RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Recognized by GOK, Approved by AICTE, New Delhi
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098
Ph:(080)28611880,28611881 URL: www.rnsit.ac.in

DEPARTMENT OF ISE

MACHINE LEARNING LAB MANUAL

(BCSL606)

(As per Visvesvaraya Technological University Course type- PCCL)

Compiled by

DEPARTMENT OF ISE
R N S Institute of Technology
Bengaluru-98

Name: _____

USN: _____

RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Recognized by GOK, Approved by AICTE, New Delhi
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098 Ph:(080)28611880,28611881
URL: www.rnsit.ac.in

DEPARTMENT OF ISE

VISION OF THE DEPARTMENT

Building Information Technology Professionals by Imparting Quality Education and Inculcating Key Competencies.

MISSION OF THE DEPARTMENT

- Provide strong fundamentals through learner centric approach.
- Instil technical, interpersonal, interdisciplinary skills and logical thinking for holistic development.
- Train to excel in higher education, research, and innovation with global perspective.
- Develop leadership and entrepreneurship qualities with societal responsibilities.

Disclaimer

Trademark

Edition: 2023- 24

Document Owner

The primary contact for questions regarding this document is:

Author(s):
1. Dr. Bhagyashree Ambore
2. Aishwarya G
3. Nivedita G Y

Department: **ISE**

Contact email ids : bhagyashree@rnsit.ac.in
aishwarya.g@rnsit.ac.in
nivedita.gy@rnsit.ac.in

COURSE OUTCOMES

Course Outcomes: At the end of this course, students are able to:
CO1- Illustrate the principles of multivariate data and apply dimensionality reduction techniques.
CO2- Demonstrate similarity-based learning methods and perform regression analysis.
CO3- Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning.
CO4- Implement the clustering algorithms to share computing resources.

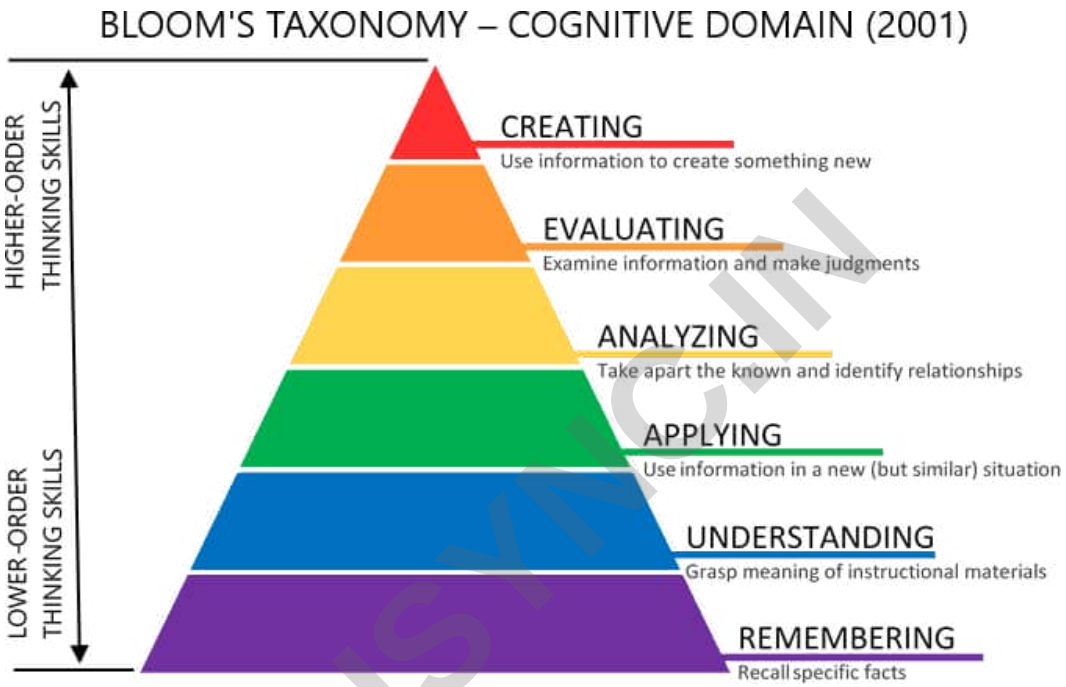
COs and POs Mapping of lab Component

COURSE OUTCOMES	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	3	2	2	1	2	-	-	-	-	-	-	3	3	2	-	-
CO2	3	3	3	1	2	-	-	-	-	-	-	3	3	3	-	-
CO3	3	3	3	1	2	-	-	-	-	-	-	3	3	3	-	-
CO4	3	3	3	3	2	-	-	-	-	-	-	3	3	3	-	-

Mapping of 'Graduate Attributes' (GAs) and 'Program Outcomes' (POs)

Graduate Attributes (GAs) (As per Washington Accord Accreditation)	Program Outcomes (POs) (As per NBA New Delhi)
Engineering Knowledge	Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems
Problem Analysis	Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
Design/Development of solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate considerations for the public health and safety and the cultural, societal and environmental consideration.
Conduct Investigation of complex problems	Use research – based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
Modern Tool Usage	Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
The engineer and society	Apply reasoning informed by the contextual knowledge to assess society, health, safety, legal and cultural issues and the consequential responsibilities relevant to the professional engineering practice.
Environment and sustainability	Understand the impact of the professional engineering solutions in societal and environmental context and demonstrate the knowledge of and need for sustainable development.
Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
Individual and team work	Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.
Communication	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
Project management & finance	Demonstrate knowledge and understanding of the engineering and management principles and apply these to ones won work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
Life Long Learning	Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

REVISED BLOOMS TAXONOMY (RBT)



PROGRAM LIST

INTRODUCTION TO MACHINE LEARNING

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that focuses on enabling machines to learn from data and make decisions or predictions without explicit programming. In simpler terms, ML allows systems to automatically improve their performance through experience. Some of the core concepts of ML are :

1. **Data:** Machine learning models learn patterns from data. This data can come in various forms: numbers, text, images, videos, etc.
2. **Algorithms:** ML uses algorithms to process data, identify patterns, and make predictions. Some popular algorithms include decision trees, neural networks, and support vector machines.
3. **Training:** During training, a machine learning model learns from a labeled dataset (for supervised learning) or an unlabeled dataset (for unsupervised learning). In supervised learning, the model is provided with both input data and the correct output (label) and learns to map inputs to outputs. In unsupervised learning, the model tries to find patterns or groupings in the data on its own.
4. **Features:** Features are the input variables or attributes used by a machine learning model to make predictions. For example, in a house price prediction model, features could include the size of the house, number of bedrooms, and location.
5. **Model Evaluation:** Once trained, the model is tested on new, unseen data to evaluate its performance. Metrics such as accuracy, precision, recall, and F1 score help assess the model's effectiveness.

Types of Machine Learning:

1. **Supervised Learning:** The model is trained on labeled data (data with known outcomes). It learns to predict the output based on input-output pairs. Common algorithms include linear regression, logistic regression, and neural networks.

2. **Unsupervised Learning:** In this case, the data doesn't come with labels. The model tries to find hidden patterns or groupings. Common techniques include clustering (e.g., K-means) and dimensionality reduction (e.g., PCA).

3. **Reinforcement Learning:** The model learns by interacting with its environment and receiving feedback in the form of rewards or penalties. It's often used in scenarios like robotics, gaming, and autonomous vehicles.

4. **Semi-supervised Learning:** This method is a combination of supervised and unsupervised learning. It uses a small amount of labeled data and a large amount of unlabeled data to build a model.

5. **Self-supervised Learning:** A method where the model generates its own labels from the input data.

Applications of ML:

- **Image Recognition:** Used in facial recognition systems, object detection, and medical imaging.
- **Natural Language Processing (NLP):** ML models are used in speech recognition, translation, and chatbots. **Recommendation Systems:** ML is used in services like Netflix and Amazon to recommend movies or products based on past behavior.
- **Autonomous Vehicles:** ML is essential for self-driving cars to navigate and make decisions in realtime.
- **Predictive Analytics:** In business, finance, and healthcare, ML helps in predicting trends, customer behavior, and disease outbreaks.

Challenges in Machine Learning:

- **Data Quality:** Poor quality data can lead to inaccurate predictions.
- **Overfitting:** A model that is too complex may perform well on training data but fail to generalize to new data.
- **Interpretability:** Some ML models, especially deep learning models, are often referred to as "black boxes" because it's difficult to interpret how they make decisions.
- **Ethical Concerns:** The use of ML can lead to biases in decision-making, privacy issues, and fairness concerns.

Common Algorithms and Techniques

- **Linear Regression:** A simple algorithm used for predicting continuous values (e.g., predicting house prices based on features like size and location).
- **Logistic Regression:** A classification algorithm used to predict binary outcomes (e.g., determining whether an email is spam or not).
- **Decision Trees:** A tree-like model used for both classification and regression tasks. It splits the data into branches based on decision rules.
- **K-Means Clustering:** An unsupervised algorithm that groups similar data points into clusters.
- **Neural Networks:** Inspired by the human brain, these models are especially good for complex tasks like image recognition, speech recognition, and language processing.

