SCHEME :K

Name :_____
Roll No.:_____ Year : 20 ___ 20 __
Exam Seat No. :_____

LABORATORY MANUAL FOR
MACHINE LEARNING (316316)

COMPUTER ENGINEERING GROUP

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

Maharashtra State Board of Technical Education, Mumbai

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

MACHINE LEARNING
(316316)

Semester–VI

(CM/CO/CW/HA/IF/IH/SE)

'K' Scheme Curriculum

Maharashtra State Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018) (ISO/IEC 27001:2013)

Maharashtra State Board of Technical Education, Mumbai
(Autonomous) (ISO 21001:2018) (ISO/IEC 27001:2013)
4th Floor, Government Polytechnic Building,
49, Kherwadi, Bandra (East), Mumbai- 400051

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

Certificate

This is to certify that Mr. / Ms……. ……. Roll No. …… of the Sixth Semester of Diploma in …… Engineering/ Technology (Program Code:-.....K) of the Institute, ……..… (Inst.Code:- ……) has completed the practical work satisfactorily for the course **Machine Learning (Course Code:- 316316)** for the academic year 20….. – 20….. as prescribed in the curriculum.

Place: …….. Enrollment No: …….

Date: ……. Exam Seat No: ……

Course Teacher **Head of the department** **Principal**

Seal of Institution

Preface

Machine Learning (316316) laboratory manual is meticulously crafted to equip sixth semester diploma engineering students with valuable practical learning experiences aligned with MSBTE 'K' Scheme Curriculum.

The primary objective of this manual is to enhance valuable skills in Machine Learning of diploma engineering students which is vital to excel at the workplace. To achieve this, each practical is mapped with prescribed theory learning outcomes (TLOs), lab learning outcomes (LLOs) and course outcomes (COs). Course facilitators can adopt suitable pedagogical methods to impart the course with an aim to achieve the prescribed course outcomes effectively.

This laboratory manual is designed to help all stakeholders, especially the students and teachers to develop in the student the pre-determined outcomes. Every practical in this manual begins by identifying the Practical Significance, Industry/Employer expected outcome, Course Level Learning Outcomes, and Laboratory Learning Outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This laboratory manual also provides guidelines to teachers and instructors to effectively facilitate student-centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students. Machine Learning is a field of Artificial Intelligence (AI) that focuses on enabling computers to understand, interpret, and respond to human language. It automating tasks, enabling predictive analysis, and personalizing experiences across many fields like finance, healthcare, and entertainment. Diploma Engineers should be equipped with machine learning knowledge can contribute to various fields by developing systems that understand different supervised and unsupervised learning algorithm task.

This laboratory manual gives the important concepts and techniques related to machine learning and enable students to build different supervised, unsupervised learning algorithm in machine learning.

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 Outcomes (POs) to be achieved through Course:

PO1 Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO2 Problem analysis: Identify and analyses well-defined engineering problems using codified standard methods.

PO3 Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4 Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO5 Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO6 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.

PO7 Life-long learning: Ability to analyses individual needs and engage in updating in the context of technological changes.

List of relevant expected Psychomotor Domain Skills

This Lab manual intends to develop expected psychomotor domain skills of students. On the successful completion of the course the students will acquire the required industry relevant skills and they will be able to:

1. Identifying appropriate tools/libraries for machine learning tasks.
2. Apply preprocessing techniques on datasets
3. Implement algorithms using libraries like scikit-learn.
4. Design and build various supervised, unsupervised machine learning models.
5. Test the different model accuracy using testing dataset.
6. Identify real-world problems suitable for AI/ML solutions.
7. Translate/ solved different business problems using Machine Learning tasks.

Practical Course Outcome Matrix

Course Outcomes (COs)

CO1	Explain the role of machine learning in AI and data science.
CO2	Implement data preprocessing.
CO3	Implement feature engineering techniques to prepare data for machine learning models.
CO4	Apply supervised learning models to train and evaluate.
CO5	Apply unsupervised learning models to train and evaluate.

Sr. No	Title of the Practical	Mapped Course Outcome				
		CO1	CO2	CO3	CO4	CO5
1	*Installation of IDE with necessary libraries	✓	-	-	-	-
2	*Implement program for Data Preprocessing Techniques	✓	✓	-	-	-
3	Implement program to read dataset (Text, CSV, JSON ,XML)	-	✓	-	-	-
4	Implement the classification algorithms on previously prepared dataset	-	✓	✓	✓	-
5	*Implement the regression model by using the suitable dataset	-	✓	-	✓	-
6	*Implement program to use clustering algorithms to find patterns in data	-	-	✓	✓	-
7	Implement program to identify the most important features that contribute to the model accuracy	-	✓	✓	-	-
8	Implement program to use k-Nearest Neighbors (KNN) model for Classification on given dataset	-	-	-	✓	-
9	*Implement program to train an SVM model on given dataset	-	-	-	✓	-
10	Implement program to use logistic regression model to classify binary outcomes	-	-	✓	-	-
11	*Implement program to use PCA technique to reduce the number of features while retaining important information	-	-	-	✓	-
12	*Implement program to use machine learning model on given dataset	-	-	-	✓	✓
13	Implement program to use Pandas, Matplotlib to analyze data of previously created model	-	-	-	✓	✓
14	Implement program to use machine learning model on Boston Housing Dataset (available in Scikit-learn)	-	-	-	✓	✓
15	Implement a program to segment customers into different groups based on their purchasing behavior features using K-Means Clustering	-	-	-	-	✓

Guidelines to Teachers

1. Teacher is expected to refer complete curriculum document and follow guidelines for implementation before start of curriculum.
2. At the beginning teacher should make the students acquainted with any of the given Simulation software environment as few practical are based on simulation.
3. Teacher should provide the guideline with demonstration of practical to the students with all features.
4. Teacher shall explain prior concepts to the students before starting of each practical
5. Involve students in performance of each practical.
6. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
7. Teachers should give opportunity to students for hands on experience after the demonstration.
8. Teacher is expected to share the skills and competencies to be developed in the students.
9. 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.

Instructions for Students

1. Listen carefully the lecture given by teacher about course, curriculum, learning structure, skills to be developed.

2. Before performing the practical student shall read lab manual of related practical to be conducted.

3. Organize the work in the group and make record of all observations.

4. Students shall develop maintenance skill as expected by industries.

5. Student shall attempt to develop related hand-on skills and gain confidence.

6. Student shall develop the habits of evolving more ideas, innovations, skills etc.

7. Student shall refer technical magazines, IS codes and data books.

8. Student should develop habit to submit the practical on date and time.

9. Student should well prepare while submitting write-up of exercise.

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	*Installation of IDE with necessary libraries.					
2	*Implement program for Data Preprocessing Techniques.					
3	Implement program to read dataset (Text, CSV, JSON ,XML)					
4	Implement the classification algorithms on previously prepared dataset.					
5	*Implement the regression model by using the suitable dataset					
6	*Implement program to use clustering algorithms to find patterns in data.					
7	Implement program to identify the most important features that contribute to the model accuracy.					
8	Implement program to use k-Nearest Neighbors (KNN) model for Classification on given dataset					
9	*Implement program to train an SVM model on given dataset					

10	Implement program to use logistic regression model to classify binary outcomes					
11	*Implement program to use PCA technique to reduce the number of features while retaining important information.					
12	*Implement program to use machine learning model on given dataset.					
13	Implement program to use Pandas, Matplotlib to analyze data of previously created model					
14	Implement program to use machine learning model on Boston Housing Dataset (available in Scikit-learn)					
15	Implement a program to segment customers into different groups based on their purchasing behavior features using K-Means Clustering					
Total Marks						

***Total marks to be transferred to proforma published by MSBTE Note:**

- '*' Marked Practical_s (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: *Installation of IDE with necessary libraries

I Practical Significance

The installation of an Integrated Development Environment (IDE) with necessary libraries is a crucial step for efficient software development. It provides a unified platform where developers can write, debug, and execute code easily. By installing the required libraries, the IDE becomes equipped with tools and packages that simplify programming tasks, ensure compatibility, and enhance productivity. This practical activity helps learners understand how to set up a proper development environment, manage dependencies, and prepare systems for real-world coding and project implementation

Scikit-learn is a powerful and user friendly library that provides a wide range of tools for data preprocessing, model training, evaluation, and tuning all with a consistent and user-friendly interface. It supports numerous essential ML algorithms for tasks such as classification, regression, and clustering, allowing users to build models with just a few lines of code. The library is widely adopted in both academia and industry due to its strong performance, seamless integration with other Python libraries (like NumPy and Pandas), and support for real-world applications. Scikit-learn simplifies complex ML workflows, making it an ideal choice for beginners and professionals alike to experiment, prototype, and deploy machine learning models effectively

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO1: Explain the role of machine learning in AI and data science.

IV Laboratory Learning Outcome(s)

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

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

1. Install Python:

Download the latest version of Python from the official Python web Visit the official page **URL: https://www.python.org/downloads/** for Python on the Windows operating system

Run the installer and ensure to check the option "Add Python to PATH" during installation.

2. Choose and Install an IDE:

i) Anaconda: A popular distribution that includes Python, many common data science libraries, and the Spyder IDE. Download the Anaconda installer and follow the instructions.

URL: https://www.anaconda.com/

ii) PyCharm: A powerful and feature-rich IDE specifically for Python. Download the Community Edition (free) and install it.

URL: https://www.jetbrains.com/pycharm/download/?section=windows

iii) VS Code: A lightweight and versatile code editor that can be extended with Python extensions to function as an IDE.

Install VS Code and then install the Python extension from the marketplace

URL: https://code.visualstudio.com/docs/languages/python

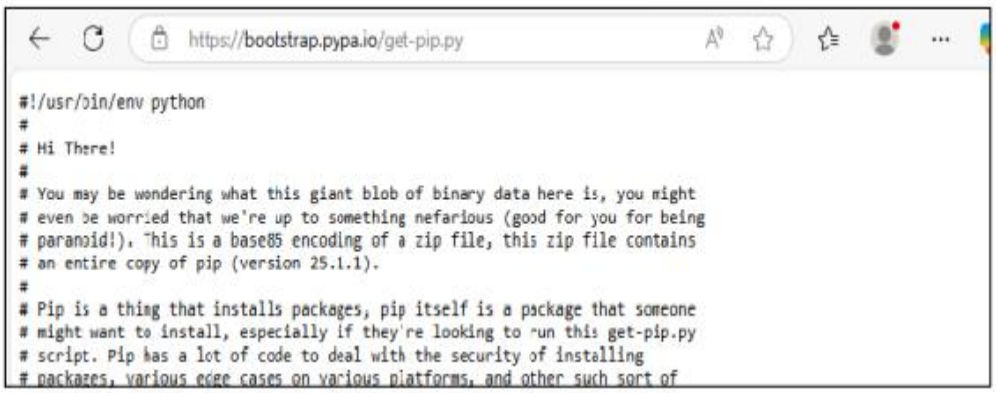
3. Install Machine Learning Libraries:

- Using pip (for PyCharm, VS Code, or a standalone Python installation):
- Open your terminal or command prompt.
- Use pip to install the desired libraries.
- Common libraries include: NumPy: For numerical operations.
- Pandas: For data manipulation and analysis.
- Matplotlib/Seaborn: For data visualization.

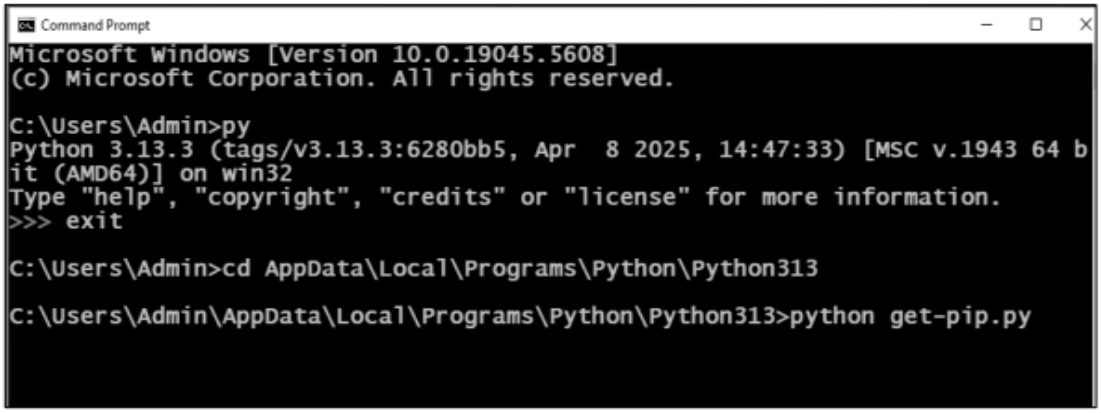
Scikit-learn: For various machine learning algorithms.

Install pip:

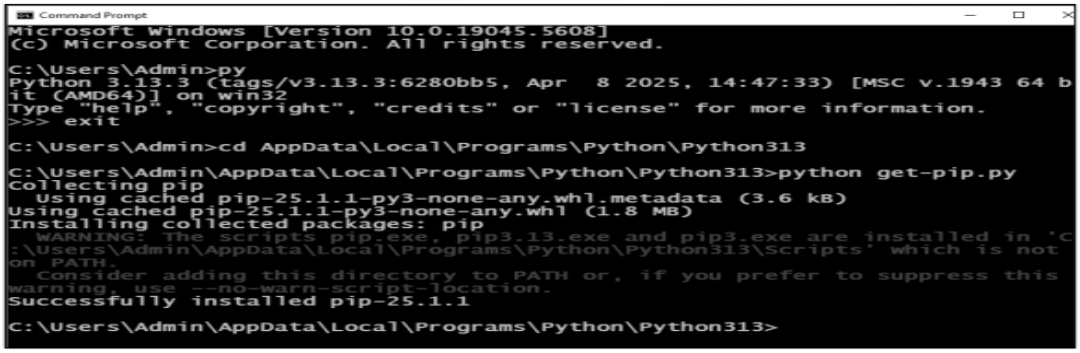
Download the get-pip.py file from https://bootstrap.pypa.io/get-pip.py and store it in the same directory as Python is installed that is wherever python.exe is available.



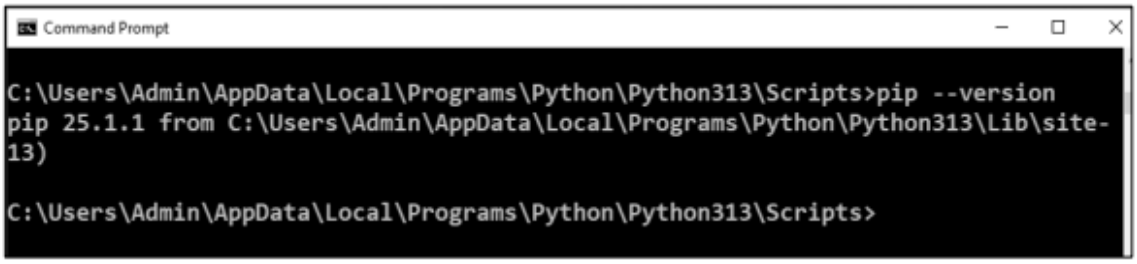
- Change the current path of the directory in the command prompt to the path of the directory where the above file exists.
- Execute the python get-pip.py command to Install pip



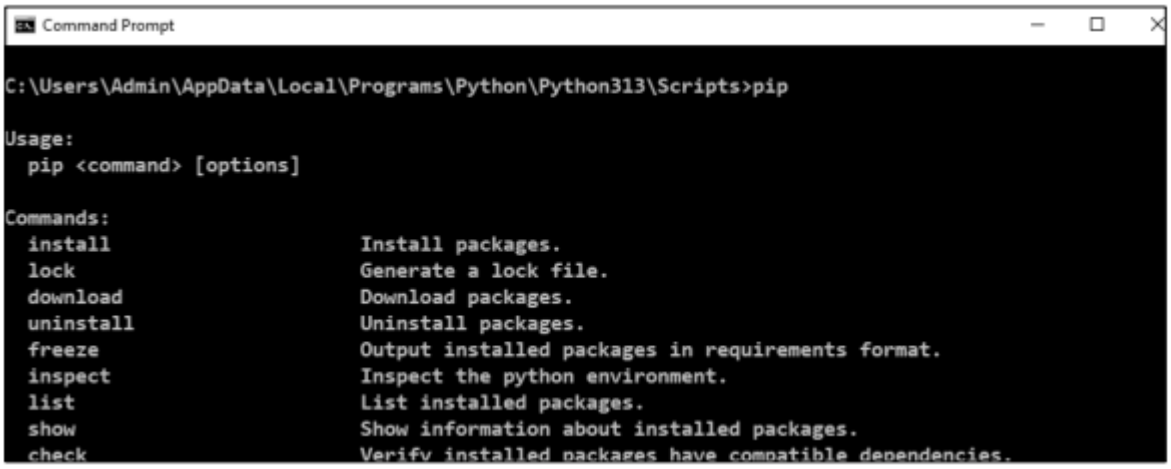
- Now wait through the installation process of pip. It takes some time



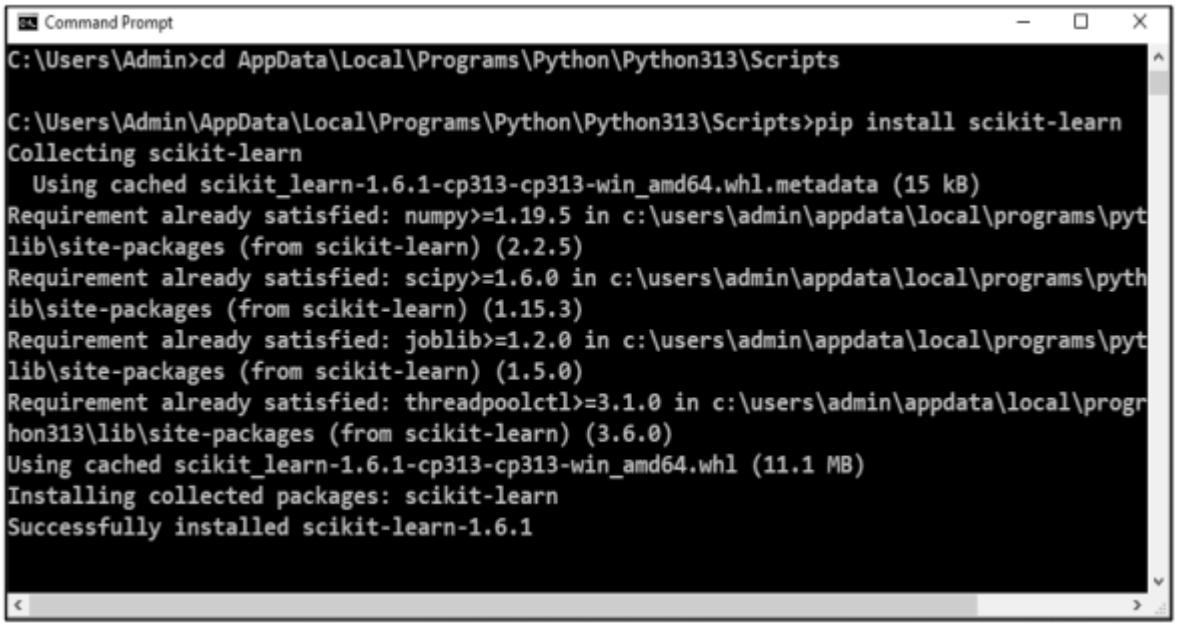
- Change the current path of the directory in the command prompt to the path of the directory where the pip.exe file exists.
- After completing above process, check the version of pip by using pip - -version command.



