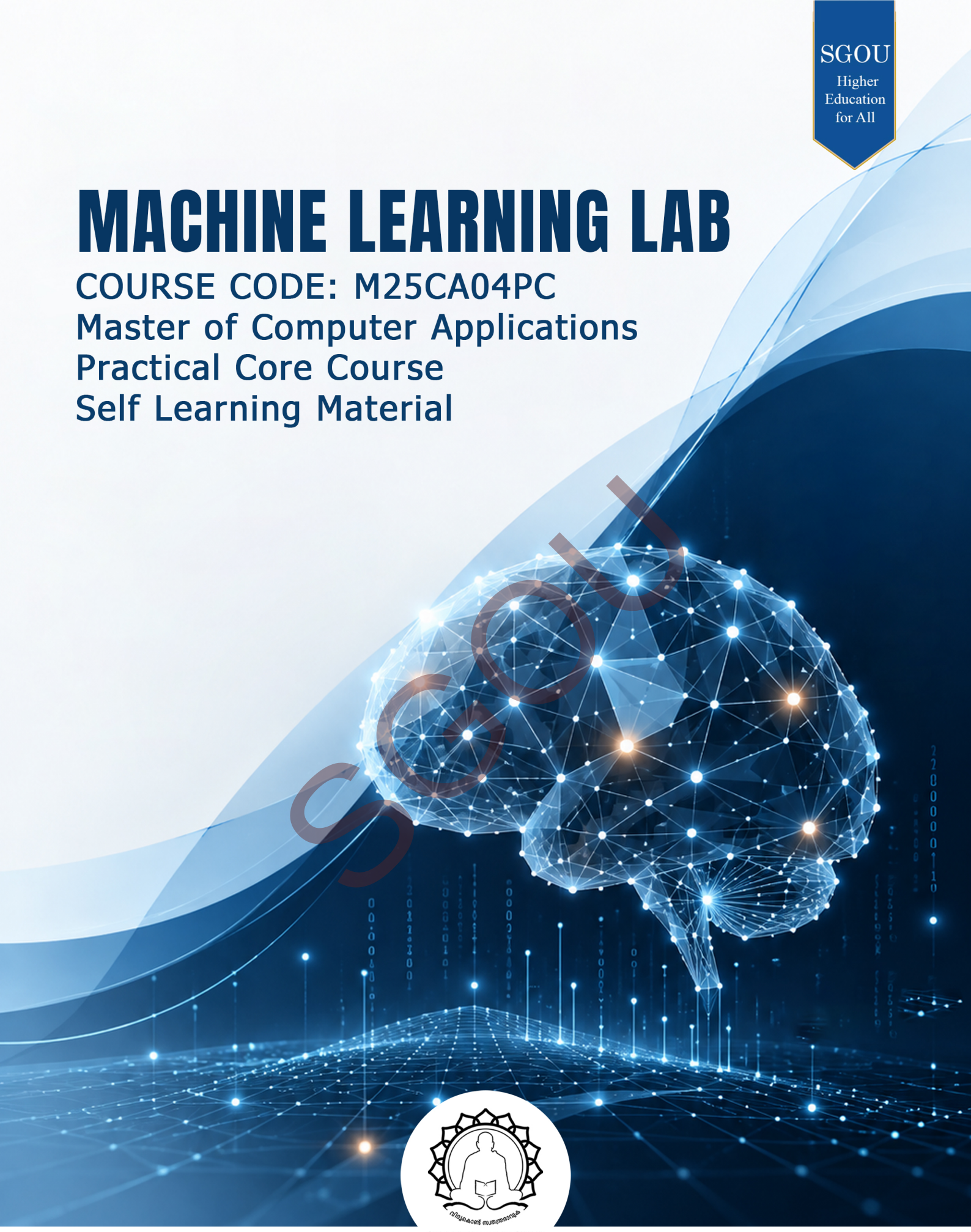
SGOU
Higher Education for All

MACHINE LEARNING LAB

COURSE CODE: M25CA04PC

Master of Computer Applications
Practical Core Course
Self Learning Material



SGOU

SREENARAYANAGURU OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU OPEN UNIVERSITY

SGOU

Vision

To increase access of potential learners of all categories to higher education, research and training, and ensure equity through delivery of high quality processes and outcomes fostering inclusive educational empowerment for social advancement.

Mission

To be benchmarked as a model for conservation and dissemination of knowledge and skill on blended and virtual mode in education, training and research for normal, continuing, and adult learners.

Pathway

Access and Quality define Equity.

Machine Learning Lab

Course Code: M25CA04PC

Semester - II

SGOU

Practical Core Course
Postgraduate Programme
Master of Computer Applications
Self Learning Material

SREENARAYANAGURU
OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU
OPEN UNIVERSITY

MACHINE LEARNING LAB

Course Code: M25CA04PC
Semester- II
Practical Core Course
Master of Computer Applications

Academic Committee

Prof. (Dr.) Sabu M.K.
Dr. G. Santhoshkumar T
Prof. (Dr.) Vinod Chandra S.S.
Dr. Vinod P.
Dr.Lajish V.L.
Sreekanth M.S.
Dr. Vivek P.
Dr. Arun K.S.
Dr. Abdul Jebbar P.

Development of the Content

Shamin S.
Greeshma P. P.
Sreerekha P. K.
Dr. Kanitha V. K.
Subi Priy Laxmi S. B. N.
Aswathy V. S.

Review and Edit

Dr. Sabeena K.

Scrutiny

Shamin S.
Greeshma P. P.
Sreerekha P. K.
Dr. Kanitha V. K.
Subi Priy Laxmi S. B. N.
Aswathy V. S.

Cover Design

Jobin J.

Co-ordination

Dr. Gopakumar C.
Head, School of ICS

Scan this QR Code for reading the SLM on a digital device.

Edition
September 2026

Copyright
© Sreenarayanaguru Open University

ISBN 978-81-181574-4-1

9 788118 157441

SGOU

All rights reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from Sreenarayanaguru Open University. Printed and published on behalf of Sreenarayanaguru Open University by Registrar, SGOU, Kollam.

www.sgou.ac.m

Visit and Subscribe our Social Media Platforms

MESSAGE FROM VICE CHANCELLOR

Dear Learner,

It gives me immense pleasure and a deep sense of pride to warmly welcome you to Sreenarayanaguru Open University—a vibrant and progressive institution committed to transforming lives through inclusive, flexible, and high-quality education.

Established in September 2020 as a forward-looking initiative of the Government of Kerala, the University stands as a beacon of opportunity for learners seeking to advance their academic and professional aspirations through the open and distance learning mode. Guided by our foundational principle that "access and quality define equity," we are steadfast in our mission to democratize education while upholding uncompromising academic standards.

Our university is inspired by the timeless vision and philosophy of Sree Narayana Guru, whose ideals of knowledge, equality, and social transformation continue to guide our academic journey. His enduring legacy instils in us the responsibility to create an educational environment that empowers individuals, nurtures critical thinking, and contributes meaningfully to society.

Understanding the dynamic needs of contemporary learners, we have adopted a robust and learner-centric blended learning model, seamlessly integrating Self-Learning Materials, Academic Counselling, and Advanced Digital Learning Platforms. This holistic approach ensures flexibility without sacrificing academic depth, enabling you to learn at your own pace while remaining meaningfully connected to a vibrant academic ecosystem.

The Master of Computer Applications (MCA) programme you are embarking upon is carefully designed to position you at the forefront of the digital revolution. It uniquely blends strong theoretical foundations with practical, industry-oriented competencies. The curriculum emphasizes algorithmic thinking, system design, programming, database management, networking, and emerging technologies such as artificial intelligence, data science, and cloud computing.
What sets this programme apart is its forward-thinking design, which offers:

- Opportunities for skill enhancement aligned with industry needs
- Multidisciplinary learning pathways for broader intellectual development
- Exposure to innovative and emerging technology domains
- Multiple specialization options tailored to evolving career landscapes
- A strong focus on employability, entrepreneurship, and global career readiness
-

Our Self-Learning Materials are meticulously developed by experts, enriched with contemporary case studies, real-world applications, and practical insights to ensure clarity, engagement, and relevance. We are committed to equipping you not just with knowledge, but with the confidence and competence to excel in a competitive global environment.

At Sreenarayanaguru Open University, you are never alone in your learning journey. Our dedicated learner support system is designed to provide continuous academic guidance, timely assistance, and effective grievance redressal. We encourage you to actively engage with us, share your concerns, and make the most of the resources and support available to you.

As you begin this important phase of your academic journey, I urge you to embrace learning with curiosity, discipline, and determination. The world of technology is ever-evolving, and your willingness to adapt, innovate, and grow will define your success.

Remember, this is not just a programme—it is a pathway to transforming your future.
I wish you a fulfilling learning experience and a successful, inspiring career ahead.

Warm regards,



Prof. (Dr.) Jagathy Raj V. P.
Vice Chancellor

SGOU

Contents

Introduction **1**

CYCLE 1 Supervised Learning and Data Preparation **16**

Experiment 1 Implement data preprocessing techniques: handle missing values, encode categorical variables, and normalize features. 17

Experiment 2 Implement Linear Regression to predict a continuous variable using Scikit-learn 25

Experiment 3 Implement Logistic Regression and evaluate results using accuracy scores 33

Experiment 4 Train a Decision Tree Classifier and compare it with Random Forest on a sample dataset 37

CYCLE 2 Advanced Models, Evaluation & Unsupervised Learning **42**

Experiment 1 Implement K-Nearest Neighbors (KNN) for classification and evaluate its performance 43

Experiment 2 Evaluate classification models using confusion matrix, precision, recall, and F1-score 47

Experiment 3 Apply K-Means clustering to discover patterns in unlabeled data 51

Experiment 4 Apply Principal Component Analysis (PCA) to reduce dimensions and visualize clusters 58

SGOU

Introduction

The Machine Learning Lab supports the Machine Learning course through a sequence of practical experiments covering data preprocessing, supervised learning, model evaluation, and unsupervised learning techniques using Python. The laboratory provides learners with hands-on experience in preparing datasets, implementing machine learning algorithms, training models, evaluating their performance, and interpreting the results obtained from experiments.