Program 1:

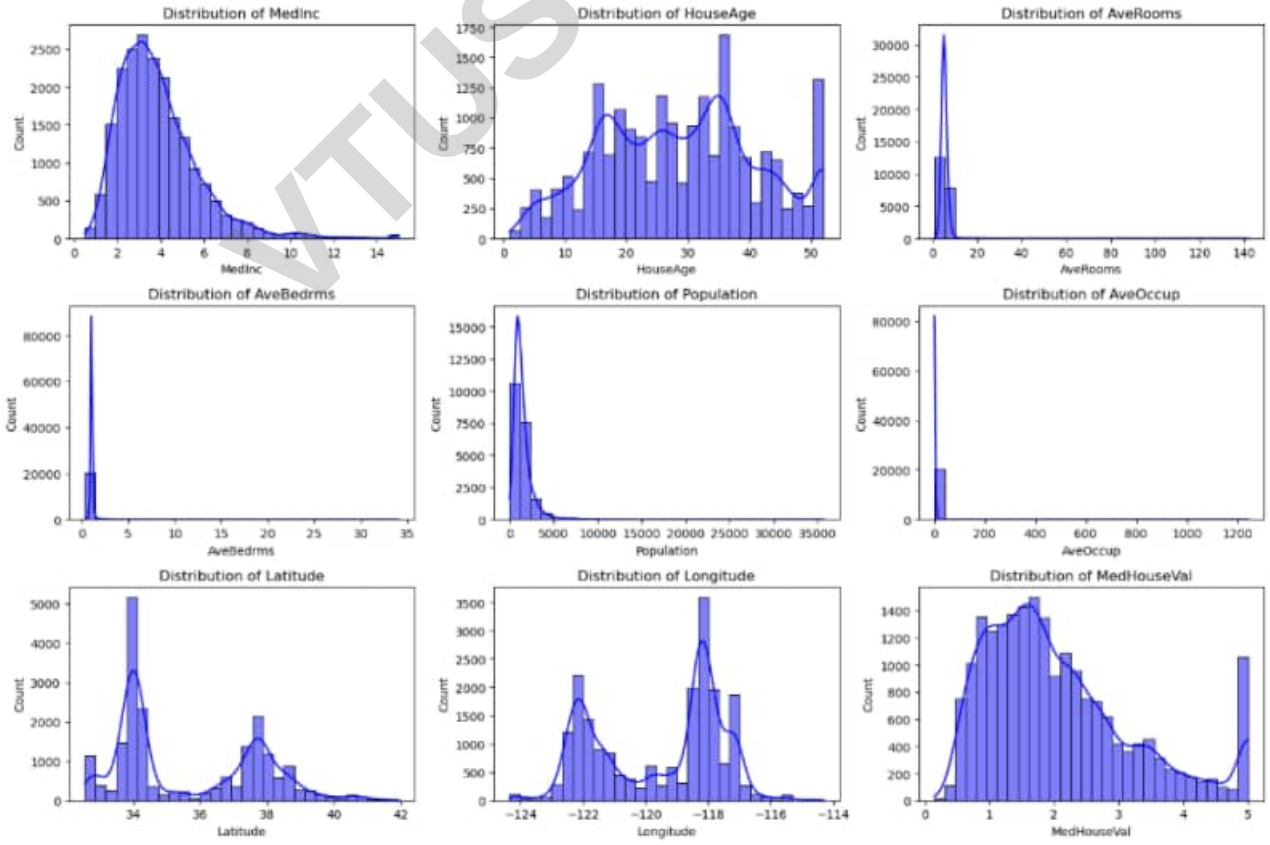
AIM: Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.

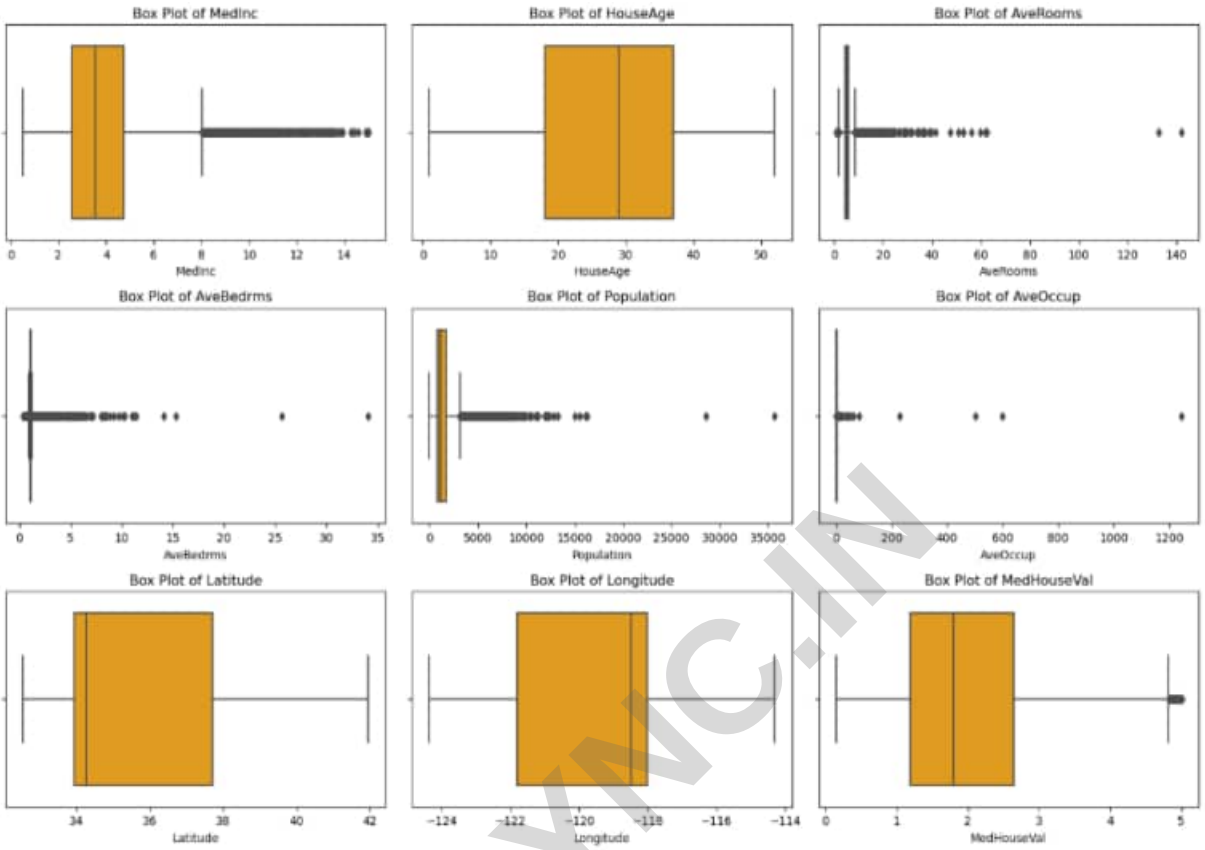
Source code:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

# Step 1: Load the California Housing dataset
data = fetch_california_housing(as_frame=True)
housing_df = data.frame

# Step 2: Create histograms for numerical features
numerical_features = housing_df.select_dtypes(include=[np.number]).columns
# Plot histograms
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.histplot(housing_df[feature], kde=True, bins=30, color='blue')
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()

# Step 3: Generate box plots for numerical features
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.boxplot(x=housing_df[feature], color='orange')
    plt.title(f'Box Plot of {feature}')
plt.tight_layout()
```

```
plt.show()
# Step 4: Identify outliers using the IQR method
print("Outliers Detection:")
outliers_summary = {}
for feature in numerical_features:
    Q1 = housing_df[feature].quantile(0.25)
    Q3 = housing_df[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = housing_df[(housing_df[feature] < lower_bound) | (housing_df[feature] > upper_bound)]
    outliers_summary[feature] = len(outliers)
    print(f"{feature}: {len(outliers)} outliers")
#Print a summary of the dataset
print("\nDataset Summary:")
print(housing_df.describe())
```

Output:






```
Outliers Detection:
MedInc: 681 outliers
HouseAge: 0 outliers
AveRooms: 511 outliers
AveBedrms: 1424 outliers
Population: 1196 outliers
AveOccup: 711 outliers
Latitude: 0 outliers
Longitude: 0 outliers
MedHouseVal: 1071 outliers

Dataset Summary:
             MedInc      HouseAge      AveRooms     AveBedrms    Population  \
count  20640.000000  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.870671     28.639486      5.429000      1.096675   1425.476744
std        1.899822     12.585558      2.474173      0.473911   1132.462122
min        0.499900      1.000000      0.846154      0.333333      3.000000
25%        2.563400     18.000000      4.440716      1.006079    787.000000
50%        3.534800     29.000000      5.229129      1.048780   1166.000000
75%        4.743250     37.000000      6.052381      1.099526   1725.000000
max       15.000100     52.000000    141.909091     34.066667  35682.000000

