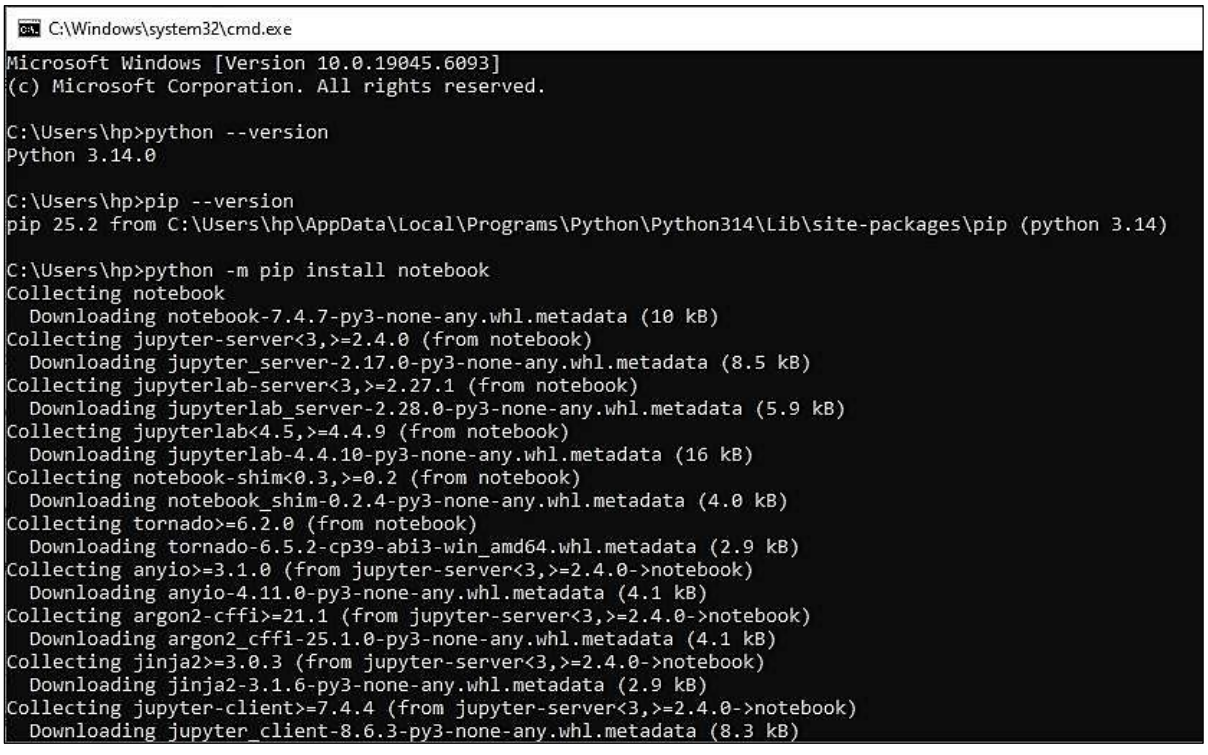
1. Using Anaconda

To install Jupyter Notebook using Anaconda: Download and install the latest Python 3 version of Anaconda. It includes Jupyter Notebook, Python, and other essential packages by default, making it an easy and recommended option.

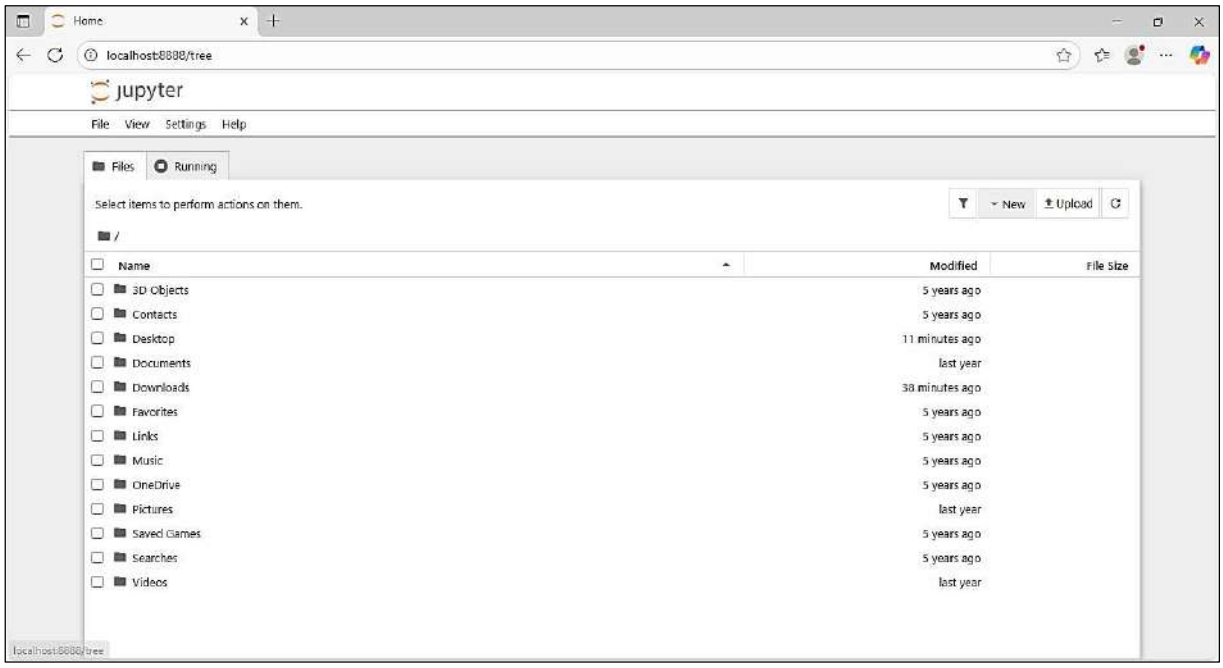
2. Using pip

a. Install pip if not installed otherwise it comes with the latest version of Python.
b. Set the path for python.exe and pip.exe in environment variable to use Python and pip.
c. Verify the installation of Python and pip by using **python --version** and **pip --version**.
d. Install Jupyter notebook by using **python –m pip install notebook** or **pip install notebook.**

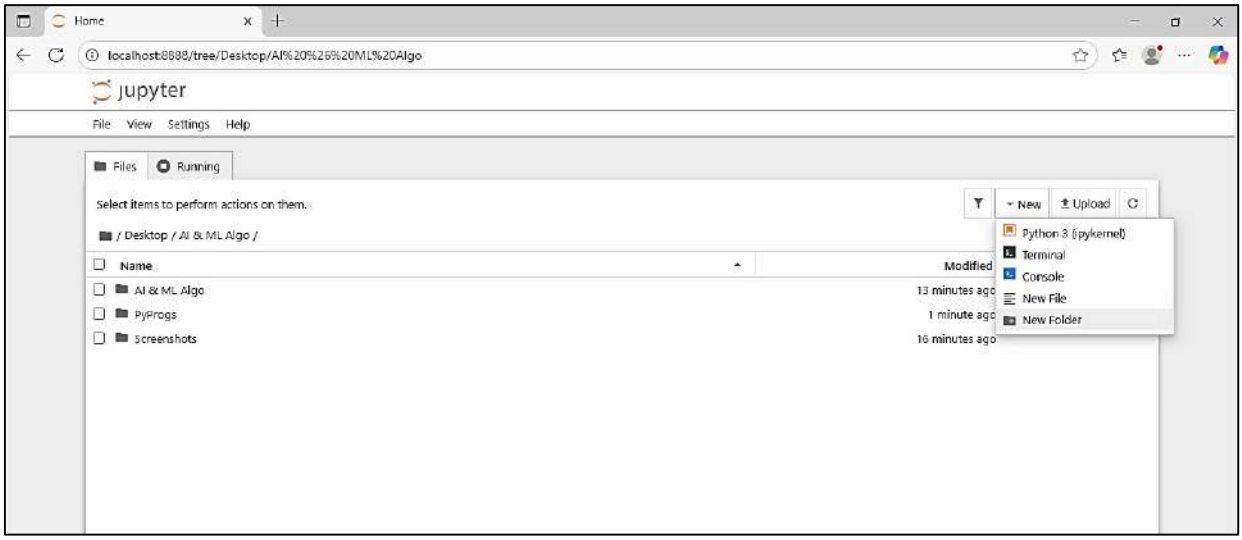




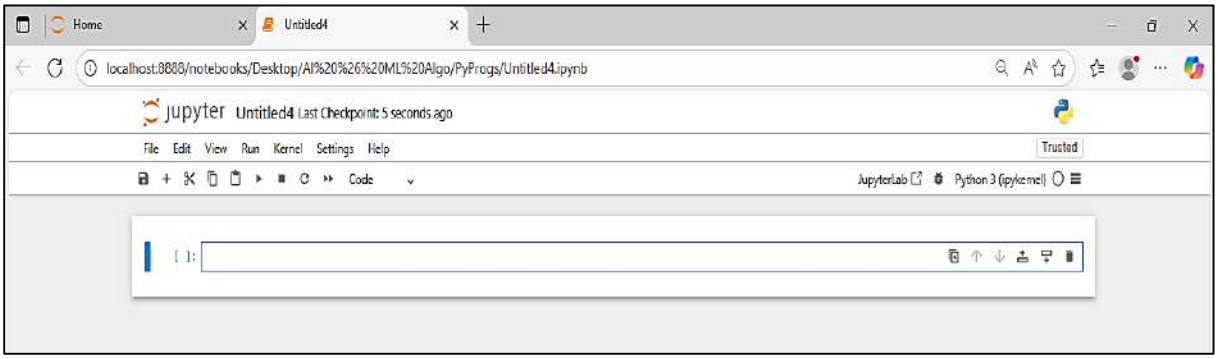
e. To launch Jupyter Notebook use **jupyter notebook** command which will open the browser as follows:

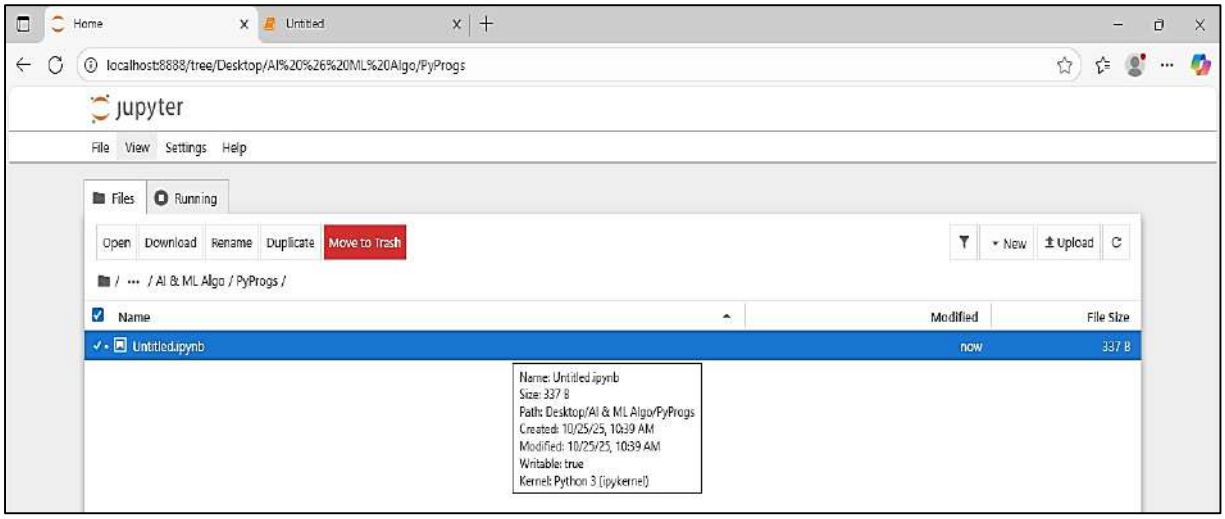


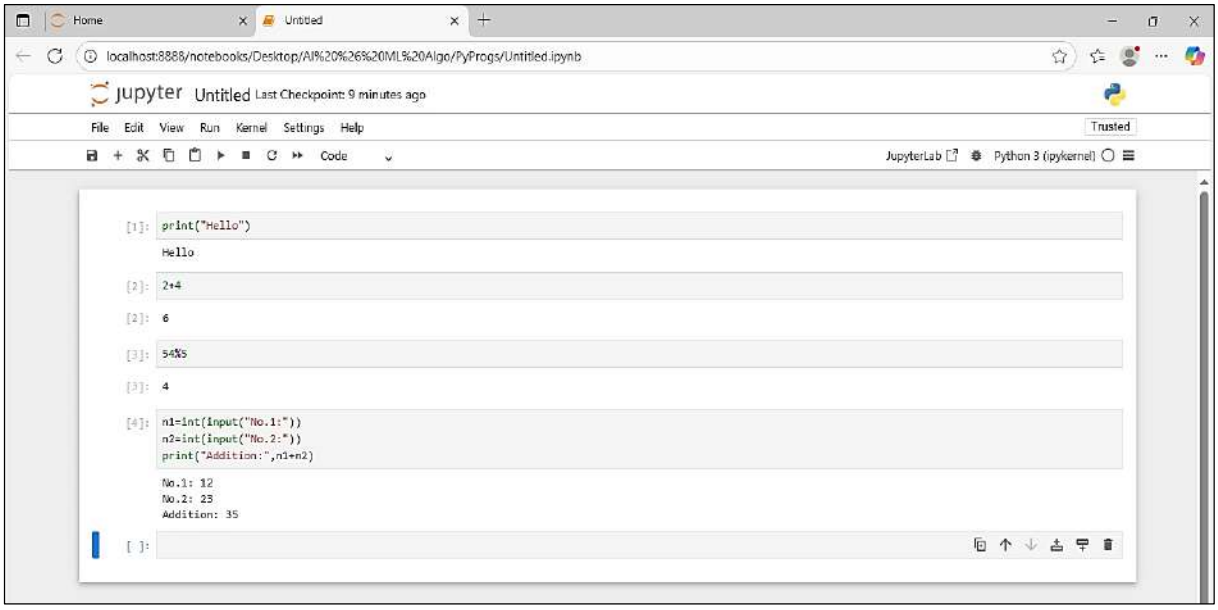
f. You can create a new folder at a particular location by exploring the folders. Click on **New** button at top right and select **New Folder.**



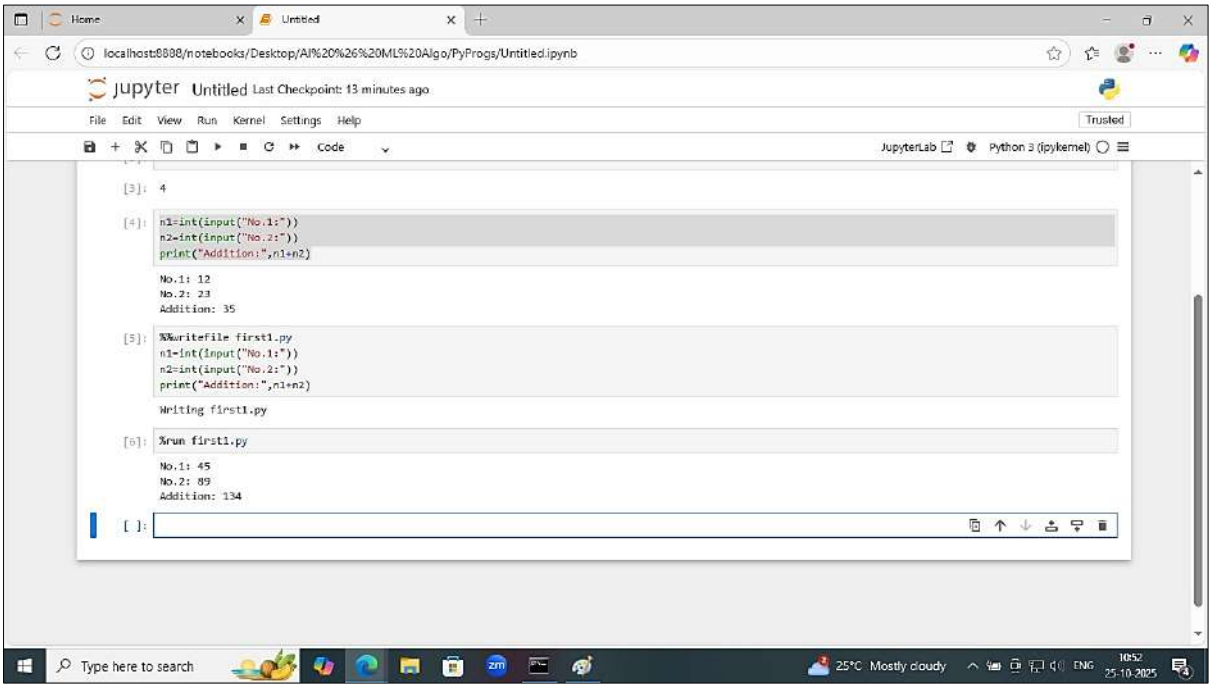
g. To create new Notebook, click on **New** button and **Python 3 (pykernel)** which will open the new tab with a cell where you can run and save Python code or script.

