EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Artificial Intelligence and Data Science

'Jnana Prabha', Virgo Nagar Post, Bengaluru-560049

Academic Year: 2024-25

LABORATORY MANUAL

SEMESTER : VI

SUBJECT : Machine Learning Lab

SUBCODE : BCSL606

NAME: ______________________________

USN: ______________________________

SECTION: ______________________________

BATCH: ______________________________

PROGRAM OUTCOMES

Engineering Graduates will able to:

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 analyse 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 consideration for the public health and safety, and the cultural, societal, and environmental considerations.

Conduct investigations 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 modelling to complex engineering activities with an understanding of the limitations.

The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, 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 and finance: Demonstrate knowledge and understanding of the Engineering and management principles and apply these to one’s own 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.

INSTITUTE VISION AND MISSION

VISION

The East Point College of Engineering and Technology aspires to be a globally acclaimed institution, recognized for excellence in engineering education, applied research and nurturing students for holistic development.

MISSION

M1: To create engineering graduates through quality education and to nurture innovation, creativity and excellence in teaching, learning and research

M2: To serve the technical, scientific, economic and societal developmental needs of our communities

M3: To induce integrity, teamwork, critical thinking, personality development and ethics in students and to lay the foundation for lifelong learning

Department of Artificial Intelligence and Data Science

DEPARTMENT VISION AND MISSION

VISION

The department orients towards identifying and exploring emerging global trends in the fields of Artificial Intelligence and Data Science through academic excellence and quality research, producing proficient professionals for a flourishing society.

MISSION

M1: To nurture students with quality education, life-long learning, values and ethics.

M2: To produce ethical and competent professionals through comprehensive and holistic methodologies that align with the global industry demands in Artificial Intelligence and Data Science.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. Graduates will possess the ability to apply their knowledge of fundamental engineering, Computer Science and Data Science.
2. Graduates will have sound intercommunication skills, ethical values and responsibilities to work and serve for the development of the society.
3. Graduates will be able to understand, interpret, model and implement the Artificial Intelligence and Data Science based solutions for real world problems.

PROGRAM SPECIFIC OUTCOMES (PSOs)

1. To cater and enhance the analytical and technical skills of the graduates in order to be ready for the professional development, research and pursue higher education.
2. To formulate solutions for the real-world problems with the application of basic engineering principles and technical skills of Artificial Intelligence and Data Science.

COURSE LEARNING OBJECTIVES

CLO 1. To become familiar with data and visualize univariate, bivariate, and multivariate data using statistical techniques and dimensionality reduction.
CLO 2. To understand various machine learning algorithms such as similarity-based learning, regression, decision trees, and clustering.
CLO 3. To familiarize with learning theories, probability-based models and developing the skills required for decision-making in dynamic environments.

COURSE OUTCOMES

CO 1: Illustrate the principles of multivariate data and apply dimensionality reduction techniques.

CO 2: Demonstrate similarity-based learning methods and perform regression analysis.

CO 3: Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning.

CO 4: Implement the clustering algorithms to share computing resources.

Sl.NO	Experiments
1	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.
2	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.
3	Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.
4	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.
5	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. a. Label the first 50 points {x1,……,x50} as follows: if (xi ≤ 0.5), then xi ∊ Class1, else xi ∊ Class1 b. Classify the remaining points, x51,……,x100 using KNN. Perform this for k=1,2,3,4,5,20,30
6	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select the appropriate data set for your experiment and draw graphs
7	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.
8	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.
9	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.
10	Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

- The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Table of Content

Program-1

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.

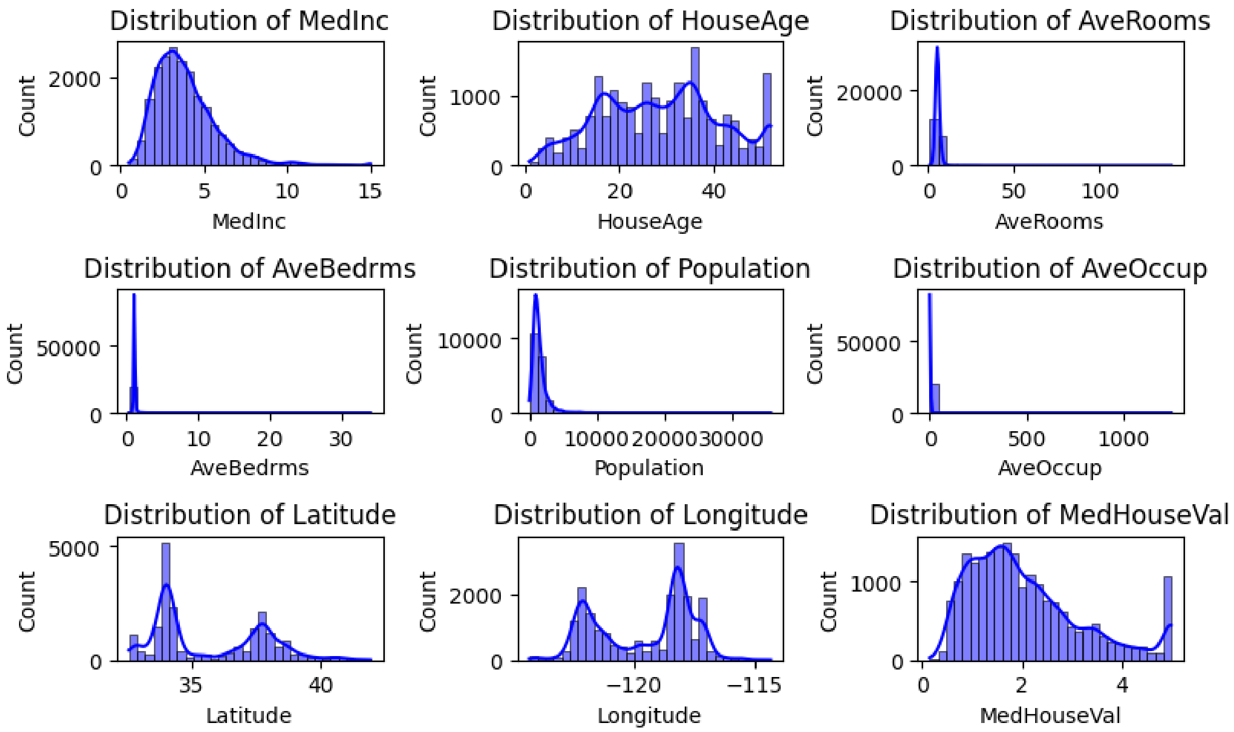
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
```