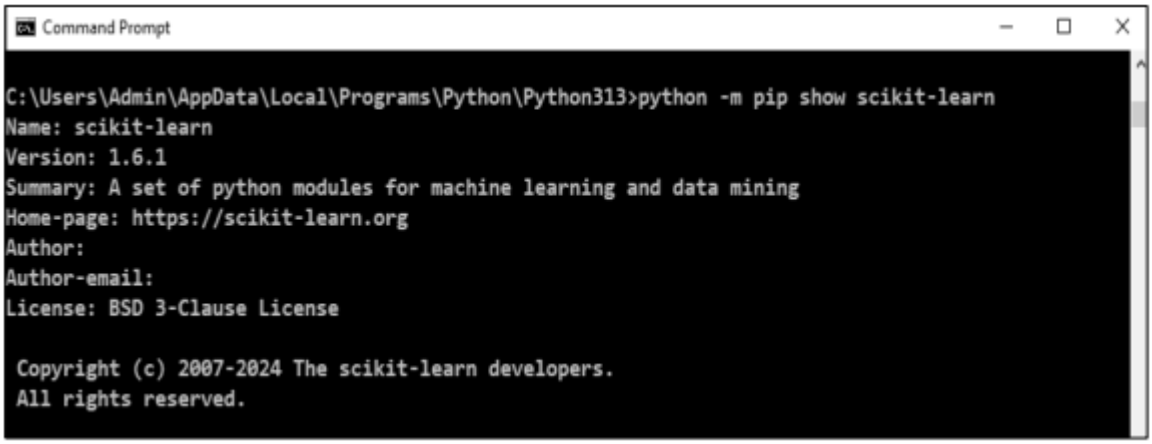
- You can use pip command to find syntax and other parameters for installing other packages or configuring them.



Installing scikit learn: Use pip install scikit-learn command to install scikit-learn library in python

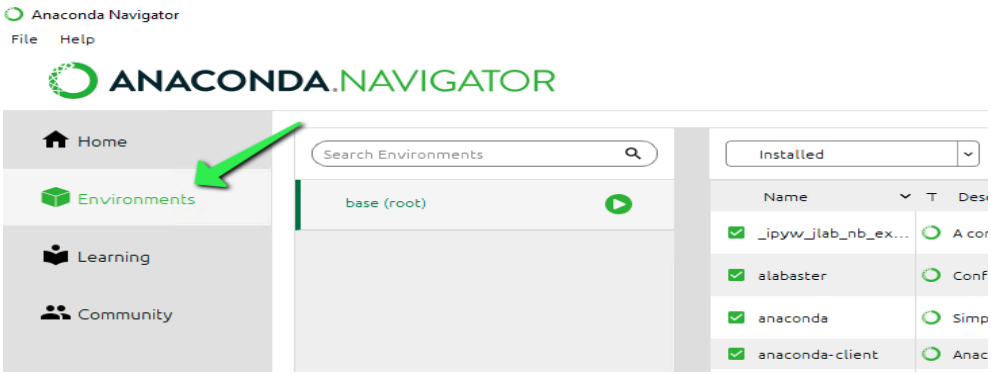


- Use python -m pip show scikit-learn to verify installation of scikit-learn library which will show following output.



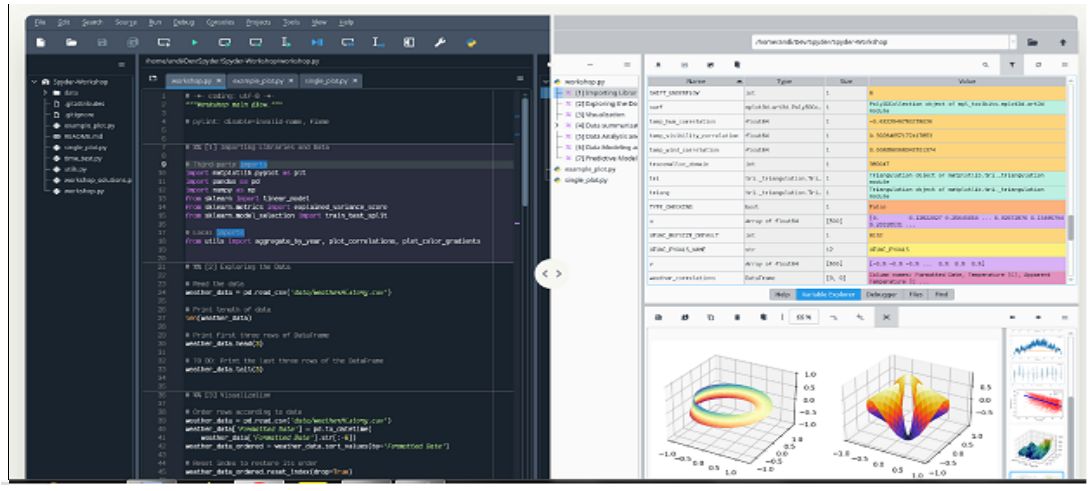
Using Anaconda Navigator (for Anaconda installations):

- Open Anaconda Navigator.
- Go to the "Environments" tab.
- Select your desired environment (e.g., base).
- Search for the libraries you need and install them through the graphical interface.



4. Verify Installation:

- Open your chosen IDE.



- Create a new Python file.

Import some of the installed libraries to ensure they are recognized. For example:

Python

```
import numpy as np
import pandas as pd
print("Libraries installed successfully!")
```

Run the script. If no errors occur, the libraries are correctly installed and accessible within your IDE.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook/,Jupyter Lab Google ColabFree/ unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

Procedure for Installation of IDE with Necessary Libraries

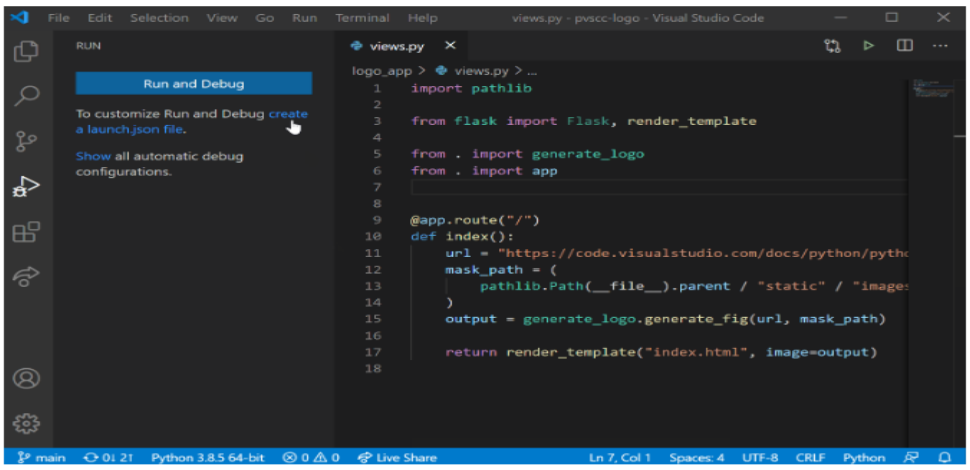
Step 1: Select an IDE

Choose an appropriate Integrated Development Environment (IDE) based on your programming language.
Examples:- Python → PyCharm, VS Code, Jupyter Notebook/Jupyter Lab

Step 2: Download the IDE

1.Visit the official website of the chosen IDE.
Example: https://code.visualstudio.com/ for VS Code

2. Download the installer suitable for your operating system (Windows/Linux/macOS)



Step 3: Install the IDE

1. Run the downloaded setup file.
2. Follow on-screen installation instructions (click Next, accept license agreement, and choose install location).
3. Once installation completes, launch the IDE.

Step 4: Install Necessary Libraries / Extensions

Depending on your programming needs:
- For Python (in VS Code or PyCharm):
Open terminal in the IDE and install libraries using pip commands:
pip install numpy pandas matplotlib

Step 5: Configure Environment

Set up path variables if required (for compilers or interpreters).
Verify installation by running a simple program (e.g., 'Hello World').

Step 6: Verify Setup

1. Open IDE and create a new project.
2. Write a sample code.
3. Run the program to ensure the IDE and libraries are working properly.

X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is an IDE?
2. What steps are involved in installing an IDE?
3. What is the use of the pip command in Python?
4. How can you check whether your IDE is correctly linked to a compiler or interpreter?
5. How can you verify if the IDE has been installed successfully?
6. What is the use of scikit learn library?
7. What are the Prerequisites for Installing Scikit-learn?
8. What are the applications of scikit learn library?
9. How to install pip? State its use.

Space for Answer

……

……

……

……

……

……

……

……

……

……

XII References:

- **https://www.python.org/downloads/release/python-31212/**
- **https://code.visualstudio.com/docs/?dv=win64user**
- **https://www.jetbrains.com/pycharm/download/?section=windows**
- **https://sourceforge.net/projects/jupyter-notebook.mirror/**
- https://machinelearningmastery.com/how-machine-learning-algorithms-work/
- **https://developers.google.com/machine-learning/crash-course/linear-regression**

XIII Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 2: *Implement program for Data Preprocessing Techniques

I Practical Significance

A data preprocessing program systematically cleans, transforms, and refines raw data into a high-quality, usable format, leading to more accurate models, reliable insights, and efficient analyses. For machine learning (ML) models, the impact of well-preprocessed data is particularly significant. Programs are essential for preparing data for consumption by ML algorithms.

Missing values are a common challenge in machine learning and data analysis. They occur when certain data points are missing for specific variables in a dataset. These gaps in information can take the form of blank cells, null values or special symbols like "NA", "NaN" or "unknown." If not addressed properly, missing values can harm the accuracy and reliability of our models. They can reduce the sample size, introduce bias and make it difficult to apply certain analysis techniques that require complete data. Efficiently handling missing values is important to ensure our machine learning models produce accurate and unbiased results.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO1: Explain the role of machine learning in AI and data science.

CO2: Implement data preprocessing.

IV Laboratory Learning Outcome(s)

LLO2.1 Write a program for handling missing values.

LLO2.2 Normalize data to make models work.

LLO2.3 Standardize data to make models work.

LLO2.4 Transform categorical data into numerical form using encoding methods.

LLO2.5 Split the dataset into train and test.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

Data preprocessing is the process of evaluating, filtering, manipulating, and encoding data so that a machine learning algorithm can understand it and use the resulting output. The major goal of data preprocessing is to eliminate data issues such as missing values, improve data quality, and make the data useful for machine learning purposes.

Data transformation, data reduction, feature selection, and feature scaling are all examples of data preprocessing approaches developers use to reorganize raw data into a format suitable for certain algorithms. This can significantly reduce the processing power and time necessary to train a new Machine learning or AI system or perform an inference against it.

Data preprocessing is the fast track to improving data quality since many of its steps mirror activities you'll find in any data quality management process, such as data cleansing, data profiling, data integration, and more.

1. Normalization:

Data normalization is a preprocessing method that resizes the range of feature values to a specific scale, usually between 0 and 1. It is a feature scaling technique used to transform data into a standard range. Normalization ensures that features with different scales or units contribute equally to the model and improves the performance of many machine learning algorithms.

Types of normalization:

a. Min-max normalization

b. Log normalization

c. Decimal scaling

d. Mean normalization (mean-centering)

2. Standardization:

Standardization, which is also called z-score scaling, transforms data to have a mean of 0 and a standard deviation of 1. This process adjusts the feature values by subtracting the mean and dividing by the standard deviation. You might have heard of 'centering and scaling' data. Well, standardization refers to the same thing: first centering, then scaling. **The formula for standardization is:**

$$X_{std} = \frac{X - \mu}{\sigma}$$

Where:

- X is the original value,
- u is the mean of the feature, and
- σ is the standard deviation of the feature.

This formula rescales the data in such a way that its distribution has a mean of 0 and a standard deviation of 1.

3. Encoding Categorical Data:

Categorical data represents discrete values or categories such as gender, country or product type. Machine learning algorithms require numerical input, making it essential to convert categorical data into a numerical format. This process is known as encoding. Categorical data is a common in many fields like marketing, finance and social sciences.

Types of Encoding:

1. Label Encoding
2. One-Hot Encoding
3. Ordinal Encoding

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook Jupyter, Notebook, Google ColabFree/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

Data preprocessing in machine learning involves a series of techniques to transform raw data into a clean, structured, and suitable format for model training. The general procedure includes:

1. Data Collection and Assessment:

Acquire the relevant dataset from various sources (e.g., databases, APIs, CSV files).

Perform an initial assessment to understand data characteristics, identify potential issues like missing values, inconsistent formats, or outliers.

2.Data Cleaning:

- **Handling Missing Values:** Address missing data points by imputation (e.g., mean, median, mode, or advanced techniques like K-Nearest Neighbors imputation) or by removing rows/columns with a high percentage of missing values if appropriate.
- **Handling Outliers:** Identify and manage extreme values (outliers) that can skew model performance. Techniques include statistical methods (Z-score, IQR), visualization (box plots), or domain-specific rules.
- **Handling Noisy Data:** Smooth out random errors or variances using techniques like binning, regression, or clustering.
- **Handling Duplicates:** Identify and remove duplicate records to ensure data integrity.

3. Data Transformation:

- **Encoding Categorical Variables:** Convert non-numeric categorical features into numerical representations understandable by machine learning models (e.g., One-Hot Encoding, Label Encoding, Ordinal Encoding).
- **Feature Scaling:** Standardize or normalize numerical features to bring them within a similar range, preventing features with larger magnitudes from dominating the learning process (e.g., Standardization using StandardScaler, Normalization using MinMaxScaler).
- **Feature Engineering:** Create new features from existing ones to improve model performance or capture more complex relationships.
- **Data Discretization:** Convert continuous numerical features into discrete categories or bins.

4. Data Reduction (Optional):

- **Dimensionality Reduction:** Reduce the number of features to mitigate the curse of dimensionality and improve computational efficiency (e.g., Principal Component Analysis (PCA)).
- **Data Compression:** Reduce the overall size of the dataset while retaining essential information.
- **Data Splitting:**

Divide the preprocessed dataset into training, validation (optional), and test sets to train the model, tune hyper parameters, and evaluate its performance on unseen data, respectively.

1 .Write a program for handling missing values.

```
import pandas as pd
import numpy as np

data = {
   'School ID': [101, 102, 103, np.nan, 105, 106, 107, 108],
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Frank', 'Grace', 'Henry'],
   'Address': ['123 Main St', '456 Oak Ave', '789 Pine Ln', '101 Elm St', np.nan, '222 Maple Rd', '444 Cedar Blvd', '555 Birch Dr'],
   'City': ['Los Angeles', 'New York', 'Houston', 'Los Angeles', 'Miami', np.nan, 'Houston', 'New York'],
   'Subject': ['Math', 'English', 'Science', 'Math', 'History', 'Math', 'Science', 'English'],
   'Marks': [85, 92, 78, 89, np.nan, 95, 80, 88],
   'Rank': [2, 1, 4, 3, 8, 1, 5, 3],
   'Grade': ['B', 'A', 'C', 'B', 'D', 'A', 'C', 'B']
}
df = pd.DataFrame(data)
print("Sample DataFrame:")
print(df)
```

Ouput: Sample data Frame with Missing value

```
Sample DataFrame:
   School ID     Name         Address         City  Subject  Marks  Rank Grade
0      101.0    Alice     123 Main St  Los Angeles     Math   85.0     2     B
1      102.0      Bob     456 Oak Ave     New York  English   92.0     1     A
2      103.0  Charlie     789 Pine Ln      Houston  Science   78.0     4     C
3        NaN    David      101 Elm St  Los Angeles     Math   89.0     3     B
4      105.0      Eva             NaN        Miami  History    NaN     8     D
5      106.0    Frank    222 Maple Rd          NaN     Math   95.0     1     A
6      107.0    Grace  444 Cedar Blvd      Houston  Science   80.0     5     C
7      108.0    Henry    555 Birch Dr     New York  English   88.0     3     B
```

Removing Rows with Missing Values:

Removing rows with missing values is a simple and straightforward method to handle missing data, used when we want to keep our analysis clean and minimize complexity.

```
df_cleaned = df.dropna()

print("\nDataFrame after removing rows with missing values:")
print(df_cleaned)
```

Ouput: Removing Rows with Missing Values

DataFrame after removing rows with missing values:

	School ID	Name	Address	City	Subject	Marks	Rank	Grade
0	101.0	Alice	123 Main St	Los Angeles	Math	85.0	2	B
1	102.0	Bob	456 Oak Ave	New York	English	92.0	1	A
2	103.0	Charlie	789 Pine Ln	Houston	Science	78.0	4	C
6	107.0	Grace	444 Cedar Blvd	Houston	Science	80.0	5	C
7	108.0	Henry	555 Birch Dr	New York	English	88.0	3	B

X Conclusion:

……

……

……

……

XI Practical related Questions:

1.What is data preprocessing, and why is it necessary in machine learning?