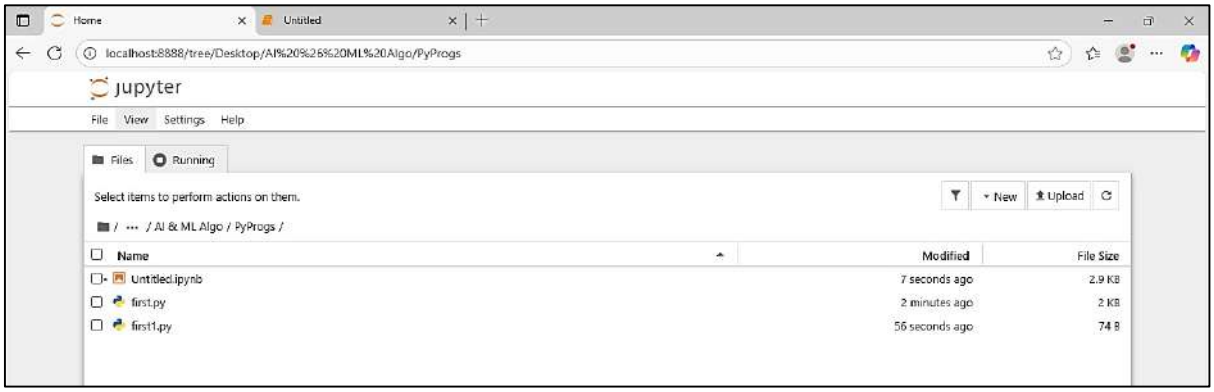

Home
Untitled4
localhost:8888/notebooks/Desktop/AI%20%26%20ML%20Algo/PyProgs/Untitled4.ipynb
jupyter Untitled4 Last Checkpoint: 5 seconds ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[ ]:


Home
Untitled
localhost:8888/tree/Desktop/AI%20%26%20ML%20Algo/PyProgs
jupyter
File View Settings Help
Files
Running
Open Download Rename Duplicate Move to Trash
New
Upload
/ AI & ML Algo / PyProgs /
Name
Modified
File Size
Untitled.ipynb
now
337 B
Name: Untitled.ipynb
Size: 337 B
Path: Desktop/AI & ML Algo/PyProgs
Created: 10/25/25, 10:39 AM
Modified: 10/25/25, 10:39 AM
Writable: true
Kernel: Python 3 (ipykernel)


Home
Untitled
localhost:8888/notebooks/Desktop/AI%20%26%20ML%20Algo/PyProgs/Untitled.ipynb
jupyter Untitled Last Checkpoint: 9 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[1]: print("Hello")
Hello
[2]: 2+4
[2]: 6
[3]: 54%5
[3]: 4
[4]: n1=int(input("No.1:"))
n2=int(input("No.2:"))
print("Addition:",n1+n2)
No.1: 12
No.2: 23
Addition: 35
[ ]:

h. Use **%%writefile scriptname.py** to create a python file and **%run scriptname.py** to execute it.





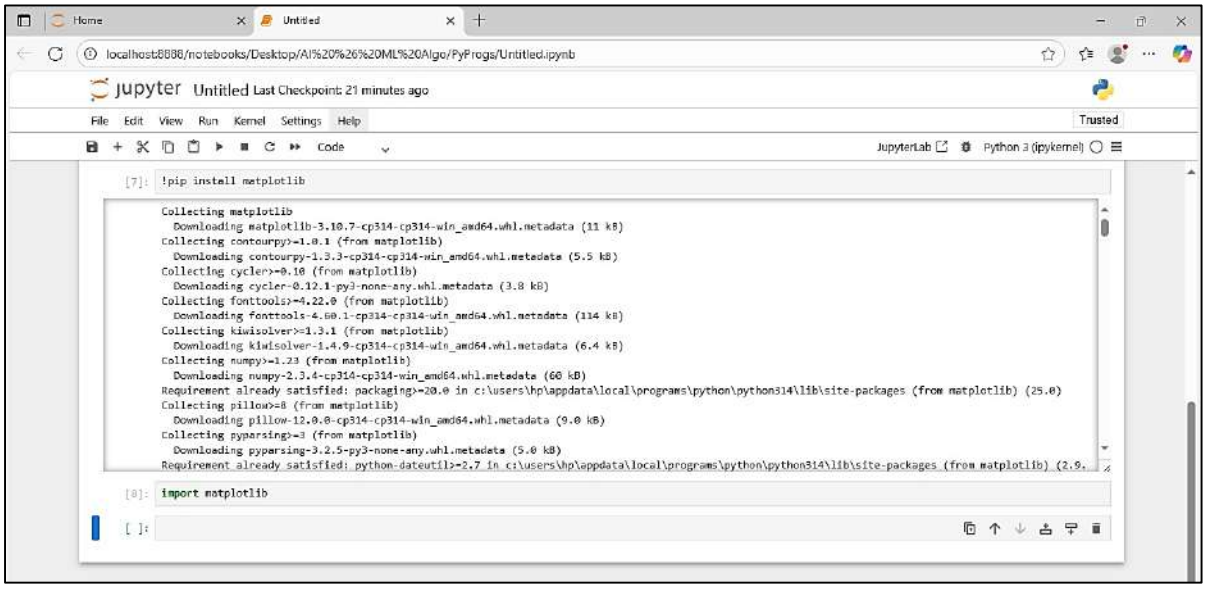
- **Installing Libraries:**

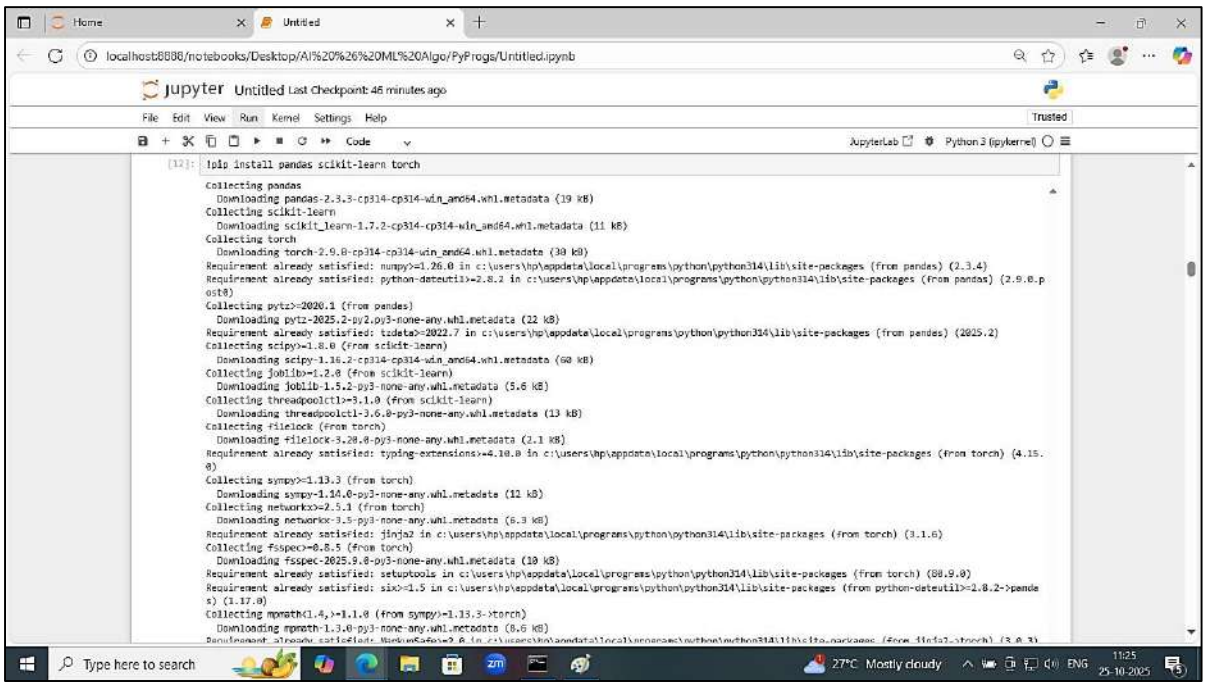
  To install Matplotlib, Numpy, Pandas, PyTorch and scikit-learn libraries, use below commands:

  a. Use **!pip install matplotlib numpy pandas torch scikit-learn** to install all libraries at once or you can use following syntax for individual installation:

  !pip install library_name

  b. Verify all above libraries by either importing them one by one or executing sample programs in which each library is imported using import statement.



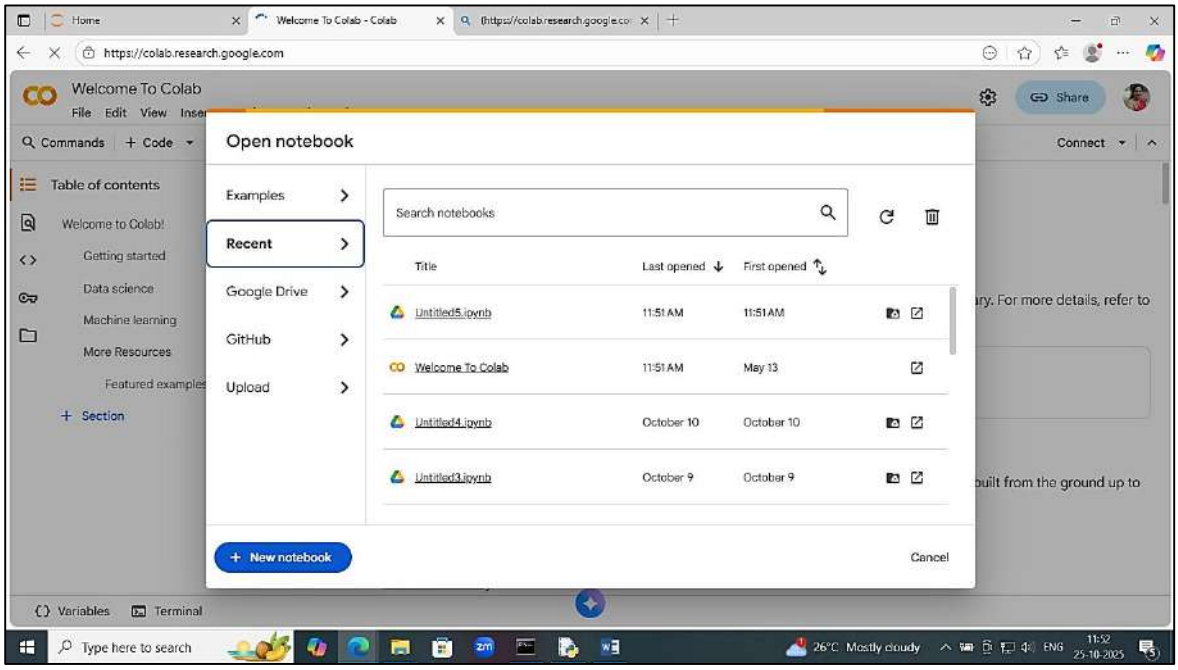


B. Use of Google Colab:

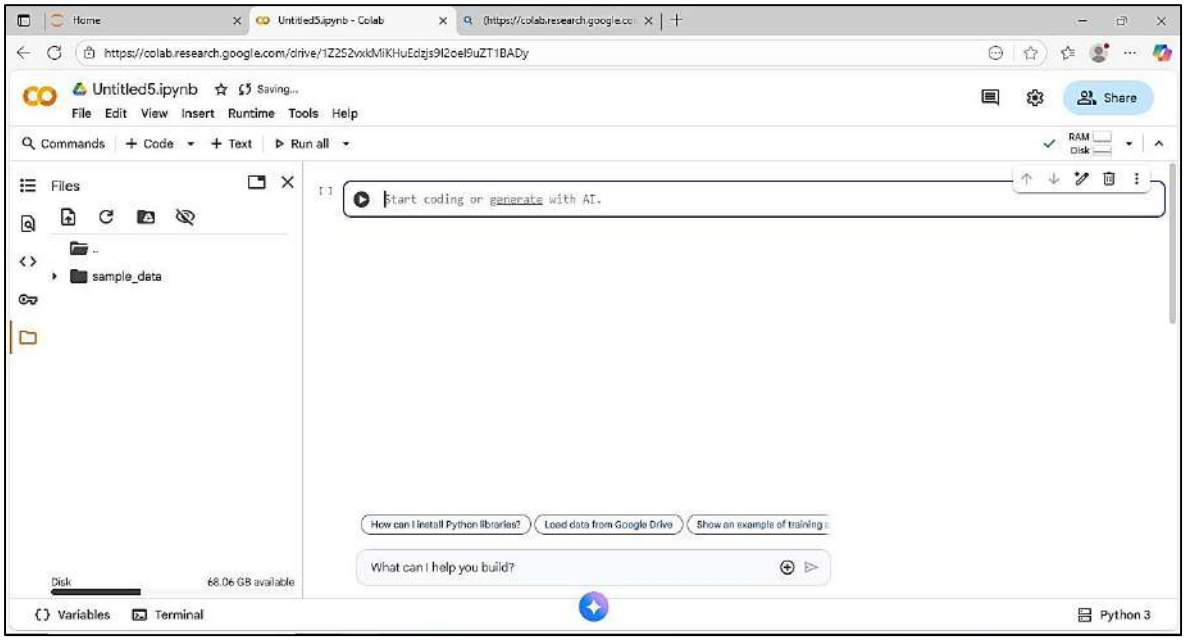
Google Colab, short for "Colaboratory," is a cloud-based platform that allows users to write and execute Python code directly in their web browsers. It provides an interactive environment called a Colab notebook, which combines executable code, rich text, images, HTML, LaTeX, and more in a single document.

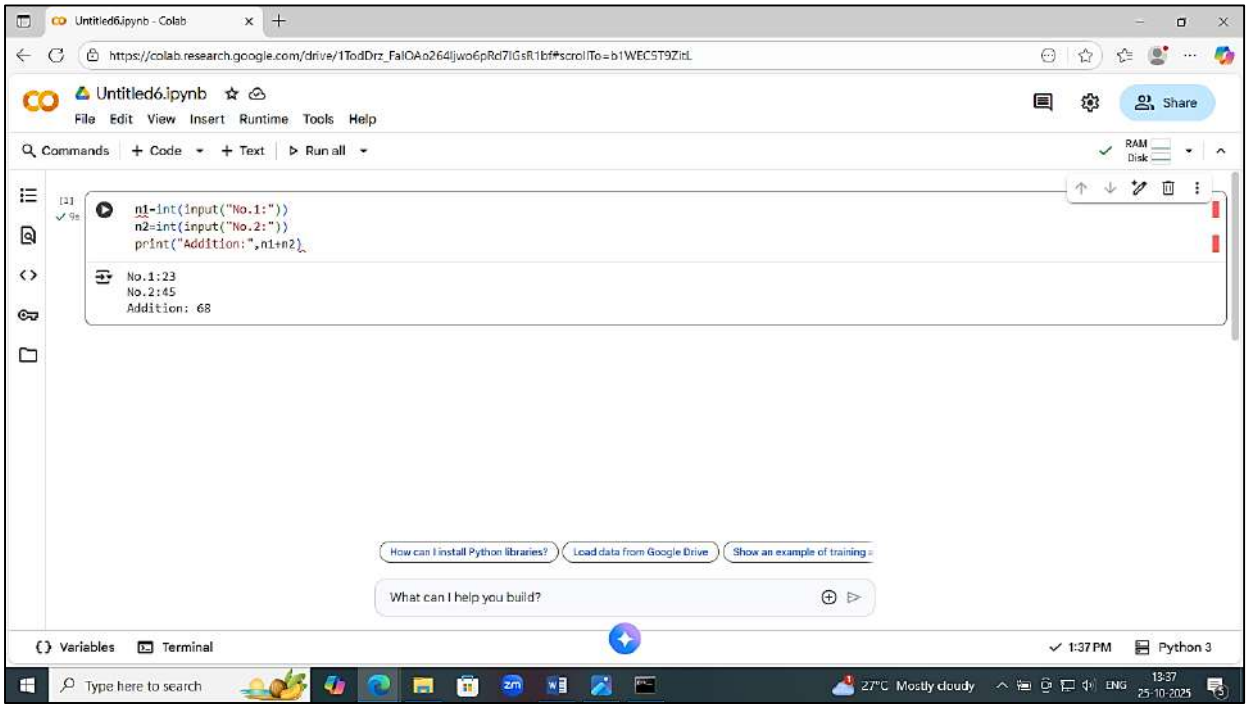
Colab is particularly popular among students, data scientists, and AI researchers. It supports various applications, including data analysis, visualization, and machine learning. For example, you can import datasets, train models, and evaluate them, all within a few lines of code.

a. To use google colab use the URL: **https://colab.research.google.com/ this will open following screen.**



b. Click on **New notebook** to open new notebook like Jupyter and start writing Python code. To run it, click on **Run all and** see the output at bottom.





VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Write steps to install Jupyter Notebook on Windows.
2. Print the version of Python, pip and libraries that you have installed.
3. List key features of Jupyter Notebook and Google Colab
4. What is the use of scikit learn library?
5. What are the Prerequisites for Installing PyTorch?
6. What are the applications of various tools in Python?
7. How to use Google Colab? Write its 2 features.

Space for Answers

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://jupyter.org/install
- https://www.python.org/downloads/
- https://pytorch.org/features/
- https://www.geeksforgeeks.org/installation-guide/install-jupyter-notebook-in-windows/

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 2

Apply filter and wrapper methods on any standard datasets

I. Practical Significance:

Applying filter and wrapper methods on standard datasets is important because these techniques help identify and select only the most relevant features that improve model performance. By removing irrelevant, noisy, or redundant data, feature selection makes machine learning models faster to train, easier to interpret, and more accurate in their predictions. Filter methods quickly eliminate unimportant features using statistical measures, while wrapper methods test different feature combinations with a learning model to find the best subset for maximum accuracy. Using these methods on standard datasets allows us to clearly observe the impact of feature selection, making it easier to build efficient and reliable machine learning models.

II. Industry/Employer expected outcome(s):

- Select the most relevant features to build faster, more accurate, and efficient machine learning models for real-world industry applications.

III. Course Level Learning Outcome (COs):

CO1 – Apply suitable Machine Learning Model for dataset feature extraction.

IV. Laboratory Learning Outcome (LLO):

LLO 2.1 Implement filter and wrapper method on dataset.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

Feature selection is an important preprocessing step in machine learning that focuses on selecting a subset of the most relevant features for model building. It helps improve model accuracy, reduce overfitting, and shorten computation time. The Filter method is based on statistical measures such as correlation, Chi-square, or ANOVA to rank features according to their relationship with the target variable. Since it evaluates each feature independently of any learning algorithm, it is fast, scalable, and suitable for high-dimensional datasets.

In contrast, the Wrapper method evaluates feature subsets by training and testing a model repeatedly, using techniques like Recursive Feature Elimination (RFE) or Sequential Feature Selection. It considers the interaction between features and the model's performance, which often leads to higher accuracy, but at a higher computational cost. Together, these methods provide a balance between efficiency and performance, helping to identify the most meaningful features before model training.

A. Filter Methods

Filter methods evaluate the relevance of features by examining their intrinsic properties independently of any predictive model. This makes them highly scalable and general-purpose.

- **Common Filter Techniques**

1. Variance Thresholding

Features with low variance across samples contain less information and can be removed.

$$\text{Var}(X_j) = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2$$

Remove X_j if $\text{Var}(X_j) < \theta$.

Syntax:

```
from sklearn.feature_selection import VarianceThreshold
selector = VarianceThreshold(threshold=value)
X_new = selector.fit_transform(X)
```

Where,

threshold = *Removes all features whose variance is less than this value (default = 0.0)*

X = *original feature matrix*

X_new = *feature matrix after feature selection*

2. Correlation Coefficient

Measures linear relationship between a feature and the target. For continuous y, use Pearson correlation:

$$\rho_{X_j,y} = \frac{\text{Cov}(X_j,y)}{\sigma_{X_j}\sigma_y}$$

Drop features with low absolute correlation $|\rho| < \theta$.

Syntax:

```
import numpy as np
import pandas as pd
# Correlation Matrix
corr_matrix = df.corr()
# Select upper triangle of the correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
# Drop features with correlation above threshold
to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
# Create new dataset
df_new = df.drop(columns=to_drop)
```

3. Chi-Squared Test (χ^2)

For categorical targets and categorical features, the Chi-squared test assesses dependence:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where O = observed frequency, E = expected frequency.

Syntax:

```
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(score_func=chi2, k=value)
X_new = selector.fit_transform(X, y)
```

Where,

k=value will Choose top `k` features

X = input features (must be non-negative for χ^2)

y = target labels

4. Mutual Information (MI)

Measures non-linear dependencies between variables. Mutual information between feature Xj and target y is:

$$I(X_j; y) = \sum_{x_j} \sum_{y} p(x_j, y) \log \left(\frac{p(x_j, y)}{p(x_j)p(y)} \right)$$

Syntax:

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif
selector = SelectKBest(score_func=mutual_info_classif, k=value)
X_new = selector.fit_transform(X, y)
```

5. F-test (ANOVA)

Used for classification problems with continuous features and categorical targets. It measures if means across groups are significantly different.

$$F = \frac{\text{Between-group variability}}{\text{Within-group variability}}$$

Syntax:

```
from sklearn.feature_selection import SelectKBest, f_classif
selector = SelectKBest(score_func=f_classif, k=value)
X_new = selector.fit_transform(X, y)
```

Where,

X = Input feature matrix

Y = Target variable (labels)

K = value = Number of best features to select

threshold=value = Minimum allowed variance

B. Wrapper Methods

How Do Wrapper Methods Work?

i. Select a subset of features.
ii. Train a model using those features.
iii. Evaluate model performance (e.g., accuracy, precision, RMSE).
iv. Repeat the process with different combinations of features.
v. Choose the subset that results in the best model performance.

1. **Recursive Feature Elimination (RFE)**
   Used to rank and remove the least important features step-by-step until only the desired number of best features remain.
   Syntax:

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
selector = RFE(estimator=model, n_features_to_select=value)
X_new = selector.fit_transform(X, y)
```

2. **Sequential Forward Selection (SFS)**
   Used to start with zero features and keep adding the most useful feature one-by-one based on model performance.
   Syntax:

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
selector = SFS(model, k_features=value, forward=True)
X_new = selector.fit_transform(X, y)
```

3. **Sequential Backward Selection (SBS) – Use**
   Used to start with all features and remove the least useful feature one-by-one.
   Best when you want to simplify the model and remove noise features that reduce accuracy.
   Syntax:

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
selector = SFS(model, k_features=value, forward=False)
X_new = selector.fit_transform(X, y)
```

   Where:
   X = Input feature matrix, y = Target labels
   Value = Number of features to select

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Explain Filter and Wrapper methods in feature selection. Compare them based on working, accuracy, and computation time.
2. What is a Wrapper Method? Write and explain any two Wrapper techniques used for feature selection with examples.
3. Why is feature selection important in Machine Learning? Explain how Filter and Wrapper methods help in improving model performance.
4. Perform feature selection using one Filter Method and one Wrapper Method on the same dataset. Compare the results and write your conclusion about which method is better and why.
5. Using Python and scikit-learn, apply Chi-Square (χ^2) feature selection on a classification dataset and display the selected top-k features. Write the output and inference.

Space for Answers

_____

_____

_____

_____

_____

_____

_____

_____

_____

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.geeksforgeeks.org/machine-learning/wrapper-methods-feature-selection/
- https://www.geeksforgeeks.org/machine-learning/feature-selection-filter-methods/
- https://www.blog.trainindata.com/feature-selection-with-wrapper-methods/

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 3

***Perform following operations: (Assume suitable data/dataset if needed).**

I. **Write program to read dataset (Text, CSV, JSON, XML)**
II. **Which of the attributes are numeric and which are categorical?**
III. **Performing Data Cleaning, Handling Missing Data, Removing Null data**
IV. **Rescaling Data vs. Encoding Data**
V. **Feature Selection**

I. Practical Significance:

Datasets are essential for data analysis, machine learning, and other data-driven processes. A dataset is a well-structured collection of data, usually represented as a table with rows and columns, where each row corresponds to a particular record and each column to a variable. Dataset attributes are classified as numeric or categorical. Numeric attributes represent measurable quantities and can be either continuous or discrete. Categorical attributes describe qualitative characteristics and are of two types: nominal and ordinal. In machine learning (ML), data cleaning is the process of locating and removing any unnecessary, duplicate, or missing data.

Feature selection is an important step in data preparation that helps choose only the most useful features from a dataset. It removes unnecessary or repetitive data, making the model simpler, faster, and more accurate. By focusing on key features, it also reduces overfitting and makes the model easier to understand.

II. Industry/Employer expected outcome(s):

- Handle, preprocess, transform, and prepare real-world datasets for analytics and machine learning applications.

III. Course Level Learning Outcome (COs):

CO1 - Apply suitable Machine learning model for dataset feature extraction.

IV. Laboratory Learning Outcome (LLO):

LLO 3.1 Implement program for Data Preprocessing Techniques.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Reading Text file**

```
file = open("data.txt", "r")
content = file.read()
print(content)
file.close()
```

- **Using Pandas library**

```
import pandas as pd
# Read text file with space-separated values
df = pd.read_csv("data.txt", sep=" ")
print(df)
```

- **Reading CSV file**

  a. Using csv Module

  To read a CSV file, you must create a csv.reader object, which lets you iterate over lines in the CSV file. The csv module comes with Python, so you can import it without having to first install it. Place data.csv in the current working directory, then write the following code :

```
import csv
data_file = open('data.csv')
data_reader = csv.reader(data_file)
data = list(data_reader)
print(data)
```

  b. Using Pandas library

```
import pandas as pd
# read csv file into a DataFrame
df = pd.read_csv('data.csv')
# display DataFrame
print(df)
```

- **JSON file**

  JSON (JavaScript Object Notation) is a text format used to store data as key–value pairs.

  To create a JSON file, open any text editor, type your data in a valid JSON format (key-value pairs within curly braces or arrays within square brackets), and save the file with a .json extension.

  Reading JSON file

  Using json Module

```
import json
# Open the JSON file
with open('data.json', 'r') as file:
    data = json.load(file)
# Display the data
print(data)
```

  To read json file using Pandas

```
import pandas as pd
df = pd.read_json('first.json')
print(df)
```

  Reading xml file

```
with open('data.xml', 'r', encoding='utf-8') as file:
    xml_data = file.read()
print(xml_data)  # shows raw XML content
```

To read xml using Pandas

```
import pandas as pd
df = pd.read_xml('data.xml')
print(df.head())
```

- **Attribute Types:** The attributes in the dataset were categorized into two types:
  1. **Numeric Attributes**: Variables that are quantitative or can be expressed as numbers. Numerical data can be of two types:
     i. **Discrete Data**: Countable numerical data are discrete data.
     ii. **Continuous Data**: This is an uncountable data type for numbers.
  2. **Categorical Attributes**: Variables that represent qualities or characteristics and can be divided into groups or categories.
     There are two primary categories of categorical data:
     i. **Nominal data**: This is the data category that names or labels its categories.
     ii. **Ordinary data**: Elements with rankings, orders, or rating scales are included in this category of categorical data.

```
import pandas as pd
# Read text file with space-separated values
df = pd.read_csv("data.csv", sep=" ")
print(df)
# Identify data types
numeric_cols = df.select_dtypes(include=['number']).columns.tolist()
categorical_cols = df.select_dtypes(include=['object', 'category']).columns.tolist()
print(f"\nNumeric attributes: {numeric_cols}")
print(f"Categorical attributes: {categorical_cols}")
```

- **Data Preprocessing**

  Handling Missing Data: Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset.

  .**isnull()**: Identifies missing values in a Series or DataFrame.

  .isna(): returns True for missing data and False for valid data.

  dropna(): Removes rows or columns with missing values with customizable options for axis and threshold.

  fillna(): Fills missing values with a specified value (like mean, median) or method (forward/backward fill).

  replace(): Replaces specified values in the DataFrame, useful for correcting or standardizing data.

  drop_duplicates(): Removes duplicate rows based on specified columns.

  unique(): Finds unique values in a Series or DataFrame.

- **Rescaling and Encoding**: To prepare the data for analysis and modeling, rescaling and encoding were performed:
  1. **Rescaling Data**: Numeric attributes were rescaled using min-max scaling to bring them within a common range, ensuring fair comparison between attributes.
  2. **Encoding Data**: Categorical attributes were encoded using one-hot encoding to convert them into a numerical format suitable for machine learning algorithms.

- **Feature Selection**
  Feature selection is the process of identifying and selecting a subset of relevant features for use in model construction. The goal is to enhance the model's performance by reducing overfitting, improving accuracy, and reducing training time. Feature selection methods have been traditionally grouped into filter methods, wrapper methods, and embedded methods.
  1. **Filter methods** select the best features based on the feature characteristics, ignoring their interaction with the machine learning model. They rank the features and then select the top-ranking ones. Ranking methods normally use statistical tests like chi-square, ANOVA, correlation, and mutual information.
  2. **Wrapper methods** wrap the search for the most relevant features around a predictive model. They generate multiple feature subsets and then evaluate their performance based on the classifier or regression model. The selected features are those from the subset that returned the best performing model.
  3. **Embedded methods** "embed" the selection procedure in the training of the predictive model. Lasso and feature importance from decision trees are the classical examples of embedded methods. The coefficients of linear models can also be used to select important features.

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
5.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
6.	Operating System	Windows 10 or later		
7.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
8.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Explain how to read dataset in different formats (text, csv, JSON, XML) using pandas library.
2. Write difference between numeric and categorical attributes in a dataset.
3. How to identify which columns are numeric and which are categorical using pandas?
4. What is data cleaning in machine learning? Discuss various techniques employed to handle missing data within datasets.
5. Differentiate between rescaling and encoding in the context of data preprocessing.
6. What is the significance of feature selection in machine learning?
7. Discuss various techniques used for feature selection.

Space for Answers

XI. References/Suggestions for further reading

- https://www.appliedaicourse.com/blog/depth-first-search-in-artificial-intelligence/
- https://favtutor.com/blogs/depth-first-search-python
- https://www.tutorialspoint.com/graph_theory/graph_theory_depth_first_search.htm
- https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
- https://www.educba.com/dfs-algorithm/
- https://youtu.be/f8luGFRtshY

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 4

*** Write a program to implement SVM for classification using suitable dataset**

I. Practical Significance:

Classification may be defined as the process of predicting class or category from observed values or given data points. The categorized output can have the form such as "Black" or "White" or "spam" or "no spam". Classification in machine learning is a supervised learning technique where an algorithm is trained with labeled data to predict the category of new data. Mathematically, classification is the task of approximating a mapping function (f) from input variables (X) to output variables (Y). It is basically belonging to the supervised machine learning in which targets are also provided along with the input data set. An example of classification problem can be the spam detection in emails. There can be only two categories of output, "spam" and "no spam"; hence this is a binary type classification. To implement this classification, we first need to train the classifier. For this example, "spam" and "no spam" emails would be used as the training data. After successfully train the classifier, it can be used to detect an unknown email.

Support Vector Machines (SVMs) are powerful yet flexible supervised machine learning algorithm which is used for both classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990 also. SVMs have their unique way of implementation as compared to other machine learning algorithms. Now a day, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

II. Industry/Employer expected outcome(s):

- Apply classification solutions to real-world problems.

III. Course Level Learning Outcome (COs):

CO2 - Implement Machine learning algorithms on given problem.

