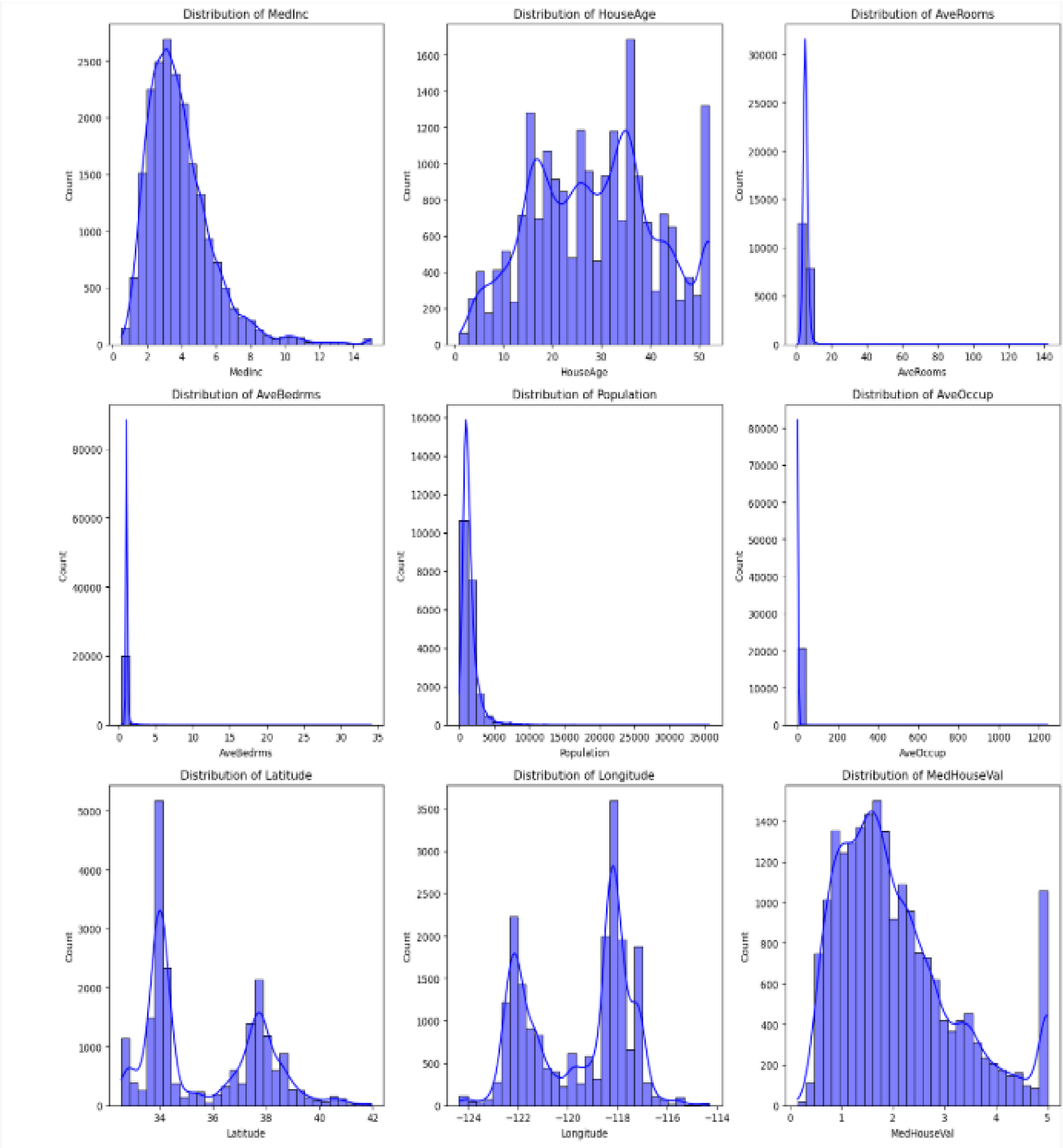
1.Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.