2.What are the main steps involved in data preprocessing?

3.Implement a preprocessing pipeline using Pipeline and ColumnTransformer.

4.Implement data preprocessing using ColumnTransformer for mixed data types.

5.Perform complete preprocessing on a dataset and show its transformation step-by-step using Python.

6.Write a program to remove outliers using the IQR (Interquartile Range) method.

XII Exercise:

1. Write a Python Program demonstrating Data Preprocessing Techniques.
2. Write a program for handling missing values

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XIII References:

- **https://www.geeksforgeeks.org/machine-learning/ml-handling-missing-values/**
- **https://developers.google.com/machine-learning/crash-course/numerical-data/normalization**
- **https://www.datacamp.com/tutorial/normalization-in-machine-learning**
- **https://www.datacamp.com/tutorial/normalization-vs-standardization**
- **https://www.geeksforgeeks.org/machine-learning/what-is-data-normalization/**
- **https://builtin.com/data-science/when-and-why-standardize-your-data**

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 3: Implement program to read dataset (Text, CSV, JSON , XML)

I Practical Significance

The practical significance of implementing a program to read datasets in formats like text, CSV, JSON, and XML for machine learning (ML) is that it enables the crucial data ingestion and preprocessing stage of the ML pipeline. Without the ability to read and parse data from diverse sources, ML projects cannot begin.

CSV (Comma-Separated Values): Simple, tabular format where data is arranged in rows and columns. Each row represents a record, and fields within a record are separated by a delimiter, typically a comma.

JSON (JavaScript Object Notation): Lightweight, human-readable, and machine-parsable format based on key-value pairs and arrays. Supports nested structures and various data types (strings, numbers, booleans, objects, arrays).

XML (Extensible Markup Language): Markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. Uses tags and attributes to define elements and their relationships, supporting hierarchical and complex data structures.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO2: Implement data preprocessing.

IV Laboratory Learning Outcome(s)

LLO3.1 Write a program to read dataset and differentiate attributes in various category.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

1.File Input/Output (I/O) Fundamentals

This is the most basic layer dealing with accessing the physical file.

- **File Abstraction:** The operating system presents a file as an abstract sequence of bytes. A program requests access obtaining a file handles or file stream.
- **Reading:** The program uses I/O functions (e.g., open, read, close) to read these bytes, usually into an in-memory buffer. Dataset reading typically employs sequential access, reading from the start to the end of the file.

- **Encoding:** The program must handle character encoding (like ASCII or UTF-8) to translate the raw bytes into meaningful characters. UTF-8 is the modern standard for supporting a wide range of global characters.

2. Data Structures and Mapping

The goal of reading a dataset is to transform the external file structure into a useful internal data structure (an **Abstract Data Type**).

3. Data Parsing Techniques

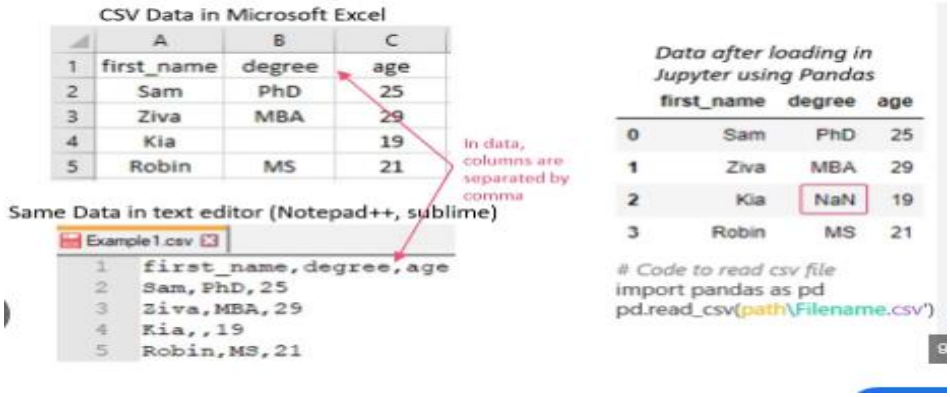
Parsing is the process of analyzing the stream of data to construct the required in-memory structure according to the format's rules.

A. Plain Text and CSV (Delimited Data)

Focus: Identifying **delimiters**.

- **Line Delimiter:** Marks the end of a record (e.g., $\text{'\textbackslash n'}$).
- **Field Delimiter:** Separates columns/fields within a record (e.g., comma, tab).

Mechanism: Involves **tokenization** (breaking the line into tokens based on the field delimiter) and handling **quoting/escaping** (where the delimiter might appear inside a field value, typically enclosed in quotes).

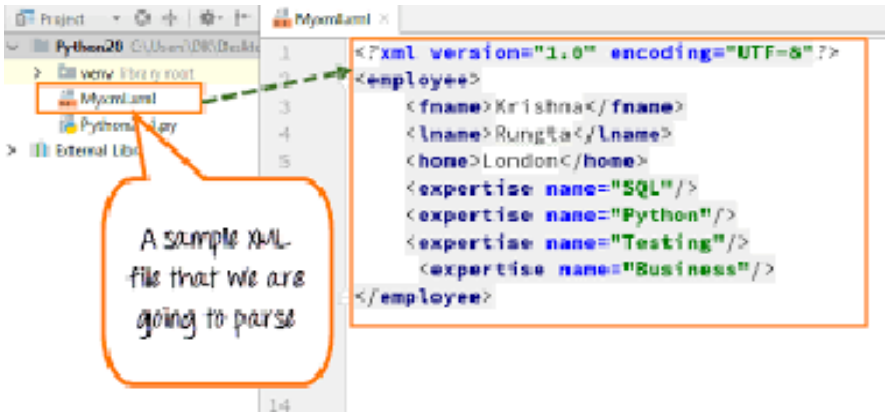


B. JSON (Key-Value/Array Data)

- **Theoretical Basis: Serialization/Deserialization**. The JSON file is a serialized string representation of data. The parser must **deserialize** it back into native programming language objects.
- **Process:** The parser performs **Lexical Analysis** (identifying tokens like `{`, `[`, `:`, string values, numbers) followed by **Syntactic Analysis** to ensure the tokens follow the JSON grammar (e.g., all objects start with `{` and end with `}`). This naturally builds the required nested dictionary/list structure.

C. XML (Hierarchical Markup)

- **Theoretical Basis:** XML is fundamentally a **tree structure** defined by parent-child element relationships.
- **Parsing Approaches:**
  1. **DOM (Document Object Model):** The entire XML file is read and loaded into memory as a **traversable tree object**. This allows for easy **random access** and modification but is memory-intensive for large files.
  2. **SAX (Simple API for XML):** An **event-driven** parser. It reads the file sequentially and generates events (e.g., "start element: book," "characters: title," "end element: book"). It is highly efficient for **very large files** because it only processes one small chunk at a time, keeping memory usage low.



4. Error Handling and Robustness

A robust program must account for issues that deviate from the format's grammar:

- **Syntax Errors:** Malformed elements (e.g., unclosed brackets in JSON, mismatched quotes in CSV).
- **Encoding Errors:** Invalid byte sequences for the specified encoding.
- **Validation:** For XML, this may involve checking the structure against a schema (**DTD** or **XML Schema**) to ensure the data is semantically correct, not just syntactically valid.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook Jupyter, Notebook, Google ColabFree/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

Here are Python implementations to read various dataset formats: Text, CSV, JSON, and XML.

1. Reading a Text File (.txt)

```
def read_text_file(filepath):
  try:
     with open(filepath, 'r') as file:
        content = file.read()
        print("Text file content:")
        print(content)
  except FileNotFoundError:
     print(f"Error: File not found at {filepath}")

# Example usage:
# Create a dummy text file
with open("sample.txt", "w") as f:
   f.write("This is a sample text file.\n")
   f.write("It contains multiple lines of text.")

read_text_file("sample.txt")
```

2. Reading a CSV File (.csv)

```
import csv

def read_csv_file_csv_module(filepath):
    try:
        with open(filepath, 'r', newline='') as file:
            reader = csv.reader(file)
            print("\nCSV file content (using csv module):")
            for row in reader:
                print(row)
    except FileNotFoundError:
        print(f"Error: File not found at {filepath}")

# Example usage:
# Create a dummy CSV file
with open("sample.csv", "w", newline='') as f:
    writer = csv.writer(f)
    writer.writerow(["Name", "Age", "City"])
    writer.writerow(["Alice", 30, "New York"])
    writer.writerow(["Bob", 24, "London"])

read_csv_file_csv_module("sample.csv")
```

3. Reading a JSON File (.json)

```
import json

def read_json_file(filepath):
    try:
        with open(filepath, 'r') as file:
            data = json.load(file)
            print("\nJSON file content:")
            print(data)
    except FileNotFoundError:
        print(f"Error: File not found at {filepath}")

# Example usage:
# Create a dummy JSON file
with open("sample.json", "w") as f:
    json.dump({"name": "Charlie", "age": 35, "occupation": "Engineer"}, f)

read_json_file("sample.json")
```

4. Reading an XML File (.xml)

```
import xml.etree.ElementTree as ET

def read_xml_file(filepath):
    try:
        tree = ET.parse(filepath)
        root = tree.getroot()
        print("\nXML file content:")
        for child in root: print(f"Tag: {child.tag}, Attributes: {child.attrib}, Text: {child.text.strip()}")
    except FileNotFoundError:
        print(f"Error: File not found at {filepath}")
    except ET.ParseError as e:
```

```
        print(f"Error parsing XML file: {e}")

# Example usage:
# Create a dummy XML file
with open("sample.xml", "w") as f:
    f.write("<data>\n")
    f.write("  <person id='1'>\n")
    f.write("    <name>David</name>\n")
    f.write("    <age>40</age>\n")
    f.write("  </person>\n")
    f.write("  <person id='2'>\n")
    f.write("    <name>Eve</name>\n")
    f.write("    <age>28</age>\n")
    f.write("  </person>\n")
    f.write("</data>\n")

read_xml_file("sample.xml")
```

IX Conclusion:

……

……

……

……

X Practical related Questions:

1. What is a dataset and why is it important in data analysis?
2. How do you read a .txt file using Python?
3. Write a Python program to read a .csv file and display the first 5 rows.
4. Write a program to read an XML file and extract specific tag values.
5. What is the difference between JSON and XML in terms of structure and use?
6. Write a program to read an XML file and extract specific tag values.

XI Exercise:

1. Write a Python program to read and display data from four different files and print the contents or summary of each.

- data.txt
- data.csv
- data.json
- data.xml

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XII References:

- https://www.w3schools.com/python/pandas/pandas_csv.asp
- https://www.kaggle.com/code/hamelg/python-for-data-10-reading-and-writing-data/notebook
- https://www.geeksforgeeks.org/python/how-to-read-data-files-in-python/
- https://www.coursera.org/learn/machine-learning-with-python
- https://machinelearningmastery.com/how-machine-learning-algo rithms-work/
- https://pandas.pydata.org/docs/reference/api/pandas.read_xml.html
- https://datascientyst.com/read-json-files-pandas/

XIII Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No.4: Implement the classification algorithms on previously prepared dataset

I Practical Significance

Implementing classification algorithms on a previously prepared dataset depends in its ability to derive actionable insights and automate decision-making in real-world applications. The raw data is transformed into a predictive, quantifiable, and actionable intelligence system.

Decision tree: is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes. A Decision Tree helps us to make decisions by mapping out different choices and their possible outcomes. It's used in machine learning for tasks like classification and prediction

K-nearest neighbors (KNN): KNN is a supervised machine learning algorithm used for both classification and regression that classifies a new data point based on the majority class of its "k" nearest neighbors or predicts a value by averaging the values of its "k" nearest neighbors

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO2: Implement data preprocessing.

CO3: Implement feature engineering techniques to prepare data for machine learning models.

CO4: Apply supervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO4.1 Write a program to implement the Decision Tree model.
LLO4.2 Write a program to implement the K-Nearest Neighbor model.
LLO 4.3 Evaluate classification performance using accuracy, precision, recall, and F1-score .

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

Implementing classification algorithms on a prepared dataset involves a systematic process of selecting an appropriate model, training it on your data, evaluating its performance, and fine-tuning it for optimal results. The theoretical framework relies on supervised learning, where a model learns from labeled data to predict categories for new and unseen data.

The classification algorithms on previously prepared dataset are

1. Logistic Regression (LR)
2. Decision Tree (DT)
3. K-Nearest Neighbors (KNN)

The steps to implement above classification algorithms on the dataset are as follows:

 Split the dataset: Separate your prepared dataset into training and testing sets. A common split is 70-80% for training and 20-30% for testing. The training set is used to train the model, and the test set evaluates its performance on new and unseen data.

 Choose a classification model: The selection of an algorithm depends on the nature of your data and the problem you are solving. Consider the data size, feature complexity, and the need for interpretability.

 Train the model: Feed the training data to the algorithm which learns the patterns and relationships between the input features (X) and the output labels (y)

 Evaluate the model: Use the test set to evaluate the trained model's performance based on specific metrics. A high-performing model accurately predicts the labels in the test set.

 Hyperparameter tuning: Adjust the model's internal settings (hyperparameters) to optimize its performance. This is an iterative process of finding the best configuration for your data.

 Refine the process: The process of model selection, training, and evaluation is often repeated to achieve the best possible results.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google ColabFree/ unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

With the dataset prepared, we can proceed with implementing various classification algorithms.

Splitting data on train dataset:

```
import numpy as np
from sklearn.model_selection import train_test_split

# Example data (replace with your actual data)
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])
y = np.array([0, 1, 0, 1, 0, 1])

#Splitting the data into train-test split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y)
```

1. Logistic Regression:

Logistic Regression is a fundamental algorithm for binary and multi-class classification. It models the probability that an instance belongs to a particular class using the logistic function. The decision boundary is a linear combination of input features, and predictions are based on a threshold.

```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


X, y = load_breast_cancer(return_X_y=True)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=23)
```

```
clf = LogisticRegression(max_iter=10000, random_state=0)
clf.fit(X_train, y_train)


acc = accuracy_score(y_test, clf.predict(X_test)) * 100
print(f"Logistic Regression model accuracy: {acc:.2f}%")
```

2.KNN: KNN is a simple and intuitive algorithm that classifies a data point based on the majority class among its k-nearest neighbors. The choice of 'k' influences the decision boundary, and it's crucial to select an optimal value for better performance.

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
# 1. Load a dataset (e.g., Iris dataset)
iris = load_iris()
X = iris.data  # Features
y = iris.target # Target labels

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

# 3. Instantiate the KNeighborsClassifier
# n_neighbors specifies the 'k' in KNN
knn = KNeighborsClassifier(n_neighbors=5)

# 4. Train the model using the training data
knn.fit(X_train, y_train)

# 5. Make predictions on the test data
y_pred = knn.predict(X_test)

# 6. Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of KNN model: {accuracy:.2f}")

# You can also predict for a new, unseen data point
new_point = np.array([[5.1, 3.5, 1.4, 0.2]]) # Example new data point
prediction = knn.predict(new_point)
print(f"Prediction for new point: {iris.target_names[prediction[0]]}")
```

3. Decision Trees: Decision Trees recursively split the dataset based on features, creating a tree-like structure. Each internal node represents a decision based on a feature, and each leaf node represents the predicted class. Decision Trees can capture complex relationships in data.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import datasets

# Load a sample dataset (e.g., Iris dataset)
iris = datasets.load_iris()
X = iris.data  # Features
y = iris.target # Target variable

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

# Create a Decision Tree Classifier object
# You can specify parameters like criterion ('gini' or 'entropy'), max_depth, etc.
dtree = DecisionTreeClassifier(criterion='entropy', random_state=42)

# Train the Decision Tree Classifier
dtree.fit(X_train, y_train)

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

# Evaluate the model (e.g., using accuracy score)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
# graph.render("iris_decision_tree") # Saves the tree visualization to a file
```

X Conclusion:

……

……

……

……

XI Practical Related Questions:

1. How do you evaluate the performance of model?
2. How you handle missing values, Outliers and Duplicated rows?
3. What is the null accuracy?
4. What is the gap between the model's accuracy on the Training Set and the Test Set?
5. Which method will be used to find the optimal model settings?
6. If the model's training accuracy is much higher than its test accuracy, what steps should be taken (e.g., adding regularization for Logistic Regression/SVM, or reducing max depth for Decision Trees)?

XII Exercise:

1. Write a program to implement the Decision Tree mode.
2. Write a program to implement the K-Nearest Neighbor model.
3. Evaluate classification performance using accuracy, precision, recall, and F1- score.

Space for Answer

……

……

XIII References:

- https://www.geeksforgeeks.org/machine-learning/top-6-machine-learning-algorithms-for-classification/
- https://medium.com/@devangchavan0204/understanding-classification-in-machine-learning-a-hands-on-project-f84167bd8cf5
- https://scikit-learn.org/stable/modules/tree.html
- https://www.geeksforgeeks.org/machine-learning/decision-tree/
- https://www.ibm.com/think/topics/decision-trees
- https://www.geeksforgeeks.org/machine-learning/decision-tree-introduction-example/

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No.5: *Implement the regression model by using the suitable dataset

I Practical Significance

Regression analysis helps us understand and predict the relationship between a *dependent variable* (target) and one or more *independent variables* (features).