            AveOccup      Latitude     Longitude   MedHouseVal
count  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.070655     35.631861   -119.569704      2.068558
std       10.386050      2.135952      2.003532      1.153956
min        0.692308     32.540000   -124.350000      0.149990
25%        2.429741     33.930000   -121.800000      1.196000
50%        2.818116     34.260000   -118.490000      1.797000
75%        3.282261     37.710000   -118.010000      2.647250
max     1243.333333     41.950000   -114.310000      5.000010
```

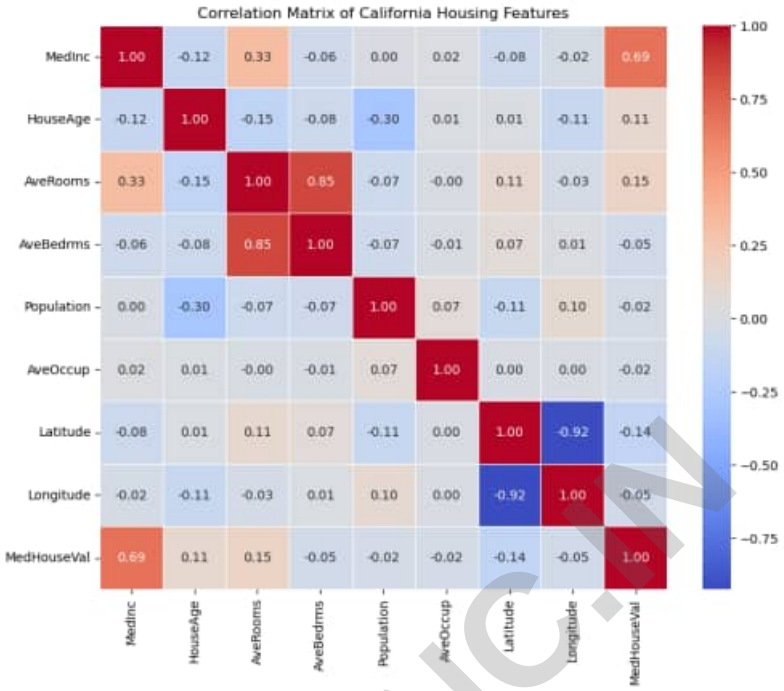
Program 2:

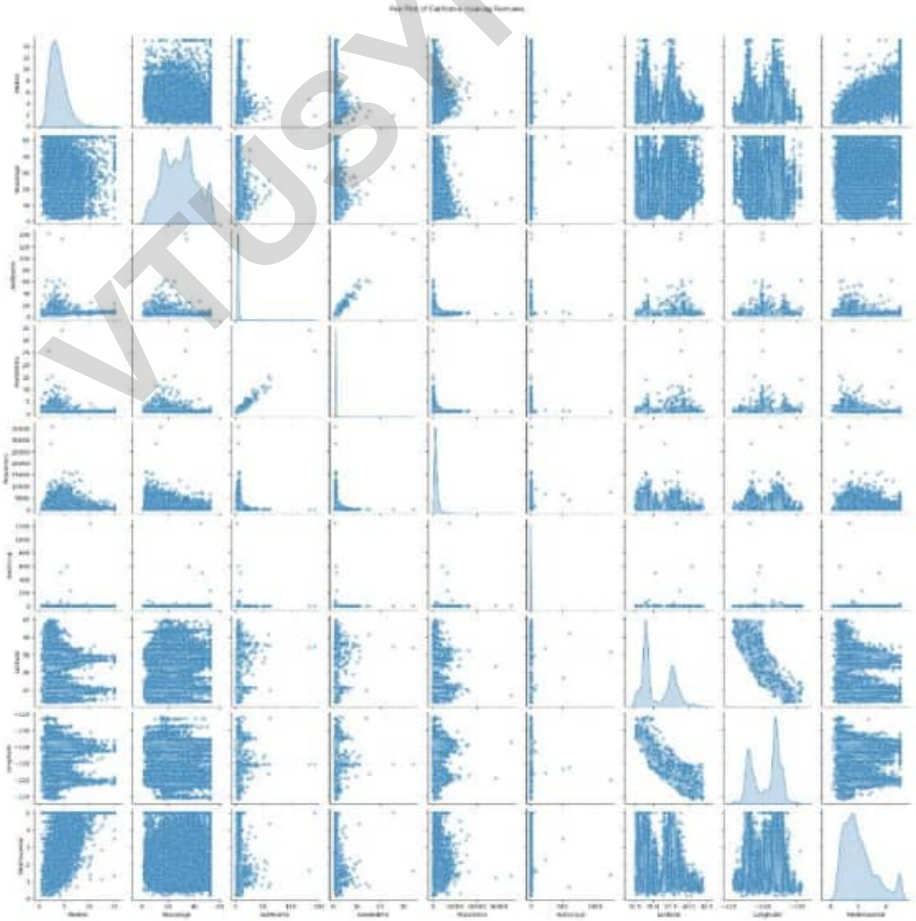
AIM: Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset.

Source Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

# Step 1: Load the California Housing Dataset
california_data = fetch_california_housing(as_frame=True)
data = california_data.frame

# Step 2: Compute the correlation matrix
correlation_matrix = data.corr()

# Step 3: Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix of California Housing Features')
plt.show()

# Step 4: Create a pair plot to visualize pairwise relationships
sns.pairplot(data, diag_kind='kde', plot_kws={'alpha': 0.5})
plt.suptitle('Pair Plot of California Housing Features', y=1.02)
plt.show()
```

Output:





Program 3:

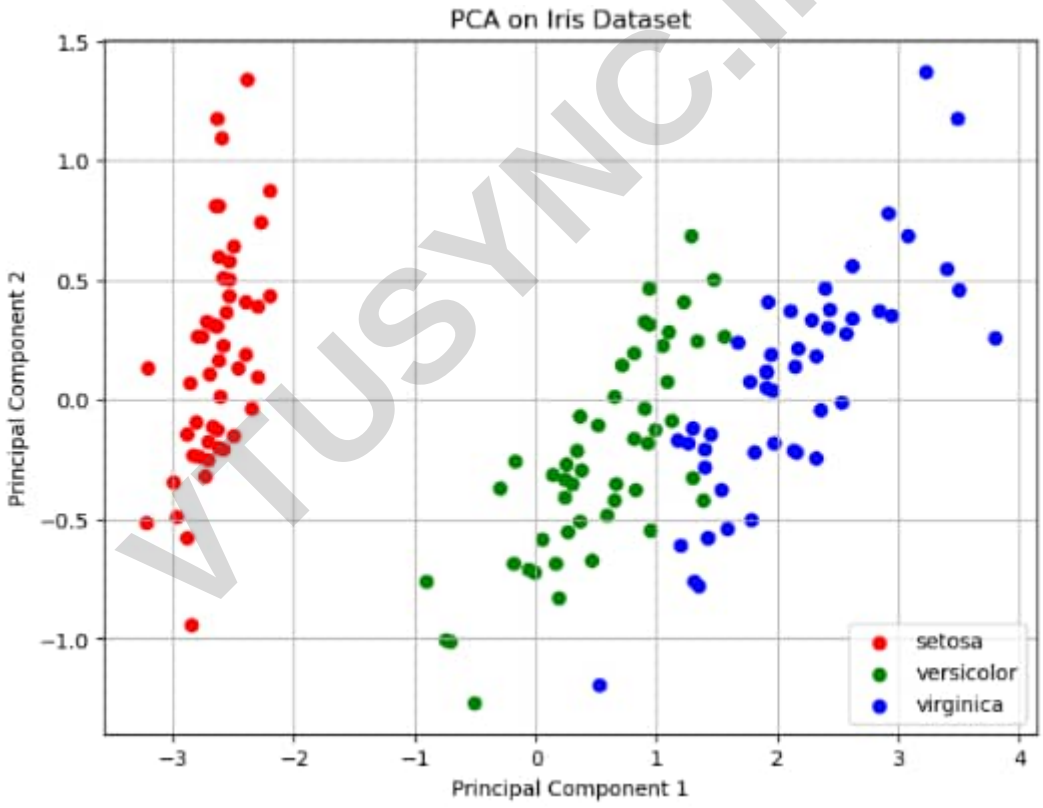
AIM: Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.