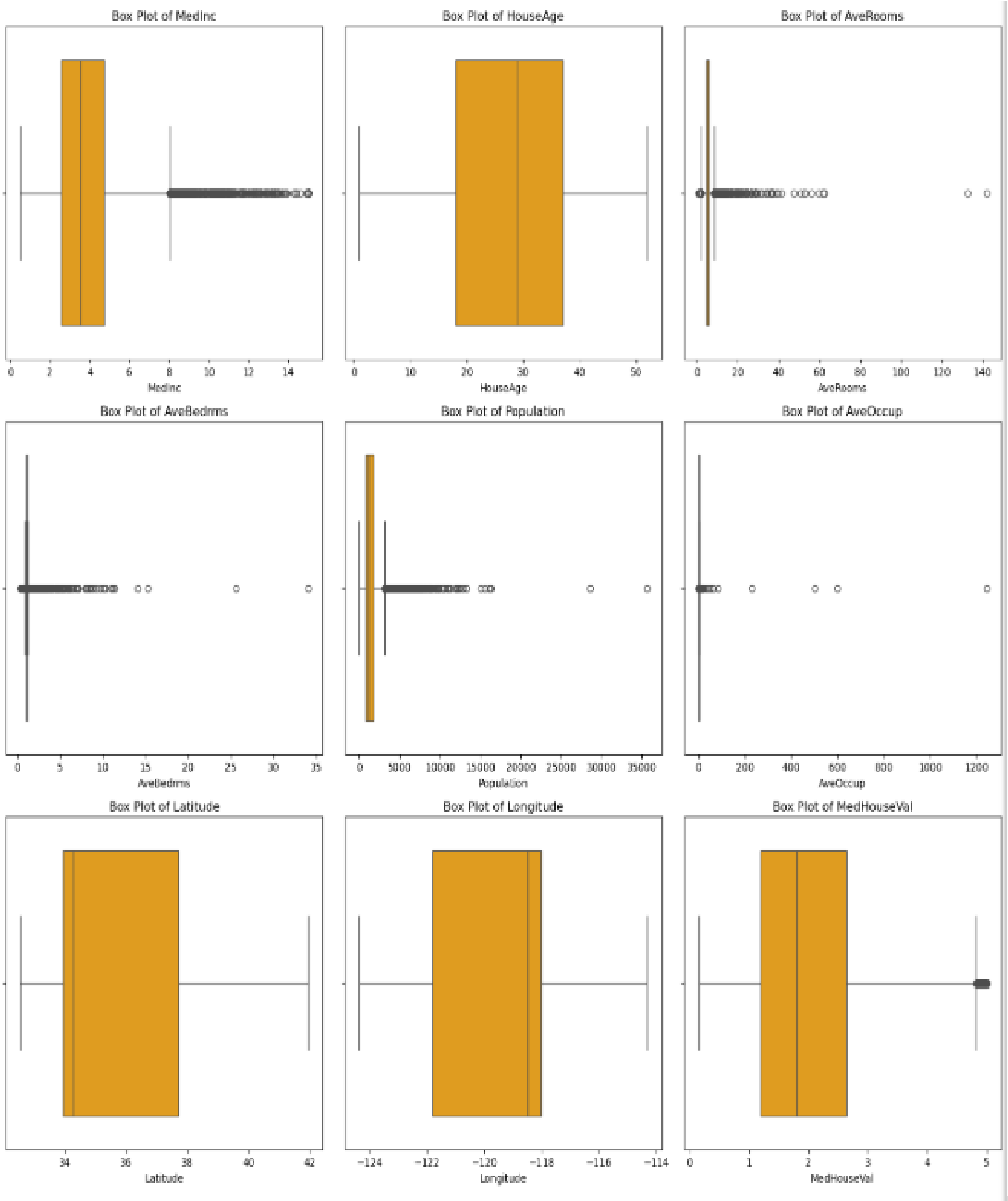
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

# 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

# Determine grid size for subplots
n_features = len(numerical_features)
n_cols = 3  # Number of columns for subplot grid
n_rows = (n_features // n_cols) + (n_features % n_cols > 0)  # Number of rows needed

# Plot histograms
plt.figure(figsize=(15, 5 * n_rows))
for i, feature in enumerate(numerical_features):
    plt.subplot(n_rows, n_cols, 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, 5 * n_rows))
for i, feature in enumerate(numerical_features):
    plt.subplot(n_rows, n_cols, 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")