A regression model is used to predict continuous numeric values based on input variables. It learns patterns from data and estimates how changes in input features affect the target variable.

Regression modeling helps in predicting continuous outcomes based on input variables.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO2: Implement data preprocessing.

CO4: Apply supervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 5.1 Write a program to implement the Linear Regression by using the suitable dataset.

LLO 5.2 Write a program to implement the logistic regression by using the suitable dataset.

LLO 5.3 Write a program to implement the Ridge Regression by using the suitable dataset.

LLO 5.4 Evaluate model performance using metrics.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background:

A regression model is a type of supervised learning algorithm that predicts a continuous numerical output based on one or more input variables (features).

Mathematically, the relationship is modelled as:

$$y=f(X)+\varepsilon$$

Where:

y: Target/dependent variable (e.g., house price)

X: Independent variables/features (e.g., income, rooms, location)

f(X): Regression function that estimates y

ε: Random error term (accounts for noise)

1. Linear Regression: If there is only one independent variable, the model becomes:

$$y=\beta0+\beta1x+\varepsilon$$

Where:

- β0: Intercept (value of y when x=0)
- β1: Slope (change in y per unit change in x)
- ε: Random error

2. Multiple Linear Regression

When there are two or more independent variables, we use:

$$y=\beta0+\beta1x1+\beta2x2+...+\beta n\ xn+\ \varepsilon$$

In matrix form:

$$y=X\beta+\varepsilon$$

3. **Logistic Regression:** Logistic Regression is a supervised learning algorithm used for classification problems. It predicts the probability of a categorical dependent variable. In binary classification, the output is either 0 or 1. It uses the sigmoid function to map predicted values to probabilities

1. Mathematical Representation:

The logistic regression model uses the sigmoid function to predict probabilities:

$$P(Y=1|X) = 1 / (1 + e^{-(b0 + b1*X1 + b2*X2 + ... + bn*Xn)})$$

2. Steps to Implement Logistic Regression

- Import necessary libraries
- Load a suitable dataset
- Split the data into training and testing sets
- Train the logistic regression model
- Make predictions
- Evaluate model performance

3. Ridge Regression: Ridge Regression is a regularized version of Linear Regression that adds a penalty term to the loss function to prevent over fitting. It is useful when predictors are highly correlated or when the model has too many features.

1. Mathematical Representation:

The Ridge Regression objective function is:

Minimize: $\Sigma(y_i - \hat{y}_i)^2 + \lambda \Sigma\beta^2$

Where:
- y_i = actual output
- $\hat{y}_i$ = predicted output
- λ = regularization parameter (controls penalty strength)
- β= model coefficients

When $\lambda = 0$, Ridge Regression becomes equivalent to Linear Regression.

2. Steps to Implement Ridge Regression

- Import required libraries
- Load a suitable dataset
- Split the data into training and testing sets
- Train the Ridge Regression model
- Make predictions on test data
- Evaluate the model performance

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter, Notebook, Google ColabFree/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

1. Tools and Libraries Required

Python (Recommended IDE: Jupyter Notebook / Google Colab / VS Code)

2.Libraries:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Step 1: Load the Dataset

Choose a dataset related to regression (e.g., House Price Prediction, Salary Data, etc.)

```
np.random.seed(42) # for reproducibility
X = 2 * np.random.rand(100, 1) # Independent variable
y = 4 + 3 * X + np.random.randn(100, 1) # Dependent variable with some noise
```

Step 2: Data Preprocessing

- Check for missing values
- Handle categorical data (if any)
- Remove irrelevant columns

```
print(data.info())
Data=data.dropna()      #Removing Missing Values
```

Step 3: Define Features (X) and Target (Y)

```
X=data [['feature1', 'feature2', 'feature3']]      # Independent variable
y= data ['target']                                 # Dependent variable
```

Step 4: Split the Dataset

Split the data into training and testing sets.

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

Step 5: Train the Regression Model

Use a suitable regression model — for example, Linear Regression

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Step 6: Make Predictions

```
y_pred = model.predict(X_test)
```

Step 7: Evaluate the Model

Use regression metrics such as Mean Squared Error (MSE) and R^2 Score.

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

    print(f"Intercept: {model.intercept_[0]:.2f}")
print(f"Coefficient: {model.coef_[0][0]:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

Step 8: Visualization

Plot the actual vs predicted values for better understanding.

```
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression')
plt.legend()
plt.show()
```

3. Result

The regression model is successfully implemented and evaluated using the given dataset. The accuracy and performance of the model are measured using MSE and R^2 score.

Source code:

Problem Statement: Write a program to implement the Linear Regression by using the suitable dataset.

Step to be followed:

1. Data Loading: Reads the dataset (e.g., Salary_Data.csv) using pandas.
2. Feature Selection: Defines X (input features) and Y (target variable).
3. Data Splitting: Divides data into training (80%) and testing (20%) sets.
4. Model Training: Fits the linear regression model using training data.
5. Prediction: Uses the trained model to predict test data.
6. Evaluation: Calculates performance metrics (MSE and R^2 Score).
7. Visualization: Plots a regression line showing actual vs predicted values.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 1. Generate a synthetic dataset
# We'll create a dataset with a clear linear relationship for demonstration.
np.random.seed(0) # for reproducibility
X = 2 * np.random.rand(100, 1) # 100 data points, 1 feature
y = 4 + 3 * X + np.random.randn(100, 1) # y = 4 + 3x + noise

# 2. Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Create and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# 4. Make predictions on the test set
y_pred = model.predict(X_test)

# 5. Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Model Intercept: {model.intercept_[0]:.2f}")
print(f"Model Coefficient (Slope): {model.coef_[0][0]:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

# 6. Visualize the results
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
```

```
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.title('Simple Linear Regression')
plt.xlabel('X (Independent Variable)')
plt.ylabel('y (Dependent Variable)')
plt.legend()
plt.grid(True)
plt.show()
```

Output:

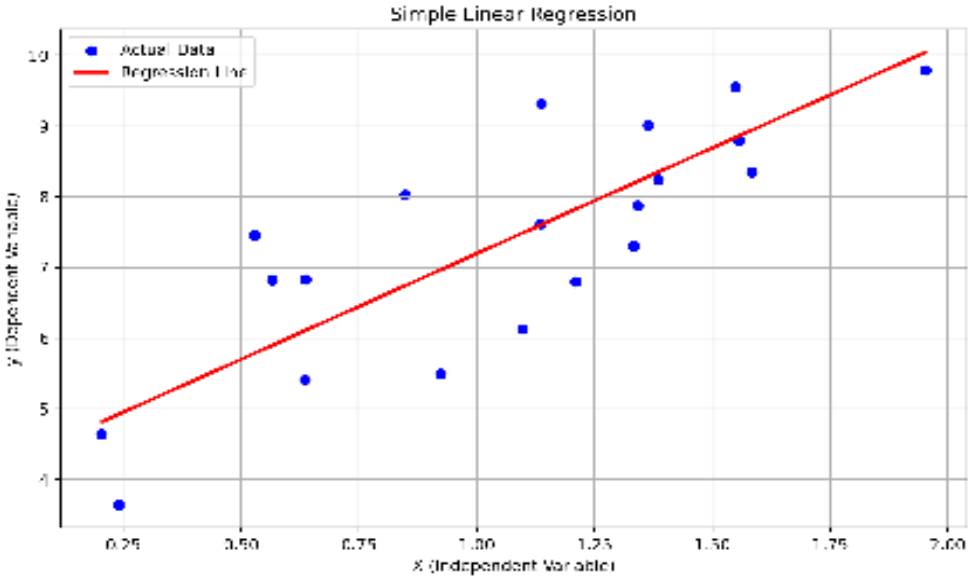
Model Intercept: 4.21

Model Coefficient (Slope): 2.99

Mean Squared Error (MSE): 0.92

R-squared (R2)

Score: 0.65



X Conclusion:

……

……

……

……

XI Practical Related Questions:

1. What is regression in machine learning?
2. What is the difference between regression and classification?
3. Give some real-life examples where regression analysis can be applied.
4. What type of data is used in regression problems?
5. Which regression algorithm did you use in your program and why?
6. What does the fit() and predict() method do in scikit-learn?
7. What is Mean Squared Error (MSE)?

XII Exercise:

1. Develop a program to demonstrate the working of Linear Regression. Use Boston Housing Dataset for Linear Regression.
2. Write a program to implement the Linear Regression by using the suitable dataset.
3. Write a program to implement the logistic regression by using the suitable dataset.
4. Write a program to implement the Ridge Regression by using the suitable dataset.
5. Evaluate model performance using metrics.

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

……

……

……

……

XIII References:

- https://machinelearningmastery.com/how-machine-learning-algorithms-work/
- https://developers.google.com/machine-learning/crash-course/linear-regression
- https://www.geeksforgeeks.org/machine-learning/
- https://www.w3schools.com/python/python_ml_getting_started.asp
- https://www.kdnuggets.com/5-free-courses-to-master-machine-l earning

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No.6: *Implement program to use clustering algorithms to find patterns in data

I Practical Significance

Clustering algorithms are unsupervised learning techniques used to identify hidden patterns or natural groupings in data. They play a vital role in data analysis, pattern recognition, and decision-making processes. Clustering helps uncover natural groupings or patterns within large datasets without prior labeling.

Clustering algorithms provide a powerful tool for pattern discovery in unlabeled datasets. They support informed decision-making, predictive analysis, and knowledge discovery.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO3: Implement feature engineering techniques to prepare data for machine learning models.

CO4: Apply supervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 6.1 Write a program to use clustering algorithms to find patterns in data.

LLO 6.2 Visualize the results using Matplotlib / Seaborn.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

Clustering algorithms are unsupervised machine learning techniques used to group data points into clusters based on their inherent similarities. These algorithms aim to uncover hidden patterns and structures within unlabeled datasets.

1. K-Means Clustering: A centroid-based algorithm that partitions data into a predefined number of 'k' clusters.

It works by iteratively assigning data points to the nearest cluster centroid and then recalculating the centroid based on the mean of the assigned points.

Example: Segmenting customers into groups based on purchasing behavior.

2. **Hierarchical Clustering:**

Builds a hierarchy of clusters by either iteratively merging (agglomerative) or splitting (divisive) clusters.

Results in a tree-like structure called a dendrogram, which can be cut at different levels to obtain desired cluster numbers.

Example: Grouping biological species based on genetic similarities.

3.Visualize the results:

Visualizing results using Matplotlib and Seaborn in Python involves importing the libraries, preparing the data, and then using their respective functions to generate various plot types.

1. Import Libraries: Begin by importing matplotlib.pyplot and seaborn.

. **2. Set Seaborn Style:** For enhanced aesthetics, you can apply Seaborn's default style.

```
sns.set_theme() # Or sns.set_style("whitegrid"),
sns.set_palette("viridis") etc.
```

3. Visualizations Example:

1. Scatter Plot (Matplotlib & Seaborn)
2. Bar Plot (Seaborn)
3. Histogram/KDE Plot (Seaborn)

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google ColabFree /unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

1. Implementation of K-Means Clustering:

We will be using blobs datasets and show how clusters are made using Python programming language.

Step 1: Importing the necessary libraries

We will be importing the following libraries.

- Numpy **:** for numerical operations (e.g., distance calculation).
- Matplotlib: for plotting data and results.
- Scikit learn**:** to create a synthetic dataset using **make_blobs**

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

Step 2: Generate sample data

```
# We create a synthetic dataset with 3 distinct "blobs" or clusters
X, y_true = make_blobs(n_samples=300, centers=3, cluster_std=0.60, random_state=0)
```

Step 3: Initialize and fit the KMeans model

```
# n_clusters: The number of clusters to form.
# random_state: Used for reproducibility of the initial centroid placement.
# n_init: 'auto' means the number of initializations to run will be chosen automatically.
kmeans = KMeans(n_clusters=3, random_state=0, n_init='auto')
kmeans.fit(X)
```

Step 4: Get the cluster labels and centroids

```
y_kmeans = kmeans.labels_  # Labels for each data point
centers = kmeans.cluster_centers_  # Coordinates of the cluster centroids
```

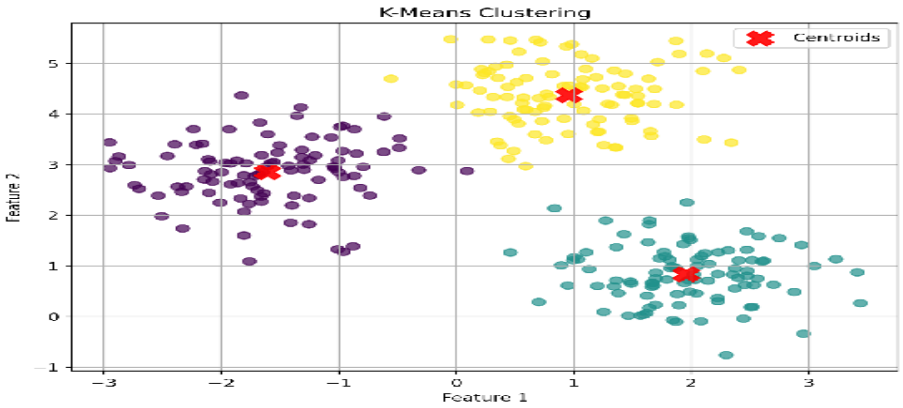
Step 5: Visualize the results

```
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis', alpha=0.7)
# Plot data points colored by cluster
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.9, marker='X', label='Centroids')
 # Plot centroids
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.grid(True)
plt.show()
```

Step 6: Predict cluster for new data points

```
new_data_points = np.array([[0, 0], [10, 10], [5, 5]])
predicted_clusters = kmeans.predict(new_data_points)
print(f"Predicted clusters for new data points: {predicted_clusters}")
```

Output:



Predicted clusters for new data points: [1 2 2]

2. Steps for Visualization:

1. Prepare your data:

Ensure your data is in a suitable format (e.g., NumPy arrays, Pandas DataFrames).

2. Choose the appropriate plot type:

Select a plot that effectively conveys the insights you want to highlight (e.g., scatter for relationships, bar for comparisons, histogram for distributions).

3. Generate the plot:

Use Matplotlib's plt.plot(), plt.scatter(), plt.bar() or Seaborn's high-level functions like sns.scatterplot(), sns.barplot(), sns.histplot(), sns.lineplot(), sns.heatmap() etc.

4. Customize (optional):

Add titles, labels, legends, change colors, adjust plot size, etc., using Matplotlib's functions (e.g., plt.title(), plt.xlabel(), plt.ylabel(), plt.legend(), plt.figure(figsize=...)).

5. Display the plot:

Use plt.show().

X Conclusion:

……

……

……

……

XI Practical Related Questions:

1. What is clustering in machine learning, and how is it different from classification?
2. How do you decide the number of clusters (k) in K-Means clustering?
3. What are the steps involved in implementing the K-Means clustering algorithm?
4. How can you visualize clusters in 2D or 3D using Python libraries like Matplotlib or Seaborn?
5. How does hierarchical clustering differ from K-Means clustering?
6. How can clustering help in real-world applications such as customer segmentation or image compression?
7. What is the difference between hard clustering and soft clustering?

XII Exercise:

1. Write a program to use clustering algorithms to find patterns in data.
2. Write a program to implement K-Means Clustering algorithms
3. Visualize the results using Matplotlib / Seaborn.

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

XIII References:

- https://ml-course.github.io/master/labs/Lab%201%20-%20Tutorial.html
- https://www.geeksforgeeks.org/machine-learning/clustering-in-machine-learning/
- https://developers.google.com/machine-learning/crash-course/classification
- https://machinelearningmastery.com/how-machine-learning-algorithms-work/
- https://www.tutorialspoint.com/machine_learning/machine_learning_k_means_clustering.htm

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No.7: Implement program to identify the most important features that contribute to the model accuracy

I Practical Significance

In machine learning, identifying the most important features means determining which input variables (features) have the strongest influence on the output or prediction of a model. This process is called feature importance analysis or feature selection. It helps improve the model's accuracy, efficiency, and interpretability by focusing only on the most impactful data attributes.

Data preprocessing involves cleaning, transforming, and organizing raw data to make it suitable for analysis or machine learning models. A typical implementation involves data collection cleaning, and integration, followed by transformation, reduction, and encoding of categorical variables**,** before finally splitting the dataset into training and testing sets.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO2: Implement data preprocessing.

CO3: Implement feature engineering techniques to prepare data for machine learning models.

IV Laboratory Learning Outcome(s)

LLO 7.1 Write a program to identify the most important features that contributes to the model accuracy.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

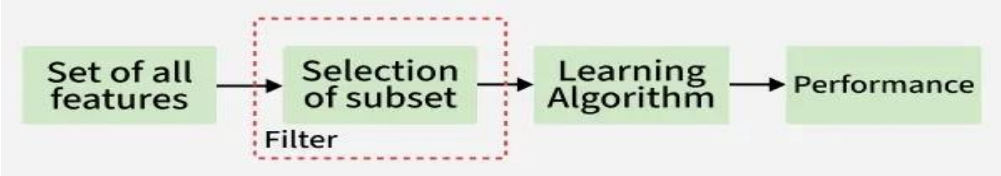
VI Relevant Theoretical Background

Feature selection is a core step in preparing data for machine learning where the goal is to identify and keep only the input features that contribute most to accurate predictions. By focusing on the most relevant variables, feature selection helps build models that are simpler, faster, less prone to overfitting and easier to interpret especially when we use datasets containing many features, some of which may be irrelevant or redundant**.**

Model accuracy is a statistic that assesses how frequently a model predicts a task's outcome accurately. It is the proportion of correctly predicted events to all predicted events. The accuracy of classification models, where the aim is to predict a category label for each input instance, is frequently used as a performance statistic in machine learning.

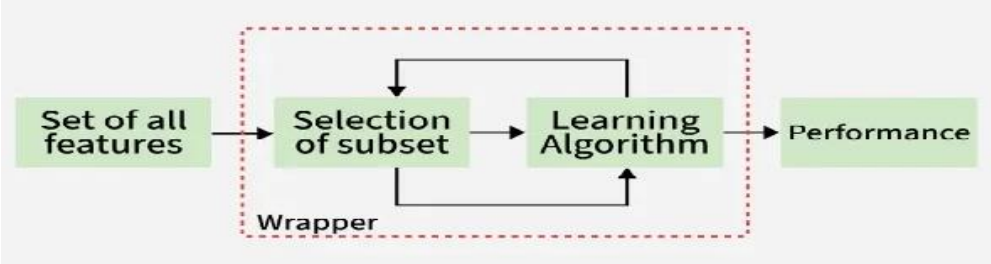
1. Filter Methods

Filter methods evaluate each feature independently with target variable. Feature with high correlation with target variable are selected as it means this feature has some relation and can help us in making predictions. These methods are used in the pre-processing phase to remove irrelevant or redundant features based on statistical tests (correlation) or other criteria.



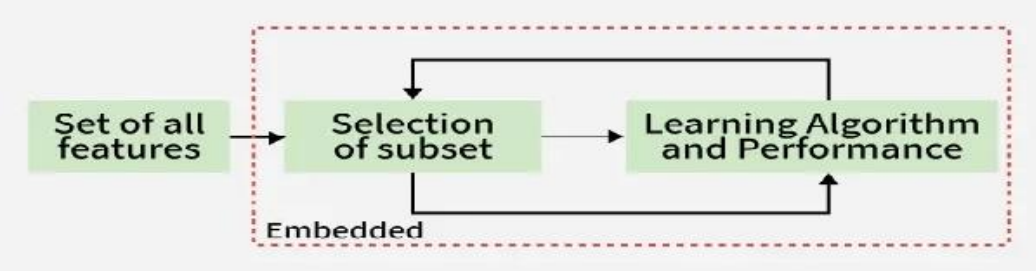
2. Wrapper methods

Wrapper methods are also referred as greedy algorithms that train algorithm. They use different combination of features and compute relation between these subset features and target variable and based on conclusion addition and removal of features are done. Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features are achieved.



3. Embedded methods

Embedded methods perform feature selection during the model training process. They combine the benefits of both filter and wrapper methods. Feature selection is integrated into the model training allowing the model to select the most relevant features based on the training process dynamically.



VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free /unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

Random Forest: Random Forest is a supervised machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (for classification) or mean prediction (for regression) of the individual trees.

Random Forest is a machine learning algorithm that uses many decision trees to make better predictions. Each tree looks at different random parts of the data and their results are combined by voting for classification or averaging for regression which makes it as ensemble learning technique. This helps in improving accuracy and reducing errors.

Working of Random Forest Algorithm:

- **Create Many Decision Trees:** The algorithm makes many decision trees each using a random part of the data. So every tree is a bit different.
- **Pick Random Features:** When building each tree it doesn't look at all the features (columns) at once. It picks a few at random to decide how to split the data. This helps the trees stay different from each other.
- **Each Tree Makes a Prediction:** Every tree gives its own answer or prediction based on what it learned from its part of the data.

- **Combine the Predictions:** For classification we choose a category as the final answer is the one that most trees agree on i.e majority voting and for regression we predict a number as the final answer is the average of all the trees predictions.
- **Why It Works Well:** Using random data and features for each tree helps avoid overfitting and makes the overall prediction more accurate and trustworthy.

Demonstrating the use of a Random Forest Classifier for a machine learning task, using the scikit-learn library:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.datasets import load_iris # Using a built-in dataset for demonstration

# 1. Load the dataset
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target

# 2. Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Initialize the Random Forest Classifier
# n_estimators: Number of trees in the forest
# random_state: For reproducibility
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# 4. Train the model
rf_classifier.fit(X_train, y_train)

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

# 6. Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=iris.target_names)
```

```
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:\n", report)

# Optional: Feature Importance
print("\nFeature Importances:")
for feature, importance in zip(X.columns, rf_classifier.feature_importances_):
    print(f"{feature}: {importance:.4f}")
```