Machine learning models depend significantly on the quality and preparation of the data provided to them. Therefore, the laboratory begins with fundamental data preprocessing techniques such as handling missing values, encoding categorical variables, and feature normalization. Learners then implement supervised learning algorithms such as Linear Regression, Logistic Regression, Decision Tree, Random Forest, and K-Nearest Neighbors. The laboratory also introduces important model evaluation measures such as accuracy, confusion matrix, precision, recall, and F1-score.

The laboratory consists of two cycles.

Cycle I – Supervised Learning and Data Preparation introduces the fundamental concepts required for developing machine learning models. The experiments begin with data preprocessing, followed by Linear Regression and Logistic Regression. The cycle then introduces Decision Tree and Random Forest classifiers. These experiments help learners understand how data is prepared, how supervised learning models are trained, and how different algorithms can be used for prediction and classification tasks.

Cycle II – Advanced Models, Evaluation and Unsupervised Learning focuses on K-Nearest Neighbors, classification model evaluation, K-Means clustering, and Principal Component Analysis. Learners examine different evaluation measures for classification models and explore how unsupervised learning techniques can be used to discover patterns in unlabeled data. PCA is introduced as a dimensionality reduction technique for simplifying high-dimensional datasets and visualizing data in a lower-dimensional space.

Python is used as the primary implementation platform throughout the laboratory. Standard libraries and frameworks such as NumPy, Pandas, Matplotlib, and Scikit-learn are used for data preparation, model implementation, visualization, and performance evaluation. Each experiment concludes with analysis and interpretation of the results obtained during the implementation.

Cycle I: Supervised Learning and Data Preparation

Cycle I introduces the fundamental concepts of data preparation and supervised machine learning. The first experiment focuses on data preprocessing, which is an essential step

SGOU

before applying machine learning algorithms. Learners work with missing values, categorical variables, and numerical features and learn how preprocessing techniques can transform raw data into a suitable format for machine learning.

The second experiment introduces Linear Regression for predicting continuous numerical values. Learners implement Linear Regression using Scikit-learn, divide the dataset into training and testing sets, train the model, make predictions, and visualize the regression relationship between input and output variables.

The third experiment focuses on Logistic Regression, which is used for classification problems. Learners implement a classification model, predict class labels, and evaluate the model using accuracy. The experiment helps learners understand the difference between predicting continuous values and predicting discrete class labels.

The fourth experiment introduces Decision Tree and Random Forest classifiers. Learners implement both models using a sample dataset and compare their performance. The experiment provides practical understanding of single-tree and ensemble-based classification approaches and helps learners observe how different algorithms behave on the same dataset.

Cycle II: Advanced Models, Evaluation and Unsupervised Learning

Cycle II extends the practical study of machine learning to additional classification, model evaluation, clustering, and dimensionality reduction techniques. The first experiment introduces the K-Nearest Neighbors (KNN) algorithm. Learners implement KNN for classification and examine the effect of different values of K on classification performance.

The second experiment focuses specifically on classification model evaluation. Learners construct and interpret a confusion matrix and calculate accuracy, precision, recall, and F1-score. The experiment emphasizes that a machine learning model should not be evaluated using a single measure alone, particularly when different types of prediction errors have different implications.

The third experiment introduces K-Means clustering, an unsupervised learning technique used to discover groups or patterns in unlabeled data. Learners implement K-Means with different values of K, calculate cluster centroids and inertia, visualize the resulting clusters, and apply the algorithm to datasets such as the Iris dataset.

The final experiment introduces Principal Component Analysis (PCA) as a dimensionality reduction technique. Learners transform high-dimensional data into a smaller number of principal components, examine explained variance, and visualize the transformed data. The experiment helps learners understand how dimensionality reduction can simplify datasets while retaining important information.

SGOU

Objectives of the Machine Learning Lab

The objectives of the Machine Learning Lab are to:

- Understand the fundamental concepts and workflow of machine learning.
- Understand the importance of data preprocessing before model development.
- Implement data cleaning and preprocessing techniques using Python.
- Handle missing values and categorical variables in datasets.
- Apply feature normalization and standardization techniques.
- Implement supervised learning algorithms for prediction and classification.
- Develop Linear Regression models for continuous-value prediction.
- Develop Logistic Regression models for classification.
- Implement Decision Tree and Random Forest classifiers.
- Implement K-Nearest Neighbors for classification.
- Understand and apply different classification evaluation measures.
- Construct and interpret confusion matrices.
- Calculate accuracy, precision, recall, and F1-score.
- Understand the basic principles of unsupervised learning.
- Implement K-Means clustering to discover patterns in unlabeled data.
- Analyze the effect of different values of K in K-Means clustering.
- Understand dimensionality reduction using Principal Component Analysis.
- Visualize machine learning results using suitable graphical techniques.
- Interpret and analyze the results produced by machine learning models.
- Develop programming, analytical, and problem-solving skills through practical experiments.

Tools and Software Used

The Machine Learning Lab uses Python as the primary programming language. Standard Python libraries and the Scikit-learn machine learning framework are used for dataset preparation, model implementation, visualization, and evaluation.

SGOU

Programming Language

Python

Python is used as the primary programming language for all laboratory experiments. Its simple syntax and extensive collection of scientific and machine learning libraries make it suitable for implementing and experimenting with machine learning algorithms.

Integrated Development Environments (IDEs)

The laboratory programs can be developed using any suitable Python development environment.

Jupyter Notebook

- Interactive environment for writing and executing Python programs.
- Supports step-by-step execution of machine learning programs.
- Suitable for data preprocessing, model implementation, visualization, and result analysis.
- Supports Markdown for documenting experiments.

Google Colab

- Cloud-based Jupyter Notebook environment.
- Allows Python programs to be executed through a web browser.
- No local installation is required.
- Provides convenient access to commonly used Python libraries.
- Supports easy sharing and collaboration.

Visual Studio Code (VS Code)

- Lightweight code editor with Python support.
- Suitable for developing and debugging machine learning programs.
- Supports Python extensions and virtual environments.
- Can be used for managing laboratory source-code files.

PyCharm

- Integrated Development Environment for Python programming.
- Provides facilities for coding, debugging, and project management.
- Suitable for organizing larger machine learning projects.

SGOU

Machine Learning Framework and Libraries

The following libraries are used in the laboratory.

NumPy

NumPy provides support for numerical computation and array operations. It is useful for creating and manipulating numerical datasets and performing mathematical operations required in machine learning experiments.

Pandas

Pandas is used for loading, creating, organizing, and preprocessing datasets. It provides DataFrame and Series structures that make tabular data manipulation convenient.

Matplotlib

Matplotlib is used for creating graphs and visualizing machine learning results. It can be used to display regression lines, clusters, data distributions, and other experimental results.

Scikit-learn

Scikit-learn is the primary machine learning library used in this laboratory. It provides implementations of algorithms and utilities for:

- Data preprocessing
- Train-test splitting
- Linear Regression
- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbors
- K-Means clustering
- Principal Component Analysis
- Model evaluation

Hardware Requirements

A computer system with the following minimum configuration is suitable for performing the laboratory experiments:

- **Processor:** Dual-core processor or better

SGOU

- **RAM:** Minimum 4 GB
- **Storage:** Sufficient free space for Python, libraries, datasets, and laboratory programs
- **Internet Connection:** Required when using Google Colab or downloading datasets and libraries

The experiments in this laboratory generally involve small and medium-sized datasets and do not require specialized hardware.

Software Installation and Configuration

Step 1: Install Python

Download and install Python from the official Python website.

Python Official Website

During installation, enable the option to add Python to the system PATH.

Step 2: Verify Python Installation

Open Command Prompt or Terminal and execute:

python --version

If Python is installed correctly, the installed Python version will be displayed.

Step 3: Install Required Libraries

The major libraries required for the laboratory can be installed using:

pip install numpy pandas matplotlib scikit-learn jupyter

If Jupyter Notebook is required, it can also be installed separately using:

pip install notebook

Step 4: Verify Library Installation

Open Python or Jupyter Notebook and execute:

import numpy

import pandas

import matplotlib

import sklearn

If no error is displayed, the required libraries have been successfully installed.

SGOU

Datasets Used in the Machine Learning Lab

Datasets are an essential component of machine learning experiments. A machine learning model learns patterns and relationships from data. Therefore, the quality, structure, and suitability of the dataset have a direct influence on the learning process and the results obtained from the model.

The laboratory uses small sample datasets as well as standard datasets available through Scikit-learn and other commonly used sources. Different datasets are selected according to the requirements of each experiment.

Types of Datasets

The experiments use different types of datasets, including:

- Tabular datasets
- Numerical datasets
- Categorical datasets
- Classification datasets
- Regression datasets
- Unlabeled datasets for clustering

Commonly Used Datasets

Iris Dataset

The Iris dataset contains measurements of iris flowers and is commonly used for classification and clustering experiments. It contains four numerical features and three flower classes:

- Setosa
- Versicolor
- Virginica

The dataset is used in experiments involving Decision Tree, Random Forest, KNN, classification evaluation, and K-Means/PCA demonstrations.

Wine Dataset

The Wine dataset contains chemical measurements of wine samples belonging to different classes. It can be used for classification experiments and for comparing machine learning algorithms.

SGOU

Breast Cancer Wisconsin Dataset

The Breast Cancer Wisconsin dataset is a binary classification dataset containing numerical features derived from breast-cancer-related observations. It can be used for demonstrating classification algorithms and model evaluation.

Sample Regression Dataset

Small sample datasets containing independent and dependent numerical variables can be created using Python and Pandas for demonstrating Linear Regression.

Sample Classification Dataset

Small datasets containing input features and class labels can be created for demonstrating Logistic Regression and other classification algorithms.

Dataset Collection and Preparation

Before applying a machine learning algorithm, the dataset should be examined and prepared appropriately. The preparation process depends on the type of data and the requirements of the selected algorithm.

Steps in Dataset Preparation

- Select a suitable dataset for the experiment.
- Load or create the dataset using Python.
- Examine the structure and attributes of the dataset.
- Identify input features and target variables where applicable.
- Check for missing values.
- Identify categorical and numerical variables.
- Apply suitable preprocessing techniques.
- Scale or normalize features when required.
- Divide the dataset into training and testing sets for supervised learning.
- Verify the prepared dataset before model training.