IV. Laboratory Learning Outcome (LLO):

LLO 4.1 Implement SVM for classification using dataset.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Working of SVM**

The goal of SVM is to find a hyperplane that separates the data points into different classes. A hyperplane is a line in 2D space, a plane in 3D space, or a higher-dimensional surface in n-dimensional space. The hyperplane is chosen in such a way that it maximizes the margin, which is

the distance between the hyperplane and the closest data points of each class. The closest data points are called the support vectors.

The distance between the hyperplane and a data point "x" can be calculated using the formula:

distance = (w. x + b) / ||w||

where "w" is the weight vector, "b" is the bias term, and "||w||" is the Euclidean norm of the weight vector. The weight vector "w" is perpendicular to the hyperplane and determines its orientation, while the bias term "b" determines its position.

The optimal hyperplane is found by solving an optimization problem, which is to maximize the margin subject to the constraint that all data points are correctly classified. In other words, we want to find the hyperplane that maximizes the margin between the two classes while ensuring that no data point is misclassified. This is a convex optimization problem that can be solved using quadratic programming.

If the data points are not linearly separable, we can use a technique called kernel trick to map the data points into a higher-dimensional space where they become separable. The kernel function computes the inner product between the mapped data points without computing the mapping itself. This allows us to work with the data points in the higher dimensional space without incurring the computational cost of mapping them.

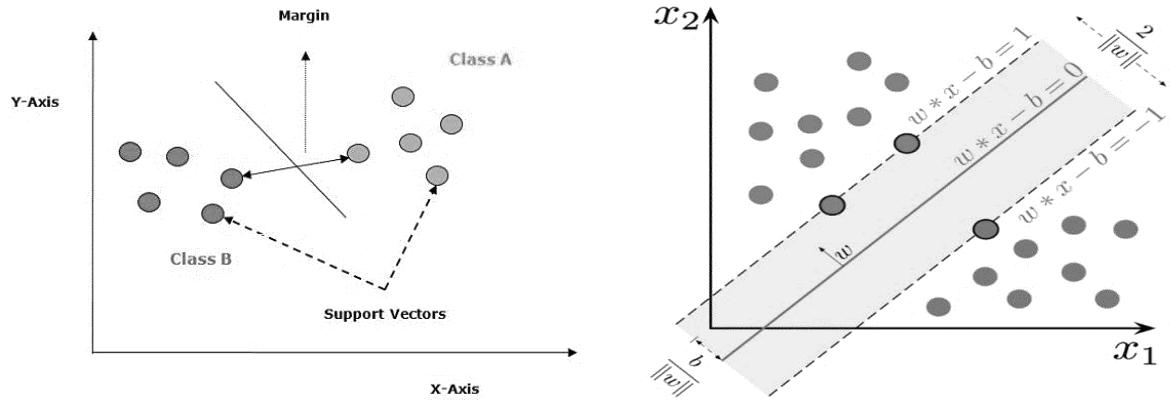


Fig.4.1: Support Vector Machine and Support Vector Margin

- **Implementing SVM Using Python**

  For implementing SVM in Python we will start with the standard libraries import as follows:

  Step 1: Import Required Libraries

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

  These libraries help us load data, split it, scale features, create an SVM model, and check accuracy.

  Step 2: Load a Sample Dataset

  Here we will use the Iris dataset (already available in sklearn).

```
iris = datasets.load_iris()
X = iris.data      # features
y = iris.target    # target or class
```

Step 3: Split Data into Training and Testing Sets

We divide data into two parts training and testing.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

70% data is used for training and 30% for testing.

Step 4: Standardize (Scale) the Data

SVM works better when features are on the same scale.

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

Step 5: Create and Train the SVM Model

```
model = SVC(kernel='linear')   # linear kernel
model.fit(X_train, y_train)
```

kernel='linear' means we are using a straight line (or plane) to separate data.

Step 6: Make Predictions or print confusion matrix

```
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

The model predicts class labels for unseen test data.

Step 7: Check Accuracy, Confusion Matrix, Classification Report, Support Vector for class and Predicted Probabilities

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nSupport Vectors:\n", model.support_vectors_)
print("\nNumber of Support Vectors for each class:", model.n_support_)
print("\nPredicted Probabilities:\n", model.predict_proba(X_test))
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. What are Support Vectors? How are they useful in SVM?
2. Differentiate between Linear SVM and Non-linear SVM.
3. List and explain any two types of kernel functions used in SVM.
4. What is the purpose of the kernel parameter in SVC()? Give an example.
5. List and explain the outputs obtained after executing an SVM program.
6. Write the step-by-step procedure to train and test an SVM model using sklearn.
7. Interpret the meaning of confusion matrix and accuracy score in SVM classification.

Space for Answers

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.techtarget.com/whatis/definition/support-vector-machine-SVM
- https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python
- https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/
- https://www.tutorialspoint.com/machine_learning/machine_learning_support_vector_machine
- https://medium.com/@RobuRishabh/support-vector-machines-svm-27cd45b74fbb

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 5

Write a program to implement unsupervised machine learning algorithm (Clustering – K Medoid) in python on dataset to cluster data. (Assume suitable dataset)

I. Practical Significance:

The K-Medoids clustering algorithm is useful because it creates robust and interpretable clusters using actual data points as centers, making it less vulnerable to outliers and noise than K-Means. Its versatility with distance measurements enables it to handle a wide range of data types, including categorical and mixed data. K-Medoids is widely used in industries for customer segmentation, fraud detection, healthcare analysis, and pattern recognition. It assists enterprises in identifying representative data points and making reliable, data-driven decisions even with faulty datasets.

II. Industry/Employer expected outcome(s):

- Understand the concept and working of the K-Medoids clustering algorithm, apply it to real datasets, and analyze cluster quality.

III. Course Level Learning Outcome (COs):

CO2 - Implement Machine learning algorithms on given problem.

IV. Laboratory Learning Outcome (LLO):

LLO 5.1 Implement unsupervised machine learning algorithm.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

K-Medoids clustering is an unsupervised machine learning algorithm used to group data into different clusters. The k-medoids algorithm is a clustering approach related to k-means clustering for partitioning a data set into k groups or clusters. In k-medoids clustering, each cluster is represented by one of the data point in the cluster. These points are named cluster medoids. The K-Medoids clustering is called a partitioning clustering algorithm. The most popular implementation of K-medoids clustering is the Partitioning around Medoids (PAM) clustering.

K-Medoids is an iterative clustering algorithm that begins by selecting k actual data points, called medoids, as the initial cluster centers. The distance between each data point and every medoid is calculated, and each point is then assigned to the cluster of the nearest medoid. Next, the algorithm computes the total cost, which is the sum of distances between all data points and their respective medoids. After this, a non-medoid point is randomly selected and swapped with one of the medoids. The cost is recalculated for this new configuration. If the new cost is lower, the swap is accepted; otherwise, it is undone. This process continues until the cost can no longer be reduced. In the end,

the medoids represent the most centrally located and best representative points of their respective clusters.

- **K-Medoids Clustering – Algorithm**

1. First, select K random data points from the dataset and use them as medoids.
2. Now, calculate the distance of each data point from the medoids. Use any of the Euclidean, Manhattan distance, or squared Euclidean distance as the distance measure.
3. Once find the distance of each data point from the medoids, assign the data points to the clusters associated with each medoid. The data points are assigned to the medoids at the closest distance.
4. After determining the clusters, calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. Let the cost be Ci.
5. Now, select a random data point Dj from the dataset and swap it with a medoid Mi. Here, Dj becomes a temporary medoid. After swapping, calculate the distance of all the non-medoid data points to the current medoid of each cluster. Let this cost be Cj.
6. If Ci>Cj, the current medoids with Dj as one of the medoids are made permanent medoids. Otherwise, undo the swap, and Mi is reinstated as the medoid.
7. Repeat 4 to 6 until no change occurs in the clusters.

- **Python Program**

```
import numpy as np
from scipy.spatial.distance import cdist     #Computes distances between sets of points efficiently .
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs   #Generates synthetic datasets for clustering.
def kmedoids(X, k, max_iterations=100):     #X:Input Dataset, k: No of cluster
    n_samples = X.shape[0]
    medoid_indices = np.random.choice(n_samples, k, replace=False)
    medoids = X[medoid_indices]
    for _ in range(max_iterations):
        # Compute distances between all points and medoids
        distances = cdist(X, medoids, metric='euclidean')
        # Assign each point to the nearest medoid
        labels = np.argmin(distances, axis=1)
        new_medoids = np.copy(medoids)
        for i in range(k):
            cluster_points = X[labels == i]
      if len(cluster_points) == 0:
          continue
      # Compute the cost for each point in the cluster to be a medoid
      cost = cdist(cluster_points, cluster_points, metric='euclidean').sum(axis=1)
      # Choose the point with the minimum cost as the new medoid
      new_medoids[i] = cluster_points[np.argmin(cost)]
   # if there is no change in medoids then break and return
```

```
        if np.all(new_medoids == medoids):
            break
        medoids = new_medoids
    return labels, medoids
X, _ = make_blobs(n_samples= 1000, centers=5, cluster_std=0.60, random_state=0)
k = 5
labels, medoids = kmedoids(X, k)
plt.figure(figsize=(6, 4))
scatter = plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(medoids[:, 0], medoids[:, 1], c='red', s=50, marker='*', label='Medoids')
plt.title('K-Medoids Clustering')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.colorbar(scatter, label='Cluster')
plt.show()
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. What is K-Medoids clustering?
2. Differentiate between K- Medoids and K-Means Clustering.
3. Write stepwise procedure of K-Medoids algorithm.
4. What is cost function?
5. Following is the dataset for K-Medoids algorithm.

Point	Coordinates
A1	(2, 6)
A2	(3, 8)
A3	(4, 7)
A4	(6, 2)
A5	(6, 4)
A6	(7, 3)
A7	(7,4)
A8	(8, 5)
A9	(7, 6)
A10	(3, 4)

Perform K-Medoids clustering with k = 2. Assume the initial medoids are A6 and A10. Assign points to clusters, calculate total cost, and update medoids.

Space for Answers

XI. References/Suggestions for further reading

- https://www.geeksforgeeks.org/machine-learning/k-means-vs-k-medoids-clustering/
- https://codinginfinite.com/k-medoids-clustering-algorithm-with-numerical-example/
- https://www.tutorialspoint.com/machine_learning/machine_learning_k_medoids_clustering.htm

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 6

*** Write a program to implement Association Rule Learning (Apriori Algorithm) on any dataset**

I. Practical Significance:

Implementing Association Rule Learning helps in discovering hidden relationships between items in large datasets. It is practically useful for identifying patterns such as products frequently bought together, enabling industries to improve sales strategies, design combo offers, and build recommendation systems. This technique supports data-driven decision-making in fields like retail, e-commerce, banking, and healthcare by revealing meaningful associations that enhance business insights and customer understanding.

The Apriori Algorithm is a data mining algorithm used in Association Rule Learning to find frequent itemsets (groups of items that appear together often) and then generate association rules from them.

II. Industry/Employer expected outcome(s):

- Identify relationships and patterns in large datasets for decision-making and business insights.

III. Course Level Learning Outcome (COs):

CO2 - Implement Machine learning algorithms on given problem.

IV. Laboratory Learning Outcome (LLO):

LLO 6.1 Implement Apriori Algorithm for Association Rule Learning.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Association Rules:**

Association rules are a fundamental concept used to find relationships, correlations or patterns within large sets of data items. They describe how often item sets occur together in transactions and express implications of the form:

$$\mathbf{X \rightarrow Y}$$

Where X and Y are disjoint sets of items. This rule suggests that when items in X appear, items in Y tend to appear as well. Association rules originated from market basket analysis and help zetailers and analysts understand customer behavior by discovering item associations in transaction data. For example, a rule stating

$$\{\text{Bread, Butter}\} \rightarrow \{\text{Milk}\}$$

indicates that customers who buy bread and butter also tend to buy milk.

- **Apriori Algorithm:**

The Apriori Algorithm is one of the main algorithms used to perform Association Rule Learning. It helps to find frequent itemsets (items that appear together often) and then generate association rules from them. Apriori helps to discover relationships between items in a large dataset. for example, which products are often bought together in a supermarket.

It's called "Apriori" Because it uses the "prior knowledge" of smaller frequent itemsets to find larger ones. It works on the Apriori property that is if an itemset is frequent, then all of its subsets must also be frequent.

Example: If {Mobile, Earphones} is frequent, then {Mobile} and {Earphones} must also be frequent.

How it Works (Step-by-Step):

1. Set a minimum support value.
   (e.g., 0.3 → means items appearing in at least 30% of transactions are frequent.)
2. Find all itemsets that meet or exceed that support — called frequent itemsets.
3. Use frequent itemsets to generate association rules like:{Mobile} → {Earphones}
4. Calculate metrics such as:
   i. Support = How often items appear together, Frequency of items appearing together.
   ii. Confidence = How often rule is true, Likelihood that item B is purchased when item A is purchased.
   iii. Lift = Strength of the rule, Strength of the relationship between A and B.
   iv.

- **For Example:**

Transaction ID	Items Purchased
T1	Mobile, Charger
T3	Mobile, Cover, Earphones
T5	Mobile
T7	Mobile, Earphones
T9	Mobile, Charger, Earphones
T2	Mobile, Earphones
T4	Charger, Earphones
T6	Earphones
T8	Cover
T10	Earphones

- **Steps to apply association rule:**

Step 1: Count occurrences

Count(Mobile) = T1, T2, T3, T5, T7, T9 → 6

Count(Earphones) = T2, T3, T4, T6, T7, T9, T10 → 7

Count(Mobile ∧ Earphones) = T2, T3, T7, T9 → 4

Total transactions = 10

Code:

```
transactions = [  {'Mobile','Charger'}, {'Mobile','Earphones'}, {'Mobile','Cover','Earphones'},
  {'Charger','Earphones'}, {'Mobile'}, {'Earphones'}, {'Mobile','Earphones'},
  {'Cover'}, {'Mobile','Charger','Earphones'}, {'Earphones'}]
total = len(transactions)
count_mobile = sum(1 for t in transactions if 'Mobile' in t)
count_earphones = sum(1 for t in transactions if 'Earphones' in t)
count_both = sum(1 for t in transactions if {'Mobile','Earphones'}.issubset(t))
```

Step 2: Calculate Support

$$Support(A \rightarrow B) = \frac{Transactions(A \wedge B)}{TotalTransactions}$$

Support({Mobile, Earphones}) = 4 / 10 = 0.4 (40%)

Code:

```
support = count_both / total
print("Support:", support)
```

Step 3: Calculate Confidence

$$Confidence(A \rightarrow B) = \frac{Support(A \wedge B)}{Support(A)}$$

= 0.4 / 0.6 = 0.67 (67%)

67% of the customers who bought a mobile also bought earphones.

Code:

```
confidence = support / (count_mobile / total)
print("Confidence:{:.2f}".format(confidence))
```

Step 4: Calculate Lift

$$Lift(A \rightarrow B) = \frac{Confidence(A \rightarrow B)}{Support(B)}$$

= 0.67 / 0.7 = 0.96

Code:

```
lift = confidence / (count_earphones / total)
print("Lift:{:.2f}".format(lift))
```

Lift < 1 means the relationship between Mobile and Earphones is slightly weaker than random chance, but still shows a mild buying trend. 40% of all customers bought both mobile and earphones together. When a mobile is bought, there is a 67% chance earphones are also purchased.

The association is moderately positive.

Output:

```
Support: 0.4
Confidence:0.67
Lift:0.95
```

The rule {Mobile} → {Earphones} shows that 40% of all customers buy both together, and 67% of mobile buyers also buy earphones. The lift value (0.96) indicates a moderate positive

relationship between the two items. Thus, the store can plan combo offers or recommend earphones when a mobile is purchased.