```
# 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
```

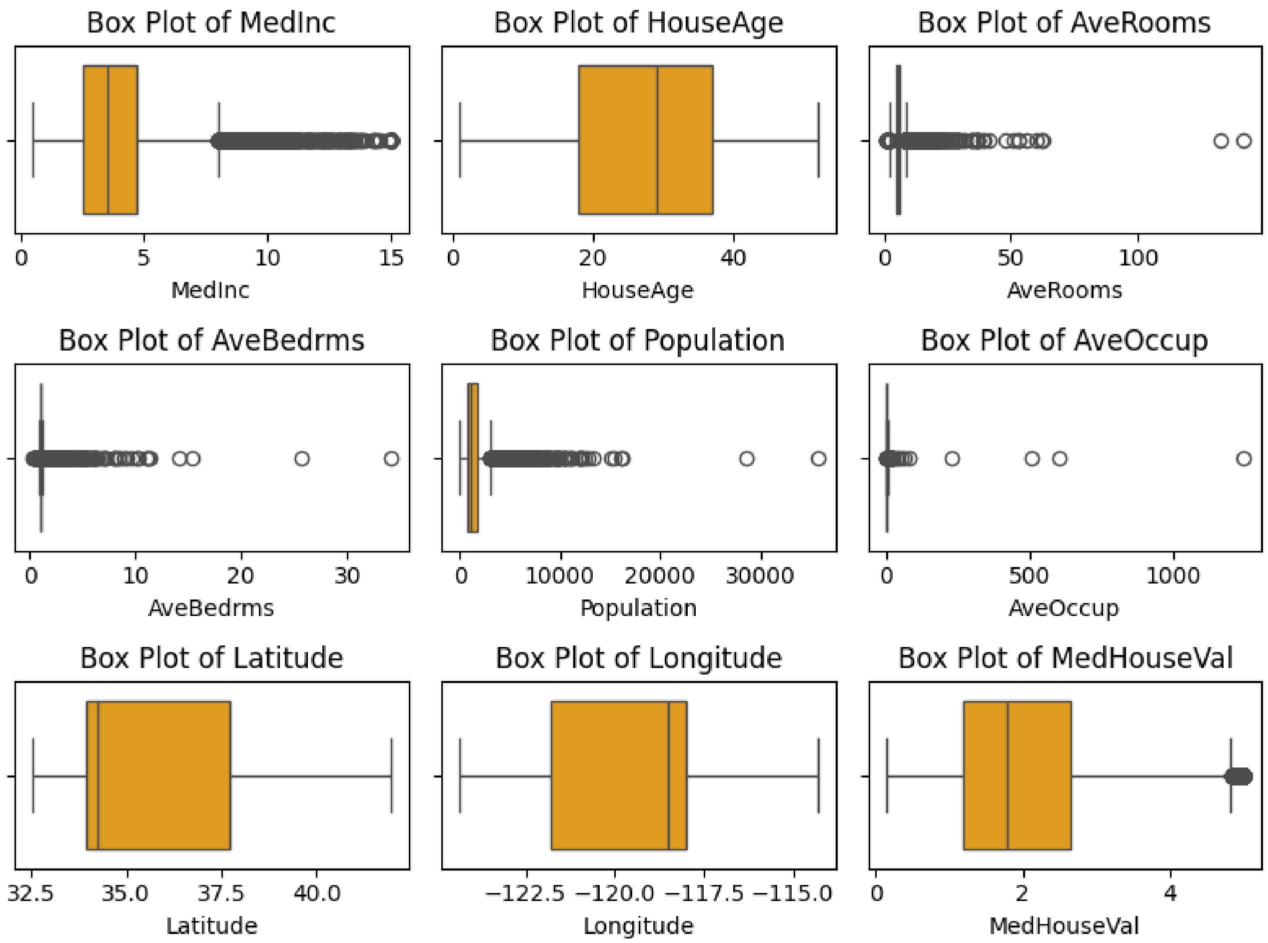
```
# Plot histograms
plt.figure(figsize=(8, 5))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.histplot(housing_df[feature], kde=True, bins=30, color='blue')
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
```

Output:



```
# Step 3: Generate box plots for numerical features
plt.figure(figsize=(8,6))
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()
```

Output:




```
# 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")
```

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
```

```
print("\nDataset Summary:")
print(housing_df.describe())
```

Output:

```
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

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.

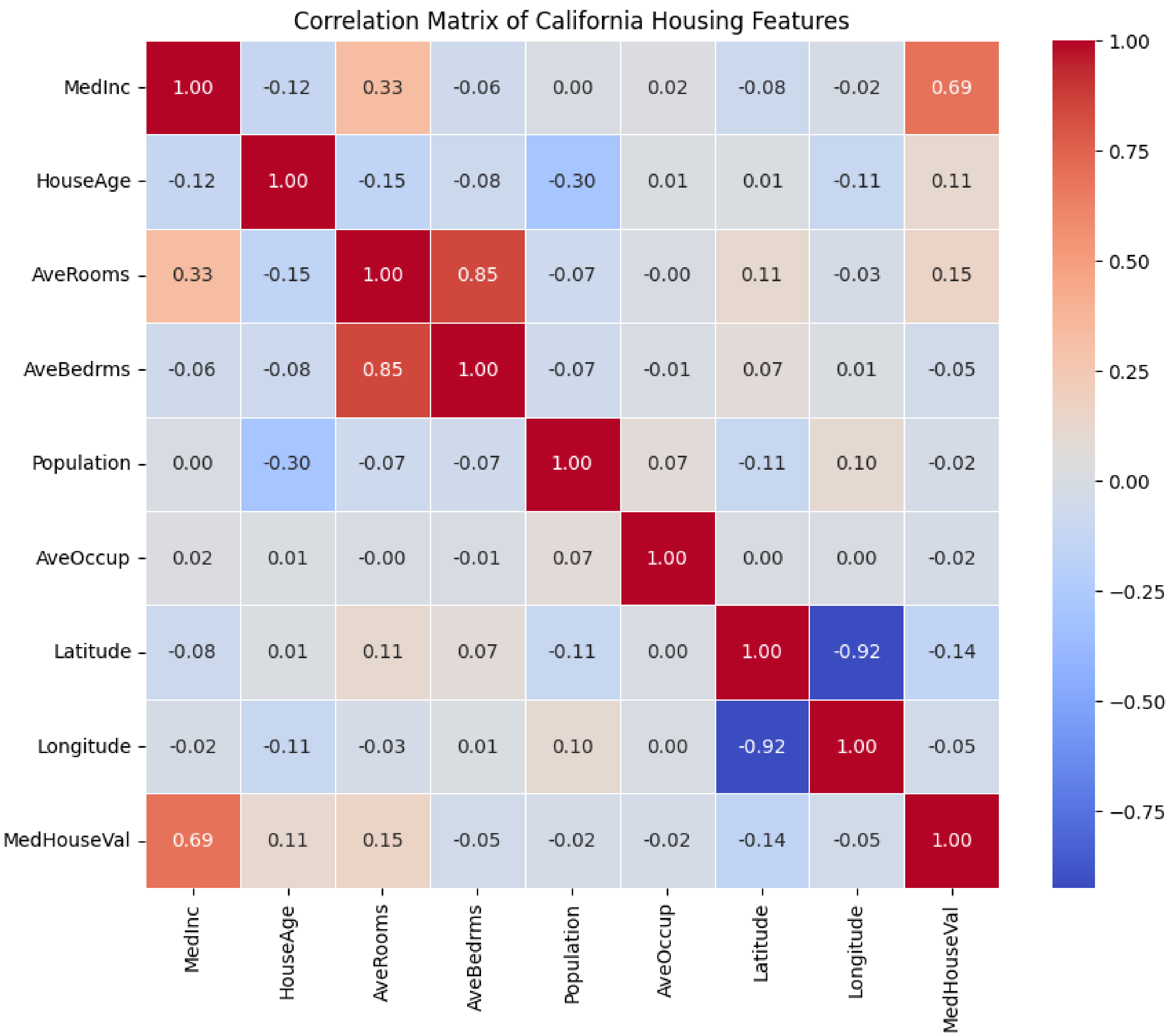
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
```

```
# 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()
```