Data Preprocessing Techniques

Data preprocessing is performed to transform raw data into a suitable form for machine learning algorithms.

SGOU

The major preprocessing techniques covered in this laboratory include:

Handling Missing Values

Missing observations can be handled by deleting selected records or replacing missing values using suitable statistical measures such as mean, median, or mode.

Encoding Categorical Variables

Categorical values such as gender, city, or product type can be converted into numerical representations using techniques such as label encoding and one-hot encoding.

Feature Scaling

Features having different numerical ranges can be transformed to comparable scales. The laboratory introduces:

- Min-Max Normalization
- Standardization

Feature scaling is particularly useful for algorithms such as KNN and K-Means that depend on distances.

Training and Testing of Machine Learning Models

Supervised machine learning models generally require labeled data for learning. The dataset is divided into training and testing portions.

Training Dataset

The training dataset is used by the algorithm to learn the relationship between input features and target values.

Testing Dataset

The testing dataset contains observations that were not used during training. It is used to evaluate how well the trained model performs on unseen data.

A typical workflow is:

Dataset → Preprocessing → Train-Test Split → Model Training → Prediction → Evaluation

The proportion of data allocated to training and testing depends on the size and nature of the dataset.

SGOU

Structure of Machine Learning Programs

Each laboratory program should follow a systematic structure. A common structure helps learners understand the machine learning workflow and makes programs easier to develop, execute, verify, and document.

General Program Structure

1. Import the required Python libraries.
2. Load or create the dataset.
3. Examine the dataset.
4. Perform necessary preprocessing.
5. Define input features and target variables.
6. Split the dataset into training and testing sets, where applicable.
7. Create the machine learning model.
8. Train the model using the training data.
9. Generate predictions using the trained model.
10. Evaluate the model using suitable performance measures.
11. Visualize the results where applicable.
12. Record the observations and final result.

Program Development Guidelines

Learners should follow the following guidelines while developing laboratory programs:

- Use meaningful variable and function names.
- Organize the program into logical sections.
- Add comments where necessary.
- Use appropriate datasets for each experiment.
- Verify the input data before model training.
- Use suitable preprocessing techniques.
- Record important model parameters.
- Execute the program and verify the output.
- Save the source code and notebook regularly.

SGOU

- Interpret the output rather than simply recording numerical values.
- Maintain originality in program development.

Model Evaluation

Model evaluation is an important stage of machine learning. A model should be evaluated using data that was not used during training.

The laboratory introduces the following classification evaluation measures:

Accuracy

Accuracy represents the proportion of correctly classified observations among all observations.

Confusion Matrix

A confusion matrix summarizes the actual and predicted class labels and helps identify correct predictions and different types of classification errors.

Precision

Precision indicates the proportion of correctly predicted positive observations among all observations predicted as positive.

Recall

Recall indicates the proportion of actual positive observations that are correctly identified by the model.

F1-Score

F1-score provides a balance between precision and recall and is useful when both types of errors need to be considered.

The appropriate evaluation measure depends on the type of machine learning problem and the characteristics of the dataset.

Visualization of Machine Learning Results

Visualization helps learners understand the behaviour of machine learning models and interpret experimental results.

Graphs can be used to represent:

- Regression relationships
- Predicted and actual values
- Classification results

SGOU

- Confusion matrices
- K-Means clusters
- Cluster centroids
- K-value versus accuracy
- PCA-transformed data
- Explained variance

Learners should use suitable graphical representations wherever visualization contributes to understanding the experiment.

Use of Jupyter Notebook

Jupyter Notebook is a suitable environment for conducting Machine Learning Lab experiments. It allows Python programs to be executed in individual cells and provides an interactive environment for experimentation and analysis.

Advantages

- Interactive program execution
- Step-by-step execution of code
- Easy visualization of results
- Convenient debugging and experimentation
- Markdown support for documentation
- Easy integration of code, output, graphs, and explanations
- Suitable for data preprocessing and machine learning experiments

Use of Google Colab Notebook

Google Colab (Collaboratory) is a cloud-based Jupyter Notebook environment that allows Python programs to be written and executed through a web browser. It can be used as an alternative to a locally installed Python environment.

Advantages of Google Colab

- No local installation is required
- Accessible through a web browser
- Provides a ready-to-use Python environment
- Supports commonly used machine learning libraries

SGOU

- Easy sharing and collaboration
- Notebooks can be stored in Google Drive
- Suitable for conducting laboratory experiments remotely

Additional Python libraries can be installed using:

pip install library_name

Expectations from Learners

Learners are expected to:

- Actively participate in all laboratory sessions.
- Read the relevant theory before performing each experiment.
- Understand the purpose of each program rather than merely copying the code.
- Complete and submit laboratory records on time.
- Execute programs and verify the generated outputs.
- Analyze and interpret experimental results.
- Maintain originality in program development.
- Follow ethical practices while using datasets and source code.
- Collaborate constructively with peers while maintaining individual understanding.
- Be prepared to explain programs during viva examinations.

Lab Guidelines and Code of Conduct

Learners should follow the instructions provided in the laboratory manual and by the instructor.

- Use authorized software, datasets, and computing resources.
- Maintain originality in laboratory work.
- Avoid plagiarism and unauthorized copying of programs.
- Maintain proper discipline during laboratory sessions.
- Save programs and notebooks regularly.
- Maintain backup copies of important work.

SGOU

- Do not modify or delete another learner's files.
- Use computing resources responsibly.
- Seek assistance from the instructor when difficulties arise.
- Maintain respectful communication with instructors, lab assistants, and peers.
- Violation of Lab Rules

Depending on the nature and seriousness of the violation, appropriate academic or disciplinary action may be taken. This may include:

- Warning
- Deduction of marks
- Requirement to repeat an experiment
- Restriction from a laboratory session in serious cases

Learners should use the laboratory as an environment for developing programming ability, analytical thinking, problem-solving skills, and practical understanding of machine learning concepts.

Result Documentation

Each laboratory experiment should be properly documented. The record should generally contain the following components:

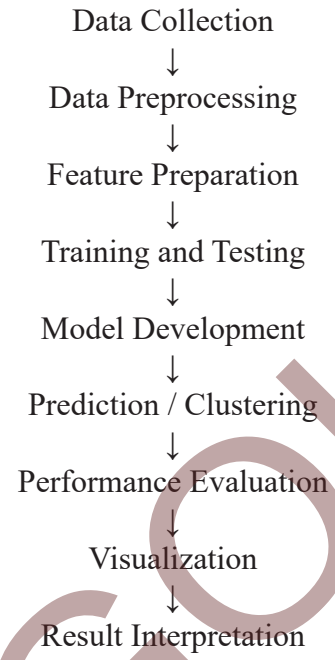
1. Experiment Number and Title
2. Objective
3. Theory
4. Dataset Used
5. Tools and Libraries Required
6. Algorithm / Procedure
7. Python Program
8. Output
9. Result
10. Observation

SGOU

The output should be verified before it is included in the laboratory record. When the output depends on random train-test splitting, initialization, or other variable conditions, the actual output obtained during execution should be recorded.

Overall Machine Learning Workflow

The experiments in this laboratory collectively demonstrate the major stages of a basic machine learning workflow:

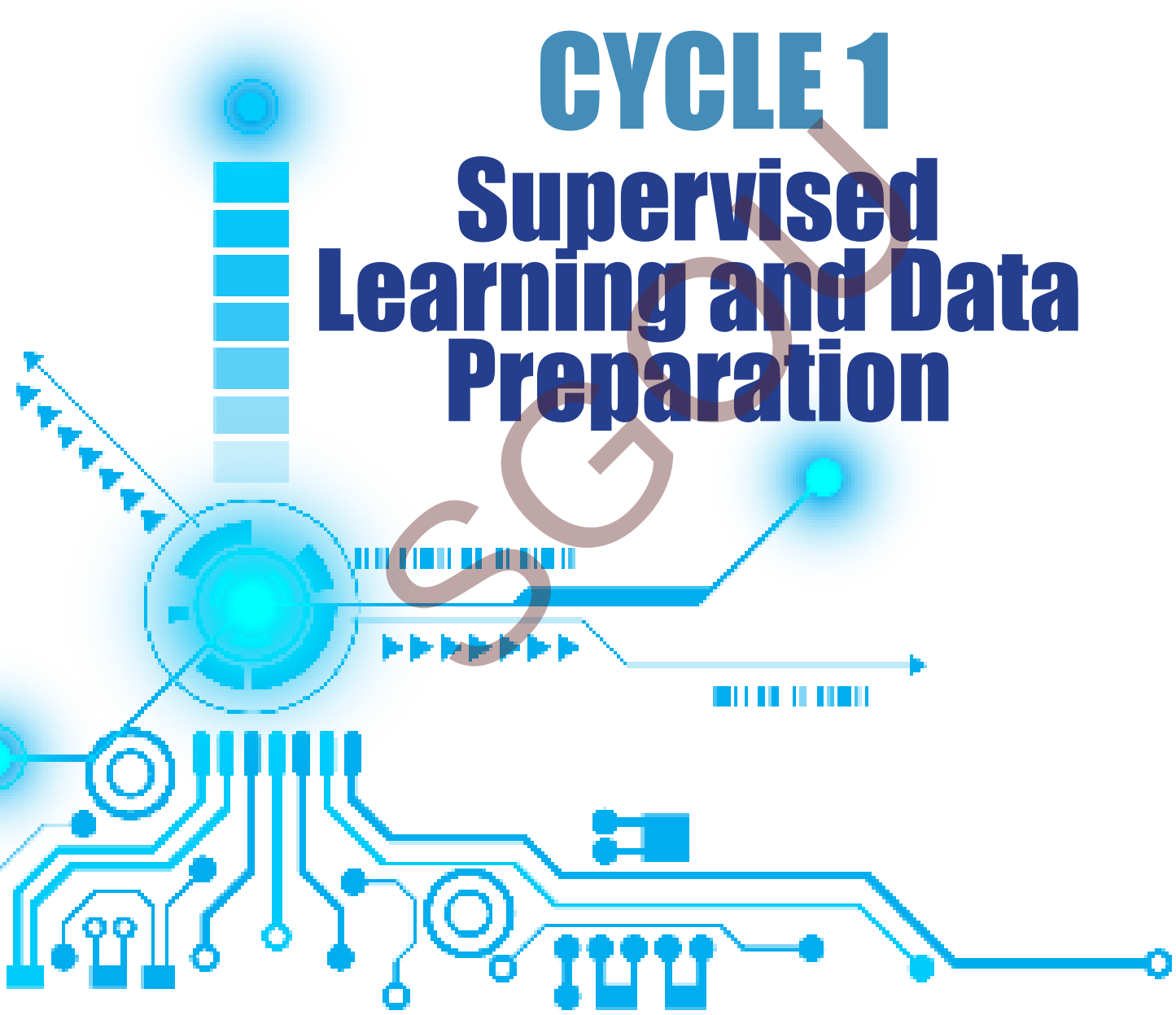


Through these experiments, learners progressively move from preparing raw data to implementing machine learning algorithms and analyzing their results. The laboratory therefore provides a practical foundation for applying machine learning techniques to real-world datasets and developing the computational skills required for further study in artificial intelligence and data science.