- **Steps to use Apriori Algorithm**

 # Import required libraries

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

 # Step 1: Create the dataset

```
dataset = [['Mobile','Charger'], ['Mobile','Earphones'], ['Mobile','Cover','Earphones'],
['Charger','Earphones'],  ['Mobile'], ['Earphones'], ['Mobile','Earphones'], ['Cover'],
['Mobile','Charger','Earphones'], ['Earphones']]
```

 # Step 2: Convert dataset to DataFrame

```
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
```

 # Step 3: Apply Apriori Algorithm

```
frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)
```

 # Step 4: Generate Association Rules

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
```

 # Step 5: Display Results

```
print("Frequent Itemsets:")
print(frequent_itemsets)
print("\nAssociation Rules:")
print(rules[['antecedents','consequents','support','confidence','lift']])
```

Output:

Frequent Itemsets:

	support	itemsets
0	0.3	(Charger)
1	0.7	(Earphones)
2	0.6	(Mobile)
3	0.4	(Earphones, Mobile)

Association Rules:

	antecedents	consequents	support	confidence	lift
0	(Earphones)	(Mobile)	0.4	0.571429	0.952381
1	(Mobile)	(Earphones)	0.4	0.666667	0.952381

Here,

Rule: {Mobile} → {Earphones}
Support = 0.4: 40% of all transactions contain both Mobile and Earphones.
Confidence = 0.67: When Mobile is bought, Earphones are also bought 67% of the time.
Lift = 1.12: Positive association (since > 1).

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Explain the relationship between Association Rule Learning and Apriori Algorithm with a neat diagram.
2. Write a Python program to implement the Apriori Algorithm on a simple dataset and display the generated rules.
3. Using a sample supermarket dataset, generate association rules and interpret the output.
4. Perform market basket analysis using Apriori Algorithm and explain one rule you obtained.
5. Implement Association Rule Learning using “mlxtend” library in Python and explain the meaning of support and confidence values.
6. State and explain Support, Confidence, and Lift used in Association Rule Learning.

Space for Answers

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.ibm.com/think/topics/apriori-algorithm
- https://www.tutorialspoint.com/what-is-association-rule-learning
- https://www.geeksforgeeks.org/machine-learning/association-rule/

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 7

Write a program to implement Association Rule Learning (Eclat Algorithm) on any dataset

I. Practical Significance:

In Association Rule Learning, the Eclat Algorithm is a crucial method for identifying significant connections among objects in huge datasets. Eclat, with its efficiency and ability to discover hidden associations in transaction data, is a valuable tool for data analysis and decision-making in various industries. By understanding the principles and implementation of ECLAT, valuable insights from the datasets are unlocked and make data-driven decisions with confidence.

II. Industry/Employer expected outcome(s):

- Apply Association Rule Learning techniques using the Eclat Algorithm to efficiently analyze large datasets and discover hidden patterns or relationships between items.

III. Course Level Learning Outcome (COs):

CO2 - Implement Machine learning algorithms on given problem.

IV. Laboratory Learning Outcome (LLO):

LLO 7.1 Implement Eclat Algorithm for Association Rule Learning.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

The ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) algorithm is a variation of the Apriori algorithm that uses a top-down approach rather than a bottom-up approach. It works by dividing the items into equivalence classes based on their support (the number of transactions in which they appear). The association rules are then generated by combining these equivalence classes in a lattice-like structure. It is a more efficient and scalable version of the Apriori algorithm.

- In order to avoid the formation of subsets when the minimal support is obtained, the support value for an item is calculated using this method.
- Each item-transaction set is compared to every other pair in the function to produce a new candidate.
- The list of common partners is expanded if this candidate is found to be common. If the frequency set includes a pair of items, the frequent item set also includes its subsets.
    - Obtain the tidlist for each database object. Here, we thoroughly search the database. The list of transactions that contain item “a” is represented by its tidlist.

- Create a new transaction list whose constituents are transactions in which both item a and item b are involved by intersecting their respective transaction lists.
- And repeat these above steps again.

Python Program

```
# Sample dataset (list of transactions)
transactions = [ ['milk', 'bread', 'butter'],
   ['bread', 'butter', 'jam'],
   ['milk', 'bread'],
   ['milk', 'bread', 'butter', 'jam'],
   ['bread', 'jam']  ]
# Minimum support threshold
min_support = 2
# Step 1: Generate item to TID mapping
item_tidset = {}
for tid, items in enumerate(transactions):
   for item in items:
      if item not in item_tidset:
         item_tidset[item] = set()
      item_tidset[item].add(tid)
# Step 2: Function to find frequent itemsets using Eclat
def eclat(prefix, items, min_support, frequent_itemsets):
   for i in range(len(items)):
      item = items[i]
      item_name, tidset = item
      support = len(tidset)
      if support >= min_support:
         new_prefix = prefix + [item_name]
         frequent_itemsets[frozenset(new_prefix)] = support
         # Build new item list by intersecting TID sets
         remaining_items = []
         for j in range(i + 1, len(items)):
            next_item, next_tidset = items[j]
            new_tidset = tidset.intersection(next_tidset)
            if len(new_tidset) >= min_support:
               remaining_items.append((next_item, new_tidset))
         # Recursive call
         eclat(new_prefix, remaining_items, min_support, frequent_itemsets)
# Step 3: Run Eclat Algorithm
frequent_itemsets = {}
items = list(item_tidset.items())
eclat([], items, min_support, frequent_itemsets)
# Step 4: Display results
print("Frequent Itemsets using Eclat Algorithm:\n")
```

```
for itemset, support in sorted(frequent_itemsets.items(), key=lambda x: (-len(x[0]), -x[1])):
    print(f"{set(itemset)} -> support: {support}")
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Consider the following transaction database, which contains five transactions and five items denoted as a, b, c, d, and e:

Transaction id	Items
t1	{a, c, d}
t2	{b, c, e}
t3	{a, b, c, e}
t4	{b, e}
t5	{a, b, c, e}

If minsup = 2, write down all the frequent itemsets and their support.

2. How does the Eclat algorithm make use of a "TID-list"?
3. What are the Eclat algorithm's main advantages over Apriori?
4. In which situations would you prefer the Eclat algorithm to Apriori?
5. Give some examples of how the Eclat Algorithm is used in the real world.

Space for Answers

XI. References/Suggestions for further reading

- https://www.i2tutorials.com/machine-learning-tutorial/machine-learning-eclat-algorithm/
- https://www.datacamp.com/tutorial/association-rule-mining-python
- https://medium.com/@gabrielreversi/association-rules-the-eclat-algorithm-96d47f32f992
- https://codinginfinite.com/eclat-algorithm-numerical-example/#google_vignette
- https://www.youtube.com/watch?v=P5LH5HCrhMU

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 8

*** Write a program to implement AND Logic Gate with 2-bit Binary Input using Perceptron algorithm**

I. Practical Significance:

A neural network link that contains computations to track features and uses Artificial Intelligence in the input data is known as Perceptron. This neural links to the artificial neurons using simple logic gates with binary outputs. An artificial neuron invokes the mathematical function and has node, input, weights, and output equivalent to the cell nucleus, dendrites, synapse, and axon, respectively, compared to a biological neuron.

Implementing an AND Logic Gate using the Perceptron algorithm demonstrates how a simple neural network can perform basic logical decision-making tasks. It shows the practical working of a perceptron, how it learns to classify inputs by adjusting weights and bias. This example helps understand the foundation of Artificial Neural Networks (ANNs) and how complex decision-making systems can be built from simple binary logic. In real-life applications, this forms the basis for designing digital circuits, control systems, and intelligent devices where logical operations are automated using learning-based models.

II. Industry/Employer expected outcome(s):

- Develop a foundation for working with advanced neural networks used in automation and robotics.

III. Course Level Learning Outcome (COs):

CO3 - Implement Artificial Neural Networks analyzing associated parameters of Deep Learning.

IV. Laboratory Learning Outcome (LLO):

LLO 8.1 Implement Perceptron algorithm for AND logic gate.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Perception Algorithm:**

Perceptron is a linear supervised machine learning algorithm. It is used for binary classification. This is a very important binary classifier, the perceptrons, which forms the basis for the most popular machine learning models nowadays – the neural networks.

Perceptron Learning Algorithm is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence. The perceptron learning algorithm is treated as the most straightforward Artificial Neural network. It is a

supervised learning algorithm of binary classifiers. Hence, it is a single-layer neural network with four main parameters, i.e., input values, weights and Bias, net sum, and an activation function.

There are four significant steps in a perceptron learning algorithm:

1. First, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

   $$\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + \ldots + w_n * x_n$$

   Add another essential term called bias 'b' to the weighted sum to improve the model performance.

   $$\sum w_i * x_i + b.$$

2. Next, an activation function is applied to this weighed sum, producing a binary or a continuous-valued output.

   $$Y = f\left(\sum w_i * x_i + b\right)$$

3. Next, the difference between this output and the actual target value is computed to get the error term, E, generally in terms of mean squared error. The steps up to this form the forward propagation part of the algorithm.

   $$E = (Y - Y_{actual})^2$$

4. We optimize this error (loss function) using an optimization algorithm. Generally, some form of gradient descent algorithm is used to find the optimal values of the hyperparameters like learning rate, weight, Bias, etc. This step forms the backward propagation part of the algorithm.

   An overview of this algorithm is illustrated in the following Figure:

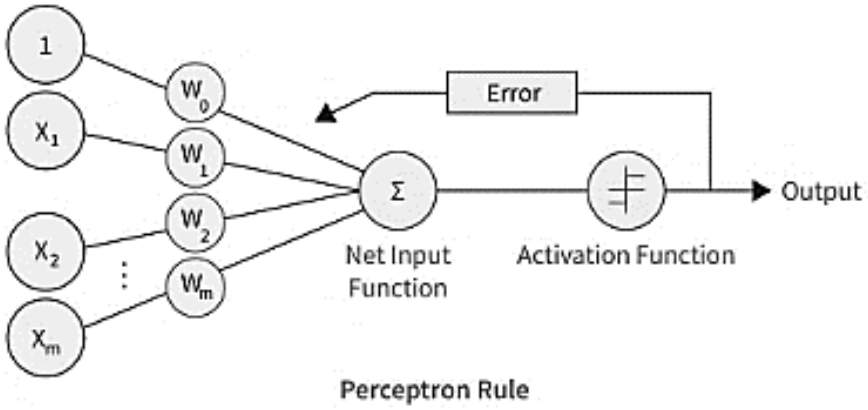


Fig.8.1: Perception Rule

During training, the Perceptron's weights are adjusted to minimize the difference between the predicted output and the actual output. This is achieved using supervised learning algorithms like the delta rule or the Perceptron learning rule.

The weight update formula is:

$$w_{i,j} = w_{i,j} + \eta(y_j - y\hat{}_j)x_i$$

Where:

- $w_{i,j}$ is the weight between the *ith* input and *jth* output neuron,
- x_i is the ith*ith* input value,
- y_j is the actual value, and $y\hat{}j$ is the predicted value,
- η is the learning rate, controlling how much the weights are adjusted.

This process enables the perceptron to learn from data and improve its prediction accuracy over time.

Example: Perceptron in Action

Let's take a simple example of classifying whether a given fruit is an apple or not based on two inputs: its weight (in grams) and its color (on a scale of 0 to 1, where 1 means red). The perceptron receives these inputs, multiplies them by their weights, adds a bias, and applies the activation function to decide whether the fruit is an apple or not.

- Input 1 (Weight): 150 grams
- Input 2 (Color): 0.9 (since the fruit is mostly red)
- Weights: [0.5, 1.0]
- Bias: 1.5

The perceptron's weighted sum would be:

$(150*0.5)+(0.9*1.0)+1.5=76.4(150*0.5)+(0.9*1.0)+1.5=76.4$

Let's assume the activation function uses a threshold of 75. Since 76.4 > 75, the perceptron classifies the fruit as an apple (output = 1).

- **Perceptron Learning Algorithm: Implementation of AND Gate**

The steps for this implementation are as follows:

1. **Import library**

   ```
   import numpy as np
   ```

2. **Define input (X) and expected output (Y)**

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

3. **Initialize weights, bias and learning rate**

   ```
   w = np.random.rand(2)
   b = np.random.rand(1)
   lr = 0.1
   ```

4. **Define activation function (step)**

   ```
   def activation(x):
   return 1 if x >= 0 else 0
   ```

5. **Training loop — outer (epochs-One full pass through all training samples)**

   ```
   for epoch in range(10):
   print(f"Epoch {epoch+1}")
   ...
   ```

6. **For each training example — compute weighted sum**

   ```
   z = np.dot(X[i], w) + b
   ```

7. **Predict using activation**

   ```
   y_pred = activation(z)
   ```

8. **Compute error**

   ```
   error = Y[i] - y_pred
   ```

9. Update rule (Perceptron learning rule)

```
w = w + lr * error * X[i]
b = b + lr * error
```

10. (Optional) Print intermediate progress

```
print(f"Input:{X[i]},Target:{Y[i]}, Predicted:{y_pred}, Updated Weights:{w}, Bias:{b}")
```

11. End of epoch separator

```
print("-" * 60)
```

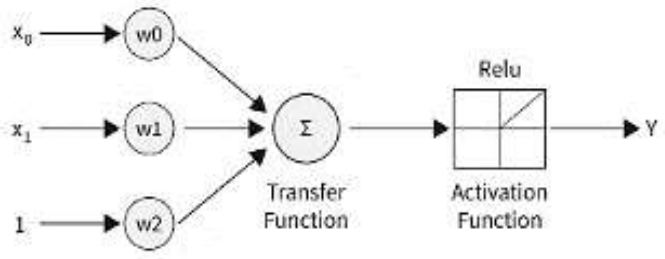
12. Testing after training

```
for i in range(len(X)):
z = np.dot(X[i], w) + b
y_pred = activation(z)
print(f"Input: {X[i]} → Output: {y_pred}")
```

Calculate Output and Activation Function using following:

AND Gate

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1



VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Write a Python program to implement OR Logic Gate using Perceptron Learning Algorithm.
2. Write and explain the steps to implement Perceptron Learning Algorithm.
3. Differentiate between Perceptron Algorithm and Perceptron Learning Algorithm.
4. What is Perceptron Learning Algorithm? Write the steps involved in it.
5. Explain the Perceptron Learning Rule with suitable example.
6. What is the role of epochs and learning rate in the Perceptron Learning Algorithm? Explain with example.

Space for Answers

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.geeksforgeeks.org/machine-learning/what-is-perceptron-the-simplest-artificial-neural-network/
- https://www.scaler.com/topics/machine-learning/perceptron-learning-algorithm/
- https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron
- https://data-intelligence.hashnode.dev/the-perceptron-algorithm-some-limitations

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 9

***Write a program to implement Perceptron Learning in Python using Iris flower dataset**

I. Practical Significance:

The Perceptron Learning Algorithm perform simple binary classification problems, it also demonstrating how machines learn from data. It clarifies the learning process by adjusting weights and defining decision boundaries. Despite its simplicity, it serves as the foundation for more powerful neural networks and aids in the solution of real-world issues such as pattern recognition, spam detection, and image classification.

II. Industry/Employer expected outcome(s):

To handle basic binary classification, understand model learning through weight updates, and apply these concepts as a foundation for advanced machine learning and neural network projects.

III. Course Level Learning Outcome (COs):

CO3 - Implement Artificial Neural Networks analyzing associated parameters of Deep Learning.

IV. Laboratory Learning Outcome (LLO):

LLO 9.1 Implement Perceptron algorithm using dataset.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

Perceptron Learning

Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron. A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.

In its simplest form, a Perceptron contains N input nodes, one for each entry in the input row of the design matrix, followed by only one layer in the network with just a single node in that layer.

- **Basic Components of Perceptron**

Perceptron is a type of artificial neural network, which is a fundamental concept in machine learning. The basic components of a perceptron are:

Input Layer: The input layer consists of one or more input neurons, which receive input signals from the external world or from other layers of the neural network.

Weights: Each input neuron is associated with a weight, which represents the strength of the connection between the input neuron and the output neuron.

Bias: A bias term is added to the input layer to provide the perceptron with additional flexibility in modeling complex patterns in the input data.