Output:

```
Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Feature Importances:
sepal length (cm): 0.1081
sepal width (cm): 0.0304
petal length (cm): 0.4400
petal width (cm): 0.4215
```

X Conclusion:

……

……

……

……

……

XI Practical Related Questions:

1. What is data preprocessing and why is it important in machine learning?
2. How do you handle missing or null values in a dataset?
3. What are different methods to determine feature importance?
4. How does correlation help in identifying important features?
5. How can Random Forest be used to find feature importance?
6. Write a Python program to identify and visualize the most important features that contribute to model accuracy.
7. How does feature importance vary across different algorithms?

XII Exercise:

1. Implement feature engineering techniques to prepare data for machine learning models.
2. Write a program to identify the most important features that contributes to the model accuracy.

Space for Answer

XIII References:

1. **https://www.geeksforgeeks.org/machine-learning/what-is-feature-engineering/**
2. **https://www.datacamp.com/tutorial/feature-engineering**
3. **https://www.geeksforgeeks.org/machine-learning/data-preprocessing-machine-learning-python/**
4. **https://www.tutorialspoint.com/write-a-machine-learning-program-to-check-model-accuracy**
5. **https://machinelearningmastery.com/how-machine-learning-algo rithms-work/**
6. **https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/**
7. **https://www.ibm.com/think/topics/random-forest**

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 8: Implement program to use k-Nearest Neighbors (KNN) model for Classification on given dataset.

I Practical Significance

The k-Nearest Neighbors (KNN) Algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It is easy to implement and understand, but has a major drawback of becoming significantly slow as the size of the data in use grows.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query and then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO4: Apply supervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 8.1 Write a program to use k-Nearest Neighbors (KNN) model for Classification on give dataset.

LLO 8.2 Experiment with different values of K and measure model performance.

V Relevant Affective Domain related outcome(s)

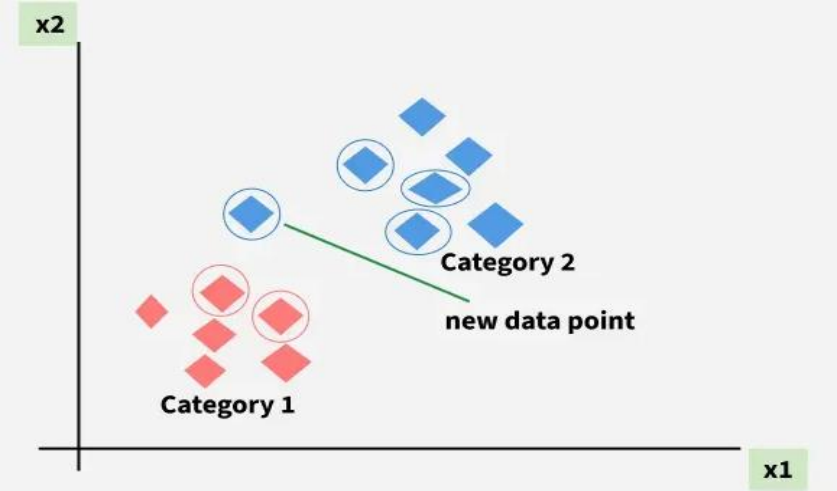
1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

k-Nearest Neighbors (KNN) is a supervised machine-learning algorithm generally used for classification but can also be used for regression tasks. It works by finding the "k" closest data points (neighbors) to a given input and makes a predictions based on the majority class (for classification) or the average value (for regression). Since KNN makes no assumptions about the underlying data distribution it makes it a non-parametric and instance-based learning method.

k-Nearest Neighbors is also called as a lazy learner algorithm because it does not learn from the training set immediately instead it stores the entire dataset and performs computations only at the time of classification.

For example, consider the following table of data points containing two features:



KNN Algorithm working visualization

The new point is classified as Category 2 because most of its closest neighbors are blue squares. KNN assigns the category based on the majority of nearby points. The image shows how KNN predicts the category of a new data point based on its closest neighbors.

- The red diamonds represent Category 1 and the blue squares represent Category 2.
- The new data point checks its closest neighbors (circled points).
- Since the majority of its closest neighbors are blue squares (Category 2) KNN predicts the new data point belongs to Category 2.

KNN works by using proximity and majority voting to make predictions. Therefore, larger k value means smother curves of separation resulting in less complex models. Whereas, smaller k value tends to over fit the data and resulting in complex models.

It's very important to have the right k-value when analyzing the dataset to avoid over fitting and under fitting of the dataset.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU: i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab, Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration.Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

- **Load Dataset:**
  - Import the dataset suitable for classification (e.g., Iris dataset in this case).
- **Split Dataset:**
  - Divide the dataset into training and testing sets:
  - Feature data (XXX): Input features of the dataset.
  - Target labels (yyy): Corresponding class labels.
- **Initialize the Model**:
  - Choose the k-value for the KNN classifier (e.g., k=1)
- **Train the Model**:
  - Fit the KNN model on the training data
- **Make Predictions:**

  For each test instance:
  - Extract the instance as input (xxx).
  - Reshape into the correct format (if needed).
  - Predict the class label using the trained model.
- **Compare Predictions:**
  - Compare the predicted label with the actual label for each test instance.
  - Optionally, map numerical class labels to class names for interpretability.

- **Evaluate the Model:**
  - Compute the model's accuracy on the test data:

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

- **Output:**
  - Display: Actual vs. predicted class labels for each test instance.
  - Overall model accuracy.

- **Source Code**

Problem Statement: To write a program to implement k-Nearest Neighbor Algorithm to classify the iris data set.

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import numpy as np
dataset=load_iris()
print(dataset)
X_train,X_test,y_train,y_test=train_test_split(dataset["data"],dataset["target"],random_state=0)
# (Mostly random state=42 is used)
kn=KNeighborsClassifier(n_neighbors=1)
kn.fit(X_train,y_train)

for i in range(len(X_test)):
  x=X_test[i]
  x_new=np.array([x])
prediction=kn.predict(x_new)
print("TARGET=",y_test[i],dataset["target_names"][y_test[i]],"PREDICTED=",prediction,dataset[
      "target_names"][prediction])
print(kn.score(X_test,y_test))
```

- **Output**

```
TARGET= 2 virginica PREDICTED= [2] ['virginica']
0.9736842105263158
```

X Conclusion:

……

……

……

……

XI Practical Related Questions:

1. How does k-NN work?
2. What distance metrics are commonly used?
3. What is the role of distance metrics in k-NN?
4. What is the effect of choosing different values of k?
5. What are the drawbacks of k-NN?
6. How do you evaluate k-NN performance?

XII Exercise:

1. Build KNN Classification model for a given dataset. Vary the number of k values as follows and compare the results:

   i. 1 ii. 3 iii. 5

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

……
……

……

XIII References:

https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/

https://ml-course.github.io/master/labs/Lab%201%20-%20Tutorial.html

https://developers.google.com/machine-learning/crash-course/classification

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 9: *Implement program to train an SVM model on given dataset.

I Practical Significance

Support vector machines (SVMs) are a set of related supervised learning methods used for classification, regression and outlier's detection. Given a set of training data, SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. It is used to classify real-world data with high accuracy, especially in cases where data classes may overlap. Also helps in understanding data patterns and making reliable predictions for decision-making. In summary, training an SVM model demonstrates the ability to extract knowledge from data and build intelligent systems that classify or predict outcomes accurately in real-world environments.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.
- Use SVM models for classification and regression tasks in industrial applications.

III Course Level Learning Outcome(s)

CO4: Apply supervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 9.1 Write a program to Train an SVM model on dataset**.**

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.

2. Follow ethical practices.

VI Relevant Theoretical Background

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier's detection. It tries to find the best boundary known as hyperplane that separates different classes in the data. It is useful when you want to do binary classification like spam vs. not spam or cat vs. dog.

The main goal of SVM is to maximize the margin between the two classes. The larger the margin the better the model performs on new and unseen data.

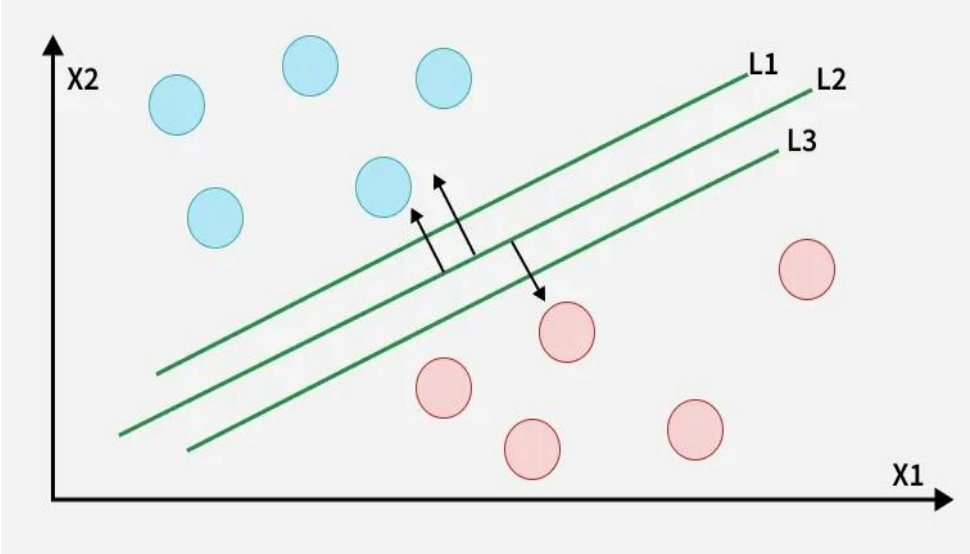
Key Concepts of Support Vector Machine

- **Hyperplane**: A decision boundary separating different classes in feature space and is represented by the equation $wx + b = 0$ in linear classification.

- **Support Vectors**: The closest data points to the hyperplane, which is crucial for determining the hyperplane and margin in SVM.
- **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
- **Kernel**: A function that maps data to a higher-dimensional space enabling SVM to handle non-linearly separable data. Eg. Linear Kernel, Radial Basis Function (RBF) Kernel, Polynomial Kernel
- **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
- **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.

Working of Support Vector Machine Algorithm:

- The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.



Multiple hyperplanes separate the data from two classes)

- The best hyperplane also known as the **"hard margin"** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So from the above figure, we choose L2 as hard margin.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

- **Import necessary libraries.**
- **Load Dataset:**
  - Import the Iris dataset.
- **Split Dataset:**
  - Divide the dataset into training and testing sets:
- **Standardize the features:**
  - Standardize the features which are important for SVM performance.
- **Train the Model**:
  - Create and train the SVM model
- **Make Predictions:**

  For each test instance reshape and make the prediction

- **Evaluate the Model:**
  - Compute the model's accuracy on the test data:
- **Output:**
  - Display the output and Overall model accuracy.

- **Source Code**

Problem Statement: Write a program to Train an SVM model on dataset.

```
# Import necessary libraries
import numpy as np
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, classification_report, confusion_matrix

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

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features (important for SVM performance)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create and train the SVM model (RBF kernel)
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
```

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

# Evaluate the model
print(" Model trained successfully!")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

- **Output**

```
Model trained successfully!
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

X Conclusion:

……

……

……

……

……

XI Practical Related Questions:

1. What is a Support Vector Machine (SVM)? Define kernel function.
2. Explain the difference between **linear** and **non-linear** SVM with examples.
3. Load any dataset and train an SVM model to classify data. Display accuracy.
4. Compare the performance of **linear** kernel vs **RBF** kernel on the same dataset. Which performs better and why?

XII Exercise:

1. Implement Support Vector Machine for a dataset and compare the accuracy by applying the following kernel functions:
   i. Linear
   ii. Polynomial
   iii. RBF

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

XIII References:

- https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/
- https://scikit-learn.org/stable/modules/svm.html
- https://ml-course.github.io/master/labs/Lab%201%20-%20Tutorial.html
- https://developers.google.com/machine-learning/crash-course/classification

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 10: Implement program to use logistic regression model to classify binary outcomes.

I Practical Significance

Regression analysis helps us understand and predict the relationship between a *dependent variable* (target) and one or more *independent variables* (features).

A regression model is used to predict continuous numeric values based on input variables. It learns patterns from data and estimates how changes in input features affect the target variable.

Logistic Regression is a supervised machine-learning algorithm used for classification problems. Implementing logistic regression allows us to classify outcomes, understand relationships between variables, estimate probabilities, and make data-driven decisions efficiently using a simple yet powerful statistical model.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.
- Use logistic regression model to classify binary outcomes.

III Course Level Learning Outcome(s)

CO3: Implement feature-engineering techniques to prepare data for machine learning models.

IV Laboratory Learning Outcome(s)

LLO 10.1 Write a program to use logistic regression model to classify binary outcomes.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.

2. Follow ethical practices.

VI Relevant Theoretical Background

Logistic Regression is a supervised machine-learning algorithm used for classification problems. Unlike linear regression, which predicts continuous values it, predicts the probability that an input belongs to a specific class. It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1.

Types of Logistic Regression:

Logistic regression can be classified into three main types based on the nature of the dependent variable:Binomial Logistic Regression, Multinomial Logistic Regression and Ordinal Logistic Regression

Working of Logistic Regression Model:

The sigmoid function is an important part of logistic regression, which is used to convert the raw output of the model into a probability value between 0 and 1.

Logistic regression model transforms the linear regression function continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

Suppose we have input features represented as a matrix:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y having only binary value i.e 0 or 1.

$$Y = \begin{cases} 0 & \text{if } Class\ 1 \\ 1 & \text{if } Class\ 2 \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

Here x_i is the i^{th} observation of X,

$w_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or Coefficient and

b is the bias term also known as intercept.

Simply this can be represented as the dot product of weight and bias.