SGOU

CYCLE 1

Supervised Learning and Data Preparation



SGOU

Experiment 1

Implement data preprocessing techniques: handle missing values, encode categorical variables, and normalize features.

Objectives

- To understand the importance of data preprocessing in data analysis
- To learn how to handle missing values in datasets
- To understand encoding of categorical variables
- To apply normalization technique on numerical features
- To implement preprocessing steps using Python

1.1.1 Theory

1.1.1.1 Introduction to Data Preprocessing

In real-world applications, data is rarely available in a form that can be directly used for analysis or machine learning. Datasets may contain missing values, duplicate records, inconsistent formats, categorical values, and numerical features with different scales. If these issues are not handled properly, they can affect the performance and reliability of machine learning models.

Data preprocessing is the process of transforming raw data into a clean, consistent, and suitable format for analysis and machine learning. It is an important step in the machine learning workflow because the quality of the input data directly influences the quality of the results produced by a model.

For example, consider a dataset containing the age, salary, and city of several individuals. Age and salary are numerical features, while city is a categorical feature. Some age or salary values may be missing, and the city values cannot be directly processed by most machine learning algorithms because they are represented as text. Preprocessing techniques can be used to address these issues.

The major preprocessing tasks covered in this experiment are:

- Handling missing values
- Encoding categorical variables
- Scaling numerical features

SGOU

These techniques help transform raw data into a format that can be effectively used for machine learning algorithms.

1.1.1.2 Handling Missing Values

Missing values occur when information about one or more observations is not available in a dataset. They may result from errors during data collection, incomplete forms, sensor failures, or unavailable information.

For example:

Age	Salary
25	50000
30	NaN
NaN	60000
22	45000

Here, NaN represents a missing value.

Missing values can cause problems because many machine learning algorithms cannot directly process incomplete data. Therefore, they need to be identified and appropriately handled before model training.

Common approaches include:

- **Deleting records:** Rows containing missing values can be removed when only a small number of records are affected.
- **Deleting features:** A column may be removed when a large proportion of its values are missing and the feature is not essential.
- **Mean imputation:** Missing numerical values are replaced with the mean of the available values.
- **Median imputation:** Missing numerical values are replaced with the median of the available values.
- **Mode imputation:** Missing categorical values are replaced with the most frequently occurring category.

The appropriate method depends on the nature of the data and the amount of missing information. Mean and median imputation are commonly used for numerical features, while mode imputation can be used for categorical features.

In this experiment, missing numerical values are replaced using the mean.

1.1.1.3 Encoding Categorical Variables

Categorical variables represent groups or categories rather than numerical quantities. Examples include gender, city, department, product type, and payment method.

SGOU

For example:

City
Kollam
Kochi
Kollam
Trivandrum

Most machine learning algorithms require numerical input. Therefore, categorical values need to be converted into numerical representations before they are provided to the model. This process is known as categorical encoding.

Two commonly used encoding techniques are:

Label Encoding:

Each category is assigned a numerical value. For example:

Gender	Encoded Value
Male	0
Female	1

Label encoding is particularly suitable for categories with a meaningful order or for certain binary categorical variables. However, assigning numbers to unordered categories may incorrectly suggest an order between them.

One-Hot Encoding:

A separate binary column is created for each category. For example, if the categories are Kochi, Kollam, and Trivandrum, three separate columns can be created. A value of 1 indicates that an observation belongs to that category, while 0 indicates that it does not.

One-hot encoding is commonly used for nominal categorical variables where there is no natural order among categories.

In this experiment, one-hot encoding is demonstrated using pandas.

1.1.1.4 Feature Scaling and Normalization

Numerical features in a dataset may have very different ranges. For example, age may range from 20 to 60, while salary may range from 20,000 to 80,000. Some machine learning algorithms may be influenced by these differences in scale.

Feature scaling transforms numerical features to comparable scales. It is particularly important for algorithms that rely on distances or numerical magnitudes, such as K-Nearest Neighbors, K-Means clustering, and many optimization-based algorithms.

Two commonly used scaling techniques are Min-Max Normalization and Standardization.

SGOU

Min-Max Normalization

Min-Max normalization transforms values to a specified range, commonly 0 to 1.

The formula is:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where:

- X is the original value
- $X_{\min}$ is the minimum value of the feature
- $X_{\max}$ is the maximum value of the feature
- X' is the normalized value

After Min-Max normalization, the minimum value becomes 0 and the maximum value becomes 1.

Standardization

Standardization transforms a feature so that it has a mean of approximately 0 and a standard deviation of 1.

The formula is:

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

where:

- X is the original value
- μ is the mean of the feature
- σ is the standard deviation
- Z is the standardized value

Unlike Min-Max normalization, standardization does not restrict values to the range 0 to 1. The resulting values can be negative or positive.

Thus, feature scaling helps prevent features with larger numerical ranges from disproportionately influencing the machine learning algorithm.

SGOU

1.1.2 Implementation

Implementation 1 : Handling Missing Values

```
import pandas as pd
import numpy as np
# Create sample data
data = {
    'Age': [25, 30, np.nan, 22],
    'Salary': [50000, np.nan, 60000, 45000]
}
df = pd.DataFrame(data)
print("Original Data:")
print(df)
# Fill missing values with mean
df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Salary'] = df['Salary'].fillna(df['Salary'].mean())
print("\nAfter Handling Missing Values:")
print(df)
```

OUTPUT

```
Original Data:
   Age   Salary
0  25.0  50000.0
1  30.0      NaN
2   NaN  60000.0
3  22.0  45000.0
After Handling Missing Values:
   Age              Salary
0  25.000000    50000.000000
1  30.000000    51666.7000000
2  25.7000000   60000.000000
3  22.000000    45000.000000
```

SGOU

Implementation 2: Encoding Categorical Data

```
import pandas as pd
# Sample data
data = {
    'City': ['Kollam', 'Kochi', 'Kollam', 'Trivandrum']
}
df = pd.DataFrame(data)
print("Original Data:")
print(df)
# One-Hot Encoding
df_encoded = pd.get_dummies(df, columns=['City'])
print("\nEncoded Data:")
print(df_encoded)
```

OUTPUT

```
Original Data:
        City
0     Kollam
1      Kochi
2     Kollam
3  Trivandrum
Encoded Data:
   City_Kochi  City_Kollam  City_Trivandrum
0           0            1                0
1           1            0                0
2           0            1                0
3           0            0                1
```

SGOU

Implementation 3 : Normalization

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
# Sample data
data = {
    'Age': [20, 30, 40, 50],
    'Salary': [20000, 40000, 60000, 80000]
}
df = pd.DataFrame(data)
print("Original Data:")
print(df)
# Apply normalization
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df)
df_scaled = pd.DataFrame(df_scaled, columns=df.columns)
print("\nNormalized Data:")
print(df_scaled)
```

OUTPUT

```
Original Data:
   Age  Salary
0   20   20000
1   30   40000
2   40   60000
3   50   80000
Normalized Data:
        Age    Salary
0  0.000000  0.000000
1  0.333333  0.333333
2  0.666667  0.666667
3  1.000000  1.000000
```

SGOU

1.1.3 Lab Practice Questions

1. Create a dataset with missing values and replace them using median.
2. Perform label encoding on a column containing gender values (Male/ Female).
3. Apply one-hot encoding on a dataset with three categories.
4. Normalize a dataset using Min-Max scaling.
5. Standardize a dataset using mean and standard deviation.
6. Combine all preprocessing steps on a single dataset.
7. Compare the range and relative scale of features before and after Min-Max scaling.

SGOU

Experiment 2

Implement Linear Regression to predict a continuous variable using Scikit-learn

Objectives

- Understand the basic concept of Linear Regression.
- Identify the independent and dependent variables in a regression problem.
- Implement Linear Regression using Scikit-learn.
- Train a regression model using a given dataset.
- Predict a continuous value for a new input.

1.2.1 Theory

1.2.1.1 Introduction to Linear Regression

Linear Regression is a supervised machine learning algorithm used to predict a continuous numerical value based on one or more input variables. It learns the relationship between the input variable and the output variable from the given training data.

For example, Linear Regression can be used to predict house price based on house area, sales based on advertising expenditure, or salary based on years of experience.

In Simple Linear Regression, only one input variable is used. The relationship between the input and output is represented using a straight-line equation:

$$y = mx + c$$

where:

- x : independent or input variable
- y : dependent or predicted variable
- m : slope of the regression line
- c : intercept of the regression line

The slope (m) indicates the change in the predicted output for a one-unit change in the input variable. The intercept (c) represents the predicted output when the input value is zero.

The values of m and c are learned from the training data.

SGOU

1.2.1.2 Working Principle of Linear Regression

The working of Linear Regression can be understood through the following steps.

Step 1 : Provide the Data

The model receives a dataset containing input values and their corresponding output values.

For example, a dataset may contain:

- House Area – input variable
- House Price – output variable

Step 2 : Identify the Variables

The input variable is represented as x, and the target variable is represented as y.

x = Input Variable

y = Target Variable

The model learns the relationship between these two variables from the available data.

Step 3 : Split the Dataset

The available dataset is divided into training data and testing data. The training data is used to train the model, while the testing data is used to evaluate how well the trained model performs on unseen data.

Step 4 : Train the Model

The training data is provided to the Linear Regression model. The model analyses the input and output values and learns suitable values for the slope and intercept. In Scikit-learn, the model is trained using the fit() method.

Step 5 : Form the Regression Line

After training, the model uses the learned slope and intercept to form the best-fit regression line. The line represents the relationship between the input and output variables and can be expressed using the equation:

$$y = mx + c$$

Step 6: Make a Prediction

A new input value is provided to the trained model. The model uses the learned regression line to estimate the corresponding continuous output value. In Scikit-learn, the predict() method is used to generate the prediction.

SGOU

1.2.1.3 Algorithm: Linear Regression Using Scikit-learn

Step 1 : Start the procedure.

Step 2 : Import the required Python libraries and the LinearRegression class from Scikit-learn.

Step 3 : Load or create the dataset containing the input feature and corresponding target values.

Step 4 : Identify the independent variable as the input feature and the dependent variable as the target variable.

Step 5 : Separate the input features into X and the target values into y.

Step 6 : Divide the dataset into training and testing subsets.

Step 7 : Initialize the Linear Regression model using the LinearRegression() class.

Step 8 : Train the model using the training data by applying the fit() method.

Step 9 : Provide a new input value to the trained model for prediction.

Step 10 : Generate the predicted continuous value using the predict() method.

Step 11 : Display the predicted value obtained from the model.

Step 12 : Plot the actual data points and the learned regression line to visualize the relationship between the input and target variables.

Step 13 : End the procedure.

1.2.2 Implementation

Write a Python program using Scikit-learn to implement Linear Regression for predicting the price of a house based on its area. Train the model using the given dataset, predict the price for a new house with an area of 2600 sq.ft., and display the regression line.

SGOU

Program

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Create the dataset
data = {
    'Area': [1000, 1200, 1400, 1600, 1800,
             2000, 2200, 2400, 2600, 2800,
             3000, 3200, 3400],
    'Price': [200, 240, 280, 320, 360,
              400, 440, 480, 520, 560,
              600, 640, 680]
}
# Convert the data into a DataFrame
df = pd.DataFrame(data)
# Display the dataset
print("Dataset:")
print(df)
# Separate input and target variables
X = df[['Area']]
y = df['Price']
# Split the dataset into training and testing data
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)
```