Source Code:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
data = iris.data
labels = iris.target
label_names = iris.target_names

# Convert to a DataFrame for better visualization
iris_df = pd.DataFrame(data, columns=iris.feature_names)

# Perform PCA to reduce dimensionality to 2
pca = PCA(n_components=2)
data_reduced = pca.fit_transform(data)

# Create a DataFrame for the reduced data
reduced_df = pd.DataFrame(data_reduced, columns=['Principal Component 1', 'Principal Component 2'])
reduced_df['Label'] = labels

# Plot the reduced data
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b']
for i, label in enumerate(np.unique(labels)):
    plt.scatter(
        reduced_df[reduced_df['Label'] == label]['Principal Component 1'],
        reduced_df[reduced_df['Label'] == label]['Principal Component 2'],
```

```
        label=label_names[label],
        color=colors[i]
    )
plt.title('PCA on Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid()
plt.show()
```

Output:



Program 4:

AIM: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.

Source Code:

```
import pandas as pd
def find_s_algorithm(file_path):
    data = pd.read_csv('C:\\Users\\user\\Downloads\\training_data.csv')
    print("Training data:")
    print(data)
    attributes = data.columns[:-1]
    class_label = data.columns[-1]
    hypothesis = ['?' for _ in attributes]
    for index, row in data.iterrows():
        if row[class_label] == 'Yes':
            for i, value in enumerate(row[attributes]):
                if hypothesis[i] == '?' or hypothesis[i] == value:
                    hypothesis[i] = value
                else:
                    hypothesis[i] = '?'
    return hypothesis
file_path = 'training_data.csv'
hypothesis = find_s_algorithm(file_path)
print("\nThe final hypothesis is:", hypothesis)
```

Output:

```
Training data:
    Outlook Temperature Humidity  Windy PlayTennis
0     Sunny         Hot     High  False         No
1     Sunny         Hot     High   True         No
2  Overcast         Hot     High  False        Yes
3      Rain        Cold     High  False        Yes
4      Rain        Cold     High   True         No
5  Overcast         Hot     High   True        Yes
6     Sunny         Hot     High  False         No

The final hypothesis is: ['Overcast', 'Hot', 'High', '?']
```

Program 5:

AIM: Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of x in the range of [0,1]. Perform the following based on dataset generated.

1. Label the first 50 points {x1,……,x50} as follows: if (xi ≤ 0.5), then xi ∈ Class1, else xi ∈ Class1

2. Classify the remaining points, x51,……,x100 using KNN. Perform this for k=1,2,3,4,5,20,30

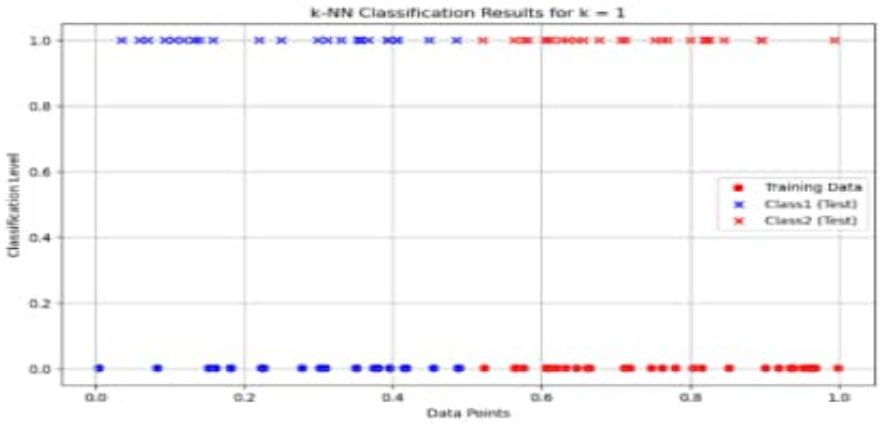
Source Code:

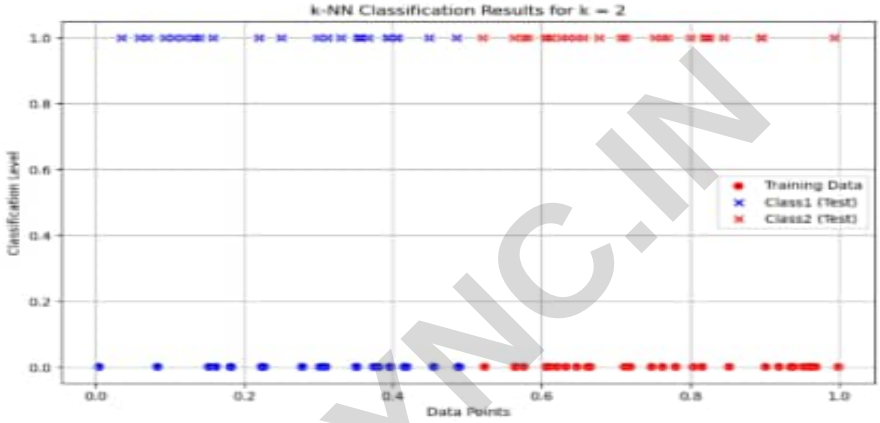
```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
data = np.random.rand(100)
labels = ["Class1" if x <= 0.5 else "Class2" for x in data[:50]]
def euclidean_distance(x1, x2):
    return abs(x1 - x2)
def knn_classifier(train_data, train_labels, test_point, k):
    distances = [(euclidean_distance(test_point, train_data[i]), train_labels[i]) for i in range(len(train_data))]
    distances.sort(key=lambda x: x[0])
    k_nearest_neighbors = distances[:k]
    k_nearest_labels = [label for _, label in k_nearest_neighbors]
    return Counter(k_nearest_labels).most_common(1)[0][0]
train_data = data[:50]
train_labels = labels
test_data = data[50:]
k_values = [1, 2, 3, 4, 5, 20, 30]
print("--- k-Nearest Neighbors Classification ---")
print("Training dataset: First 50 points labeled based on the rule (x <= 0.5 -> Class1, x > 0.5 -> Class2)")
print("Testing dataset: Remaining 50 points to be classified\n")
results = {}
for k in k_values:
    print(f"Results for k = {k}:")
    classified_labels = [knn_classifier(train_data, train_labels, test_point, k) for test_point in test_data]
    results[k] = classified_labels
```

```
        for i, label in enumerate(classified_labels, start=51):
            print(f"Point x{i} (value: {test_data[i - 51]:.4f}) is classified as {label}")
    print("\n")
print("Classification complete.\n")
for k in k_values:
    classified_labels = results[k]
    class1_points = [test_data[i] for i in range(len(test_data)) if classified_labels[i] == "Class1"]
    class2_points = [test_data[i] for i in range(len(test_data)) if classified_labels[i] == "Class2"]
    plt.figure(figsize=(10, 6))
    plt.scatter(train_data, [0] * len(train_data), c=["blue" if label == "Class1" else "red" for label in train_labels],
                label="Training Data", marker="o")
    plt.scatter(class1_points, [1] * len(class1_points), c="blue", label="Class1 (Test)", marker="x")
    plt.scatter(class2_points, [1] * len(class2_points), c="red", label="Class2 (Test)", marker="x")
    plt.title(f"k-NN Classification Results for k = {k}")
    plt.xlabel("Data Points")
    plt.ylabel("Classification Level")
    plt.legend()
    plt.grid(True)
    plt.show()
```

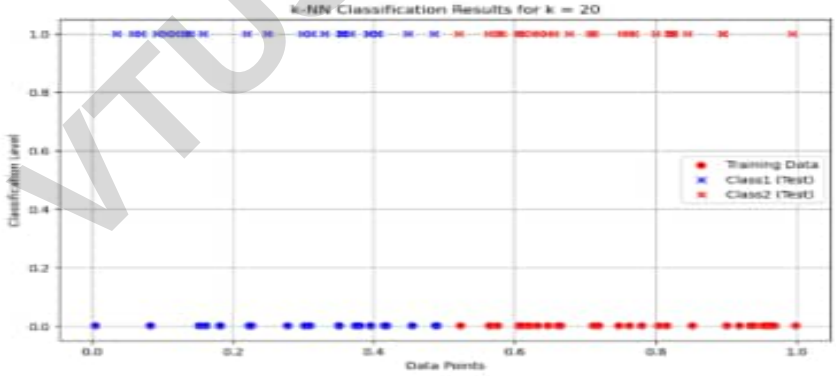
Output:

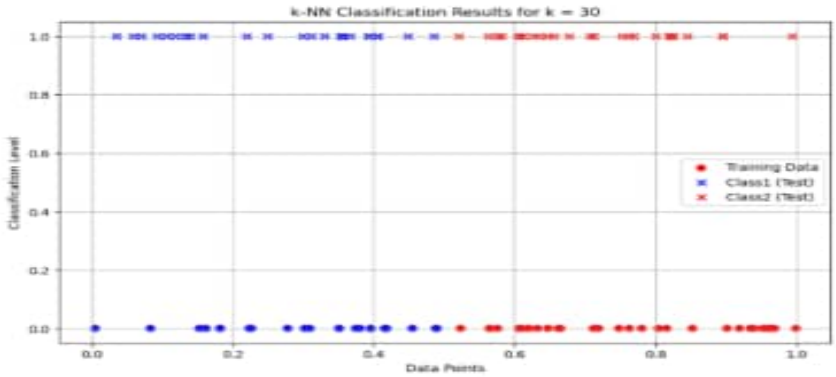
```
--- k-Nearest Neighbors Classification ---
Training dataset: First 50 points labeled based on the rule (x <= 0.5 -> Class1, x > 0.5 -> Class2)
Testing dataset: Remaining 50 points to be classified