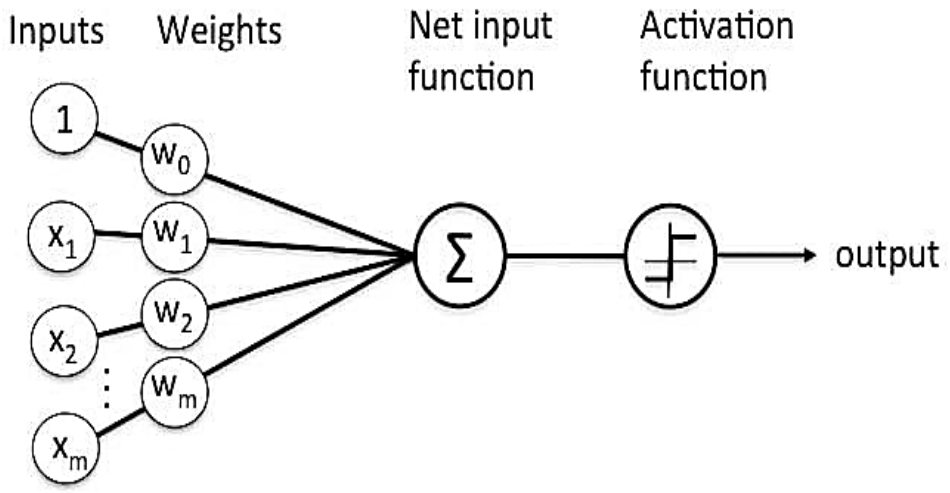


Fig. 9.1: Components of Perceptron

Activation Function: The activation function determines the output of the perceptron based on the weighted sum of the inputs and the bias term. Common activation functions used in perceptrons include the step function, sigmoid function, and ReLU function.

Output: The output of the perceptron is a single binary value, either 0 or 1, which indicates the class or category to which the input data belongs.

Training Algorithm: The perceptron is typically trained using a supervised learning algorithm such as the perceptron learning algorithm or backpropagation. During training, the weights and biases of the perceptron are adjusted to minimize the error between the predicted output and the true output for a given set of training examples.

The perceptron model begins with multiplying all input values and their weights, then adds these values to create the weighted sum. Further, this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f.' This step function or Activation function is vital in ensuring that output is mapped between (0,1) or (-1,1). Take note that the weight of input indicates a node's strength. Similarly, an input value gives the ability the shift the activation function curve up or down.

There are four significant steps in a perceptron learning algorithm:

1. First, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows: $\sum w_i * _x i = x_1 * w_1 + x_2 * w_2 + \ldots + w_n * x_n$. Add another essential term called bias 'b' to the weighted sum to improve the model performance. $\sum w_i * x_i + b$.
2. Next, an activation function is applied to this weighed sum, producing a binary or a continuous-valued output. $Y = f(\sum w_i * x_i + b)$
3. Next, the difference between this output and the actual target value is computed to get the error term, E, generally in terms of mean squared error. The steps up to this form the forward propagation part of the algorithm. $E = (Y - Y_{actual})^2$

4. Optimize this error (loss function) using an optimization algorithm. Generally, some form of gradient descent algorithm is used to find the optimal values of the hyperparameters like learning rate, weight, Bias, etc. This step forms the backward propagation part of the algorithm.

Perception Function

Perceptron learning algorithm function $f(x)$ is represented as the product of the input vector (x) and the learned weight vector (w). In mathematical notion, it can be described as:

$$f(x) = 1, if w.x + b > 0 \; f(x) = 0, otherwise$$

Where–

- w represents the weight vector which consists of a set of real-valued weights.
- b represents the bias vector.
- x represents the input vector which consists of the input feature values.

Python Program

```
# Import libraries
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
# Use only two classes (0: setosa, 1: versicolor)
X = X[y != 2]
y = y[y != 2]

# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Initialize and train Perceptron
clf = Perceptron(max_iter=1000, eta0=0.1, random_state=42)
clf.fit(X_train, y_train)

# Predict on test set
y_pred = clf.predict(X_test)
# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Perceptron Test Accuracy: {accuracy * 100:.2f}%")
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Define perceptron.
2. Draw the simple perceptron model.
3. What are the characteristics of Perceptron Algorithms?
4. Identify the parameters in a perceptron network and its significance.
5. List different Types of Perceptron Models?
6. Explain single layer and Multiple layer Perceptron Model.
7. What is activation function?
8. Illustrate Perceptron Learning Algorithm with an Example

Space for Answers

_____

_____

_____

_____

XI. References/Suggestions for further reading

- https://pyimagesearch.com/2021/05/06/implementing-the-perceptron-neural-network-with-python/
- https://www.scaler.com/topics/machine-learning/perceptron-learning-algorithm/
- https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron
- https://www.upgrad.com/blog/perceptron-learning-algorithm-how-it-works/

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 10

Write a program to implement Gradient Descent in PyTorch

I. Practical Significance:

Implementing Gradient Descent in PyTorch efficiently optimizes machine learning and deep learning models through automatic differentiation and dynamic computation. PyTorch's Autograd feature automatically computes gradients, eliminating manual derivative calculations and reducing errors. This enables faster and more accurate training as model parameters are updated iteratively to minimize the loss function. Additionally, PyTorch supports GPU acceleration for large-scale computations, making the optimization process highly efficient. Overall, Gradient Descent in PyTorch is essential for enabling models to learn from data, improve prediction accuracy, and form the foundation for various real-world AI applications such as image recognition, speech processing, and natural language understanding.

II. Industry/Employer expected outcome(s):

- To understand the basic working of model optimization to solve real-world problems using machine learning techniques
- To be capable of analyzing and improving model performance.

III. Course Level Learning Outcome (COs):

CO3 - Implement Artificial Neural Networks analyzing associated parameters of Deep Learning.

IV. Laboratory Learning Outcome (LLO):

LLO 10.1 Develop Gradient Descent in PyTorch.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Gradient descent:**

Gradient descent is like climbing down a mountain in foggy weather. If you can only see a few steps ahead, you must carefully adjust your path based on the slope beneath your feet. In deep learning, this "slope" is the gradient, and the goal is to reach the lowest point of the loss function, where the model makes the best predictions.

Gradient descent is an optimization technique used in ML techniques to fine-tune a model's parameters, ensuring it learns from data effectively. The algorithm iteratively adjusts weights and biases according to loss function gradient, aiming to minimize errors in predictions. Gradient descent is considered as the backbone of deep learning optimization as it allows models to reduce a loss function by iteratively updating their parameters. This section will explain how gradient descent works and why it is essential for training generative models in PyTorch.

- **How Gradient Descent Works?**

The process follows four key steps:

a. **Calculate Loss:** The model measures how far its predictions deviate from actual values using a loss function. The most common examples are Binary Cross-Entropy for classification tasks and Mean Squared Error (MSE) for regression models.

b. **Compute Gradients:** Loss function gradient is determined using backpropagation, which calculates how much each parameter contributes to the overall error.

c. **Update Parameters:** The model updates its weights by moving in the opposite direction of the gradient, gradually reducing the loss with each step.

d. **Iterate Until Convergence:** This cycle continues for multiple iterations until the model converges to an optimal solution.

By carefully tuning the learning rate and optimizing gradients, gradient descent enables deep learning models to improve accuracy and generalization over time. Different variations, such as stochastic, mini-batch, and full-batch gradient descent, offer flexibility in handling large datasets efficiently.

- **The Role of Loss Functions in Gradient Descent**

Loss functions measure the difference between a model's predictions and the actual values, providing a benchmark for optimization during training. The choice of loss function influences how gradients are calculated and updated:

- **Mean Squared Error (MSE):** Common in regression problems, MSE penalizes larger errors more heavily, make i useful for models where precise numerical predictions matter.
- **Cross-Entropy Loss:** This loss function is used for classification tasks; this loss function helps adjust weights based on how confidently the model predicts each class.
- **Wasserstein Loss:** Particularly useful for GANs, Wasserstein loss stabilizes training by ensuring a smoother gradient update compared to traditional adversarial loss functions.

In our case MSE will be represented by the MSE loss function, which looks like this:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- **Choosing the Right Batch Size: Mini-Batch vs. Full-Batch Gradient Descent**

The way data is processed during training also affects optimization:

- **Full-Batch Gradient Descent:** Uses all data at once, leading to stable but computationally expensive updates.
- **Mini-Batch Gradient Descent:** Processes smaller chunks of data, balancing computational efficiency with stable convergence. This is the most widely used approach in deep learning.

By understanding how loss functions and batch sizes impact training, we can fine-tune radient descent for more efficient and accurate deep learning models.

The job of the loss function is to assess how far the predictions of the model are from the actual targets. The farthest they are, the greater will be the loss. And usually, since we start with a model whose weights are initialized randomly, at the beginning the value of the loss function is likely to be very high. This is where the optimization process steps in!

- **PyTorch**

  PyTorch provides built-in optimizers, implementing gradient descent manually helps in understanding its inner workings. The following are the steps used to train a simple model using gradient descent in PyTorch.

  Step 1: Import Required Libraries

  Step 2: Define a Simple Model

  Step 3: Define Loss Function and Initialize Parameters

  Step 4: Implement Manual Gradient Descent

  Step 5: Evaluate the Model

- **Example: Implementing Gradient Descent in PyTorch**

```
import torch
# Step 1: Initialize variable (parameter)
x = torch.tensor([5.0], requires_grad=True)
# Step 2: Set learning rate
learning_rate = 0.1
# Step 3: Perform Gradient Descent iterations
for i in range(10):
    # Function: y = x^2 (we want to minimize this)
    y = x**2
    # Compute gradient (dy/dx)
    y.backward()
    # Update parameter using Gradient Descent rule
    with torch.no_grad():
        x -= learning_rate * x.grad
    # Reset gradients for next iteration
    x.grad.zero_()
    # Display progress
    print(f"Iteration {i+1}: x = {x.item():.4f}")
```

Here,

- **Function:** $y=x^2$
- **Goal:** Minimize y by updating x using Gradient Descent.
- **Gradient (dy/dx):** 2x
- **Update rule:** $x= x – \eta \times$ gradient
  where η is the learning rate.

Output:

```
Iteration 1: x = 4.0000
Iteration 2: x = 3.2000
Iteration 3: x = 2.5600
Iteration 4: x = 2.0480
Iteration 5: x = 1.6384
Iteration 6: x = 1.3107
Iteration 7: x = 1.0486
```

Iteration 8: x = 0.8389
Iteration 9: x = 0.6711
Iteration 10: x = 0.5369

Here, The value of x keeps decreasing toward **0**, which is the minimum of the function $y=x^2$. This demonstrates how **Gradient Descent** helps the model minimize error and reach an optimal solution.

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Write a PyTorch program to implement Gradient Descent for minimizing the function
2. Demonstrate how to use torch.autograd to compute gradients and update parameters manually.
3. Implement a linear regression model using PyTorch and train it using Gradient Descent.
4. Differentiate between Batch, Mini-Batch, and Stochastic Gradient Descent.
5. What is the learning rate in Gradient Descent? How does it affect the model's performance?
6. Explain the working of the .backward() and .step() functions in PyTorch with reference to Gradient Descent.

Space for Answers

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.scaler.com/topics/pytorch/deep-learning-with-pytorch/
- https://magnimindacademy.com/blog/gradient-descent-in-pytorch-optimizing-generative-models-step-by-step-a-practical-approach-to-training-deep-learning-models/
- https://docs.pytorch.org/tutorials/beginner/examples_autograd/polynomial_custom_function.html
- https://www.scaler.com/topics/pytorch/model-optimization-pytorch/
- https://medium.com/@my_key/gradient-descent-in-pytorch-bed6de03da07

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 11

Write a program to implement/Simulate Back propagation/feed forward neural network

I. Practical Significance:

The use of a Feed Forward Neural Network (FFNN) with Back Propagation is important because it explains how machines learn from data by modifying weights to reduce errors. This experiment lays a solid foundation for deep learning and gives you hands-on experience training models for prediction and classification problems.

II. Industry/Employer expected outcome(s):

Incorporate the capacity to create and build fundamental neural network models for practical uses and comprehend the methods employed in AI systems for data preparation, model training, and performance assessment.

III. Course Level Learning Outcome (COs):

CO4 - Build a Convolutional Neural Network for given context.

IV. Laboratory Learning Outcome (LLO):

LLO 11.1 Implement Back propagation/feed forward neural network.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

A feed forward neural network (FFNN) is an artificial neural network in which information goes in a single direction, from the input layer through hidden layers to the output layer, with no loops or feedback. It is mostly utilized for pattern recognition applications such as picture and audio classification.

Structure of FFNN

The layered structure of feedforward neural networks allows data to move through each layer in order.

Input Layer: The neurons that receive the input data make up the input layer. An input data characteristic is represented by each neuron in the input layer.

Hidden Layers: In between the input and output layers are one or more hidden layers. The complex patterns in the data are learned by these levels. A non-linear activation function is used after a weighted sum of inputs by each neuron in a hidden layer.

Output Layer: The output layer gives the network's final output. In a classification task, the number of classes, and in a regression problem, the number of outputs, is the number of neurons in this layer.

Weights and Biases

The weights assigned to each neuronal connection in these layers are modified during training in order to reduce prediction error.

Weights and biases in a neural network's feedforward are learnable parameters that are modified during training to minimize a certain loss function. Weights are the characteristics that govern the strength of connections between neurons in different levels. They are used to scale input signals before they are transmitted via a neuron's activation function. In other words, weights control how much influence a specific input has on a neuron's output. Biases are the characteristics that determine a neuron's offset, or baseline activity level. They move the input signal along the y-axis before passing it via the activation function. They contribute to preventing all outputs from becoming zero when the input is zero.

Activation Function

In a neural network feedforward, an activation function is a mathematical function that is applied to the output of a neuron. By adding non-linearity, it enables the network to learn and simulate increasingly complex input-output interactions. A neural network would be linear, less powerful, and less expressive than a non-linear model without the activation function.

There are many different activation functions that we can use in a neural networks feedforward;

Sigmoid:

The sigmoid activation function maps any input value to a value between 0 and 1, which is useful for binary classification problems.

$f(x) = \frac{1}{1 + e^{-x}}$ where x represents the input, and e is Euler's number

The rectified linear unit (ReLU):

It is a popular choice in neural networks. It is defined as

$f(x)=\max(0,x)$

where x is the input to the function. For any input x, the output of the ReLU function is x if x is positive and 0 if x is negative. This activation function is simple to implement computationally and faster than other non-linear activation function like tanh or sigmoid.

tanh (hyperbolic tangent):

Similar to sigmoid, the tanh maps values from -1 to 1.

Softmax:

The softmax activation function maps the input value to a probability distribution, which is useful for multi-class classification problems.

Back Propagation

An important machine learning method for optimizing artificial neural networks is backpropagation. It makes it easier to adjust network weights using gradient descent techniques. It involves adjusting a neural network's weights according on the error rate (or loss) of the preceding epoch (or iteration). Lower error rates are guaranteed by properly calibrating the weights, which also increases the model's generalization and reliability.

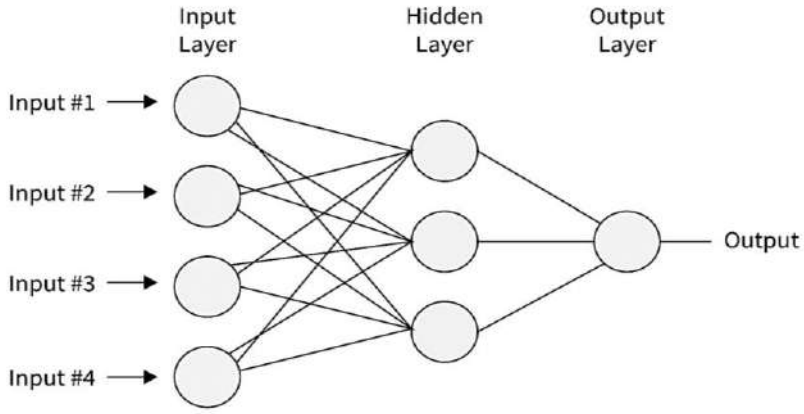


Fig.11.1: Back propagation in Neural Network

Suppose we have:

- Input vector x= [x1, x2,..., xn]
- Weights w_{ij} connecting neuron i to neuron j
- Bias b_j
- Activation Function f(.)

Then, for a hidden neuron j:

$$z_j=\sum w_{ij}x_i+b_j$$

$$a_j=f(z_j)$$

The same applies layer by layer until the output layer.

- **Back Propagation Steps:**

1. **Forward Pass:**

Compute the output for given inputs using current weights.

Loss Calculation:

Compute the error (loss) between the predicted output $\tilde{y}$ and the true label y, e.g.

$L=\frac{1}{2}(y+\tilde{y})^2$

(for mean squared error).

2. **Backward Pass:**

Compute the gradient of the loss L with respect to each weight using the chain rule:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a}\cdot\frac{\partial a}{\partial z}\cdot\frac{\partial z}{\partial w}$$

3. **Weight Update:**

Adjust each weight to reduce error:

$W \leftarrow W - \eta \frac{\partial L}{\partial w}$ where η is the learning rate.

- **Python Program**