SGOU

```
# Create the Linear Regression model
model = LinearRegression()
# Train the model
model.fit(X_train, y_train)
# Display the slope and intercept
print("\nSlope:", model.coef_[0])
print("Intercept:", model.intercept_)
# Predict the price for a new house
new_house = pd.DataFrame({
    'Area': [2600]
})
predicted_price = model.predict(new_house)[0]
print("\nPredicted Price for 2600 sq.ft.:", predicted_price)
# Plot the actual data points
plt.scatter(
    X,
    y,
    color='blue',
    label='Actual Data'
)
# Plot the regression line
plt.plot(
    X,
    model.predict(X),
    color='red',
    label='Regression Line'
)
```

SGOU

```
# Mark the new input and predicted value
plt.scatter(
    [2600],
    [predicted_price],
    color='green',
    s=120,
    label='Prediction'
)
# Draw a vertical dashed line from the prediction to the x-axis
plt.plot(
    [2600, 2600],
    [0, predicted_price],
    linestyle='--',
    color='green'
)
# Add annotation to the prediction point
plt.annotate(
    f'2600 sq.ft. → {predicted_price:.0f}',
    xy=(2600, predicted_price),
    xytext=(2100, 570),
    arrowprops=dict(arrowstyle='->'),
    fontsize=11
)
# Add labels and title
plt.xlabel('House Area (sq.ft.)')
plt.ylabel('House Price')
plt.title('Linear Regression: House Area vs House Price')
# Display legend and grid
```

SGOU

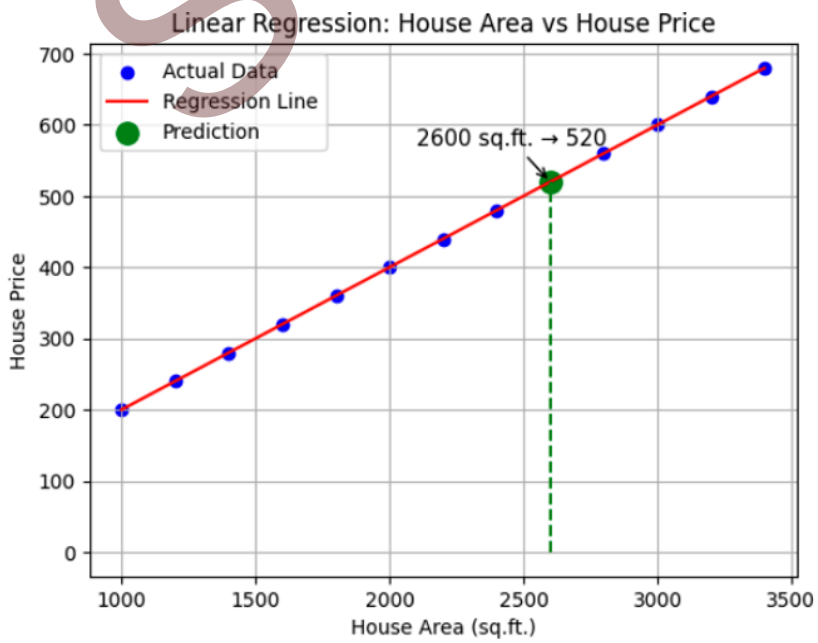
```
plt.legend()
plt.grid(True)
# Display the graph
plt.show()
```

OUTPUT

```
Dataset:
    Area  Price
0   1000    200
1   1200    240
2   1400    280
3   1600    320
4   1800    360
5   2000    400
6   2200    440
7   2400    480
8   2600    520
9   2800    560
10  3000    600
11  3200    640
12  3400    680

Slope: 0.2
Intercept: 0.0

Predicted Price for 2600 sq.ft.: 520.0
```



SGOU

1.2.3 Lab Practice Questions

1. Implement Linear Regression using Scikit-learn to predict the salary of an employee based on years of work experience. Create a suitable dataset, train the model, and predict the salary for a new employee with a given number of years of experience.

2. Modify the given Linear Regression program to predict the price of a house based on its area. Train the model using a suitable dataset and predict the price of a new house with an area of 3500 sq.ft. Display the predicted value and the regression graph.

3. Implement Linear Regression using Scikit-learn to predict a student's examination marks based on the number of hours studied. Create a suitable dataset, train the model, and predict the marks for a student who studies for a given number of hours.

4. Implement Linear Regression using Scikit-learn to predict the sales of a product based on advertising expenditure. Create a suitable dataset, train the model, and predict the expected sales for a given advertising expenditure. Display the predicted value and the regression graph.

5. Implement Linear Regression using Scikit-learn to predict the electricity consumption of a household based on the number of hours of appliance usage. Create a suitable dataset, train the model, and predict the electricity consumption for a new usage value. Display the predicted output and visualize the regression line.

SGOU

Experiment 3

Implement Logistic Regression and evaluate results using accuracy scores

Objectives

- To understand the concept of Logistic Regression
- To implement Logistic Regression using Python
- To perform classification on a dataset
- To evaluate model performance using accuracy score

1.3.1 Theory

1.3.1.1 Introduction

Logistic Regression is a supervised machine learning algorithm used for classification tasks. It predicts the probability that a given input belongs to a particular class.

The output of Logistic Regression lies between 0 and 1, making it suitable for binary classification problems such as:

- Spam vs Not Spam
- Pass vs Fail
- Disease vs No Disease

Instead of fitting a straight line (like Linear Regression), Logistic Regression uses a Sigmoid function to produce probabilities.

1.3.1.2 Types of Logistic Regression

Binary Logistic Regression

- Output has two classes (0 or 1)
- Example: Yes/No

Multinomial Logistic Regression

- More than two categories
- Example: Classifying types of fruits

SGOU

Ordinal Logistic Regression

- Categories with order
- Example: Rating (Low, Medium, High)

1.3.1.3 Working Principle

Logistic Regression uses the Sigmoid Function:

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

Where:

- $z = wX + b$
- Output is a probability between 0 and 1

Decision Rule:

- If output ≥ 0.5 → Class 1
- If output < 0.5 → Class 0

Thus logistic regrevion can be used for binary classification. It can also be extended to handle multi-class classification

1.3.1.4 Algorithm

1. Import necessary libraries
2. Load dataset
3. Separate features (X) and target (y)
4. Split dataset into training and testing sets
5. Create Logistic Regression model
6. Train the model using training data
7. Predict using test data
8. Evaluate using accuracy score

SGOU

1.3.2 Implementation

```
# Import required libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
# Load dataset
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
# Define features and target
X = data.data
y = data.target
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# Create Logistic Regression model
model = LogisticRegression(max_iter=5000)
# Train the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
# Display result
print("Accuracy Score:", accuracy)
```

SGOU

OUTPUT

```
Accuracy Score: 0.95
```

1.3.3 Lab Practice Questions

Use the Iris dataset to classify flowers based on their measurements.

Tasks:

1. Implement Logistic Regression .
2. Change the test size (e.g., 0.3) and compare accuracy.
3. Compute and display the confusion matrix.
4. Compare results with another classifier like KNN.
5. Perform Logistic Regression on a custom dataset.
6. Experiment with different values of max_iter and observe results.
7. Plot the ROC Curve

Use the Diabetes dataset available in Scikit-learn or another suitable dataset.

Tasks:

1. Load the Iris dataset.
2. Display the first five records.
3. Select appropriate input features and target variables.
4. Split the dataset into training and testing sets.
5. Train a Logistic Regression classifier.
6. Predict the class of the test samples.
7. Calculate the accuracy score.
8. Display the predicted and actual classes.

SGOU

Experiment 4

Train a Decision Tree Classifier and compare it with Random Forest on a sample dataset

Objectives

- To understand the working of Decision Tree and Random Forest algorithms.
- To implement both classifiers using a sample dataset.
- To evaluate and compare their performance using accuracy.
- To analyze which model performs better and why.

1.4.1 Theory

Introduction

In machine learning, classification algorithms are used to predict categories or labels based on input data. Two widely used classification techniques are Decision Tree and Random Forest. A Decision Tree is simple and interpretable, while Random Forest improves performance by combining multiple decision trees. This lab helps students gain practical knowledge of both methods and understand their differences through implementation.