Results for k = 1:
Point x51 (value: 0.1060) is classified as Class1
Point x52 (value: 0.0353) is classified as Class1
Point x53 (value: 0.2499) is classified as Class1
Point x54 (value: 0.7516) is classified as Class2
Point x55 (value: 0.3306) is classified as Class1
Point x56 (value: 0.3663) is classified as Class1
Point x57 (value: 0.3923) is classified as Class1
Point x58 (value: 0.3521) is classified as Class1
Point x59 (value: 0.5638) is classified as Class2
Point x60 (value: 0.1184) is classified as Class1
Point x61 (value: 0.6035) is classified as Class2
Point x62 (value: 0.0937) is classified as Class1
Point x63 (value: 0.8000) is classified as Class2
Point x64 (value: 0.8451) is classified as Class2
Point x65 (value: 0.7070) is classified as Class2
Point x66 (value: 0.3940) is classified as Class1
Point x67 (value: 0.7615) is classified as Class2
Point x68 (value: 0.1323) is classified as Class1
Point x69 (value: 0.1401) is classified as Class1
Point x70 (value: 0.6064) is classified as Class2
Point x71 (value: 0.6272) is classified as Class2
Point x72 (value: 0.8260) is classified as Class2
Point x73 (value: 0.6556) is classified as Class2
Point x74 (value: 0.3558) is classified as Class1
Point x75 (value: 0.5809) is classified as Class2
Point x76 (value: 0.4860) is classified as Class1
Point x77 (value: 0.1584) is classified as Class1
Point x78 (value: 0.6354) is classified as Class2
Point x79 (value: 0.2991) is classified as Class1
Point x80 (value: 0.0707) is classified as Class1
Point x81 (value: 0.8208) is classified as Class2
Point x82 (value: 0.9941) is classified as Class2
Point x83 (value: 0.8944) is classified as Class2
Point x84 (value: 0.6120) is classified as Class2
Point x85 (value: 0.8190) is classified as Class2
Point x86 (value: 0.2201) is classified as Class1
Point x87 (value: 0.1331) is classified as Class1
Point x88 (value: 0.4481) is classified as Class1
Point x89 (value: 0.0591) is classified as Class1
Point x90 (value: 0.4043) is classified as Class1
Point x91 (value: 0.6775) is classified as Class2
Point x92 (value: 0.7685) is classified as Class2
Point x93 (value: 0.8957) is classified as Class2
Point x94 (value: 0.5207) is classified as Class2
Point x95 (value: 0.7131) is classified as Class2
Point x96 (value: 0.5756) is classified as Class2
Point x97 (value: 0.6460) is classified as Class2
Point x98 (value: 0.4063) is classified as Class1
Point x99 (value: 0.3120) is classified as Class1
Point x100 (value: 0.3575) is classified as Class1

.
.
.
.
.
.

Classification complete.
```


k-NN Classification Results for k = 1
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)


k-NN Classification Results for k = 2
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)

•
•
•


k-NN Classification Results for k = 20
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)


k-NN Classification Results for k = 30
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)

Program 6:

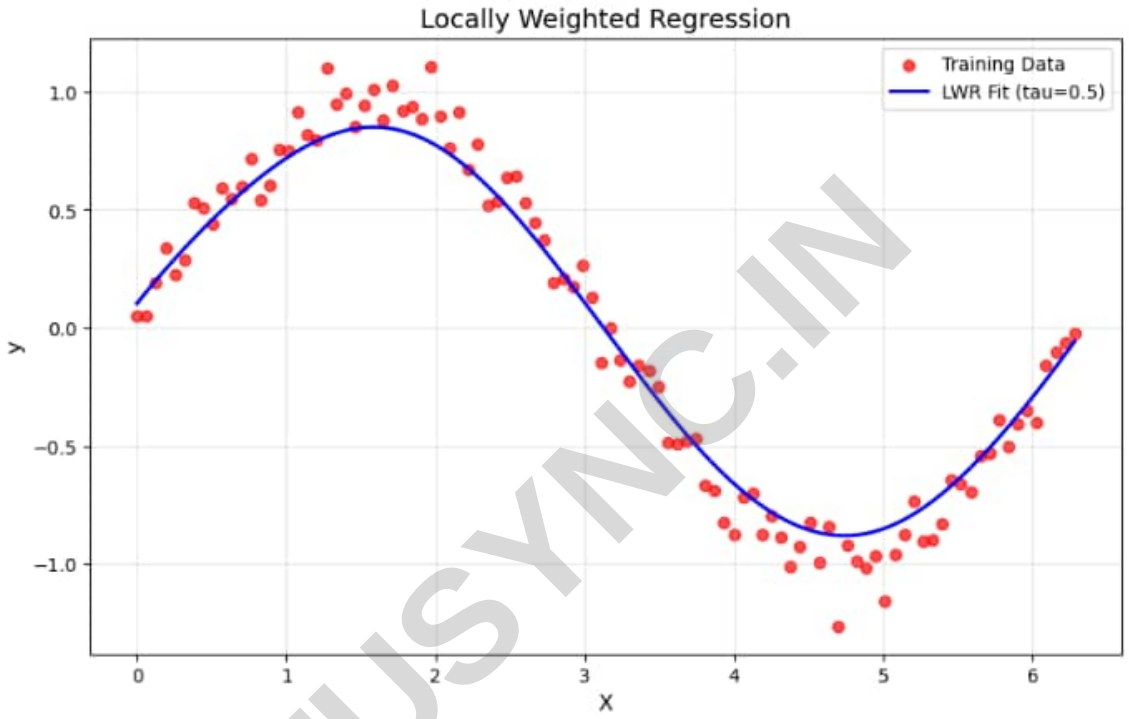
AIM: Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Source Code:

```
import numpy as np
import matplotlib.pyplot as plt
def gaussian_kernel(x, xi, tau):
    return np.exp(-np.sum((x - xi) ** 2) / (2 * tau ** 2))
def locally_weighted_regression(x, X, y, tau):
    m = X.shape[0]
    weights = np.array([gaussian_kernel(x, X[i], tau) for i in range(m)])
    W = np.diag(weights)
    X_transpose_W = X.T @ W
    theta = np.linalg.inv(X_transpose_W @ X) @ X_transpose_W @ y
    return x @ theta
np.random.seed(42)
X = np.linspace(0, 2 * np.pi, 100)
y = np.sin(X) + 0.1 * np.random.randn(100)
X_bias = np.c_[np.ones(X.shape), X]
x_test = np.linspace(0, 2 * np.pi, 200)
x_test_bias = np.c_[np.ones(x_test.shape), x_test]
tau = 0.5
y_pred = np.array([locally_weighted_regression(xi, X_bias, y, tau) for xi in x_test_bias])
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='red', label='Training Data', alpha=0.7)
plt.plot(x_test, y_pred, color='blue', label=f'LWR Fit (tau={tau})', linewidth=2)
plt.xlabel('X', fontsize=12)
plt.ylabel('y', fontsize=12)
```

```
plt.title('Locally Weighted Regression', fontsize=14)
plt.legend(fontsize=10)
plt.grid(alpha=0.3)
plt.show()
```

Output:



Program 7:

AIM: Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.

Source Code:

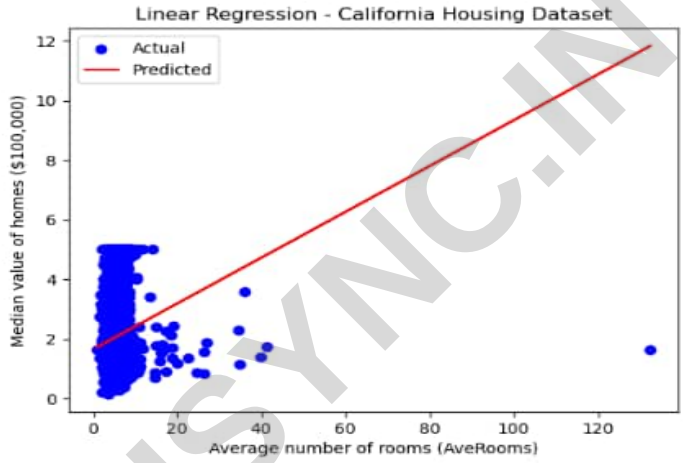
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score
def linear_regression_california():
    housing = fetch_california_housing(as_frame=True)
    X = housing.data[["AveRooms"]]
    y = housing.target
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

plt.scatter(X_test, y_test, color="blue", label="Actual")

    plt.plot(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Average number of rooms (AveRooms)")
```

```
    plt.ylabel("Median value of homes ($100,000)")
    plt.title("Linear Regression - California Housing Dataset")
    plt.legend()
    plt.show()
    print("Linear Regression - California Housing Dataset")
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))
def polynomial_regression_auto_mpg():
    url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
    column_names = ["mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model_year",
"origin"]
    data = pd.read_csv(url, sep='\s+', names=column_names, na_values="?")
    data = data.dropna()
    X = data["displacement"].values.reshape(-1, 1)
    y = data["mpg"].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    poly_model = make_pipeline(PolynomialFeatures(degree=2), StandardScaler(), LinearRegression())
    poly_model.fit(X_train, y_train)
    y_pred = poly_model.predict(X_test)
plt.scatter(X_test, y_test, color="blue", label="Actual")
    plt.scatter(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Displacement")
    plt.ylabel("Miles per gallon (mpg)")
    plt.title("Polynomial Regression - Auto MPG Dataset")
    plt.legend()
    plt.show
print("Polynomial Regression - Auto MPG Dataset)
```