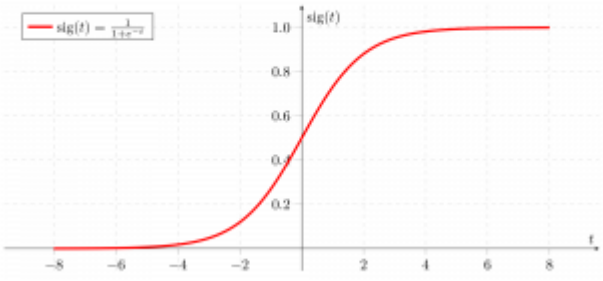
$$z = w \cdot X + b$$

At this stage, z is a continuous value from the linear regression.

Logistic regression then applies the sigmoid function to z to convert it into a probability between 0 and 1 which can be used to predict the class.

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Sigmoid function

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

We will use Breast cancer datasets for example.

- **Import necessary libraries.**

- **Load Dataset:**

Import the Breast cancer dataset

- **Split Dataset:**
  - Divide the dataset into training and testing sets:
- **Standardize the features:**
  - Standardize the features which are important for regression performance.
- **Train the Model**:
  - Create and train the logistic regression model.
- **Make Predictions:**

  For each test instance reshape and make the prediction
- **Evaluate the Model:**
  - Compute the model's accuracy on the test data:
- **Output:**
  - Display the output and Overall model accuracy.

Problem Statement: Write a program to use logistic regression model to classify binary outcomes

- **Source Code**

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Load dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split dataset into training & testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler()
```

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

# Create Logistic Regression model
model = LogisticRegression()

# Train model
model.fit(X_train, y_train)
# Predict
y_pred = model.predict(X_test)
# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

- **Output**

```
Accuracy: 0.9736842105263158

Confusion Matrix:
 [[41  2]
 [ 1 70]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        43
           1       0.97      0.99      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is Logistic Regression?
2. Which function is used to split dataset into train and test in Python?
3. Interpret the role of the **sigmoid function** in logistic regression.
4. What is the difference between linear regression and logistic regression?
5. Why is logistic regression preferred over linear regression for binary classification?

XII Exercise:

1. Modify the given program to use your own dataset instead of breast-cancer dataset.
2. Change the test size to 30% and find the new accuracy
3. Apply logistic regression to identify whether a student will pass or fail based on his marks & study hours.

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XIII References:

- https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/
- https://www.ibm.com/think/topics/logistic-regression
- https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/
- https://developers.google.com/machine-learning/crash-course

XIVAssessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 11: *Implement program to use PCA technique to reduce the number of features while retaining important information.

I Practical Significance

PCA (Principal Component Analysis) is a dimensionality reduction technique used in data analysis and machine learning. PCA helps reduce the number of variables without losing important information. It makes machine-learning models faster, more stable and easier to interpret. It is a powerful statistical technique that maps high-dimensional data to a smaller space while preserving essential information. It improves computational efficiency, supports better visualization and enhances machine learning model performance.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.

III Course Level Learning Outcome(s)

CO4: Apply supervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 11.1 Write a program to use PCA technique to reduce the number of features while retaining important information.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

Principal Component Analysis (PCA) is a dimensionality reduction technique used in data analysis and machine learning that transforms high-dimensional data into a lower-dimensional form while preserving as much variance as possible It helps you to reduce the number of features in a dataset while keeping the most important information. It changes your original features into new features these new features don't overlap with each other and the first few keep most of the important differences found in the original data.

PCA is commonly used for data pre-processing for use with machine learning algorithms. It helps to remove redundancy, improve computational efficiency and make data easier to visualize and analyze especially when dealing with high-dimensional data.

Working of Principal Component Analysis:

PCA uses linear algebra to transform data into new features called principal components. It finds these by calculating eigenvectors (directions) and eigenvalues (importance) from the covariance

matrix. PCA selects the top components with the highest eigenvalues and projects the data onto them simplify the dataset.

Here is how it works step by step:

Step 1: Standardize the Data

Different features may have different units and scales like salary vs. age. To compare them fairly PCA first standardizes the data by making each feature have, a mean of 0 and standard deviation of 1.

$$Z = \frac{X - \mu}{\sigma}$$

where:

- Z is Z-score.
- X is Individual data point.
- μ is the mean of independent features $\mu = \{\mu_1, \mu_2, \cdots, \mu_m\}$
- σ is the standard deviation of independent features $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$

Step 2: Calculate Covariance Matrix

Next PCA calculates the covariance matrix to see how features relate to each other whether they increase or decrease together. The covariance between two features x1 and x2 is:

$$\text{cov}(\text{x1}, \text{x2}) = \frac{\sum_{i=1}^{n} (x1i - \overline{x1})\ (\text{x2i} - \overline{x2})}{n - 1}$$

Where:

- $\overline{x1}$and $\overline{x2}$ are the mean values of features x1and x2.
- n is the number of data points

The value of covariance can be positive, negative or zeros.

Step 3: Find the Principal Components

PCA identifies **new axes** where the data spreads out the most:

1st Principal Component (PC1): The direction of maximum variance (most spread).

2nd Principal Component (PC2): The next best direction, *perpendicular to PC1* and so on.

These directions come from the **eigenvectors** of the covariance matrix and their importance is measured by **eigenvalues**.

For a square matrix 'A' an **eigenvector** X (a non-zero vector) and its corresponding **eigenvalue** λ satisfy:

$$AX = \lambda X$$

This means:

- When *A* acts on X it only stretches or shrink's X by the scalar λ.
- The direction of X remains unchanged hence eigenvectors define "stable directions" of A.

Eigenvalues help rank these directions by importance.

Step 4: Pick the Top Directions & Transform Data

After calculating the eigenvalues and eigenvectors PCA ranks them by the amount of information they capture. We then:

1. **Select the top k components** that capture most of the variance like 95%.
2. **Transform the original dataset** by projecting it onto these top components.

This means we reduce the number of features (dimensions) while keeping the important patterns in the data.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

- **Input:**
  - Dataset with multiple features (XXX) and labels (yyy).
- **Preprocessing:**
  - Standardize the dataset to ensure all features have equal importance.
- **Compute Covariance Matrix:**
  - Calculate the covariance matrix of the standardized data.

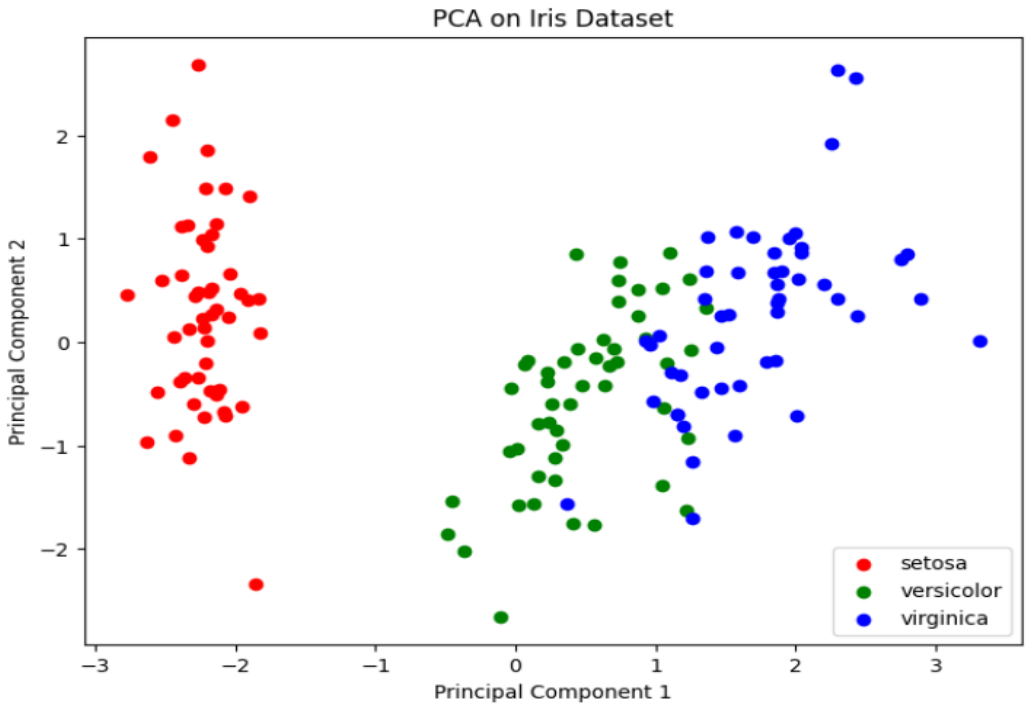
- **Eigen Decomposition**:
  - Compute eigenvalues and eigenvectors of the covariance matrix.
  - Select the top k eigenvectors corresponding to the largest eigenvalues.
- **Project Data:**
  - Transform the original data to the lower-dimensional space using the selected eigenvectors
- **Visualization:**
  - Plot the transformed data in 2D or 3D for visualization.
- **Source Code**

  Problem Statement: To implement Principal Component Analysis (PCA) for dimensionality reduction and visualize the reduced data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# Load Iris Dataset
from sklearn.datasets import load_iris
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.DataFrame(data.target, columns=["Target"])
# Standardize the Data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
# Visualize Reduced Data
plt.figure(figsize=(8, 6))
for target, color in zip([0, 1, 2], ['red', 'green', 'blue']):
  plt.scatter(X_pca[y["Target"] == target, 0], X_pca[y["Target"] == target, 1], c=color,
label=data.target_names[target])
```

```
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA on Iris Dataset')
plt.legend()
plt.show()
```

- **Output:**



X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is PCA and why is it used?
2. What are eigenvalues and eigenvectors and how do they relate to PCA?
3. Why do we standardize data before applying PCA?
4. How is PCA different from Linear Discriminant Analysis (LDA)?
5. What are the limitations of PCA?
6. How does PCA handle correlated features in a dataset?

XII Exercise:

1. To implement Principal Component Analysis (PCA) for dimensionality reduction, visualize the reduced data and evaluate its effectiveness using a classification algorithm.

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XIII References:

- https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/
- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- https://ml-course.github.io/master/labs/Lab%201%20-%20Tutorial.html

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 12: *Implement program to use machine-learning model on given dataset.

I Practical Significance

Classification involves two main stages: a training phase and a prediction phase. During training, a model is built using labeled data to learn patterns and relationships. In the prediction phase, this model is applied to new or unseen data to determine the correct output or category. One of the simplest and most commonly used algorithms for classification is the Decision Tree. This method is widely favored for its ability to visually represent decision-making steps, making the results easy to interpret. Decision trees can also be applied to both classification and regression tasks, offering versatility in solving various types of problems.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.
- Train decision tree on the dataset to solve a specific business problem.

III Course Level Learning Outcome(s)

CO4: Apply supervised learning models to train and evaluate.

CO5: Apply unsupervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 11.1 Write a program to use any machine learning model on given dataset.

LLO 11.2 Preprocess, analyze, and build models to get the meaningful insights.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.

2. Follow ethical practices.

VI Relevant Theoretical Background

A Decision Tree is a type of supervised learning algorithm commonly used in machine learning tasks. It works by organizing decisions in a tree-like structure, where each internal node represents a test on a specific feature and each branch shows the outcome of that test. The leaf nodes represent the final output or target value. The process begins at the root node, where a data sample is evaluated. At every node, a condition based on one of the sample's features determines the path to the next node. This continues until the sample reaches a leaf, where a prediction is made. The algorithm learns by identifying the most effective conditions (or rules) at each step to divide the data, using criteria such as information gain or Gini index to decide how to split the data efficiently.

The decision trees can be divided, with respect to the target values into:

- **Classification** trees used to classify samples, assign to a limited set of values - classes. In scikit-learn it is DecisionTreeClassifier.
- **Regression** trees used to assign samples into numerical values within the range. In scikit-learn it is DecisionTreeRegressor.

Decision trees are considered a fundamental tool in machine learning. They provide logical insights into complex datasets. It has a hierarchical tree structure with a root node, branches, internal nodes, and leaf nodes. Following are the Decision trees terminologies:

a. **Root node:** The root node is the beginning point of a decision tree where the whole dataset starts to divide based on different features present in the dataset.

b. **Decision nodes:** Nodes with children nodes represent a decision to be taken. The root node (if having children) is also a decision node.

c. **Leaf nodes:** Nodes that indicate the final categorization or result when additional splitting is not possible. Terminal nodes are another name for leaf nodes.

d. **Branches or subtrees:** A branch or subtree is a component of the decision tree that is part of the larger structure. Within the tree, it symbolizes a certain decision-making and result-oriented path.

e. **Pruning:** It is the practice of eliminating or chopping down particular decision tree nodes to simplify the model and avoid overfitting.

f. **Parent and child nodes:** In a decision tree, a node that can be divided is called a parent node and nodes that emerge from it are called its child nodes. The parent node represents a decision or circumstance, and the child nodes represent possible outcomes or additional decisions based on that situation.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

- **Load Dataset:**
  - The Iris dataset, a classic for classification tasks, is loaded using load_iris() from sklearn.datasets. The features (X) and target variable (y) are extracted.
- **Split Dataset:**
  - Divide the dataset into training and testing sets:
  - train_test_split divides the data into training and testing sets, allowing for evaluation of the model's performance on unseen data. test_size=0.3 means 30% of the data is used for testingand random_state ensures reproducibility.
- **Initialize the Decision Tree Classifier**:
  - A DecisionTreeClassifier object is created. You can customize its behavior by setting hyperparameters like max_depth (to control tree depth and prevent overfitting) or criterion (gini for Gini impurity or entropy for information gain).
- **Train the Model**:
  - The fit() method trains the decision tree on the training data
- **Make Predictions:**
  - The trained model predicts the class labels for the test set (X_test) using the predict() method.
- **Evaluate the Model:**
  - accuracy_score calculates the proportion of correctly classified instances.
  - classification_report provides a comprehensive summary including precision, recall, f1-score, and support for each class.
- **Output:**
  - Display: Decision Tree Classifier Performance and Overall model accuracy.

- **Source Code**

Problem Statement: Write a program to use decision tree model on given dataset.

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# 1. Load the dataset
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target

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

# 3. Initialize the Decision Tree Classifier
# You can adjust hyperparameters like max_depth, criterion (gini or entropy)
dt_classifier = DecisionTreeClassifier(random_state=42)

# 4. Train the model
dt_classifier.fit(X_train, y_train)

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

# 6. Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=iris.target_names)
print("Decision Tree Classifier Performance:")
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(report)
```

- **Output**

Decision Tree Classifier Performance:

Accuracy: 1.00

Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	19
versicolor	1.00	1.00	1.00	13
virginica	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is a Decision Tree?
2. Name any two real-world applications of decision trees.
3. What is Gini Index?
4. What is Entropy in decision tree classification?
5. Explain the concept of information gain in Decision Trees.
6. Why do decision trees require pruning?
7. Differentiate between Gini Index and Entropy

XII Exercise:

1. Modify the above program to use criterion="entropy" instead of "gini".
2. Build a simple decision-support system for medical diagnosis using a decision tree.

Space for Answer

……

……

……

……

……

XIII References:

- Python Decision Tree Classification Tutorial: Scikit-Learn DecisionTreeClassifier | DataCamp
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://colab.research.google.com/github/gumption/Python_for_Data_Science/blob/master/4_Python_Simple_Decision_Tree.ipynb
- https://www.ibm.com/think/topics/decision-trees

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 13: Implement program to use Pandas, Matplotlib to analyze data of previously created model.

I Practical Significance

Pandas is an open-source Python library widely used for data manipulation and analysis. It provides powerful and flexible data structures, particularly DataFrames and Series, designed to make working with structured data both easy and intuitive.

Matplotlib is a widely-used Python library used for creating static, animated and interactive data visualizations. It is built on the top of NumPy and it can easily handle large datasets for creating various types of plots such as line charts, bar charts, scatter plots, etc.

This practical approach allows us to interpret model output, identify patterns, detect errors or biases, and make data-driven decisions to improve the model. By visualizing accuracy, loss curves, feature importance, or prediction comparisons, users can better evaluate the effectiveness of the machine-learning model, communicate results clearly, and enhance future model development.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.
- Interpret model output, patterns and make data- driven decisions to improve the industry model.

III Course Level Learning Outcome(s)

CO4: Apply supervised learning models to train and evaluate.

CO5: Apply unsupervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 13.1 Write a program to use Pandas, Matplotlib to analyze data of previously created model.

LLO 13.2 Plot histograms, box plots, scatterplots, and correlation heatmaps to understand relationships between variables.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Follow ethical practices.

VI Relevant Theoretical Background

Pyplot is a module in Matplotlib that provides a simple interface for creating plots. It allows users to generate charts like line graphs, bar charts and histograms with minimal code.

1. Line Chart

Line chart is one of the basic plots and can be created using **plot()** function. It is used to represent a relationship between two data X and Y on a different axis.

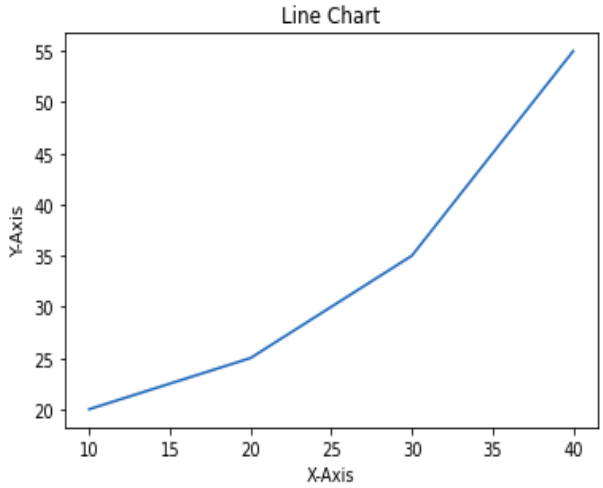
Syntax:

matplotlib.pyplot.plot(x, y)

Parameter: x, y Coordinates for data points.

Example:

```
import matplotlib.pyplot as plt


x = [10, 20, 30, 40]
y = [20, 25, 35, 55]


plt.plot(x, y)
plt.title("Line Chart")
plt.ylabel('Y-Axis')
plt.xlabel('X-Axis')
plt.show()
```



2. Histogram

Histogram shows the distribution of data by grouping values into bins. The **hist()** function is used to create it, with X-axis showing bins and Y-axis showing frequencies.