1.4.1.1 Decision Tree Classifier

A Decision Tree is a supervised learning algorithm that splits the dataset into smaller subsets based on feature values. It creates a tree-like structure where:

- Each internal node represents a feature (attribute)
- Each branch represents a decision rule
- Each leaf node represents an output (class label)

Advantages:

- Easy to understand and visualize
- Requires less data preprocessing

SGOU

Limitations:

- Prone to overfitting
- Sensitive to noise in data

1.4.1.2 Random Forest Classifier

A Random Forest is an ensemble learning technique that builds multiple decision trees and combines their outputs. It uses:

- Bagging (Bootstrap Sampling) to create different datasets
- Random Feature Selection to introduce diversity

Final prediction is made using:

- Majority voting (classification)

Advantages:

- Reduces overfitting
- Higher accuracy compared to a single decision tree

Limitations:

- More computationally expensive
- Less interpretable

1.4.2 Implementation

1.4.2.1 Tools Required

- Python
- Jupyter Notebook / Google Colab
- Libraries: pandas, scikit-learn

1.4.2.2 Dataset Used

We use the wine quality dataset, which contains 11 features.

SGOU

1.4.2.3 Implementation step1: Import Libraries

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

1.4.2.4 Implementation step2: Load the Dataset

```
data = load_wine()
X = data.data
y = data.target
```

1.4.2.5 Implementation step3: Split the Dataset

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

1.4.2.6 Implementation step4: Train Decision Tree Model

```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_predictions)
print("Decision Tree Accuracy:", dt_accuracy)
print(classification_report(y_test, dt_predictions))
```

SGOU

1.4.2.7 Implementation step5 : Train Random Forest Model

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Random Forest Accuracy:", rf_accuracy)
print(classification_report(y_test, rf_predictions))
```

1.4.2.8 Implementation step6 : Comparison of result

```
print("Comparison:")
print("Decision Tree Accuracy:", dt_accuracy)
print("Random Forest Accuracy:", rf_accuracy)
```

OUTPUT

```
Decision Tree Accuracy: 0.9629
Random Forest Accuracy: 1.0
```

Observation: Random Forest generally gives higher accuracy due to ensemble learning.

The Random Forest Classifier performs better than the Decision Tree Classifier in terms of accuracy and generalization. It reduces overfitting by combining multiple trees.

Feature Comparison

Feature	Decision Tree	Random Forest
Model Type	Single Tree	Multiple Trees
Overfitting	High	Low
Accuracy	Moderate	High
Speed	Fast	Slower

1.4.3 Lab Practice Questions

1. Train both models on a different dataset (e.g., Breast cancer wisconsin dataset) and compare their performance.
2. Modify the number of trees (n_estimators) in Random Forest and analyze its effect on accuracy.

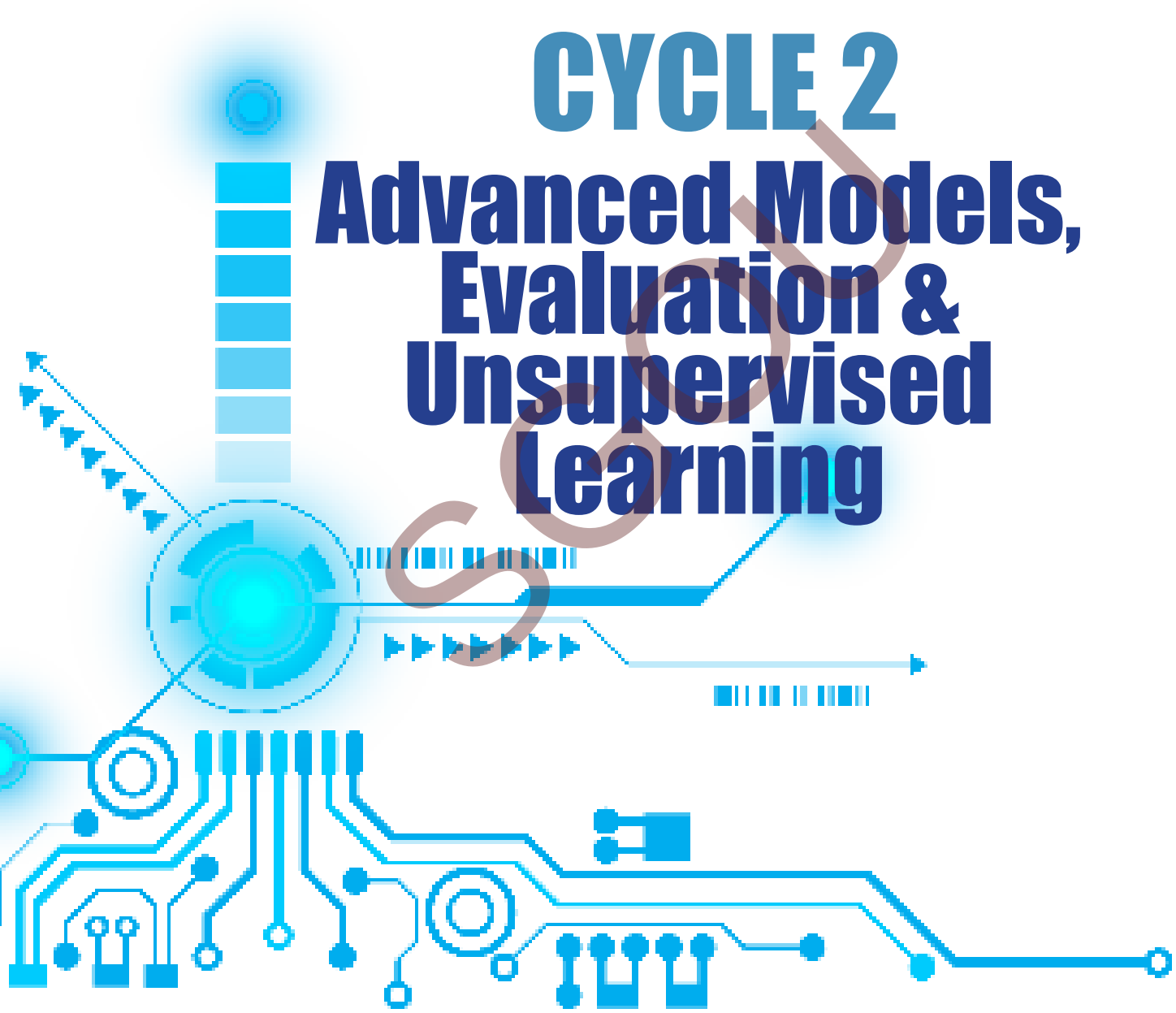
SGOU

3. Compare Decision Tree performance with and without limiting its depth (max_depth).

SGOU

CYCLE 2

Advanced Models, Evaluation & Unsupervised Learning



Experiment 1

Implement K-Nearest Neighbors (KNN) for classification and evaluate its performance

Objectives

- To understand the working principle of the K-Nearest Neighbors (KNN) classification algorithm.
- To implement KNN using a suitable programming language (e.g., Python).
- To analyze the effect of different values of K on classification accuracy.
- To evaluate the performance of the KNN model using metrics such as accuracy, precision, recall, and confusion matrix.

2.1.1 Theory

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It is a non-parametric and lazy learning algorithm, meaning it does not build a model during training but instead stores the training data.

In classification, KNN assigns a class label to a new data point based on the majority class among its K nearest neighbors in the feature space.

Key Concepts

- **Distance Metric:** Measures similarity (e.g., Euclidean distance).
- **K Value:** Number of nearest neighbors considered.
- **Majority Voting:** Determines the final class label.

2.1.1.1 Algorithm

1. Load the dataset.
2. Choose the value of K.
3. Calculate the distance between the test data point and all training data points.
4. Sort the distances in ascending order.
5. Select the top K nearest neighbors.

SGOU

6. Assign the class based on majority voting.

7. Repeat for all test samples.

2.1.1.2 Requirements

Hardware

- Computer with minimum 4GB RAM

Software

- Python (3.x)
- Libraries: NumPy, Pandas, Matplotlib, Scikit-learn

2.1.2 Implementation

```
# Import libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_
report
# Load dataset (Example: Iris dataset)
from sklearn.datasets import load_iris
data = load_iris()
X = data.data
y = data.target
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
# Create KNN model
k = 5
model = KNeighborsClassifier(n_neighbors=k)
# Train model
model.fit(X_train, y_train)
```

SGOU

```
# Predictions
y_pred = model.predict(X_test)
# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT

```
Accuracy: 0.9333333333333333
Confusion Matrix:
[[15  0  0]
 [ 0 11  1]
 [ 0  2 16]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.85      0.92      0.88        12
           2       0.94      0.89      0.91        18

    accuracy                           0.93        45
   macro avg       0.93      0.94      0.93        45
weighted avg       0.94      0.93      0.93        45
```

The KNN classifier was successfully implemented and tested. The performance was evaluated using accuracy and other metrics, showing effective classification based on nearest neighbors.

SGOU

2.1.3 Lab Practice Questions

1. Write a program to implement the K-Nearest Neighbors (KNN) algorithm for a given dataset (e.g., Iris dataset). Classify the test data and compute the accuracy of the model.

2. Implement KNN for different values of *K* (e.g., 1, 3, 5, 7, 9). Compare the classification accuracy for each value and identify the optimal *K*. Plot a graph of *K vs Accuracy* and interpret the results.

3. Using a dataset, apply KNN classification and evaluate the model using:

    - Confusion Matrix
    - Precision
    - Recall
    - F1-Score

Analyze the performance and discuss how well the model classifies different classes.

SGOU

Experiment 2

Evaluate classification models using confusion matrix, precision, recall, and F1-score

Objectives

- To understand the concept of classification model evaluation.
- To construct and interpret a confusion matrix.
- To compute precision, recall, and F1-score for a classification model.
- To analyze the effectiveness of a model using different evaluation metrics.

2.2.1 Theory

Classification is a supervised machine learning task in which a model learns to assign observations to predefined classes. Examples include classifying an email as spam or not spam, identifying whether a transaction is fraudulent or genuine, and classifying a flower into one of several species.

After training a classification model, it is necessary to evaluate how well the model performs on data that was not used during training. Evaluating the model only on the training data may give an overly optimistic estimate of its performance. Therefore, the available dataset is generally divided into training and testing sets. The training set is used to build the model, while the testing set is used to evaluate its performance on unseen data.