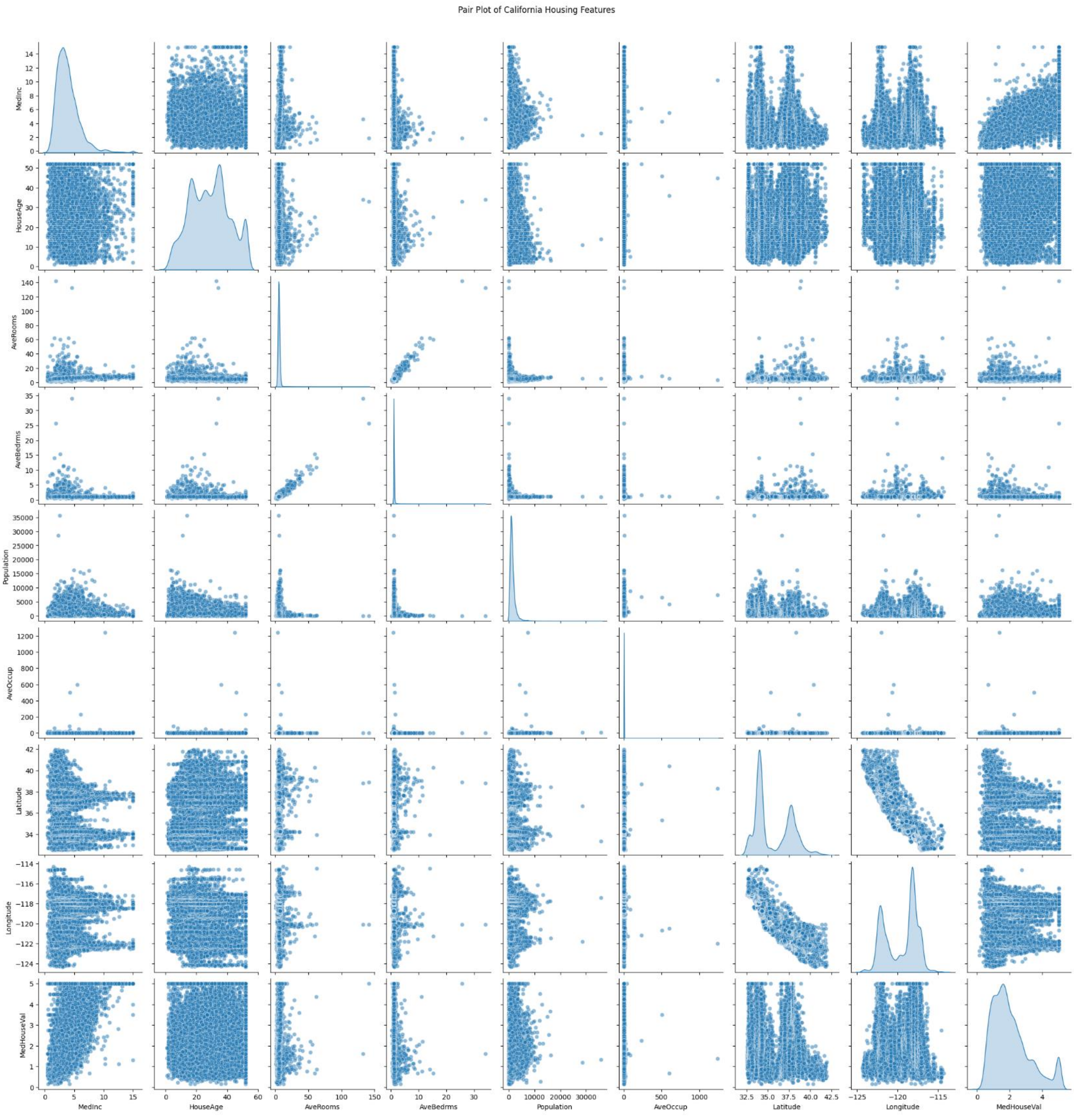
```
# Step 3: Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title('Correlation Matrix of California Housing Features')
plt.show()
```

Output:




```python
# 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:

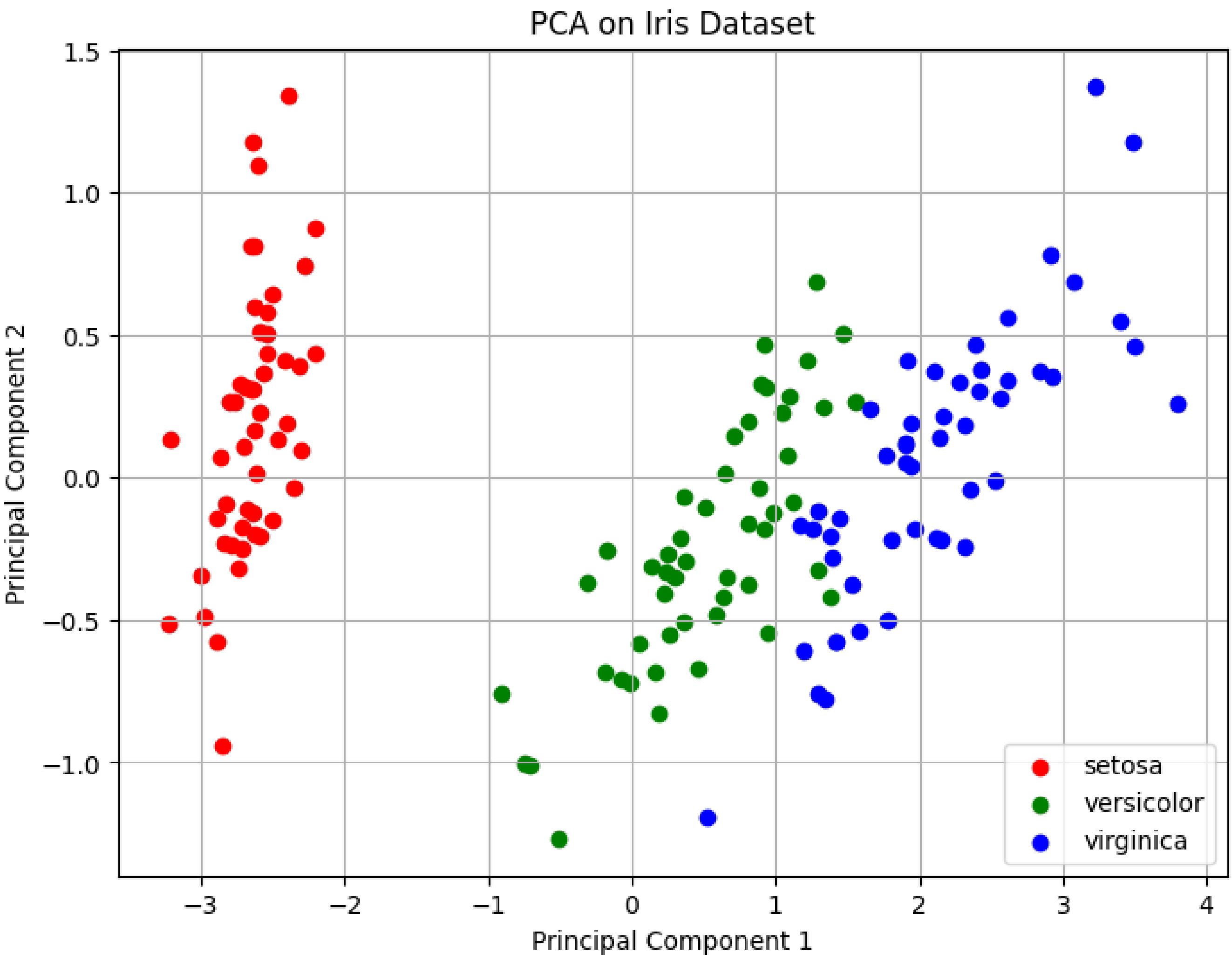


Pair Plot of California Housing Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal

Program-3

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