Syntax:

matplotlib.pyplot.hist(x, bins=None)

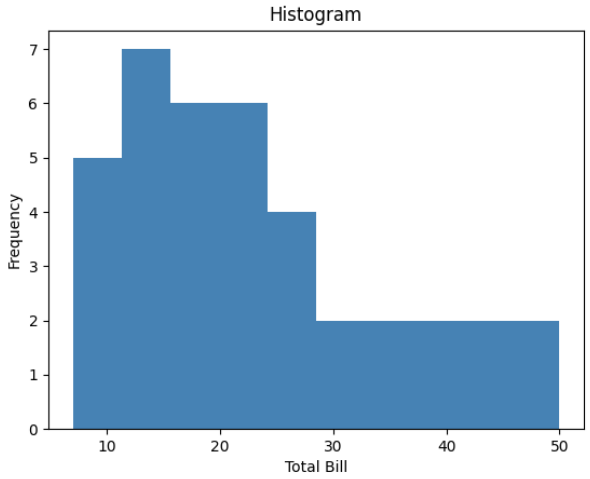
Parameter:

x: Input data.

bins: Number of bins (intervals) to group data.

Example: This code plots a histogram to show frequency distribution of total bill values from the list x. It uses 10 bins and adds axis labels and a title for clarity.

```
import matplotlib.pyplot as plt

x = [7, 8, 9, 10, 10, 12, 12, 12, 13, 14, 14, 15, 16, 16, 17, 18, 18, 19, 20, 20,21, 22, 23, 24, 25, 25, 26, 28, 30, 32, 35, 36, 38, 40, 42, 44, 48, 50]


plt.hist(x, bins=10, color='steelblue')
plt.title("Histogram")
plt.xlabel("Total Bill")
plt.ylabel("Frequency")
plt.show()
```



3. Scatter Plot

Scatter plots are used to observe relationships between variables. The **scatter()** method in the matplotlib library is used to draw a scatter plot.

Syntax:

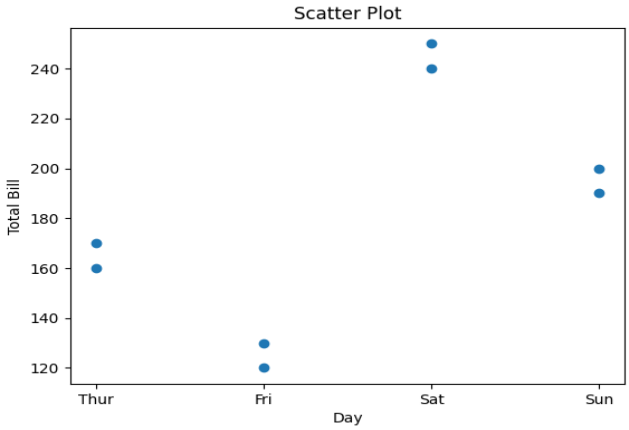
matplotlib.pyplot.scatter(x, y)

Parameter: x, y Coordinates of the points.

Example: This code creates a scatter plot to visualize the relationship between days and total bill amounts using **scatter()**.

```
import matplotlib.pyplot as plt
```

```
x = ['Thur', 'Fri', 'Sat', 'Sun', 'Thur', 'Fri', 'Sat', 'Sun']
y = [170, 120, 250, 190, 160, 130, 240, 200]

plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("Day")
plt.ylabel("Total Bill")
plt.show()
```



4. Box Plot

Box plot is a simple graph that shows how data is spread out. It displays the minimum, maximum, median and quartiles and also helps to spot outliers easily.

Syntax:

matplotlib.pyplot.boxplot(x, notch=False, vert=True)

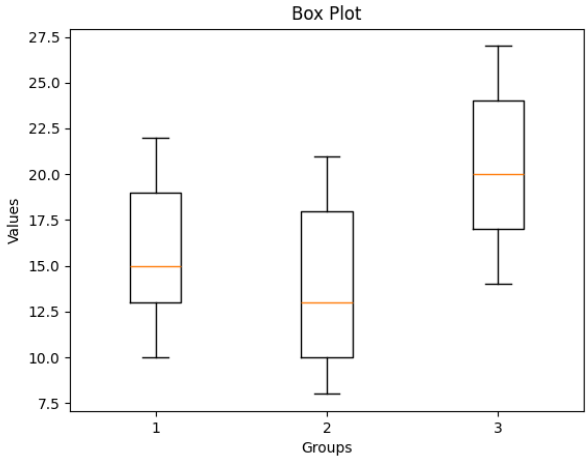
Parameter:

- **x:** Data for which box plot is to be drawn (usually a list or array).
- **notch:** If True, draws a notch to show the confidence interval around the median.
- **vert:** If True, boxes are vertical. If False, they are horizontal.

Example: This code creates a box plot to show the data distribution and compare three groups using matplotlib

```
import matplotlib.pyplot as plt

data = [ [10, 12, 14, 15, 18, 20, 22],
        [8, 9, 11, 13, 17, 19, 21],
        [14, 16, 18, 20, 23, 25, 27]]

plt.boxplot(data)
plt.xlabel("Groups")
```

```
plt.ylabel("Values")
plt.title("Box Plot")
plt.show()
```



5. Heatmap

Heatmap is a graphical representation of data where values are shown as colors. It helps visualize patterns, correlations or intensity in a matrix-like format. It is created using **imshow()** method in Matplotlib.

Syntax:

matplotlib.pyplot.imshow(X, cmap='viridis')

Parameter:

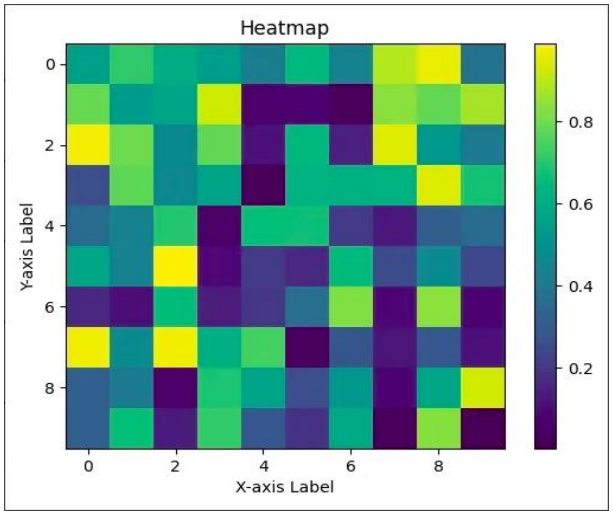
X: 2D array (data to display as an image or heatmap).

cmap: Sets the color map.

Example: This code creates a heatmap of random 10×10 data using **imshow()**. It uses '**viridis**' color map and **colorbar()** adds a color scale.

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

np.random.seed(0)
data = np.random.rand(10, 10)
plt.imshow(data, cmap='viridis', interpolation='nearest')
plt.colorbar()
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
```

```
plt.title('Heatmap')
plt.show()
```



Explanation:

np.random.seed(0): Ensures same random values every time (reproducibility).

np.random.rand(10, 10): Generates a 10×10 array of random numbers between 0 and 1.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

Step 1: Create a synthetic dataset for demonstration.

Step 2: Plot Histograms.

Step 3: Plot Box Plots.

Step 4: Plot Scatterplots to understand relationships.

Step 5: Plot Correlation Heatmap.

Problem Statement: Write a program to use Pandas and Matplotlib (with Seaborn for enhanced aesthetics and correlation heatmaps) to analyze data and visualize relationships between variables in a machine learning context.

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

# 1. Create a synthetic dataset for demonstration
# In a real scenario, load your data using pd.read_csv(), pd.read_excel(), etc.
np.random.seed(42) # for reproducibility
data = {
    'Feature1': np.random.rand(100) * 10,
    'Feature2': np.random.rand(100) * 5 + 2,
    'Feature3': np.random.randn(100) * 3 + 15,
    'Target': (np.random.rand(100) * 10) + (np.random.rand(100) * 5) # Simulate some relationship
}
df = pd.DataFrame(data)

print("First 5 rows of the DataFrame:")
print(df.head())
print("\nDataFrame Info:")
```

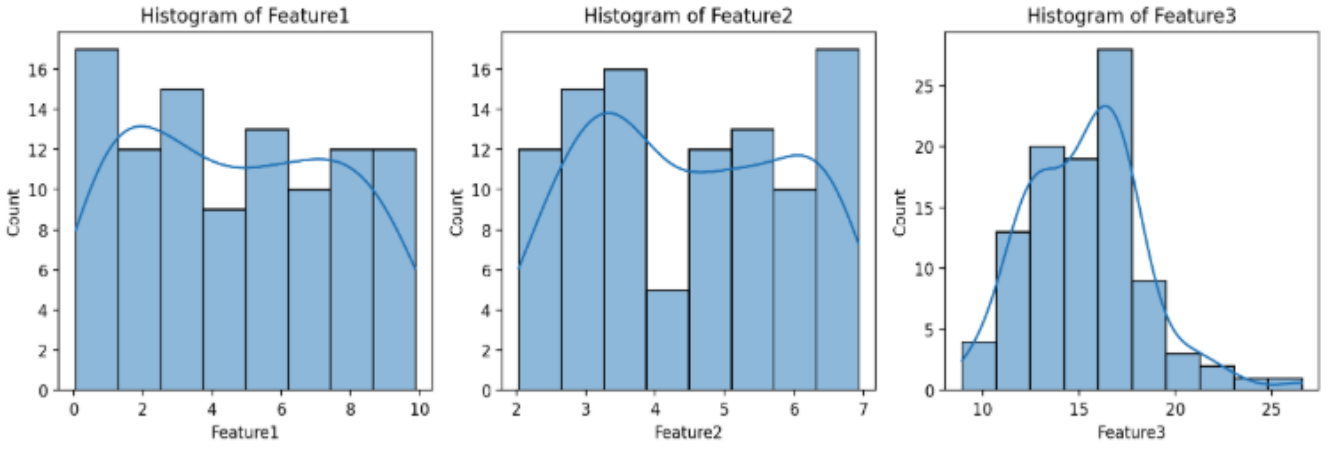
```
df.info()
print("\nDescriptive Statistics:")
print(df.describe())

# 2. Plot Histograms
plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
sns.histplot(df['Feature1'], kde=True)
plt.title('Histogram of Feature1')

plt.subplot(1, 3, 2)
sns.histplot(df['Feature2'], kde=True)
plt.title('Histogram of Feature2')

plt.subplot(1, 3, 3)
sns.histplot(df['Feature3'], kde=True)
plt.title('Histogram of Feature3')
plt.tight_layout()
plt.show()

# 3. Plot Box Plots
plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
sns.boxplot(y=df['Feature1'])
plt.title('Box Plot of Feature1')

plt.subplot(1, 3, 2)
sns.boxplot(y=df['Feature2'])
plt.title('Box Plot of Feature2')

plt.subplot(1, 3, 3)
sns.boxplot(y=df['Feature3'])
plt.title('Box Plot of Feature3')
plt.tight_layout()
plt.show()
```

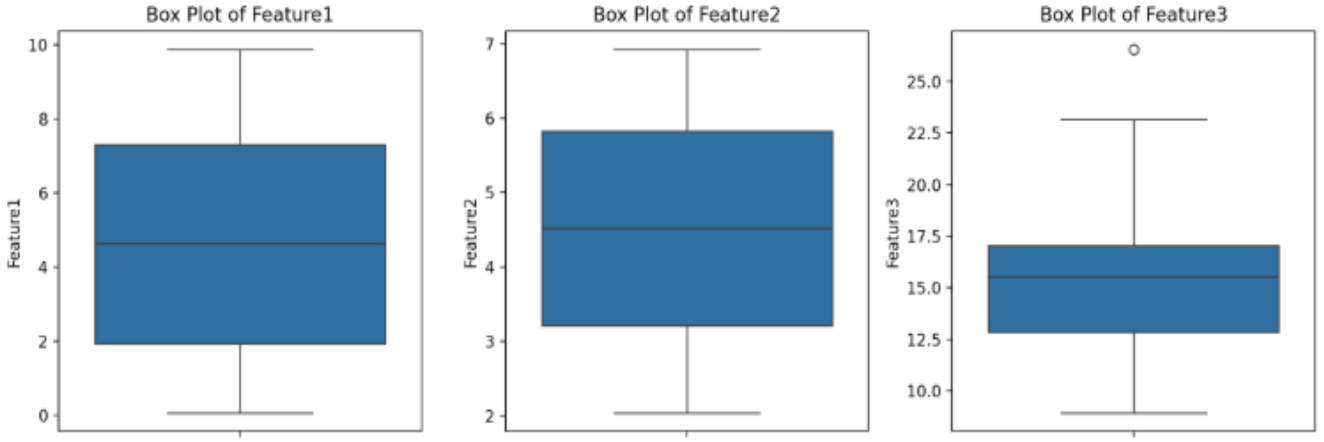
```
# 4. Plot Scatterplots to understand relationships
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
sns.scatterplot(x=df['Feature1'], y=df['Target'])
plt.title('Feature1 vs Target')
plt.subplot(1, 2, 2)
sns.scatterplot(x=df['Feature2'], y=df['Target'])
plt.title('Feature2 vs Target')
plt.tight_layout()
plt.show()
# 5. Plot Correlation Heatmap
plt.figure(figsize=(8, 6))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Features and Target')
plt.show()
```

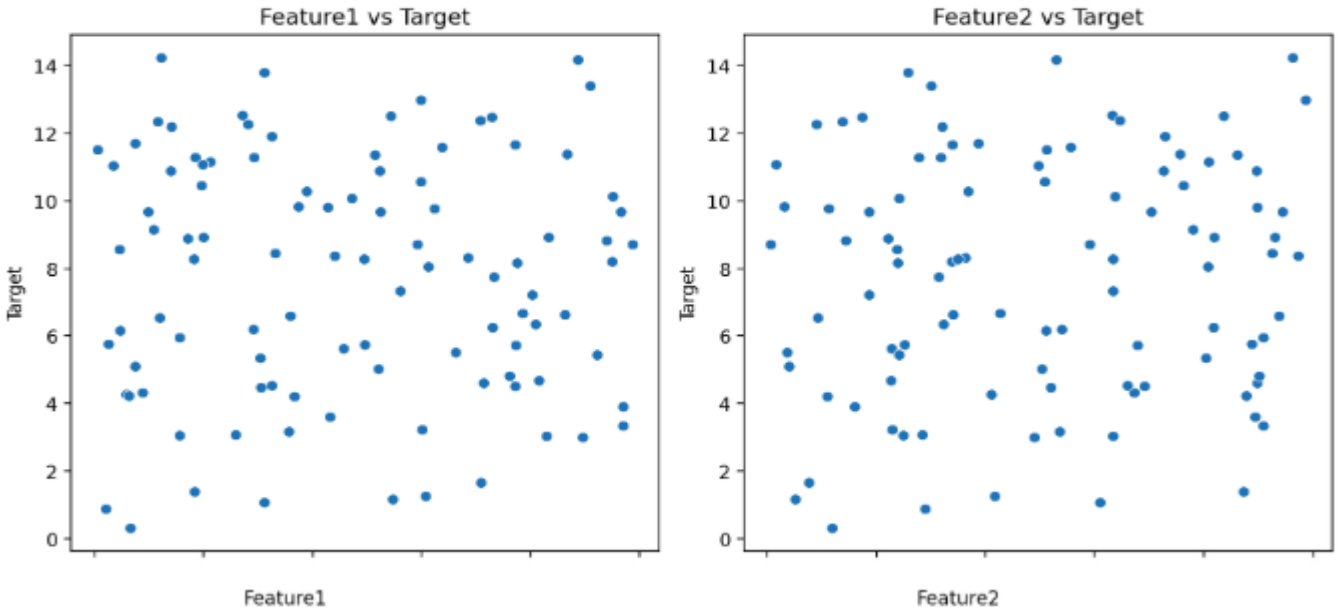
- **Output**

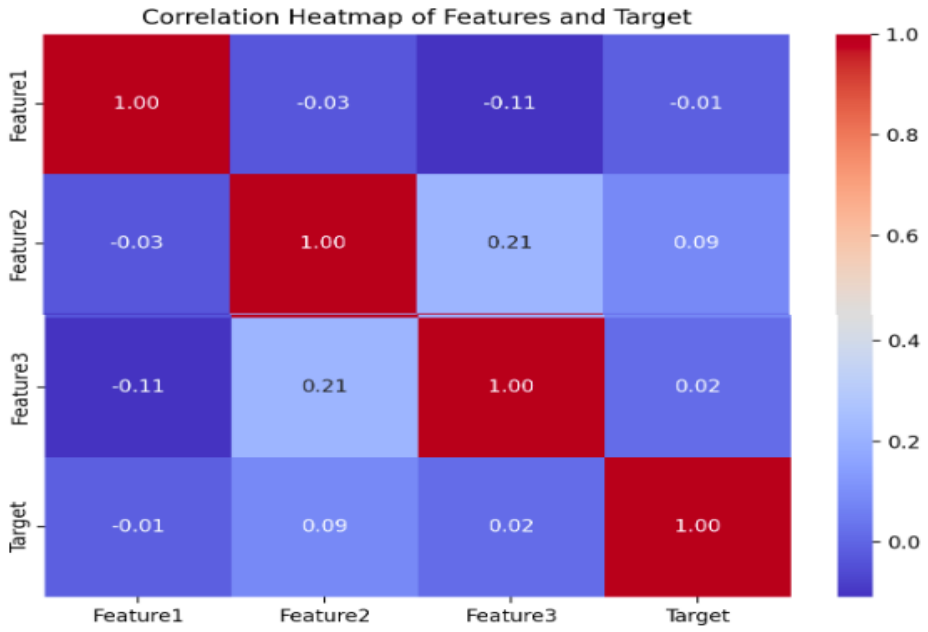
```
First 5 rows of the DataFrame:
   Feature1  Feature2   Feature3     Target
0  3.745401  2.157146  12.959926   9.836447
1  9.507143  5.182052  15.696761  10.126614
2  7.319939  3.571780  15.879217   7.751193
3  5.986585  4.542853  12.856946  10.555728
4  1.560186  6.537832  20.597324   5.950576

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Feature1  100 non-null    float64
 1   Feature2  100 non-null    float64
 2   Feature3  100 non-null    float64
 3   Target    100 non-null    float64
dtypes: float64(4)
memory usage: 3.3 KB
```

```
Descriptive Statistics:
         Feature1    Feature2    Feature3      Target
count  100.000000  100.000000  100.000000  100.000000
mean     4.701807    4.489159   15.324100    7.715094
std      2.974894    1.465556    3.017157    3.543891
min      0.055221    2.034761    8.924572    0.321881
25%      1.932008    3.210023   12.856232    4.780472
50%      4.641425    4.528124   15.539453    8.237004
75%      7.302031    5.830918   17.043790   10.888519
max      9.868869    6.928252   26.558194   14.225511
```









X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is the purpose of using Pandas in data analysis?
2. Define a histogram. What does it represent?
3. Which Python library is used to create visualizations like scatterplots and box plots?
4. What is a correlation heatmap?
5. Write the function used in Pandas to load a CSV file.
6. Describe the advantage of using a scatter plot to identify relationships between variables.
7. How does a correlation heatmap help in feature selection?