Accuracy is one of the commonly used classification metrics. However, accuracy alone may not provide sufficient information about the types of errors made by a model, particularly when the classes are imbalanced or when false positive and false negative errors have different consequences. Therefore, additional measures such as precision, recall, and F1-score are used to obtain a more detailed evaluation.

Confusion Matrix

A Confusion Matrix is a table used to describe the performance of a classification model.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

SGOU

2.2.1.1 Evaluation Metrics

1. Precision

Precision measures how many of the predicted positive instances are actually positive.

Precision=TP/(TP+FP)

2. Recall

Recall measures how many actual positive instances are correctly identified.

Recall=TP/(TP+FN)

3. F1-Score

F1-score is the harmonic mean of precision and recall.

F1=2×(Precision×Recall)/(Precision+Recall

2.2.1.2 Algorithm

4. Load the dataset.
5. Split the dataset into training and testing sets.
6. Train a classification model (e.g., logistic regression, KNN).
7. Predict the output for test data.
8. Construct the confusion matrix.
9. Calculate precision, recall, and F1-score.
10. Interpret the results.

2.2.1.3 Requirements

Hardware

- Computer with minimum 4GB RAM

Software

- Python (3.x)
- Libraries: NumPy, Pandas, Scikit-learn

SGOU

2.2.2 Implementation

```
# Import libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
# Load dataset
data = load_iris()
X = data.data
y = data.target
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
# Train model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
# Predictions
y_pred = model.predict(X_test)
# Evaluation
cm = confusion_matrix(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print("Confusion Matrix:\n", cm)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

SGOU

OUTPUT

```
Confusion Matrix:

[[15  0  0]

 [ 0 15  0]

 [ 0  2 13]]

Accuracy: 0.9555555555555556

Precision: 0.9607843137254902

Recall: 0.9555555555555556

F1 Score: 0.9553571428571429
```

The classification model was evaluated successfully using a confusion matrix and performance metrics. The results provide a detailed understanding of model performance beyond simple accuracy.

2.2.3 Lab Practice Questions

1. Compute the confusion matrix and evaluation metrics for a binary classification dataset.
2. Compare precision and recall for two different models and interpret the results.
3. Modify the program to evaluate another classifier (e.g., logistic regression) and compare performance.

SGOU

Experiment 3

Apply K-Means clustering to discover patterns in unlabeled data

Objectives

- To implement the K-Means clustering algorithm on an unlabeled dataset
- To group data points into clusters based on similarity
- To understand centroid calculation and distance-based clustering
- To visualize clusters using graphical representation

2.3.1 Theory

2.3.1.1 Introduction to Clustering

Clustering is a fundamental technique in Machine Learning, specifically belonging to the category of unsupervised learning. Unlike supervised learning, where models are trained using labeled data, clustering works with unlabeled data and aims to discover hidden structures or patterns within it.

The primary goal of clustering is to group similar data points together while ensuring that points in different groups are as dissimilar as possible. These groups are called clusters.

Key Characteristics of Clustering

- No prior labels or categories are provided.
- The algorithm automatically identifies patterns in data.
- It is exploratory in nature and helps in data understanding.

Objective of Clustering

- **Maximize intra-cluster similarity**: Data points within the same cluster should be highly similar.
- **Minimize inter-cluster similarity**: Data points in different clusters should be as different as possible.

SGOU

Real-Life Example

Consider a supermarket trying to understand customer behavior. Without predefined categories, clustering can group customers based on purchasing patterns such as:

- Frequent buyers
- Occasional buyers
- High-value customers

This helps businesses make better decisions in marketing and product placement.

2.3.1.2 K-Means Clustering Algorithm

K-Means is one of the most widely used partition-based clustering algorithms. It divides a dataset into K distinct clusters, where K is a predefined number.

Each cluster is represented by a centroid, which is the mean (average) position of all data points in that cluster.

Working of K-Means Algorithm

The K-Means algorithm follows an iterative refinement process:

Step 1: Initialization

- Select the number of clusters K.
- Randomly initialize K centroids.

Step 2: Assignment Step

- Assign each data point to the nearest centroid based on a distance measure (usually Euclidean distance).

Step 3: Update Step

- Recalculate the centroids by taking the mean of all data points assigned to each cluster.

Step 4: Repeat

- Repeat the assignment and update steps until:
    - Centroids do not change significantly, or
    - Maximum number of iterations is reached.

SGOU

2.3.1.3 Objective Function

K-Means tries to minimize the Within-Cluster Sum of Squares (WCSS):

$$WCSS = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$

Where:

- C_i = cluster
- μ_i = centroid of cluster
- x = data point

This ensures clusters are compact and well-separated.

Advantages of K-Means

- Simple and easy to implement
- Computationally efficient for large datasets
- Works well when clusters are spherical and evenly sized

Limitations

- Requires predefining the number of clusters (K)
- Sensitive to initial centroid selection
- Not suitable for non-spherical or overlapping clusters
- Affected by outliers

2.3.1.4 Distance Measure

In clustering, distance metrics are used to determine the similarity or dissimilarity between data points and centroids.

The most commonly used distance measure is the Euclidean Distance.

- It calculates the straight-line distance between two points in a 2D plane.
- For higher dimensions, the formula extends similarly.

If two points are:

- A(2, 3)
- B(5, 7)

SGOU

Then,

$$d = \sqrt{(5-2)^2 + (7-3)^2}$$

$$= \sqrt{9+16}$$

$$= \sqrt{25}$$

$$= 5$$

2.3.2 Implementation

1. Write a Python program to implement K-Means clustering for K = 2 and display the clusters.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
X = np.array([
  [1, 2], [1.5, 1.8], [5, 8],
  [8, 8], [1, 0.6], [9, 11]
])
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
print("Cluster Labels:", labels)
print("Centroids:\n", centroids)
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='X')
plt.show()
```

SGOU

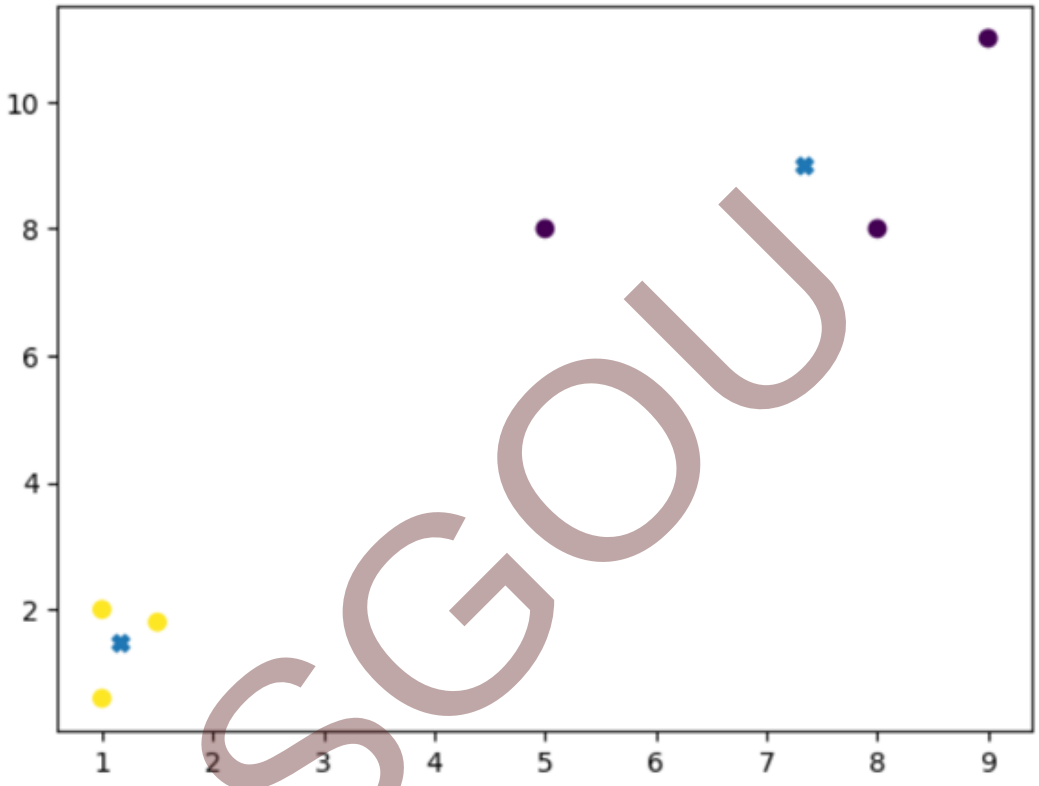
OUTPUT

```
Cluster Labels: [1 1 0 0 1 0]

Centroids:

[[7.33333333 9.        ]

[1.16666667 1.46666667]]
```



Graph shows 2 clusters with centroids marked as X

2. Write a Python program to implement K-Means clustering for K = 3 and observe the change in clustering.

```
kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

labels = kmeans.labels_

centroids = kmeans.cluster_centers_

print("Cluster Labels:", labels)

print("Centroids:\n", centroids)
```

SGOU

OUTPUT

```
Cluster Labels: [1 1 0 0 1 2]
Centroids:
[[ 6.5        8.        ]
 [ 1.16666667  1.46666667]
 [ 9.         11.        ]]
```

3. Write a Python program to calculate inertia (Within-Cluster Sum of Squares).

```
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
print("Inertia:", kmeans.inertia_)
```

OUTPUT

```
Inertia: 15.98
```

4. Write a Python program to display cluster labels for each data point.

```
for i, point in enumerate(X):
  print(f"Point {point} belongs to Cluster {labels[i]}")
```

OUTPUT

```
Point [1. 2.] belongs to Cluster 1
Point [1.5 1.8] belongs to Cluster 1
Point [5. 8.] belongs to Cluster 0
Point [8. 8.] belongs to Cluster 0
Point [1.  0.6] belongs to Cluster 1
Point [9. 11.] belongs to Cluster 2
```

SGOU

5. Write a Python program to apply K-Means clustering on the Iris dataset.

```
from sklearn.datasets import load_iris

data = load_iris()

X = data.data

kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

print("Cluster Labels:\n", kmeans.labels_)
```

OUTPUT

```
Cluster Labels:

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2

2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2

2 0]
```

2.3.3 Lab Practice Questions

1. Write a Python program to implement K-Means clustering for K = 4.
2. Modify the dataset and observe how clustering results change.
3. Write a program to visualize clusters using different color maps.
4. Implement K-Means and display centroids separately.
5. Write a program to compare results for different values of K.
6. Apply K-Means clustering on a real-world dataset and interpret the output.
7. Write a program to calculate inertia for different K values.
8. Implement clustering and display results in tabular form.