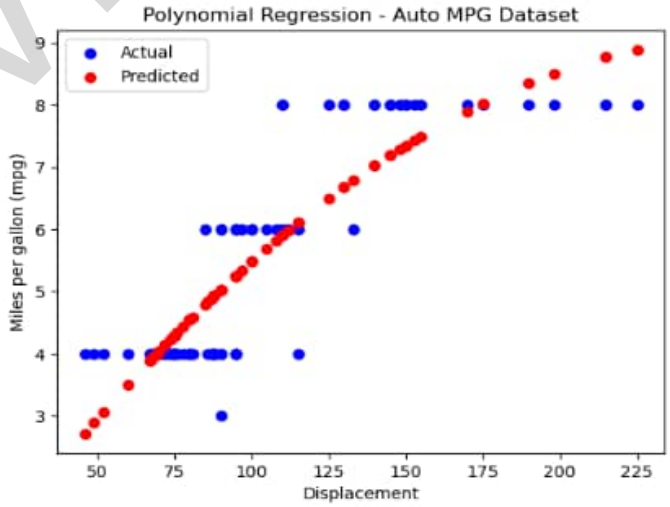
```
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))
if __name__ == "__main__":
    print("Demonstrating Linear Regression and Polynomial Regression\n")
    linear_regression_california()
    polynomial_regression_auto_mpg()
```

Output:

```
Demonstrating Linear Regression and Polynomial Regression
```




```
Linear Regression - California Housing Dataset
Mean Squared Error: 1.2923314440807299
R^2 Score: 0.013795337532284901
```




```
Polynomial Regression - Auto MPG Dataset
Mean Squared Error: 0.7431490557205862
R^2 Score: 0.7505650609469626
```

Program 8:

AIM: Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample.

Source Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
# Load the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# 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)

# Train a Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

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

# Calculate and print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Predict the class for a new sample (first sample in the test set)
new_sample = X_test[0].reshape(1, -1)  # Reshape to match the input format
```

prediction = clf.predict(new_sample)

 # Map the prediction to the class name

prediction_class = "Benign" if prediction == 1 else "Malignant"

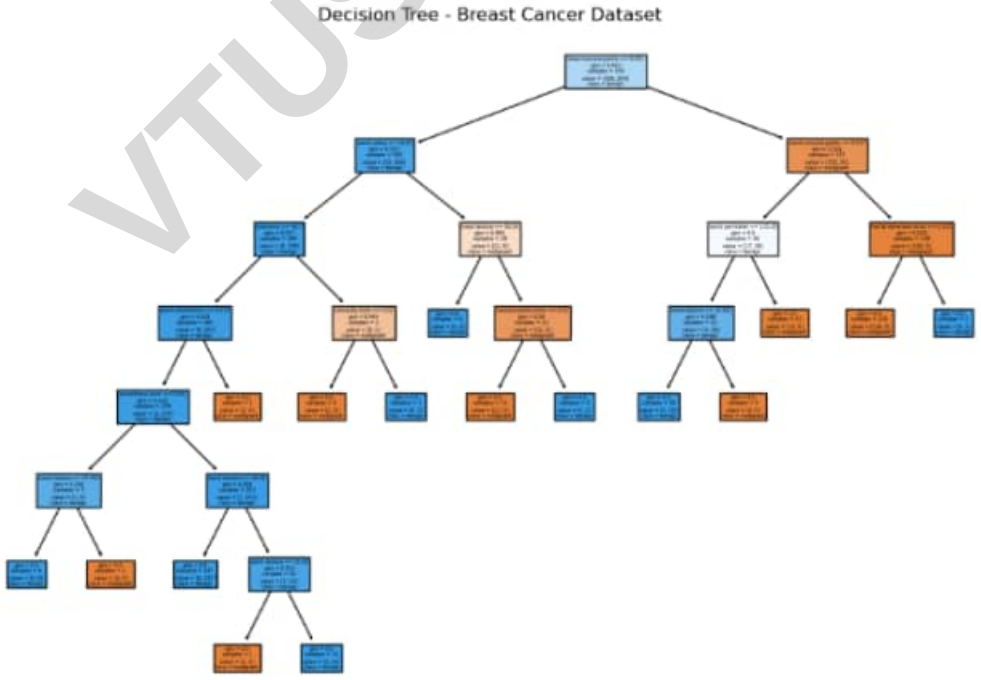
print(f"Predicted Class for the new sample: {prediction_class}")

 # Plot the decision tree

```
plt.figure(figsize=(12, 8))
tree.plot_tree(
    clf,
    filled=True,
    feature_names=data.feature_names.tolist(),  # Convert NumPy array to list
    class_names=data.target_names.tolist()     # Convert NumPy array to list
)
plt.title("Decision Tree - Breast Cancer Dataset")
plt.show()
```

Output:

```
Model Accuracy: 94.74%
Predicted Class for the new sample: Benign
```



Program 9:

AIM: Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.

Source Code:

```
import numpy as np
from sklearn.datasets import fetch_olivetti_faces
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt

data = fetch_olivetti_faces(shuffle=True, random_state=42)
X = data.data
y = data.target

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

gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=1))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
cross_val_accuracy = cross_val_score(gnb, X, y, cv=5, scoring='accuracy')
print(f'\nCross-validation accuracy: {cross_val_accuracy.mean() * 100:.2f}%')


fig, axes = plt.subplots(3, 5, figsize=(12, 8))
for ax, image, label, prediction in zip(axes.ravel(), X_test, y_test, y_pred):
    ax.imshow(image.reshape(64, 64), cmap=plt.cm.gray)
    ax.set_title(f"True: {label}, Pred: {prediction}")
    ax.axis('off')
plt.show()
```

Output:

```
downloading Olivetti faces from https://ndownloader.figshare.com/files/5976027 to C:\Users\user\scikit_learn_data
Accuracy: 80.83%

Classification Report:
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         2
           1       1.00      1.00      1.00         2
           2       0.33      0.67      0.44         3
           3       1.00      0.00      0.00         5
           4       1.00      0.50      0.67         4
           5       1.00      1.00      1.00         2
           7       1.00      0.75      0.86         4
           8       1.00      0.67      0.80         3
           9       1.00      0.75      0.86         4
          10       1.00      1.00      1.00         3
          11       1.00      1.00      1.00         1
          12       0.40      1.00      0.57         4
          13       1.00      0.80      0.89         5
          14       1.00      0.40      0.57         5
          15       0.67      1.00      0.80         2
          16       1.00      0.67      0.80         3
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         3
          19       0.67      1.00      0.80         2
          20       1.00      1.00      1.00         3
          21       1.00      0.67      0.80         3
          22       1.00      0.60      0.75         5
          23       1.00      0.75      0.86         4
          24       1.00      1.00      1.00         3
          25       1.00      0.75      0.86         4
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         5
          28       0.50      1.00      0.67         2
          29       1.00      1.00      1.00         2
          30       1.00      1.00      1.00         2
          31       1.00      0.75      0.86         4
          32       1.00      1.00      1.00         2
          34       0.25      1.00      0.40         1
          35       1.00      1.00      1.00         5
          36       1.00      1.00      1.00         3
          37       1.00      1.00      1.00         1
          38       1.00      0.75      0.86         4
          39       0.50      1.00      0.67         5

    accuracy                           0.81       120
   macro avg       0.89      0.85      0.83       120
weighted avg       0.91      0.81      0.81       120


Confusion Matrix:
[[2 0 0 ... 0 0 0]
 [0 2 0 ... 0 0 0]
 [0 0 2 ... 0 0 1]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 3 0]
 [0 0 0 ... 0 0 5]]