```
import numpy as np
def sigmoid(x):
   return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
   # derivative of sigmoid: f'(x) = f(x) * (1 - f(x))
```

```
    return x * (1 - x)
X = np.array([    [0, 0],    [0, 1],    [1, 0],    [1, 1]])
y = np.array([    [0],    [1],    [1],    [0]])
input_size = 2
hidden_size = 2
output_size = 1
learning_rate = 0.5
epochs = 10000
np.random.seed(42)
W1 = np.random.randn(input_size, hidden_size)     # weights for input -> hidden
b1 = np.zeros((1, hidden_size))                   # bias for hidden layer
W2 = np.random.randn(hidden_size, output_size)    # weights for hidden -> output
b2 = np.zeros((1, output_size))                   # bias for output layer
for epoch in range(epochs):
    # ---- Forward pass ----
    z1 = np.dot(X, W1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, W2) + b2
    a2 = sigmoid(z2)

    # ---- Compute loss (Mean Squared Error) ----
    loss = np.mean((y - a2) ** 2)

    # ---- Backpropagation ----
    d_a2 = (a2 - y)
    d_z2 = d_a2 * sigmoid_derivative(a2)
    d_W2 = np.dot(a1.T, d_z2)

    d_b2 = np.sum(d_z2, axis=0, keepdims=True)
    d_a1 = np.dot(d_z2, W2.T)
    d_z1 = d_a1 * sigmoid_derivative(a1)
    d_W1 = np.dot(X.T, d_z1)
    d_b1 = np.sum(d_z1, axis=0, keepdims=True)

    # ---- Update weights ----
    W2 -= learning_rate * d_W2
    b2 -= learning_rate * d_b2
    W1 -= learning_rate * d_W1
    b1 -= learning_rate * d_b1

    # ---- Print loss occasionally ----
    if epoch % 1000 == 0:
        print(f"Epoch {epoch}, Loss: {loss:.4f}")
print("\nFinal predictions after training:")
for i in range(len(X)):
```

```
z1 = np.dot(X[i], W1) + b1
a1 = sigmoid(z1)
z2 = np.dot(a1, W2) + b2
a2 = sigmoid(z2)
print(f"Input: {X[i]} → Predicted: {float(a2):.4f}")
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Define a Feedforward Neural Network (FNN).
2. Explain the structure of a Feedforward Neural Network.
3. Describe the different layers in a feedforward neural network and their functions?
4. Discuss the significance of weights and biases in a neural network.
5. Explain the purpose of an activation function in a neural network.
6. List and explain some real-world applications of Feedforward Neural Networks trained using the Backpropagation algorithm.
7. Describe the forward pass and backward pass in the backpropagation algorithm.
8. Outline the step-by-step procedure for implementing a Feedforward Neural Network trained using Backpropagation.

9. Explain how the Backpropagation algorithm helps in minimizing the loss function.

Space for Answers

XI. References/Suggestions for further reading

- https://www.geeksforgeeks.org/nlp/feedforward-neural-network/
- https://www.scaler.com/topics/deep-learning/introduction-to-feed-forward-neural-network/
- https://www.ibm.com/think/topics/backpropagation
- https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 12

Build a small CNN model consisting of 5 convolution layers

I. Practical Significance:

Convolutional Neural Networks (CNNs) are essential for image processing and computer vision applications. Implementing a small CNN model with 5 convolutional layers helps to understand how deep learning models extract hierarchical image features from edges and textures to objects and shapes. This experiment is helpful to learn feature extraction, convolutional filters, activation functions, and pooling layers, which form the foundation of modern AI applications like face recognition, medical image analysis, and object detection.

II. Industry/Employer expected outcome(s):

- Build and train small CNNs for classification or recognition tasks.

III. Course Level Learning Outcome (COs):

CO4 - Build a Convolutional Neural Network for given context.

IV. Laboratory Learning Outcome (LLO):

LLO 12.1 Apply given CNN architecture.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Convolutional Neural Network (CNN):**

Convolutional Neural Networks (CNNs) are a type of deep learning model used for image recognition, processing, and classification. With basic CNN architecture, you can automatically and efficiently extract features from input data. But what is CNN in machine learning?

CNNs are a key technique in machine learning and deep learning, specializing in processing grid-like data such as images. Unlike traditional models like decision trees or SVMs, CNNs use filters to detect patterns like edges or shapes automatically.

This efficient feature extraction allows CNNs to handle complex visual data, making them ideal for tasks like image classification over other algorithms. The working of basic CNN architecture is like solving a puzzle. It first identifies individual pieces (comparable to identifying features like edges or shapes in an image) and then puts them to get the full picture (similar to classification or output). CNN algorithm helps streamline this process of extracting and learning from visual data efficiently.

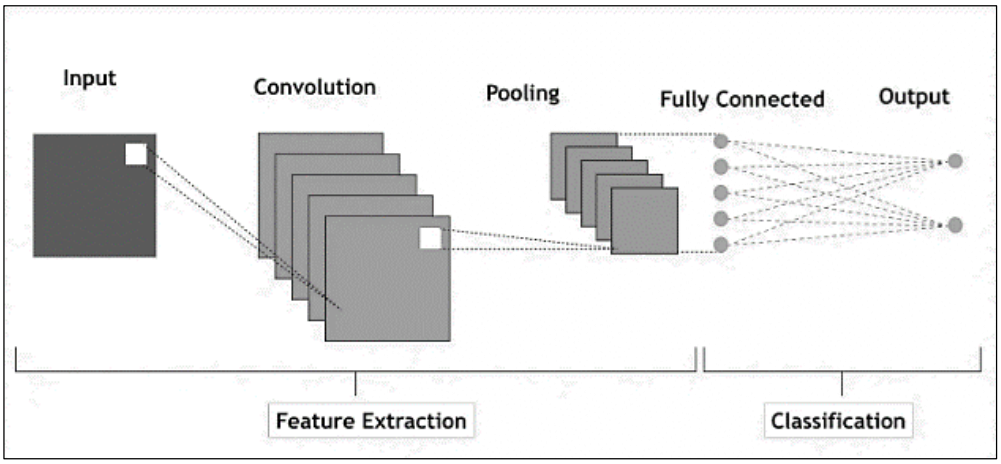


Fig.12.1: The architecture of CNN

Currently, CNNs are widely used for purposes such as video recognition (e.g., facial recognition), medical imaging (e.g., detecting cancerous tumors), self-driving cars (e.g., identifying road signs), and natural language processing (e.g., text classification). These are some examples of CNN types, where CNNs are applied across different domains.

The CNN architecture can be divided into five components. Here's an overview of the five layers.

- **Feature Extraction through Convolutional Layers**

  Convolutional layers scan the input data using filters (kernels) to detect patterns like edges, textures, or shapes.

  1. **Pooling Layers**

     Pooling layers preserve key features while reducing computational complexity. This helps reduce multiple dimensions.
  2. **Activation Layers**

     Activation layers apply non-linear functions like ReLU to introduce non-linearity. This enables the network to learn complex patterns.
  3. **Flattening and Fully Connected Layers**

     Once the feature has been extracted, the data is converted into a vector and passed through fully connected layers for classification.
  4. **Output Layer**

     The output layer provides the final prediction using a Softmax function for classification tasks.

Step:

1. import the necessary libraries
2. set the parameter
3. define the kernel
4. Load the image and plot it.
5. Reformat the image
6. Apply convolution layer operation and plot the output image.
7. Apply activation layer operation and plot the output image.
8. Apply pooling layer operation and plot the output image.

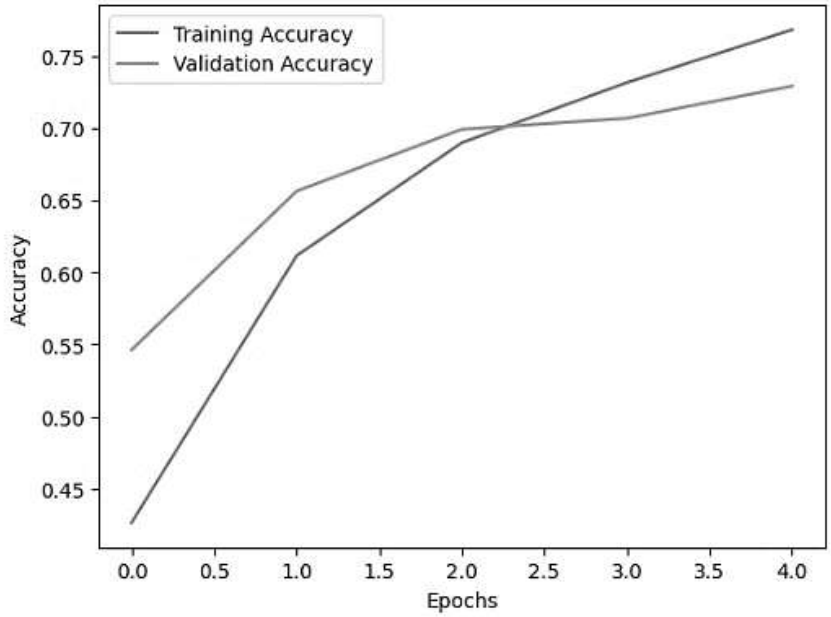
- **Stepwise Program Code:**

```
# Step 1: Import libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

# Step 2: Load and preprocess dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Step 3: Build CNN model with 5 Conv layers
model = models.Sequential([layers.Conv2D(32, (3,3), activation='relu', padding='same',
input_shape=(32, 32, 3)), layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    layers.MaxPooling2D((2,2)), layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    layers.Flatten(), layers.Dense(64, activation='relu'), layers.Dense(10, activation='softmax')])

# Step 4: Compile model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Step 5: Train model
history = model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

# Step 6: Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'\n Test Accuracy: {test_acc:.4f}')

# Step 7: Plot accuracy graph
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Output: (Execution time may be more than 15 minutes)

```
Epoch 1/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 203s 128ms/step - accuracy:
0.3229 - loss: 1.8014 - val_accuracy: 0.5463 - val_loss: 1.2324
Epoch 2/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 203s 130ms/step - accuracy:
0.5856 - loss: 1.1455 - val_accuracy: 0.6563 - val_loss: 0.9619
```

Epoch 3/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **213s** 136ms/step - accuracy: 0.6820 - loss: 0.8997 - val_accuracy: 0.6991 - val_loss: 0.8697
Epoch 4/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **201s** 129ms/step - accuracy: 0.7321 - loss: 0.7609 - val_accuracy: 0.7067 - val_loss: 0.8508
Epoch 5/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **201s** 129ms/step - accuracy: 0.7710 - loss: 0.6464 - val_accuracy: 0.7290 - val_loss: 0.7948
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **13s** 40ms/step - accuracy: 0.7399 - loss: 0.7763
Test Accuracy: 0.7290



VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. How does the accuracy of the CNN model change as the number of epochs increases? Support your answer with an example.
2. Explain the architecture of a Convolutional Neural Network (CNN) with the function of each layer.
3. Differentiate between Convolution layer and Pooling layer in a CNN.
4. What is the role of the ReLU activation function in CNN models?
5. Describe the working of the Flatten and Dense layers in a CNN model.
6. Explain how CNN automatically extracts features from images without manual feature selection.
7. Write any four important applications of CNN in real-world industries.

Space for Answers

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.upgrad.com/blog/basic-cnn-architecture/
- https://www.ibm.com/think/topics/convolutional-neural-networks
- https://www.tensorflow.org/tutorials/images/cnn
- https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 13

***Write a program to implement CIFAR 10- CNN using PyTorch**

I. Practical Significance:

Convolutional neural networks (CNNs) are useful because they can automatically extract and learn meaningful features from images and other spatial data without the need for feature engineering. CNNs are extensively utilized in applications including image classification, object detection, facial recognition, and medical image analysis, and they have completely transformed the area of computer vision. They are able to efficiently process and interpret vast amounts of visual data, recognizing intricate patterns like edges, textures, and objects at various levels of abstraction.

II. Industry/Employer expected outcome(s):

- The ability to create and use deep learning models for practical visual tasks like pattern recognition, object detection, and picture recognition.

III. Course Level Learning Outcome (COs):

CO4 - Build a Convolutional Neural Network for given context.

IV. Laboratory Learning Outcome (LLO):

LLO 13.1 Implement CIFAR 10- CNN.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

Convolutional Neural Networks (CNNs), sometimes referred to as ConvNets, are a subset of deep learning algorithms that are mostly used for tasks involving object recognition, such as picture categorization, detection, and segmentation. The primary purpose of Convolutional Neural Networks (CNNs), an advanced version of artificial neural networks (ANNs), is to extract features from grid-like matrix datasets.

CNNs consist of multiple layers like the input layer, Convolutional layer, pooling layer, and fully connected layers. CNN's complexity grows with each layer, it can recognize larger areas of the image. Earlier layers emphasize basic elements like borders and colors. As the image input moves through the CNN's layers, it begins to identify the item's bigger forms or elements before identifying the target object.

Convolutional layer : It requires a few components, which are input data, a filter and a feature map. At the most basic level, the input to a convolutional layer is a two-dimensional array which can be the input image to the network or the output from a previous layer in the network. Convolutional layers use filters to process the input data.

Pooling layers, also known as downsampling, conducts dimensionality reduction, reducing the number of parameters in the input.

Fully-connected layer : The name of the full-connected layer aptly describes itself. In the fully-connected layer, each node in the output layer connects directly to a node in the previous layer. This layer performs the task of classification based on the features extracted through the previous layers and their different filters.

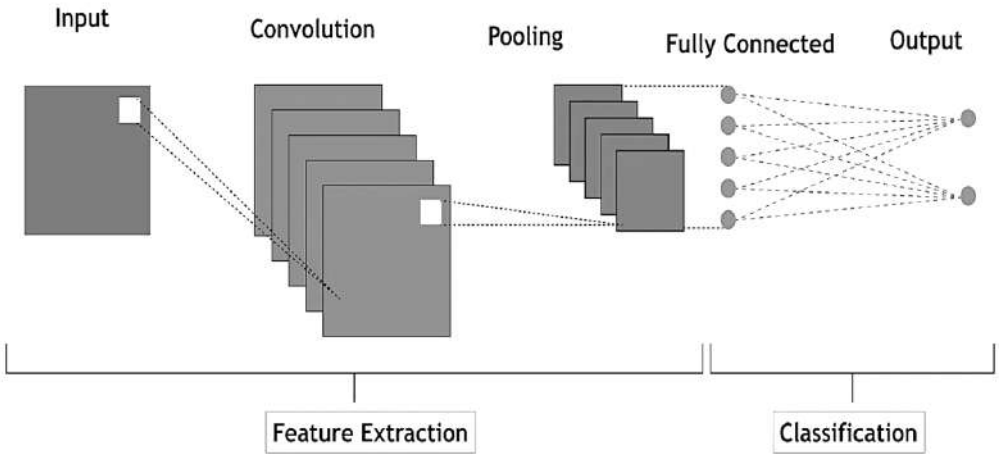


Fig.13.1: The Architecture of CNN

The 6 steps of CNN are:

1. Input Layer: Receives the raw input data, usually images.
2. Convolutional Layer: Applies convolutional filters to detect features.
3. Activation Layer: Introduces non-linearity to the system.
4. Pooling Layer: Reduces spatial dimensions, retaining critical information.
5. Flattening: Converts the pooled feature maps into a vector.
6. Fully Connected Layer: Connects every neuron to every neuron in the subsequent layer.

Steps:

1. Load and normalize the CIFAR10 training and test datasets using torchvision
2. Define a Convolutional Neural Network
3. Define a loss function
4. Train the network on the training data
5. Test the network on the test data

Load and normalize CIFAR10

```
import torch
import torchvision
import torchvision.transforms as transforms
```

The output of torchvision datasets are PILImage images of range [0, 1]. We transform them to Tensors of normalized range [-1, 1].