SGOU

Experiment 4

Apply Principal Component Analysis (PCA) to reduce dimensions and visualize clusters

Objectives

- To implement Principal Component Analysis (PCA) on a dataset
- To reduce high-dimensional data into lower dimensions
- To understand variance and feature transformation
- To visualize data in 2D or 3D space

2.4.1 Theory

2.4.1.1 Introduction to Dimensionality Reduction

Dimensionality reduction is a key concept in Machine Learning that involves reducing the number of input variables (features) in a dataset while preserving important information. High-dimensional data often leads to increased computational complexity and difficulty in visualization. Dimensionality reduction techniques help in simplifying data without losing significant patterns.

The main goal is to transform the original dataset into a new set of variables that retain most of the useful information.

2.4.1.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical technique used to transform high-dimensional data into a lower-dimensional space. It does this by identifying directions, called principal components, along which the variance in the data is maximized.

These principal components are:

- Orthogonal (independent of each other)
- Ordered by the amount of variance they capture

The first principal component captures the maximum variance, followed by the second, and so on.

SGOU

2.4.1.3 Working of PCA

PCA follows a systematic process:

Step 1: Standardization

The data is standardized so that each feature has a mean of zero and unit variance.

Step 2: Covariance Matrix Computation

A covariance matrix is computed to understand relationships between variables.

Step 3: Eigenvalues and Eigenvectors

Eigenvalues represent the amount of variance captured, and eigenvectors represent directions of principal components.

Step 4: Feature Vector Formation

Top eigenvectors are selected to form a feature vector.

Step 5: Data Projection

The original data is projected onto the new feature space.

Objective of PCA

PCA aims to:

- Maximize variance in reduced dimensions
- Minimize information loss
- Remove redundancy in features
- Improve visualization and efficiency

2.4.1.4 Advantages and Limitations

Advantages

- Reduces dimensionality effectively
- Removes correlated features
- Improves computational efficiency
- Helps in visualization

Limitations

- Loss of interpretability
- Sensitive to scaling

SGOU

- Assumes linear relationships
- May discard useful information

2.4.2 Implementation

1. Write a Python program to apply PCA and reduce data to 2 dimensions.

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Sample dataset
X = np.array([
  [2.5, 2.4],
  [0.5, 0.7],
  [2.2, 2.9],
  [1.9, 2.2],
  [3.1, 3.0],
  [2.3, 2.7],
  [2, 1.6],
  [1, 1.1],
  [1.5, 1.6],
  [1.1, 0.9]
])
# Standardize data
X_scaled = StandardScaler().fit_transform(X)
# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
print("Transformed Data:\n", X_pca)
```

SGOU

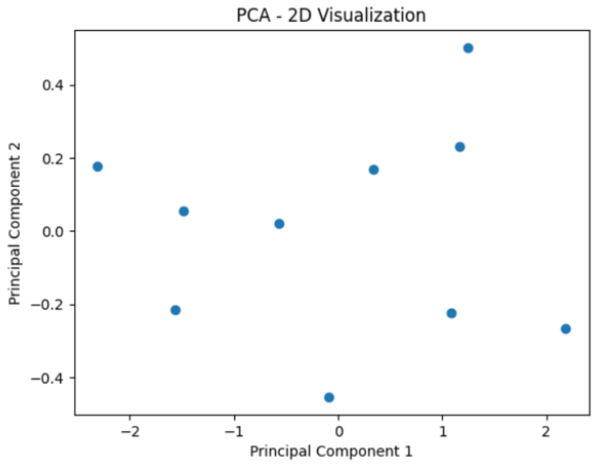
OUTPUT

```
Transformed Data:
[[ 1.08643242 -0.22352364]
[-2.3089372   0.17808082]
[ 1.24191895  0.501509  ]
[ 0.34078247  0.16991864]
[ 2.18429003 -0.26475825]
[ 1.16073946  0.23048082]
[-0.09260467 -0.45331721]
[-1.48210777  0.05566672]
[-0.56722643  0.02130455]
[-1.56328726 -0.21536146]]
```

2. Write a Python program to visualize PCA output in 2D.

```
import matplotlib.pyplot as plt
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.title("PCA - 2D Visualization")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

OUTPUT



Graph shows data projected onto 2 principal components

SGOU

3. Write a Python program to display explained variance ratio.

```
print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```

OUTPUT

```
Explained Variance Ratio: [0.96318131 0.03681869]
```

4. Write a Python program to apply PCA on Iris dataset.

```
from sklearn.datasets import load_iris
data = load_iris()
X = data.data
# Standardize
X_scaled = StandardScaler().fit_transform(X)
# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
print("Reduced Shape:", X_pca.shape)
```

OUTPUT

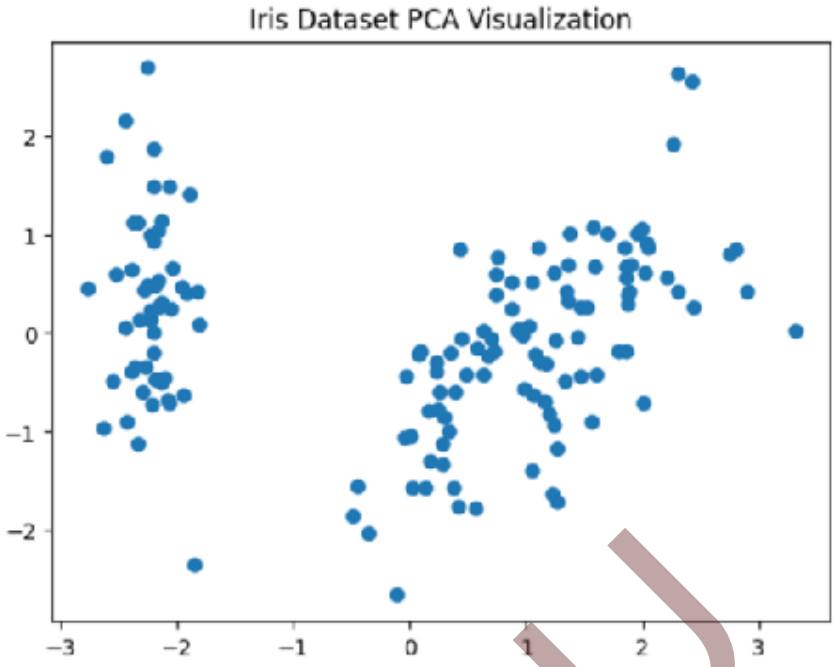
```
Reduced Shape: (150, 2)
```

5. Write a Python program to visualize Iris dataset using PCA.

```
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.title("Iris Dataset PCA Visualization")
plt.show()
```

SGOU

OUTPUT



Graph shows clusters of Iris data in 2D space

2.4.3 Lab Practice Questions

1. Write a Python program to reduce a dataset to 3 principal components.
2. Modify the program to visualize PCA in 3D.
3. Write a program to compare original and reduced dimensions.
4. Implement PCA and display eigenvalues and eigenvectors.
5. Apply PCA on a real-world dataset and interpret the results.
6. Write a program to calculate cumulative explained variance.
7. Modify PCA code using different scaling techniques.
8. Compare PCA with clustering techniques like K-Means.

SGOU

സർവ്വകലാശാലാഗീതം

വിദ്യയാൽ സ്വതന്ത്രരാകണം
വിശ്വപൗരരായി മാറണം
ഗ്രഹപ്രസാദമായ് വിളങ്ങണം
ഗുരുപ്രകാശമേ നയിക്കണേ

കൂരിരുട്ടിൽ നിന്നു ഞങ്ങളെ
സൂര്യവീഥിയിൽ തെളിക്കണം
സ്നേഹദീപ്തിയായ് വിളങ്ങണം
നീതിവൈജയന്തി പാറണം

ശാസ്ത്രവ്യാപ്തിയെന്നുമേകണം
ജാതിഭേദമാകെ മാറണം
ബോധരശ്മിയിൽ തിളങ്ങുവാൻ
ജ്ഞാനകേന്ദ്രമേ ജ്വലിക്കണേ

കുരീപ്പുഴ ശ്രീകുമാർ

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

Regional Centres

Kozhikode
Govt. Arts and Science College
Meenchantha, Kozhikode,
Kerala, Pin: 673002
Ph: 04952920228
email: rckdirector@sgou.ac.in

Thalassery
Govt. Brennen College
Dharmadam, Thalassery,
Kannur, Pin: 670106
Ph: 04902990494
email: rctdirector@sgou.ac.in

Tripunithura
Govt. College
Tripunithura, Ernakulam,
Kerala, Pin: 682301
Ph: 04842927436
email: rcedirector@sgou.ac.in

Pattambi
Sree Neelakanta Govt. Sanskrit College
Pattambi, Palakkad,
Kerala, Pin: 679303
Ph: 04662912009
email: rcpdirector@sgou.ac.in

SAY NO TO RAGGING

BEFORE YOU EVEN THINK OF RAGGING



Humiliation

Blacklisting

Ruined Career

Download

ANTI RAGGING

App

Don't just stand and watch. Stop Ragging! Show Character

Visit UGC Website i.e. www.ugc.Ac.in & www antinaging.in to UGC Anti Ragging regulations.

Ministry of Education
Government of India

Are You Being Ragged ?

विश्वविद्यालय अनुदान आयोग
University Grants Commission
quality higher education for all

DON'T LET IT BE TOO LATE

SAY NO TO DRUGS

LOVE YOURSELF AND ALWAYS BE HEALTHY



SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SGOU

MACHINE LEARNING LAB

COURSE CODE: M25CA04PC

SGOU

Sreenarayanaguru Open University
Kollam, Kerala Pin- 691601, email: info@sgou.ac.in, www.sgou.ac.in Ph: +91 474 2966841

ISBN 978-81-181574-4-1