Cross-validation accuracy: 87.25%
```


True: 18, Pred: 18
True: 0, Pred: 0
True: 5, Pred: 5
True: 22, Pred: 22
True: 22, Pred: 22
True: 27, Pred: 27
True: 16, Pred: 16
True: 18, Pred: 18
True: 31, Pred: 31
True: 35, Pred: 35
True: 12, Pred: 12
True: 5, Pred: 5
True: 22, Pred: 22
True: 0, Pred: 0
True: 25, Pred: 25

Program 10:

AIM: Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

Source Code:

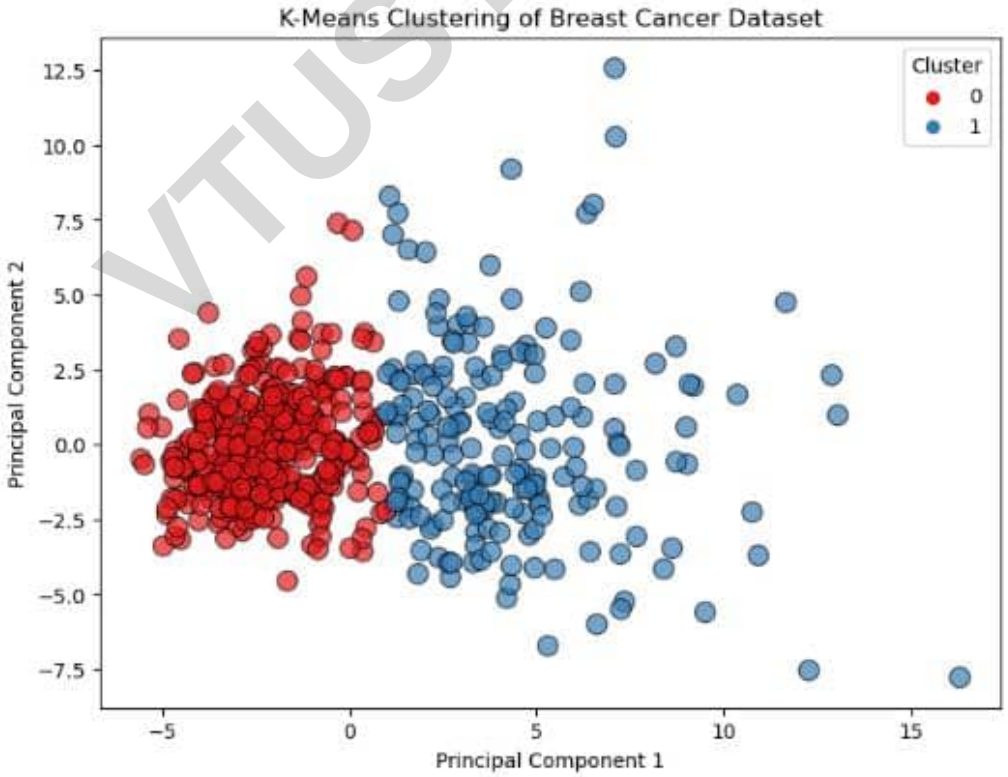
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, classification_report

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

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=2, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)

print("Confusion Matrix:")
print(confusion_matrix(y, y_kmeans))
print("\nClassification Report:")
print(classification_report(y, y_kmeans))
```

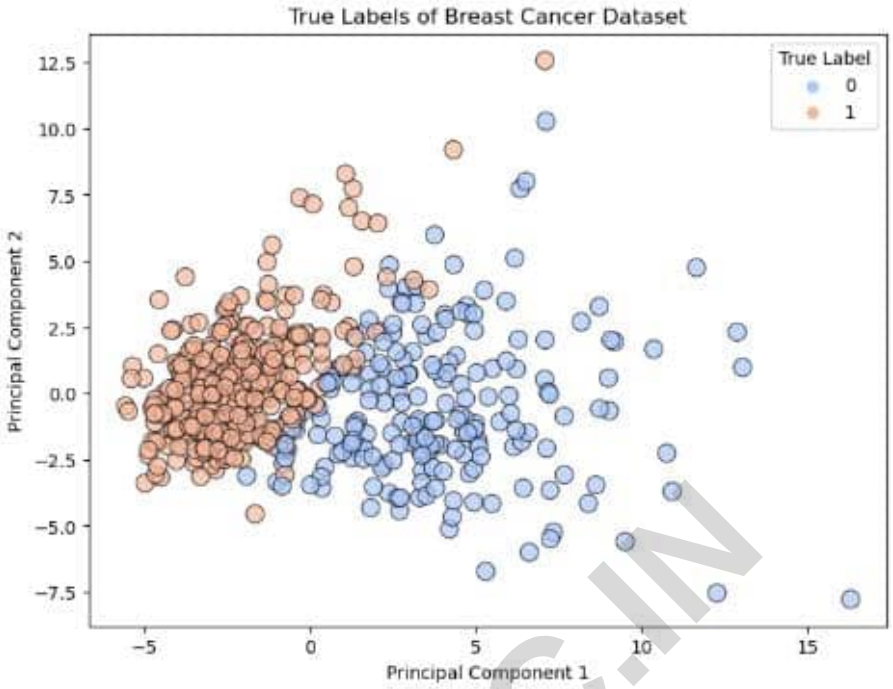
```
pca = PCA(n_components=2)
X_pca = pca. fit_transform(X_scaled)

df = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
df['Cluster'] = y_kmeans
df['True Label'] = y

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100, edgecolor='black', alpha=0.7)
plt.title('K-Means Clustering of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='True Label', palette='coolwarm', s=100, edgecolor='black',
alpha=0.7)
plt.title('True Labels of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="True Label")
plt.show()

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100, edgecolor='black', alpha=0.7)
centers = pca.transform(kmeans.cluster_centers_)
plt.scatter(centers[:, 0], centers[:, 1], s=200, c='red', marker='X', label='Centroids')
plt.title('K-Means Clustering with Centroids')
```

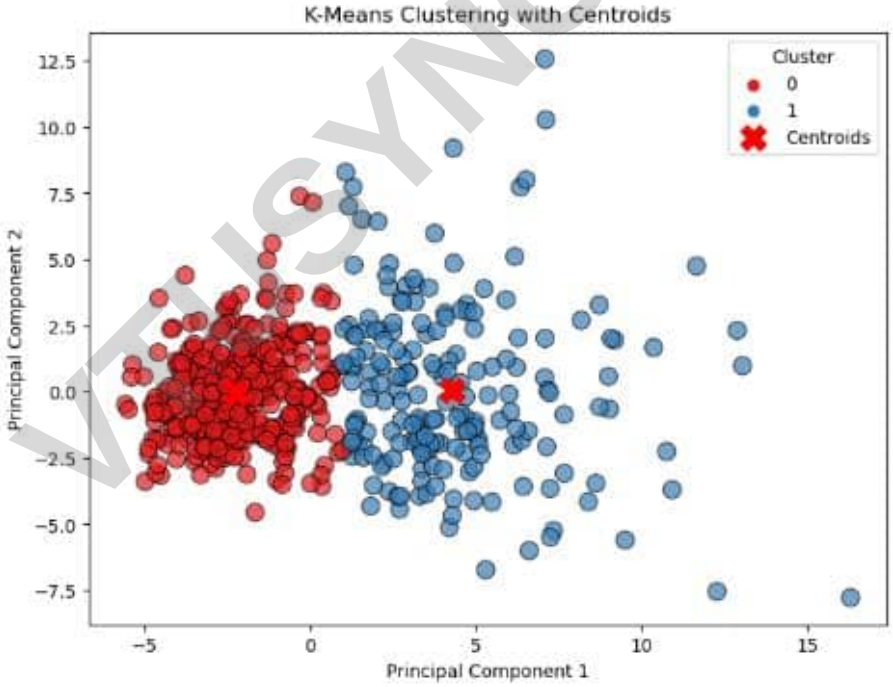
```
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()
```

Output:

```
Confusion Matrix:
[[ 36 176]
 [339  18]]

Classification Report:
              precision    recall  f1-score   support

           0       0.10      0.17      0.12       212
           1       0.09      0.05      0.07       357

    accuracy                           0.09       569
   macro avg       0.09      0.11      0.09       569
weighted avg       0.09      0.09      0.09       569
```




True Labels of Breast Cancer Dataset
True Label
0
1
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15


K-Means Clustering with Centroids
Cluster
0
1
Centroids
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

Viva Questions:

1. **What is machine learning, and how does it differ from traditional programming?**

   Machine learning enables computers to learn patterns from data and make decisions without explicit programming. Traditional programming follows explicit rules coded by humans.

2. **What are the different types of machine learning?**

   Supervised learning, unsupervised learning, and reinforcement learning.

3. **Define supervised and unsupervised learning with examples.**

   - Supervised learning: Data has labels (e.g., spam detection in emails).
   - Unsupervised learning: No labels; models find patterns (e.g., customer segmentation using clustering).

4. **What are the main steps in a machine learning workflow?**

   Data collection, preprocessing, feature selection, model training, evaluation, and deployment.

5. **Explain overfitting and underfitting in machine learning.**

   - Overfitting: The model learns noise and performs well on training data but poorly on new data.
   - Underfitting: The model is too simple and fails to capture patterns.

6. **What is the bias-variance tradeoff?**

   High bias (underfitting) ignores patterns; high variance (overfitting) learns noise. A balance improves generalization.

7. **How do you handle missing values in a dataset?**

   Imputation (mean/median/mode), removal, or using algorithms that handle missing data.

8. **What is feature engineering, and why is it important?**

   Creating and selecting meaningful features to improve model accuracy.

9. **What is the curse of dimensionality**?

   Increased dimensions make computations complex and models less effective due to sparsity.

10. **How can dimensionality reduction help in machine learning?**

    Reduces computation cost, improves model performance, and prevents overfitting.

11. **Why is data visualization important in machine learning?**

    Helps identify patterns, outliers, and relationships between features.

12. **What are histograms used for in data analysis?**

    Show frequency distribution of numerical variables.