```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
batch_size = 4
```

```
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size, shuffle=True,
num_workers=2)
testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True,
transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size, shuffle=False,
num_workers=2)
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

Define a Convolutional Neural Network

```
import torch.nn as nn
import torch.nn.functional as F
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
net = Net()
```

Define a Loss function and optimizer

```
import torch.optim as optim
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Train the network

```
for epoch in range(2):  # loop over the dataset multiple times
  running_loss = 0.0
  for i, data in enumerate(trainloader, 0):
     # get the inputs; data is a list of [inputs, labels]
      inputs, labels = data
            # zero the parameter gradients
            optimizer.zero_grad()
            # forward + backward + optimize
            outputs = net(inputs)
```

```
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            # print statistics
            running_loss += loss.item()
            if i % 2000 == 1999:    # print every 2000 mini-batches
                print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
                running_loss = 0.0
print('Finished Training')
```

Test the network on the test data

```
dataiter = iter(testloader)
images, labels = next(dataiter)

# print images
imshow(torchvision.utils.make_grid(images))
print('GroundTruth: ', ' '.join(f'{classes[labels[j]]:5s}' for j in range(4)))
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

_____

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Define Convolutional Neural Network (CNN).
2. Where is the CNN algorithm used?
3. What are the components of CNN?
4. How does pooling work, and why is it used?
5. What are some common CNN architectures and their characteristics?
6. Explain what a stride is and how it affects the output size of the convolution layer?

Space for Answers

XI. References/Suggestions for further reading

- https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/
- https://www.ibm.com/think/topics/convolutional-neural-networks/
- https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
- https://clanx.ai/glossary/convolutional-neural-networks-cnns
- https://docs.pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://www.youtube.com/watch?v=sgL7RrqhGKI

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 14

Basic Time Series Forecasting with LSTM

I. Practical Significance:

Basic Time Series Forecasting using Long Short-Term Memory (LSTM) networks enables accurate prediction of future trends based on past data. LSTM models are capable of learning long-term dependencies and patterns, making them highly effective for applications where data changes over time. This technique is widely used in industries such as finance, sales, energy, weather prediction, and healthcare to forecast stock prices, product demand, power consumption, or patient vitals. By implementing LSTM forecasting, we can handle sequential data, preprocess time series, train deep learning models and interprets real-world prediction results which are directly applicable in modern data analytics and AI-driven industries.

II. Industry/Employer expected outcome(s):

- To apply Long Short-Term Memory (LSTM) networks to analyze and forecast time-dependent data.

III. Course Level Learning Outcome (COs):

CO5 - Classify Sequential and Image Data using Deep Learning.

IV. Laboratory Learning Outcome (LLO):

LLO 14.1 Implement a basic LSTM model for forecasting future values based on past data.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

- **Time Series:**

A time series data is any temporal data with a discrete time interval and almost equidistant time steps. The general task is to estimate the function used to generate such a time series. If we can estimate the function correctly, we can predict future points the model has yet to encounter. Examples of time series include heart rate data, stock market data, and many others.

RNNs are a family of models that take entire series as inputs and return series as outputs. Unlike a simple Convolutional network that uses Backpropagation, an RNN uses a modified variant called Backpropagation through time (BPTT) to embed temporal data. This algorithm is a sequential process and contains hidden states that model the underlying data. An RNN is considered Turing complete and is used in domains such as Natural Language Processing, Computer Vision, Robotics, and many others. The RNN architecture is made up of gates.

RNNs suffer from a variety of problems due to their sequential nature.

1. RNNs fail to model longer sequences. This property makes it very hard to use for data that has a long temporal span.
2. The classical RNN also had an issue with exploding and vanishing gradients due to how the underlying architecture worked. These problems make an RNN very unstable.

- **LSTM:**

LSTMs are a modified version of RNNs with different gates, enabling the architecture to model much longer sequences. The LSTMs use gated connections that learn which features to forget and which to remember. The ability to choose what to forget makes them much better than a classical RNN. LSTMs are also much more stable and less likely to explode or vanish gradients.

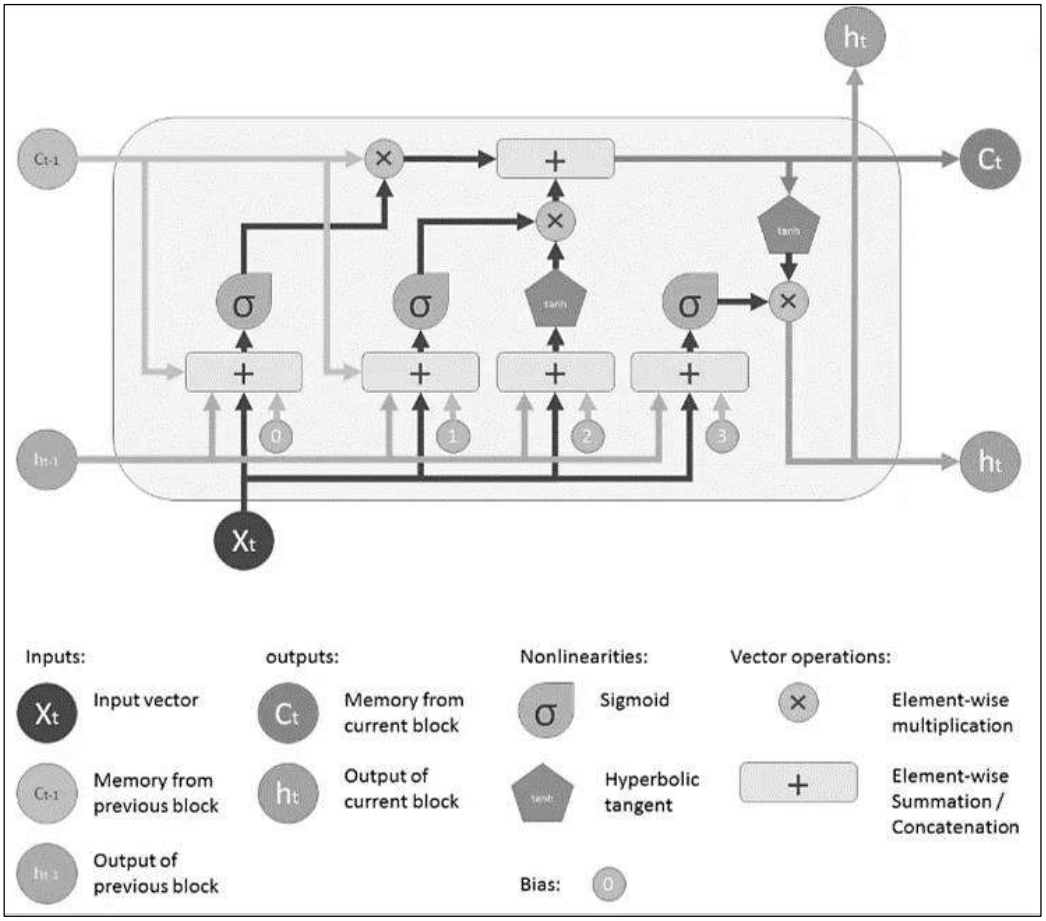


Fig.14.1: Structure of LSTM

LSTM works kind of like this, it has:

1. **Memory:** LSTM has a "notebook" where it can store important information (like the recipe steps). This notebook can keep track of things for a long time, helping it remember what's necessary and forget the rest. It's called (Memory cells) and can store information for long periods. These cells help the network learn which data to remember or forget
2. **Gates:** Think of these as decision-makers who control the flow of information. LSTM has three gates that help it decide:
   i. **Input Gate:** Should I write this new information in my notebook? which corresponds to how much new information flows into the memory cell
   ii. **Forget Gate:** Should I erase something I don't need anymore? — how much of the previous cell state should be forgotten

iii. **Output Gate:** Should I use this piece of information right now? — how much information from the cell state flows out to the next time step

We need many libraries to implement an `LSTM time series model. These are listed below.

- TensorFlow for building the LSTM time series model architecture.
- NumPy for numerical processing.
- Pandas for handling tabular data.
- Matplotlib and seaborn for plotting.
- The RC module in Matplotlib to configure plots.

Steps:

```
# Step 1: Import Required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
# Step 2: Load and Prepare the Dataset
# Here we create a sample sine wave as time series data
data = pd.DataFrame({'Value': np.sin(np.arange(0, 100, 0.1))})
# Plot the original data
plt.figure(figsize=(8,4))
plt.plot(data)
plt.title("Sample Time Series Data")
plt.xlabel("Time")
plt.ylabel("Value")
plt.show()
# Step 3: Normalize the Data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
# Step 4: Create Sequences for Training
def create_dataset(dataset, time_step=10):
    X, Y = [ ], [ ]
    for i in range(len(dataset)-time_step-1):
        X.append(dataset[i:(i+time_step), 0])
        Y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(Y)
time_step = 10
X, Y = create_dataset(scaled_data, time_step)

# Step 5: Reshape Data for LSTM
X = X.reshape(X.shape[0], X.shape[1], 1)
# Step 6: Build the LSTM Model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step, 1)))
```

```
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
# Step 7: Train the Model
model.fit(X, Y, epochs=20, batch_size=16, verbose=1)
# Step 8: Make Predictions
train_predict = model.predict(X)
# Step 9: Inverse Transform to Original Scale
train_predict = scaler.inverse_transform(train_predict)

# Step 10: Visualize the Results
plt.figure(figsize=(8,4))
plt.plot(data['Value'], label='Original Data')
plt.plot(range(time_step, len(train_predict) + time_step), train_predict, label='Predicted Data', color='red')
plt.title("Time Series Forecasting using LSTM")
plt.xlabel("Time")
plt.ylabel("Value")
plt.legend()
plt.show()
```

Here, the program creates a sample sine wave as time series data. It then normalizes and converts data into input sequences for LSTM. The LSTM model is trained to learn the sequence patterns. Finally, it predicts and visualizes future values compared to the original data. An output will be a graph showing blue line which is Original data and red line which contains LSTM predicted values, these two curves will closely follow each other, showing successful forecasting.

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

X. Practical related Questions:

(Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)

1. Modify the given LSTM model to predict future stock prices or temperature data.
2. Implement and execute above program and observe the output. Also train the LSTM model for different numbers of epochs and note the effect on accuracy and prediction curve.
3. Explain the architecture of LSTM and its role in time series forecasting.
4. Differentiate between Recurrent Neural Network (RNN) and LSTM. Why is LSTM preferred for long-term forecasting?
5. What is the importance of normalization in time series forecasting using LSTM?
6. Explain how LSTM overcomes the vanishing gradient problem in traditional RNNs.

Space for Answers

XI. References/Suggestions for further reading

- https://colab.research.google.com/drive/
- https://www.scaler.com/topics/deep-learning/lstm-time-series/
- https://medium.com/data-science-collective/future-forecasting-of-time-series-using-lstm-a-quick-guide-for-business-leaders-370661c574c9
- https://www.mathworks.com/help/deeplearning/ug/time-series-forecasting-using-deep-learning.html

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		

Practical No. 15

*Classification of images using imagenet dataset

I. Practical Significance:

The ImageNet dataset is extremely useful for image classification. ImageNet is a large and well-labeled image collection for training and testing deep learning models, particularly Convolutional Neural Networks (CNNs). It establishes a consistent baseline for measuring model performance. Models are used for transfer learning, which involves adapting pretrained models to specific tasks using smaller datasets. Overall, ImageNet-based image classification has set the foundation for modern AI, improving the accuracy, efficiency, and dependability of visual identification technology.

II. Industry/Employer expected outcome(s):

To develop intelligent systems that can automatically identify, analyze, and categorize visual information with high accuracy.

III. Course Level Learning Outcome (COs):

- CO5 - Classify Sequential and Image Data using Deep Learning.

IV. Laboratory Learning Outcome (LLO):

- LLO 15.1 Classify image data using pre trained model.

V. Relevant Affective Domain related Outcome(s):

- Follow safety practices.
- Demonstrate working as a leader or a team leader.
- Follow ethical practices.
- Translate problems into ML tasks.

VI. Relevant Theoretical Background:

Image classification is the process of detecting and categorizing images based on their content. Image classification is supervised machine learning (ML) in which an algorithm labels an image based on its visual information. It involves building a model on a labeled dataset so that it can learn to categorize previously unseen images into predetermined groups. The classification process entails feature extraction, pattern identification, and decision-making with ML or deep learning models.

Pre-trained models are a critical component of current deep learning. These models are originally trained using large general-purpose datasets such as ImageNet. They learn to recognize a variety of elements, including simple edges, complicated textures, and objects.

There are several advantages to using pre-trained models, including the ability to leverage the knowledge and experience of others, the ability to save time and resources, and the ability to improve model performance. Pre-trained models are often trained on large, diverse datasets and have been trained to recognize a wide range of patterns and features. As a result, they can provide a strong foundation for fine-tuning and can significantly improve the performance of the model.

Several pre-trained models have become standards in image classification due to their performance and reliability.
Here are the key models:

- ResNet (Residual Networks)
- Inception (GoogLeNet)
- VGG (Visual Geometry Group)
- EfficientNet
- DenseNet (Dense Convolutional Network)
- MobileNet
- NASNet (Neural Architecture Search Network)
- Xception (Extreme Inception)
- AlexNet
- Vision Transformers (ViT)

Classifying image data with a pre-trained model involves using a neural network built on a large dataset, such as ImageNet, to perform image classification tasks. This method, also known as transfer learning, saves a large amount of time and computational resources when compared to building a model from scratch.
ImageNet is a freely available large-scale collection of annotated photos designed to be utilized in a variety of computer vision tasks. It contains over 14 million photos, each annotated with WordNet synonym sets. It is one of the most extensive resources for building deep learning models to perform image recognition tasks.

- **Python Code:**

```
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input,
decode_predictions
import numpy as np

def classify_image_with_pretrained_model(image_path):
    # Load the pre-trained ResNet50 model with ImageNet weights
    model = ResNet50(weights='imagenet')

    # Load and preprocess the image
    img = image.load_img(image_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    img_array = preprocess_input(img_array)  # Apply ResNet50 specific preprocessing

    # Make predictions
    predictions = model.predict(img_array)
    # Decode the predictions
    decoded_predictions = decode_predictions(predictions, top=5)[0] # Get top 3 predictions
    return decoded_predictions
if __name__ == "__main__":
```

```
# Example usage:
# Create a dummy image file for demonstration
from PIL import Image
dummy_image = Image.new('RGB', (224, 224), color = 'red')
dummy_image.save('cat.jpg')
image_to_classify = 'cat.jpg' # Replace with your image path
results = classify_image_with_pretrained_model(image_to_classify)
print(f"Classification results for '{image_to_classify}':")
for i, (imagenet_id, label, score) in enumerate(results):
    print(f"{i + 1}: {label} ({score:.2f})")
```

VII. Required Resources:

Sr. No.	Name of Resource	Specification	Quantity	Remark (if any)
1.	Computer System	Computer with Processor i5 with 8 GB or more RAM	01 for each student	
2.	Operating System	Windows 10 or later		
3.	Software	Editor: Python Interpreter/ IDLE OR Any other open source IDLE such as Jupyter Notebook, Google Colab etc. PyTorch tool.		
4.	Printer	Laser/Inkjet Printer A4 size	01 for Batch	

VIII. Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion:

_____

_____

_____

_____

X. Practical related Questions:

(*Note: Teacher shall assign any questions to students from following. Teachers must design more such questions if required to achieve identified CO)*

1. How is the ImageNet dataset structured, and why is it important?
2. What is the image classification?
3. What is object detection and how does it differ from classification?
4. What is the role of Convolutional Neural Networks (CNNs) in image classification?
5. Mention any two popular models trained on the ImageNet dataset.
6. What are the main steps involved in image classification using deep learning?

Space for Answers

XI. References/Suggestions for further reading

- https://www.geeksforgeeks.org/computer-vision/top-pre-trained-models-for-image-classification/
- https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/
- https://encord.com/glossary/pre-trained-model-definition/
- https://www.appliedaicourse.com/blog/image-classification-using-machine-learning/
- https://viso.ai/deep-learning/imagenet/?utm_source=chatgpt.com

XII. Assessment Scheme: 25 Marks

Sr. No.	Performance Indicators	Weightage	Marks Obtained
A.	**Process Related: 15 Marks**	**60%**	
1.	Logic formation	40%	
2.	Debugging ability	10%	
3.	Correctness of program code	10%	
B.	**Product Related: 10 Marks**	**40%**	
3.	Expected output	10%	
4.	Timely completion of practical	10%	
5.	Answer to sample questions	20%	
C.	**Total marks obtained out of 25 Marks**		
D.	**Dated signature of Course Teacher**		