```
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()
```



PCA on Iris Dataset
1.5
1.0
0.5
0.0
−0.5
−1.0
Principal Component 2
−3
−2
−1
0
1
2
3
4
Principal Component 1
setosa
versicolor
virginica

Program-4

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.

```
import pandas as pd
import numpy as np



from google.colab import files
data_to_load = files.upload()
```

```
import io

df = pd.read_csv(io.StringIO(data_to_load['training_data.csv'].decode('utf-8')))
df
```

Output:

	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

```
columnLength= df.shape[1]                #obtain number of columns
print (df)
```

```
    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
```

```
h = ['0']*(columnLength-1)
hp=[]       #initialize list hp i.e list of hypotheses for positive training examples
hn=[]
```

```
for trainingExample in df.values: #this loop is used to build the hypothesis list for every row.
    if trainingExample[-1]!='no':   #if the trainingExample is positive, then it is appended to hp else to hn
        hp.append(list(trainingExample))
    else:
        hn.append(list(trainingExample))
```

```
for i in range (len(hp)):             #update the hypothesis h from most specific to maximally specific
    for j in range(columnLength-1):  #if the hypothesis attribute value is 0, it is updated to the attributes in the first hypothesis
        if (h[j]=='0'):
            h[j]=hp[i][j]
        if (h[j]!=hp[i][j]):        #if the attribute value in the hypothesis is not same as the attribute value in the successive hypotheses
            h[j]='?'                #then it is updated to '?' indicating that any value is acceptable.
        else:                       #if the attribute in the hypothesis is the same as the attribute value in the successive hypotheses
            h[j]=hp[i][j]
```

```
print('\nThe positive Hypotheses are\n')
print(hp)
print('\nThe negative Hypotheses are')
print(hn)
print('\nThe Maximally Specific Hypothesis h is')
print(h)
```

Output:

The positive Hypotheses are

[['Sunny', 'Hot', 'High', False, 'No'], ['Sunny', 'Hot', 'High', True, 'No'], ['Overcast', 'Hot', 'High', False, 'Yes'], ['Rain', 'Cold', 'High', False, 'Yes'], ['Rain', 'Cold', 'High', True, 'No'], ['Overcast', 'Hot', 'High', True, 'Yes'], ['Sunny', 'Hot', 'High', False, 'No']]

The negative Hypotheses are
[]

The Maximally Specific Hypothesis h is
['?', '?', 'High', '?']

Program-5

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.
a. Label the first 50 points {x1,……,x50} as follows: if (xi ≤ 0.5), then xi ∊ Class1, else xi ∊ Class1
b. Classify the remaining points, x51,……,x100 using KNN. Perform this for k=1,2,3,4,5,20,30

```
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")
```

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 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")
```

Output:

Results for k = 1:
Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2
Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2
Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class1
Point x58 (value: 0.4221) is classified as Class1
Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class1
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class1
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class1
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class1
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class1
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1

Point x100 (value: 0.4541) is classified as Class1

Results for k = 2:
Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2
Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2
Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class1
Point x58 (value: 0.4221) is classified as Class1
Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class1
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class1
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class1
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class1
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class1
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1
Point x100 (value: 0.4541) is classified as Class1

Results for k = 3:

Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2
Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2
Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class1
Point x58 (value: 0.4221) is classified as Class1
Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class1
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class2
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class1
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class2
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class1
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1
Point x100 (value: 0.4541) is classified as Class1

Results for k = 4:
Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2

Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2
Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class1
Point x58 (value: 0.4221) is classified as Class1
Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class1
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class1
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class1
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class1
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class1
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1
Point x100 (value: 0.4541) is classified as Class1

Results for k = 5:
Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2
Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2

Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class2
Point x58 (value: 0.4221) is classified as Class1
Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class2
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class2
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class2
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class2
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class2
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1
Point x100 (value: 0.4541) is classified as Class2

Results for k = 20:
Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2
Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2
Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class1
Point x58 (value: 0.4221) is classified as Class1

Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class1
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class2
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class2
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class2
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class1
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1
Point x100 (value: 0.4541) is classified as Class1

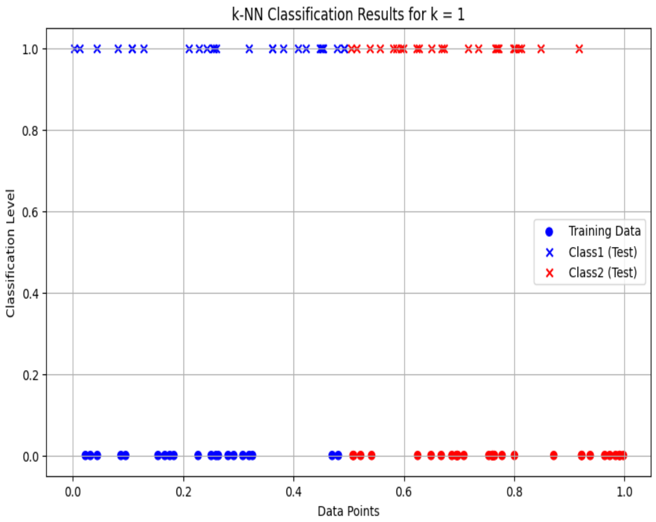
Results for k = 30:
Point x51 (value: 0.6280) is classified as Class2
Point x52 (value: 0.0018) is classified as Class1
Point x53 (value: 0.5994) is classified as Class2
Point x54 (value: 0.1290) is classified as Class1
Point x55 (value: 0.7166) is classified as Class2
Point x56 (value: 0.4091) is classified as Class1
Point x57 (value: 0.4479) is classified as Class1
Point x58 (value: 0.4221) is classified as Class1
Point x59 (value: 0.1073) is classified as Class1
Point x60 (value: 0.2282) is classified as Class1
Point x61 (value: 0.0439) is classified as Class1