13. **How do box plots help identify outliers in data**

    Display data distribution, highlighting outliers beyond whiskers.

14. What does a correlation matrix represent?

Shows relationships between variables using correlation coefficients.

15. How does a heatmap help visualize correlation?

Uses color gradients to represent correlation strength.

16. What is a pair plot, and when should it be used?

Displays scatter plots of feature pairs; useful for understanding relationships.

17. What is Principal Component Analysis (PCA), and why is it used?

A dimensionality reduction technique that transforms data into principal components.

18. How does PCA reduce dimensionality while retaining information?

It captures the most variance using eigenvalues and eigenvectors.

19. What are eigenvalues and eigenvectors in PCA?

Eigenvalues represent variance captured; **eigenvectors** define principal directions.

20. How do you decide how many principal components to retain?

By analyzing the explained variance ratio.

21. Explain the working of the Find-S algorithm.

A simple learning algorithm that finds the most specific hypothesis from positive examples.

22. What are the limitations of the Find-S algorithm?

Works only for consistent and noiseless data.

23. How does the k-Nearest Neighbors (k-NN) algorithm work?

Classifies based on the majority class of the k nearest data points.

24. What is the significance of the k-value in k-NN?

A small k makes the model sensitive to noise, while a large k smoothens decision boundaries.

25. How do you choose the best value for k in k-NN?

Using cross-validation.

26. What are the advantages and disadvantages of k-NN?

- **Pros**: Simple, non-parametric.
- **Cons**: Computationally expensive, sensitive to irrelevant features.

27. What is Locally Weighted Regression?

A regression model that assigns different weights to data points based on proximity.

28. How does Locally Weighted Regression differ from standard regression?

It gives more importance to nearby points rather than fitting a global function.

29. What is the difference between Linear and Polynomial Regression?

Linear Regression fits a straight line; Polynomial Regression fits a curved line.

30. How do you evaluate the performance of a regression model?

Using metrics like R^2 score, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

31. Explain how the decision tree algorithm works.

Splits data recursively based on attribute conditions to create a tree-like structure.

32. What are entropy and information gain in decision trees?

Entropy measures data impurity; information gain decides the best split.

33. How does pruning help in decision tree learning?

Prevents overfitting by removing unnecessary branches.

34. What is the Naïve Bayes classifier, and why is it called "naïve"?

A probabilistic model assuming feature independence.

35. How does the Naïve Bayes classifier handle continuous features?

Uses Gaussian (Normal) distribution.

36. What assumptions does the Naïve Bayes classifier make?

Features are independent given the class.

37. What is k-Means clustering?

A partitioning method that groups data into k clusters.

38. How does k-Means Clustering work?

Initializes k centroids, assigns data points, recalculates centroids iteratively.

39. What are some applications of k-Means clustering?

Customer segmentation, image compression, anomaly detection.

40. How do you determine the optimal number of clusters in k-Means?

Using the Elbow Method or Silhouette Score.

41. What is the difference between k-Means and hierarchical clustering?

- k-Means: Divides into fixed k groups.
- Hierarchical: Creates a tree of clusters.

42. How does k-Means handle outliers?

Poorly, as outliers can distort centroids.

43. What are precision, recall, and F1-score?

- **Precision**: True Positives / (True Positives + False Positives).
- **Recall**: True Positives / (True Positives + False Negatives).
- **F1-score**: Harmonic mean of precision and recall.

44. What is cross-validation, and why is it important?

Splits data multiple times for better model evaluation.

45. What is the difference between training, validation, and test datasets?

- **Training**: Model learns.
- **Validation**: Tunes hyperparameters.
- **Test**: Evaluates performance.

46. Why do we use normalization and standardization in machine learning?

To scale data for better model performance.

47. How do you handle class imbalance in a dataset?

Oversampling, undersampling, or using weighted loss functions.

48. What are the common challenges in implementing machine learning models?

Data quality, overfitting, scalability.

49. How would you improve the accuracy of a machine learning model?

Feature engineering, hyperparameter tuning, ensemble learning.

50. How do you decide which machine learning model to use for a given problem?

Based on data type, problem type (classification, regression, clustering), and evaluation metrics.

Sample Programs:

1. **Write a program to predict the exam score.**

```
from sklearn.linear_model import LinearRegression
 X = [[1], [2], [3], [4], [5]]  # Hours studied
 y = [50, 55, 65, 70, 75]  # Exam score
 # Train the model
 model = LinearRegression()
 model.fit(X, y)
 # Predict for 6 hours of study
 prediction = model.predict([[6]])
 print(f"Predicted score: {prediction[0]:.2f}")
```

Output: Predicted score: 82.50

2. **Write a program on Sentiment Analysis.**

```
from textblob import TextBlob
text = "I love this product!"
sentiment = TextBlob(text).sentiment.polarity
if sentiment > 0:
   print("Positive sentiment 😊")
elif sentiment < 0:
   print("Negative sentiment 👿")
else:
   print("Neutral 😐")
```

Output: Positive sentiment ☺

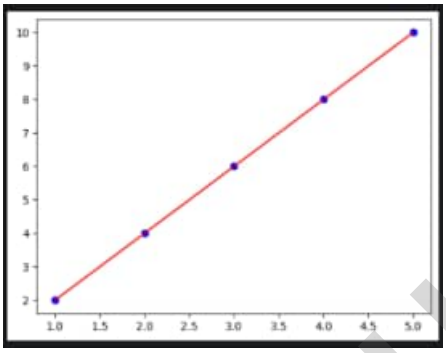
3. **Write a program to predict next value or future value.**

```
from sklearn.linear_model import LinearRegression
X = [[1], [2], [3], [4], [5]]  # Feature values
y = [2, 4, 6, 8, 10]           # Target values
model = LinearRegression().fit(X, y)  # Train model
print(model.predict([[6]]))  # Predict for X=6   Output: [12.]
```

4. **Write a program to predict linear regression for the given data points.**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 6, 8, 10])
model = LinearRegression().fit(X, y)
y_pred = model.predict(X)
plt.scatter(X, y, color="blue")  # Plot data points
plt.plot(X, y_pred, color="red")  # Plot regression line
plt.show()
```

Output:



5. **Write a program for gender prediction.**

```
from sklearn.tree import DecisionTreeClassifier
# Sample data (features: height, weight, shoe size), labels: 0 = Male, 1 = Female
X = [[180, 80, 44], [170, 70, 42], [160, 60, 38], [150, 50, 36]]
y = [0, 0, 1, 1]
# Train model
model = DecisionTreeClassifier()
model.fit(X, y)
# Predict gender for new data
print(model.predict([[165, 60, 40]]))
```

Output: [1]

6. **Write a program for gender prediction using SVM.**

```
from sklearn import svm
```

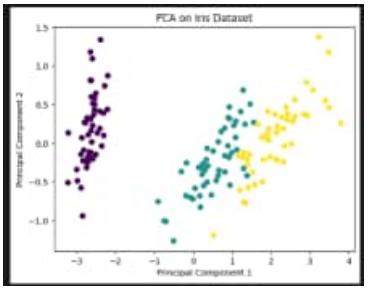
```
# Training data (height, weight) and labels (0 = Male, 1 = Female)
X = [[180, 80], [170, 70], [160, 60], [150, 50]]
y = [0, 0, 1, 1]
# Train SVM model
model = svm.SVC(kernel="linear")  # Linear kernel
model.fit(X, y)
# Predict gender for a new person
print(model.predict([[175, 75]]))
```

Output: [0]

7. **Write a program to predict and plot using PCA for iris dataset.**

```
from sklearn.decomposition import PCA
from sklearn import datasets
import matplotlib.pyplot as plt
# Load iris dataset
iris = datasets.load_iris()
X = iris.data
# Reduce dimensions from 4 to 2
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
# Scatter plot of reduced data
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=iris.target, cmap="viridis")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("PCA on Iris Dataset")
plt.show()
```

Output:



8. **Develop a program to predict whether a customer will make a purchase based on their age and salary**.

```
from sklearn.ensemble import RandomForestClassifier
# Sample labeled data: [Age, Salary], Labels: 0 = No Purchase, 1 = Purchase
X = [[25, 30000], [30, 40000], [35, 50000], [40, 60000], [45, 70000]]
y = [0, 0, 1, 1, 1]
# Train model
model = RandomForestClassifier()
model.fit(X, y)
# Predict purchase decision
print(model.predict([[33, 45000]]))
```

Output: [1]

9. **Develop a program to predict the height based on the given age.**

```
from sklearn.linear_model import LinearRegression
import numpy as np
# Training data: age vs. height
ages = np.array([10, 15, 18, 22]).reshape(-1, 1)
heights = np.array([4.5, 5.2, 5.8, 6.1])
# Train the model
model = LinearRegression()
model.fit(ages, heights)
# Predict height for a 17-year-old
predicted_height = model.predict([[17]])
print(f"Predicted height for 17 years old: {predicted_height[0]:.2f} feet")
```

Output: 5.50 feet