XII Exercise:

1. Write a program to use Pandas, Matplotlib to analyze data of previously created model. Also plot histograms, box plots, scatterplots, and correlation heatmaps to understand relationships between variables.

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

……

……
……

……

……

……

XIII References:

- https://www.geeksforgeeks.org/data-visualization/data-visualization-using-matplotlib/
- https://ourcodingclub.github.io/tutorials/pandas-python-intro/
- https://medium.com/@waziriphareeyda/data-visualization-and-plotting-techniques-with-pandas-in-python-a-comprehensive-guide-to-8a0d5bab3189
- https://www.datacamp.com/tutorial/types-of-data-plots-and-how-to-create-them-in-python

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 14: Implement program to use machine learning model on Boston Housing Dataset (available in Scikit-learn).

I Practical Significance

The Boston Housing dataset, which is used in regression analysis, provides insights into the housing values in the suburbs of Boston. This dataset has been a staple for algorithm demonstration, from simple linear regression to more complex machine learning models in predictive analytics.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.
- Use machine learning models for predictive analytics using available datasets.

III Course Level Learning Outcome(s)

CO4: Apply supervised learning models to train and evaluate.

CO5: Apply unsupervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 14.1 Write a program to use any machine learning model on given dataset.

LLO 14.2 Predict results based on various features using a regression model.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.

2. Follow ethical practices.

VI Relevant Theoretical Background

Boston housing dataset:

The Boston Housing dataset is a collection of data from the 1970s on housing prices in various Boston districts, commonly used in machine learning to demonstrate regression analysis.

Originally curated by the U.S. Census Service, it includes 506 instances, each with 13 features, and the target variable is the median value of owner-occupied homes in $1000s.

Understanding Boston Dataset

These datasets are pre-built datasets in sklearn. It consists of:

Samples total - 506

Dimensionality - 13

Features - real, positive

Targets - real 5. - 50.

Description of Boston Dataset in Sklearn

The Boston Housing dataset contains several columns that are used to describe various aspects of residential homes in Boston. Here is a description of each column in the dataset:

- CRIM: Per capita crime rate by town. It indicates the level of crime in the area.
- ZN: Proportion of residential land zoned for lots over 25,000 sq.ft. This feature reflects the area's residential density.
- INDUS: Proportion of non-retail business acres per town. This is an indicator of the commercial use of land away from residential areas.
- CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise). This indicates whether the property is near the Charles River, which may add to the aesthetic value of the neighborhood.
- NOX: Nitric oxides concentration (parts per 10 million). It represents the level of industrial pollutants in the area.
- RM: Average number of rooms per dwelling. More rooms typically indicate more spacious accommodation.
- AGE: Proportion of owner-occupied units built prior to 1940. Older structures might lack newer amenities or could be considered more prestigious depending on the architecture and condition.
- DIS: Weighted distances to five Boston employment centres. This feature measures the accessibility to workplaces, which can influence housing prices.
- RAD: Index of accessibility to radial highways. Higher values indicate easier access to major roadways.
- TAX: Full-value property-tax rate per $10,000. This reflects the annual property tax rate.
- PTRATIO: Pupil-teacher ratio by town. Lower values typically indicate better educational facilities, which is a significant factor for families when choosing a home.
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

Use Cases of Boston Housing data

The Boston Housing dataset is a well-known dataset commonly used in machine learning and data science for regression tasks. Here are some typical use cases in Python:

- **Predicting Housing Prices:**

  The primary use case for the Boston Housing dataset is to predict housing prices based on various features such as crime rate, number of rooms, accessibility to highways, and more.

- **Exploratory Data Analysis (EDA):**

  The dataset can be used to perform EDA to understand the relationships between different features and the target variable (housing prices). Techniques include visualizations (scatter plots, histograms, box plots), summary statistics, and correlation analysis.

- **Feature Engineering:**

  The dataset provides a good platform to practice feature engineering techniques like creating new features, handling missing values, scaling features, and transforming variables.

- **Regression Models:**

  You can use the dataset to train and evaluate various regression models like Linear Regression, Ridge Regression, Lasso Regression, and Polynomial Regression.

- **Model Evaluation and Validation:**

  The dataset can be used to practice different model evaluation techniques like cross-validation, train-test split, and performance metrics (RMSE, MAE, R^2).

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

Step1: Load Boston Housing data in sklearn:

To download the Boston housing dataset, use Python's scikit-learn library to load it using sklearn.datasets.load_boston().

Method 1: Load with scikit-learn (Python)

- **Import the datasets module:** from sklearn import datasets.
- **Load the dataset:** Call the load_boston() function to create a dataset object.
- **Note on deprecation**: Be aware that load_boston() is deprecated in scikit-learn version 1.2 and above due to ethical concerns.
- **Access the data:** The object contains separate keys for the features ('data'), the target variable ('target'), and column names ('feature_names').

Alternatively, for educational purposes, you can load it directly as a CSV file from online repositories like Kaggle.

Method 2: Download as a CSV file

- **Find a trusted online repository:** Search for "Boston housing dataset csv" on platforms like Kaggle or other data science websites.
- **Download the CSV file:** Download the file directly to your machine. Students can download the Boston Housing dataset from the following link:

  https://gist.github.com/nnbphuong/def91b5553736764e8e08f6255390f37
- **Load in Python:** Use libraries like Pandas to read the CSV file into a DataFrame for analysis.

Step 2: Create a DataFrame from the dataset

```
import pandas as pd

boston_df = pd.DataFrame(boston.data, columns=boston.feature_names)
boston_df['PRICE'] = boston.target
```

Add the target variable (house prices)

Step 3: Define features (X) and target (y)

```
X = boston_df.drop('PRICE', axis=1)
y = boston_df['PRICE']
```

Step 4: Split the data into training and testing sets

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 5: Initialize and train the Linear Regression model

```
# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Step 6: Make predictions on the test set

```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

Step 7: Evaluate the model

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

# Predict results based on custom features
# Example: Predict the price of a house with specific characteristics
# (Ensure the order of features matches the training data)
sample_features = pd.DataFrame([[0.03, 15.0, 2.0, 0, 0.45, 6.5, 30.0, 4.0, 2, 250, 18.0, 390.0, 5.0]],
columns=boston.feature_names)

predicted_price = model.predict(sample_features)
print(f"\nPredicted price for the sample features: ${predicted_price[0]*1000:.2f}")
```

Step 7: Display feature coefficients

```
# Display feature coefficients
print("\nFeature Coefficients:")
coefficients_df = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
print(coefficients_df)
```

X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is the purpose of the Boston Housing dataset?
2. Name any three features (independent variables) present in the dataset.
3. What is the target variable in the Boston Housing dataset?
4. Which Python library provides the Boston Housing dataset?
5. Why do we use a correlation heatmap in data analysis?
6. Interpret what a high R^2 score means in a regression model.

XII Exercise:

1. Write a Python program to visualize the relationship between LSTAT and MEDV using a scatter plot.
2. Design a predictive model using **Decision Tree** or **Random Forest** for the same dataset and compare results with Linear Regression.

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

……

……

……

……

……

……

……

……

……

……

……

XIII References:

- https://www.geeksforgeeks.org/machine-learning/how-to-load-boston-housing-data-in-python/
- https://www.geeksforgeeks.org/machine-learning/boston-dataset-in-sklearn/
- https://www.kaggle.com/code/shreayan98c/boston-house-price-prediction

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		

Practical No. 15: Implement a program to segment customers into different groups based on their purchasing behavior features using K-Means Clustering.

I Practical Significance

Clustering algorithms are unsupervised learning techniques used to identify hidden patterns or natural groupings in data. They play a vital role in data analysis, pattern recognition, and decision-making processes. K-means clustering is an unsupervised learning algorithm used for data clustering which groups un-labeled data points into groups or clusters. K-means is an iterative, centroid-based clustering algorithm that partitions a dataset into similar groups based on the distance between their centroids.

II Industry/Employer Expected Outcome(s)

- Apply machine learning effectively across various domains.
- Guide data-driven decision-making to achieve specific business goals.

III Course Level Learning Outcome(s)

CO5: Apply unsupervised learning models to train and evaluate.

IV Laboratory Learning Outcome(s)

LLO 15.1 Write a program to make groups based on their features using K-Means Clustering.

V Relevant Affective Domain related outcome(s)

1. Follow safety practices.

2. Follow ethical practices.

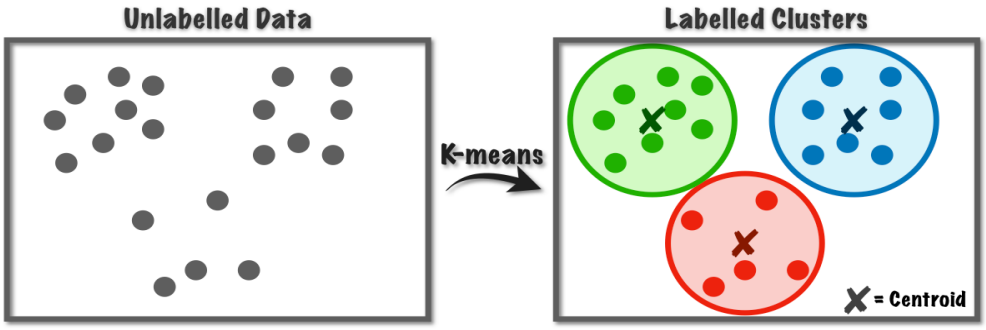
VI Relevant Theoretical Background

Clustering algorithms are unsupervised machine learning techniques used to group data points into clusters based on their inherent similarities. These algorithms aim to uncover hidden patterns and structures within unlabeled datasets.

K-Means Clustering:

K-Means Clustering is an unsupervised machine learning algorithm that helps group data points into clusters based on their inherent similarity. Unlike supervised learning, where we train models using labeled data, K-Means is used when we have data that is not labeled and the goal is to uncover hidden patterns or structures. For example, an online store can use K-Means to segment customers into groups like "Budget Shoppers," "Frequent Buyers," and "Big Spenders" based on their purchase history.

Example: Segmenting customers into groups based on purchasing behavior.



K-Means Clustering

Working of K-Means Clustering

Suppose we are given a data set of items with certain features and values for these features like a vector. The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm. "k" represents the number of groups or clusters we want to classify our items into.

The algorithm will categorize the items into "k" groups or clusters of similarity. To calculate that similarity we will use the Euclidean distance as a measurement.

The algorithm works as follows:

1. **Initialization:** We begin by randomly selecting k cluster centroids.
2. **Assignment Step:** Each data point is assigned to the nearest centroid, forming clusters.
3. **Update Step:** After the assignment, we recalculate the centroid of each cluster by averaging the points within it.
4. **Repeat:** This process repeats until the centroids no longer change or the maximum number of iterations is reached.

VII Resources Required:

Sr. No.	Name of the Resources	Specifications	Remark if Any
1.	Computer System	Computer System with CPU : i3-i5 preferable, RAM 8 GB or more, with Windows 8 or Above or Linux OS.	
2.	Python	Python 3.7 above	
3.	Python Interpreter / IDE	Any open source IDE such IDLE, Colab Notebook, Jupyter Notebook, Google Colab Free/unpaid Account/ Anaconda.	
4.	Other Useful Tools:	VS Code / PyCharm – Alternative Python development environments. Git & GitHub – For version control and project collaboration. Kaggle – Free datasets and cloud computing for ML experiments.	

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 Procedure:

- **Import necessary libraries.**
- **Load Dataset:**
  - Define the sample Dataset (Customer Purchases: Annual Income vs Spending Score)
- **Selecting features**
  - Select the features from sample dataset.
- **Standardize the features:**
  - Standardize the features which are important for clustering performance.
- **Apply K-Means with 3 clusters**:
  - Apply the K- Means and label the cluster
- **Plotting the clusters:**
  - Compute the model's accuracy on the test data:
- **Output:**

  Display the output and plot the clusters.

- **Source Code**

Problem Statement: Write a program to make groups based on their features using K-Means Clustering.

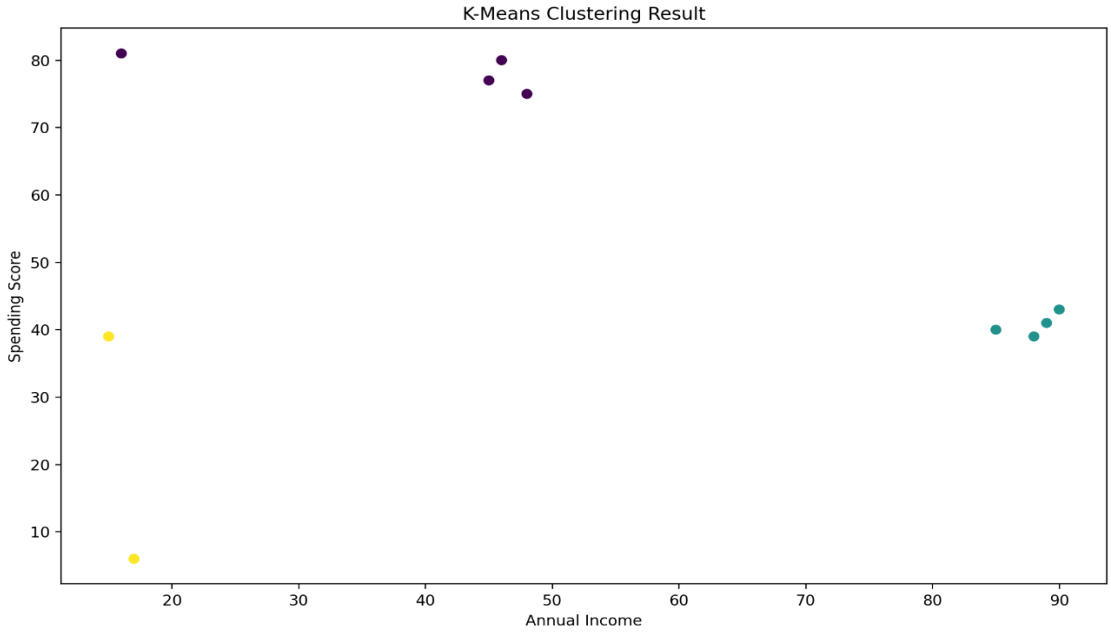
```
# Import required libraries
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```
# Sample Dataset (Customer Purchases: Annual Income vs Spending Score)
data = {
   'Annual_Income': [15, 16, 17, 45, 46, 48, 85, 89, 88, 90],
   'Spending_Score': [39, 81, 6, 77, 80, 75, 40, 41, 39, 43]
}
df = pd.DataFrame(data)

# Selecting features
X = df[['Annual_Income', 'Spending_Score']]

# Scaling the data (important for K-Means)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)

# Adding cluster labels to dataset
df['Cluster'] = kmeans.labels_

print("Clustered Data:")
print(df)

# Plotting the clusters
plt.scatter(df['Annual_Income'], df['Spending_Score'], c=df['Cluster'])
plt.title("K-Means Clustering Result")
plt.xlabel("Annual Income")
plt.ylabel("Spending Score")
plt.show()
```

- **Output**

Clustered Data:

	Annual_Income	Spending_Score	Cluster
0	15	39	2
1	16	81	0
2	17	6	2
3	45	77	0
4	46	80	0
5	48	75	0
6	85	40	1
7	89	41	1
8	88	39	1
9	90	43	1



X Conclusion:

……

……

……

……

XI Practical related Questions:

1. What is K-Means clustering?
2. Define centroid in K-Means.
3. What does the value **K** represent?
4. Explain how K-Means clustering algorithm works.
5. Write the formula for Euclidean distance.
6. Describe the difference between a cluster and a class.

XII Exercise:

1. Perform clustering on student performance dataset to group students by learning patterns.

2. Design a clustering-based system to segment users for an e-commerce marketing strategy.

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

……

……

……

……

……

……

XIII References:

- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
- https://www.ibm.com/think/topics/k-means-clustering
- https://www.geeksforgeeks.org/machine-learning/k-means-clustering-introduction/
- https://cocalc.com/features/jupyter-notebook

XIV Assessment Scheme (25 Marks)

Weightage- Process related: 60%		**15 Marks**
1	Practical Implementation with Specified time:40%	
2	Handling of Network components:10%	
3	Follow Ethical Practices:10%	
Weightage- Product related: 40%		**10 Marks**
4	Correctness of Practical:15%	
5	Timely Submission:15%	
6	Answer to Sample questions:10%	
Total marks (out of 25)		
Dated Signature of Course Teacher		