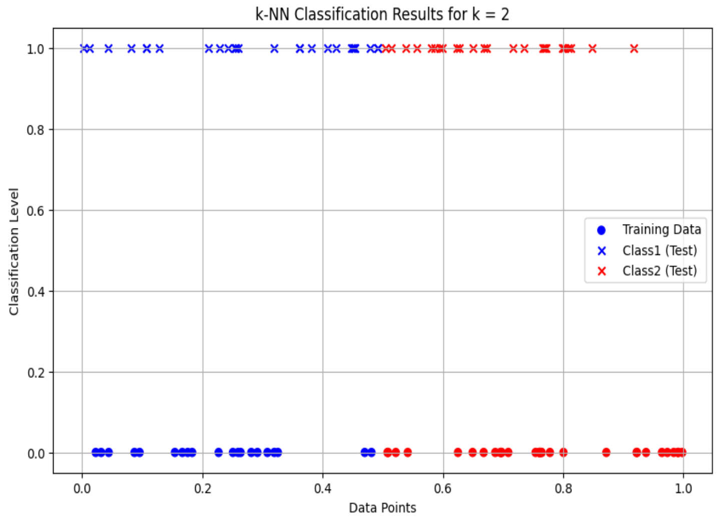
Point x62 (value: 0.5049) is classified as Class2
Point x63 (value: 0.4531) is classified as Class1
Point x64 (value: 0.0124) is classified as Class1
Point x65 (value: 0.0821) is classified as Class1
Point x66 (value: 0.7692) is classified as Class2
Point x67 (value: 0.3622) is classified as Class1
Point x68 (value: 0.4916) is classified as Class2
Point x69 (value: 0.6733) is classified as Class2
Point x70 (value: 0.4786) is classified as Class2
Point x71 (value: 0.5859) is classified as Class2
Point x72 (value: 0.7345) is classified as Class2
Point x73 (value: 0.3816) is classified as Class1
Point x74 (value: 0.7713) is classified as Class2
Point x75 (value: 0.6234) is classified as Class2
Point x76 (value: 0.3193) is classified as Class1
Point x77 (value: 0.1071) is classified as Class1
Point x78 (value: 0.8124) is classified as Class2
Point x79 (value: 0.8487) is classified as Class2
Point x80 (value: 0.8017) is classified as Class2
Point x81 (value: 0.5954) is classified as Class2
Point x82 (value: 0.5389) is classified as Class2
Point x83 (value: 0.5141) is classified as Class2
Point x84 (value: 0.7983) is classified as Class2
Point x85 (value: 0.9180) is classified as Class2
Point x86 (value: 0.4914) is classified as Class2
Point x87 (value: 0.6681) is classified as Class2
Point x88 (value: 0.2552) is classified as Class1
Point x89 (value: 0.2594) is classified as Class1
Point x90 (value: 0.2099) is classified as Class1
Point x91 (value: 0.3616) is classified as Class1
Point x92 (value: 0.5575) is classified as Class2
Point x93 (value: 0.8080) is classified as Class2
Point x94 (value: 0.7659) is classified as Class2
Point x95 (value: 0.4509) is classified as Class1
Point x96 (value: 0.2539) is classified as Class1
Point x97 (value: 0.5810) is classified as Class2
Point x98 (value: 0.6504) is classified as Class2
Point x99 (value: 0.2422) is classified as Class1
Point x100 (value: 0.4541) is classified as Class1

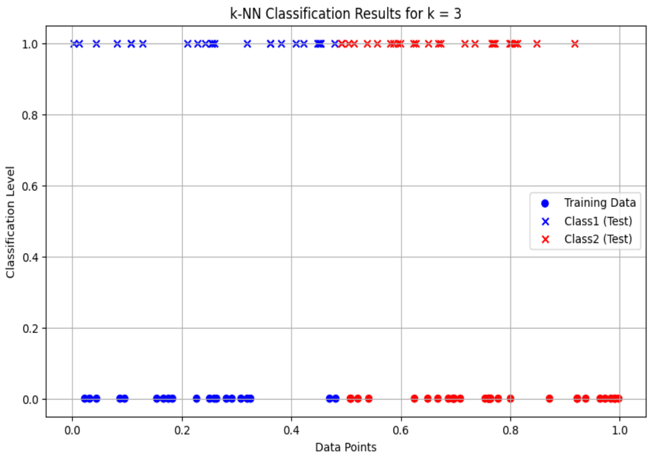
Classification complete.

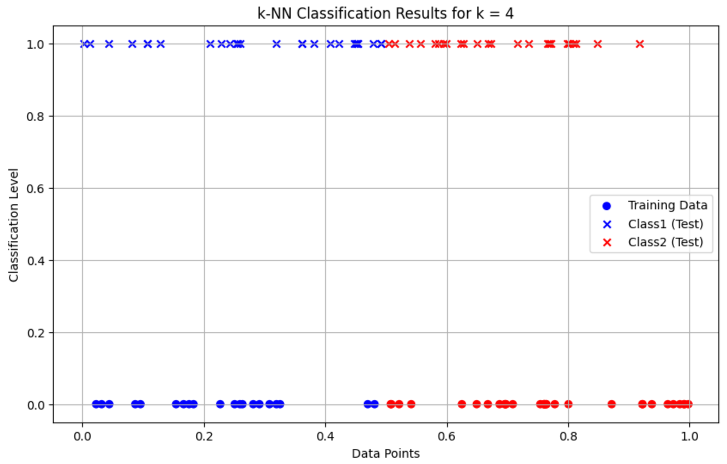
```
for k in k_values:
    classified_labels = results[k]
    class1_points = [test_data[i] for i in range(len(test_data)) if 
classified_labels[i] == "Class1"]
    class2_points = [test_data[i] for i in range(len(test_data)) if 
classified_labels[i] == "Class2"]

    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")
```

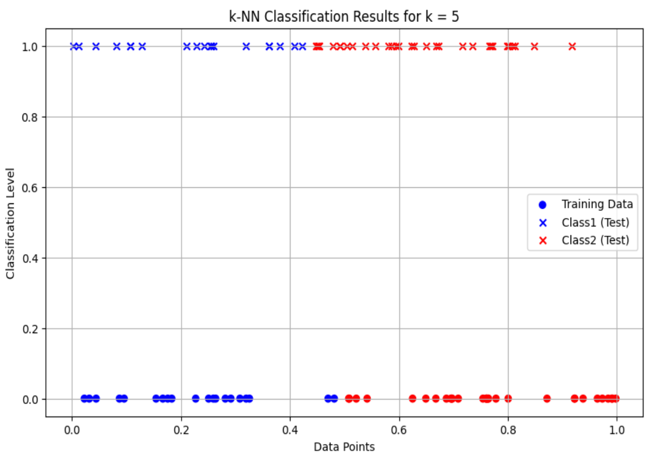
```python
    plt.scatter(class1_points, [1] * len(class1_points), c="blue", label="Class1 (Test)", marker="x")
    plt.scatter(class2_points, [1] * len(class2_points), c="red", label="Class2 (Test)", marker="x")

    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()
```

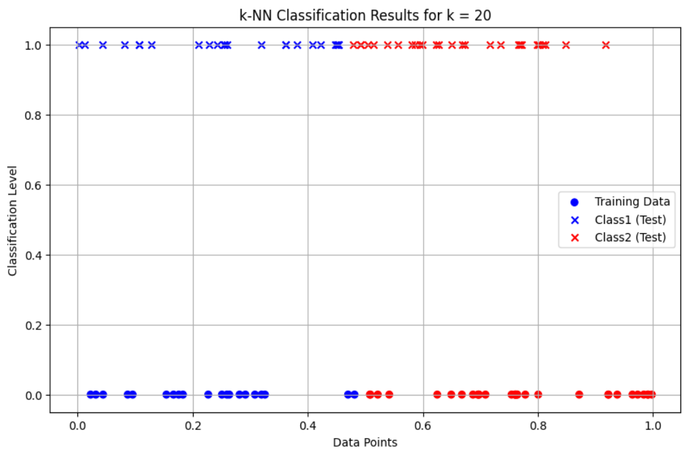


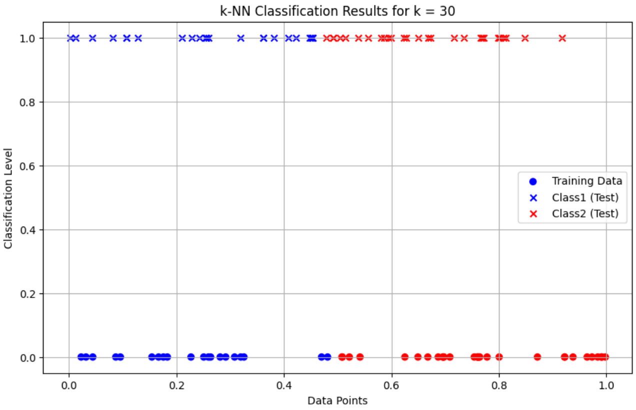








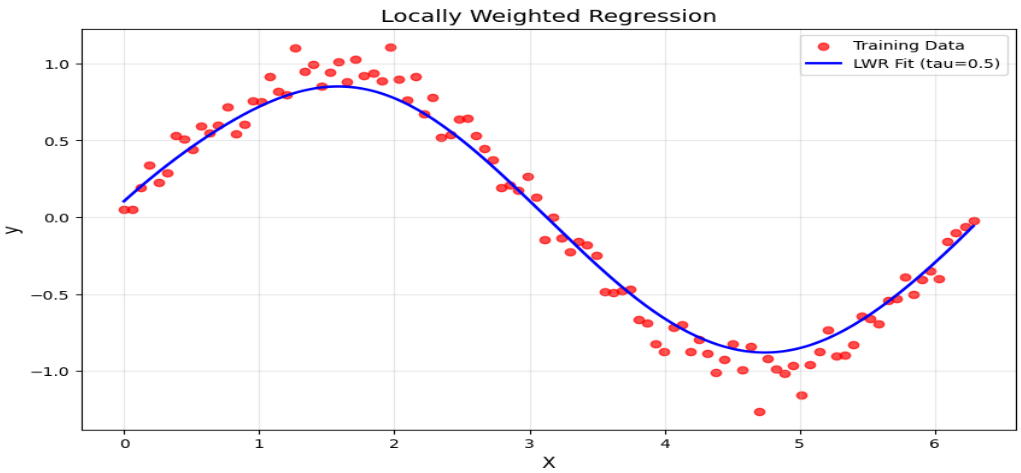



k-NN Classification Results for k = 30
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Program-6

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

```
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()
```



Program-7

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.

```
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
```

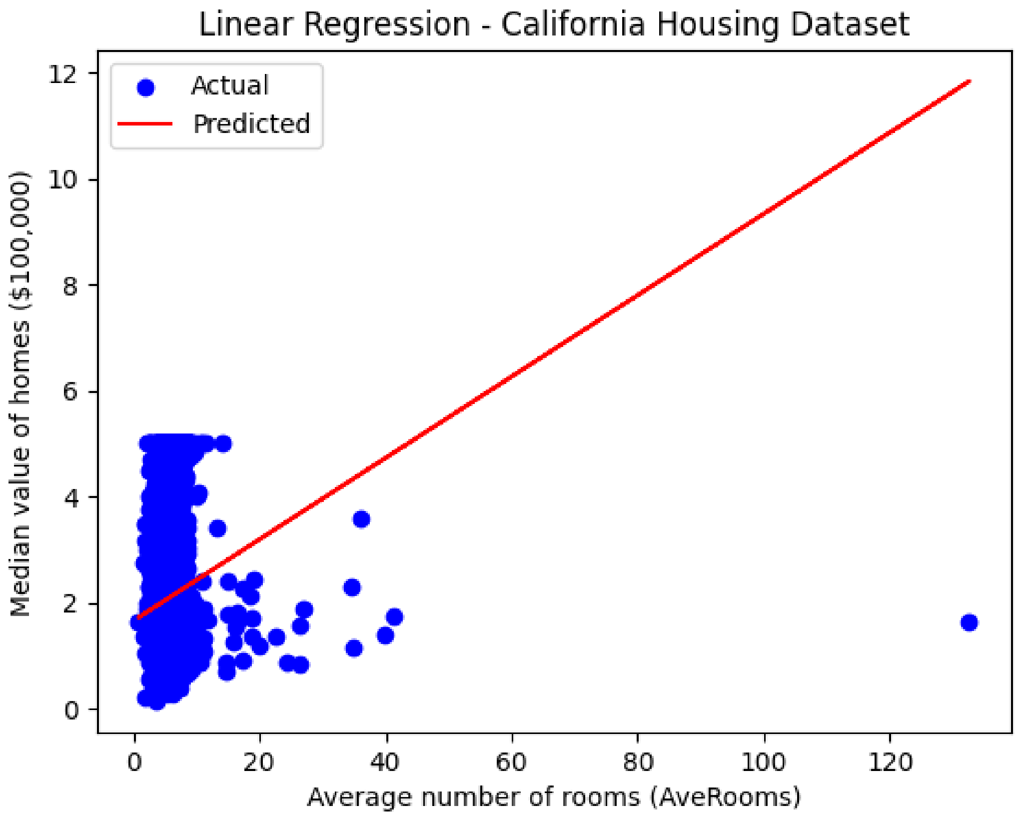
```
def linear_regression_california():
    housing = fetch_california_housing(as_frame=True)
    X = housing.data[["AveRooms"]]
    y = housing.target

    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)
```

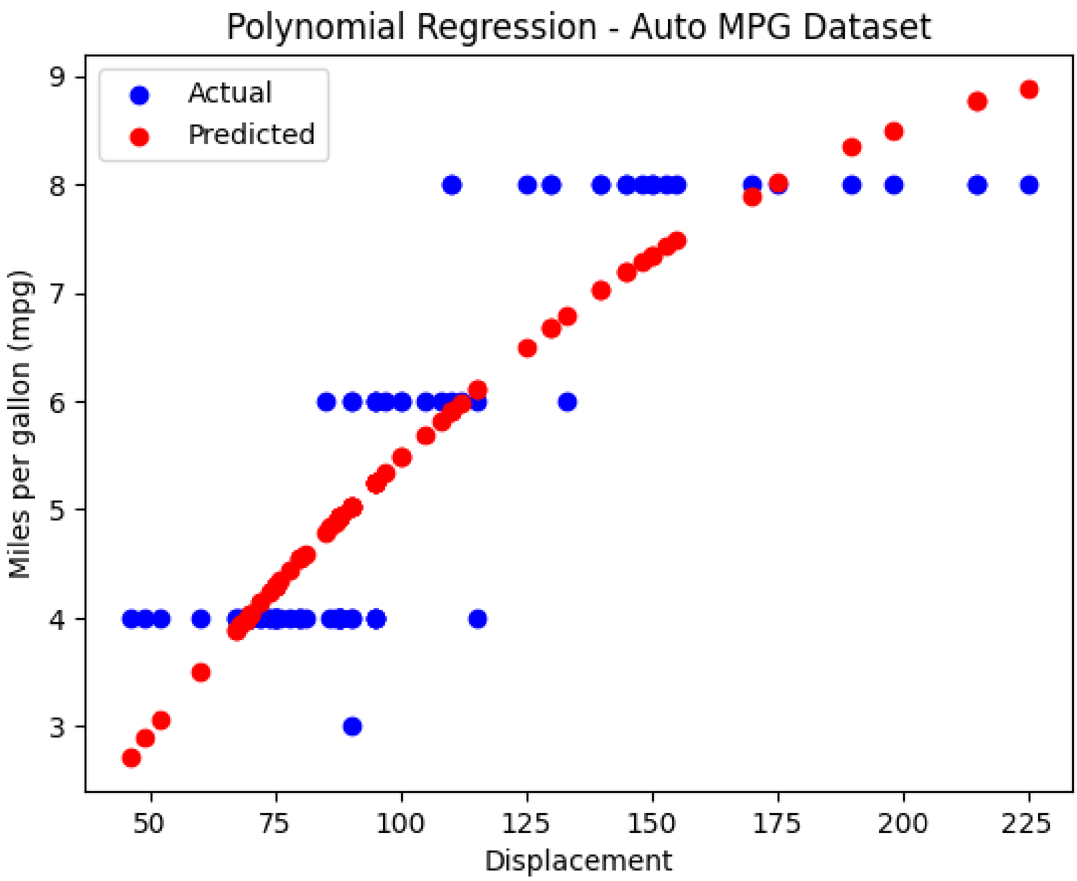
```python
    poly_model = make_pipeline(PolynomialFeatures(degree=2), StandardScaler(),
LinearRegression())
    poly_model.fit(X_train, y_train)

    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: 0.7431490557205861
R^2 Score: 0.7505650609469626
```

Program-8

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.

```
# Importing necessary libraries
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
```

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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
new_sample = np.array([X_test[0]])
prediction = clf.predict(new_sample)
```

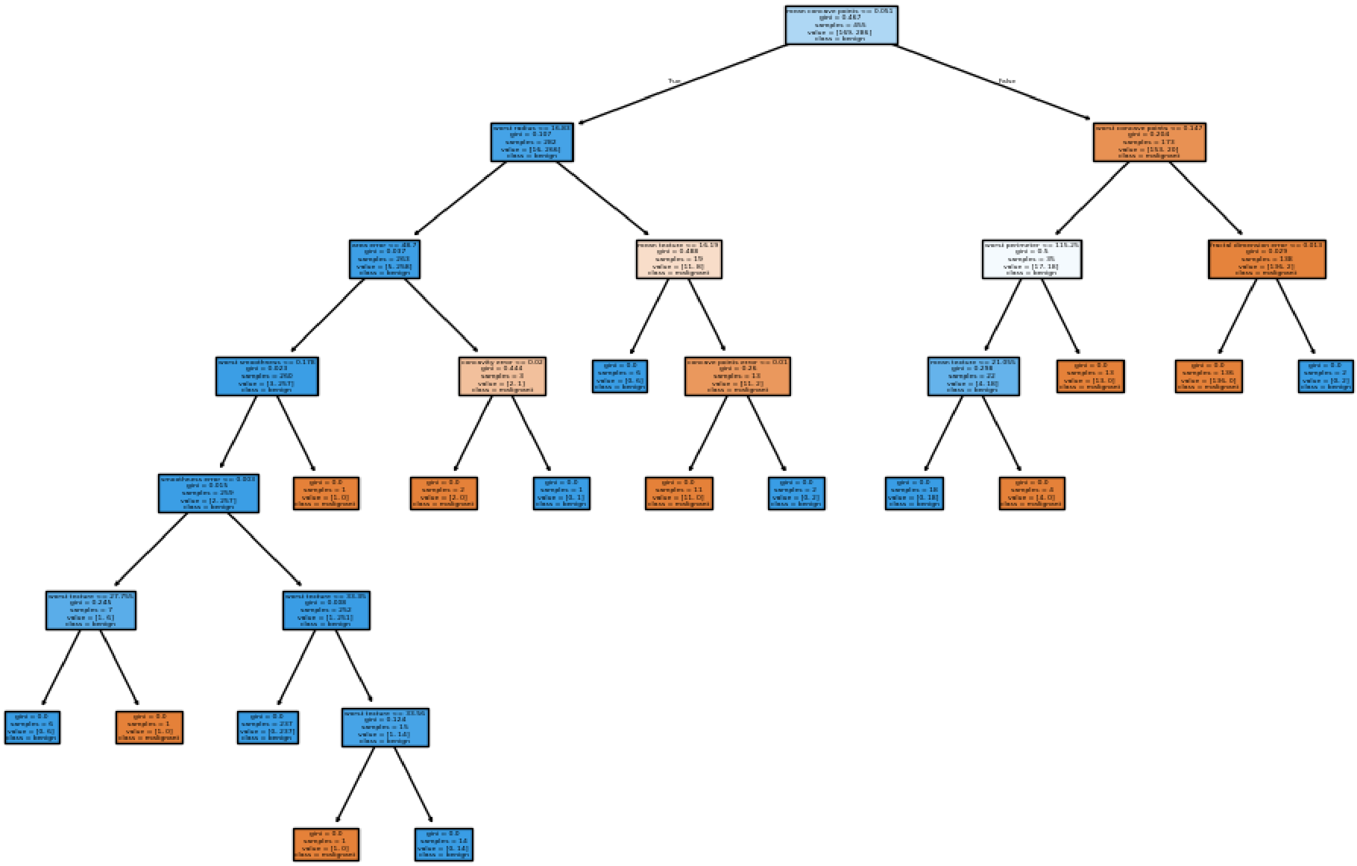
Output:

Model Accuracy: 94.74%

```
prediction_class = "Benign" if prediction == 1 else "Malignant"
print(f"Predicted Class for the new sample: {prediction_class}")
```

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

Decision Tree - Breast Cancer Dataset



Program-9

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.

```
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}%')
```

Output:
Accuracy: 80.83%

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

Output:

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
```

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

Output:

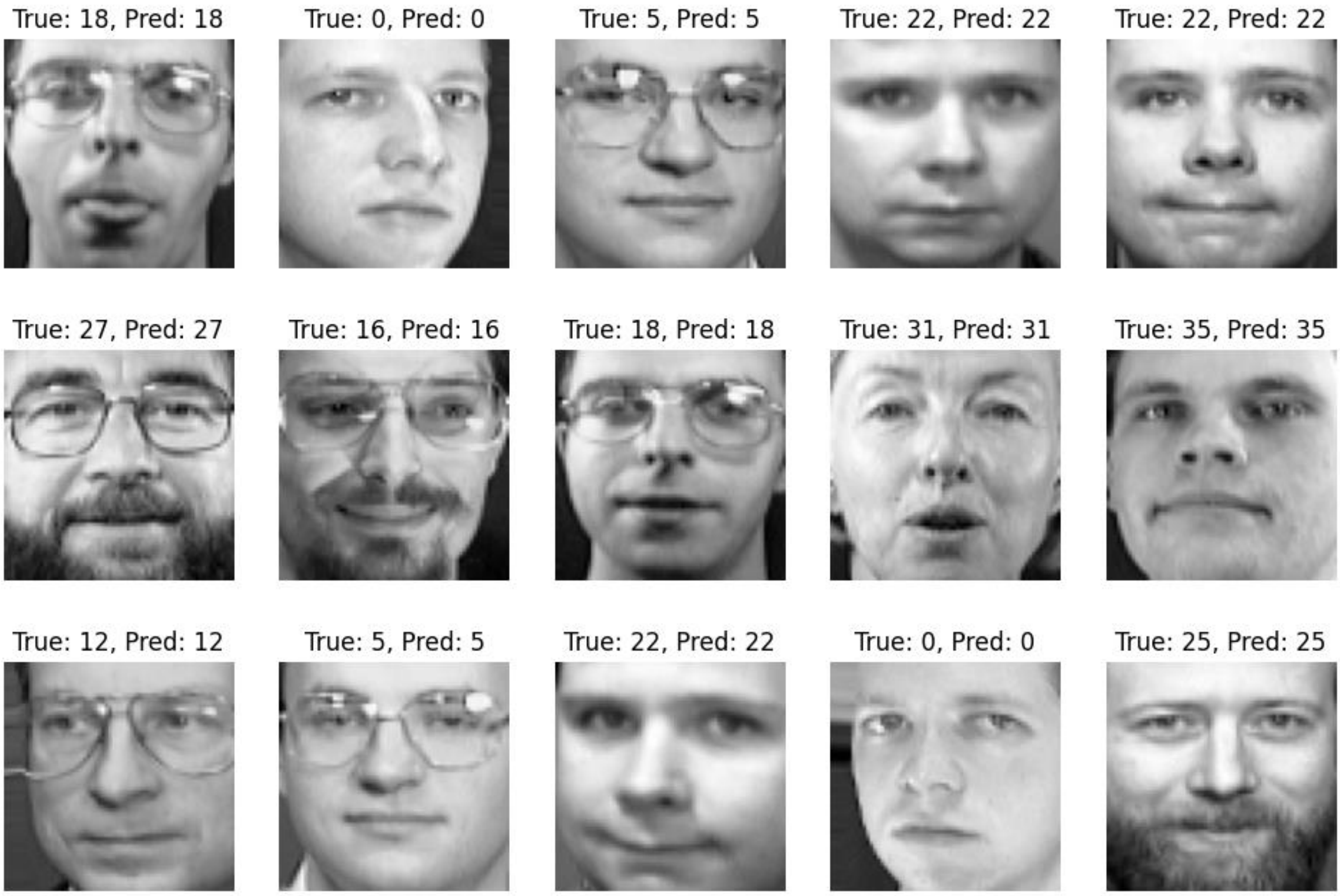
```
Confusion Matrix:
[[2 0 0 ... 0 0 0]
 [0 2 0 ... 0 0 0]
 [0 0 2 ... 0 0 1]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 3 0]
 [0 0 0 ... 0 0 5]]
```

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

Output:

```
Cross-validation accuracy: 87.25%
```

```
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()
```


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

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

```
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))
```

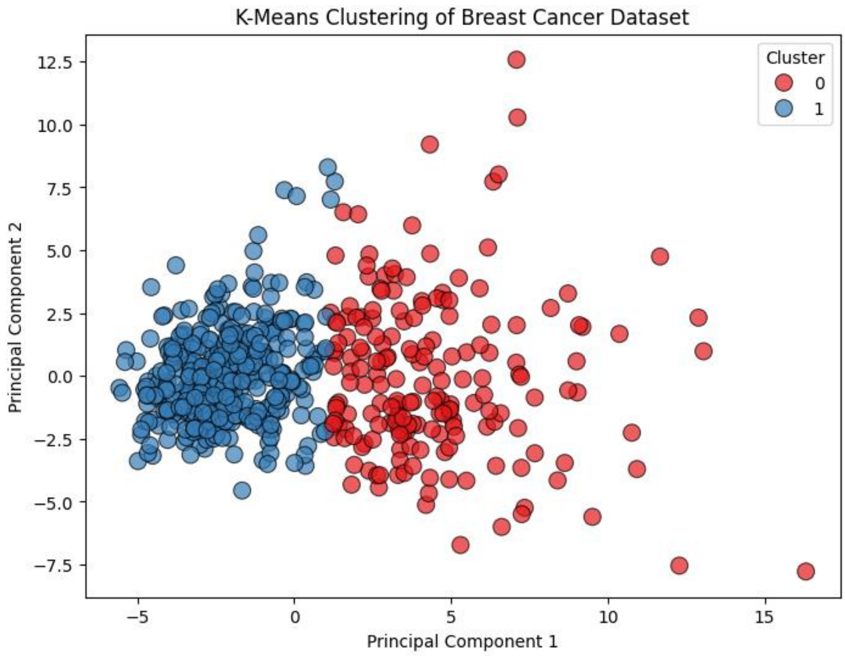
Output:

```
Confusion Matrix:
[[175  37]
 [ 13 344]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.83      0.88       212
           1       0.90      0.96      0.93       357

    accuracy                           0.91       569
   macro avg       0.92      0.89      0.90       569
weighted avg       0.91      0.91      0.91       569
```

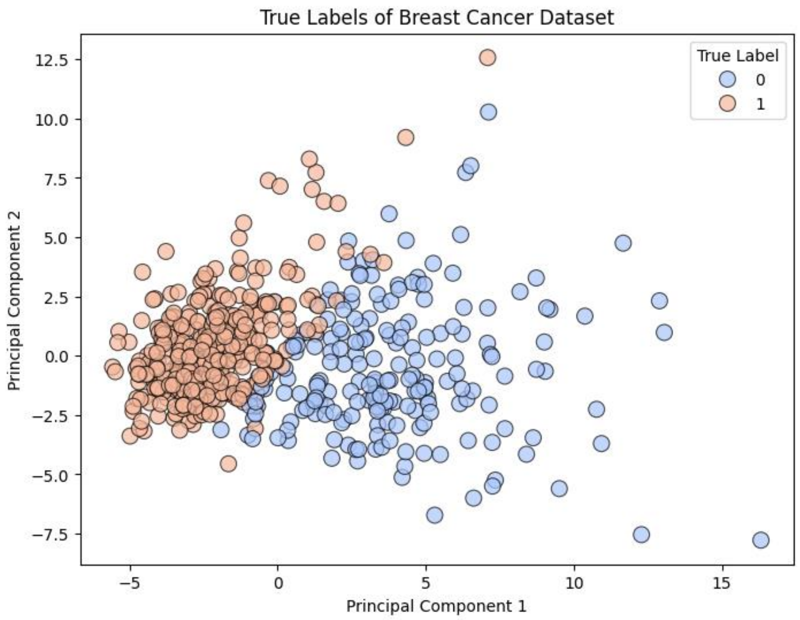
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

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

```python
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()
```




```python
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()
```