# Optional: Print a summary of the dataset
print("\nDataset Summary:")
print(housing_df.describe())
```

OUTPUT



Distribution of MedInc
MedInc
Count
Distribution of HouseAge
HouseAge
Count
Distribution of AveRooms
AveRooms
Count
Distribution of AveBedrms
AveBedrms
Count
Distribution of Population
Population
Count
Distribution of AveOccup
AveOccup
Count
Distribution of Latitude
Latitude
Count
Distribution of Longitude
Longitude
Count
Distribution of MedHouseVal
MedHouseVal
Count



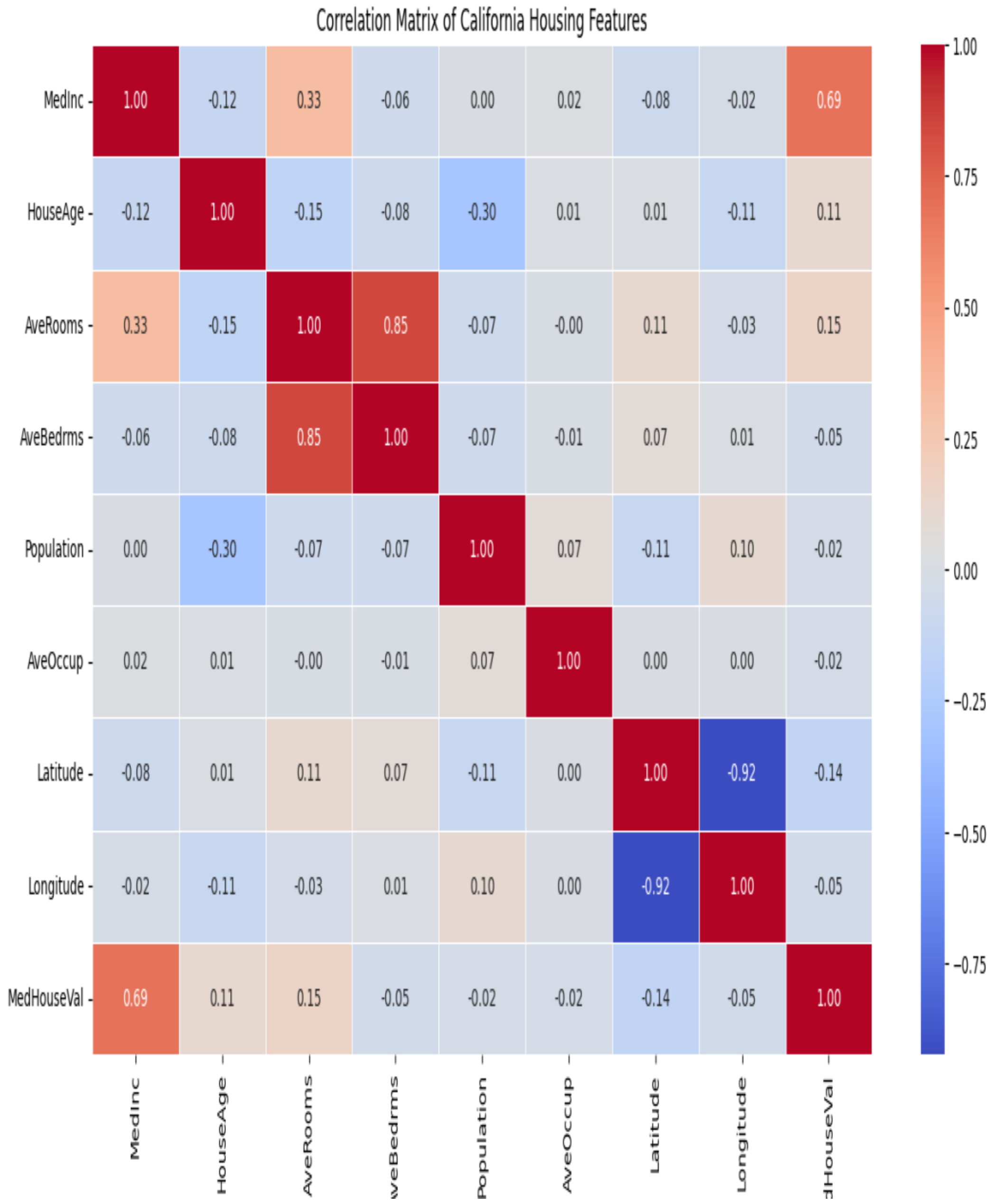
Box Plot of MedInc
MedInc
Box Plot of HouseAge
HouseAge
Box Plot of AveRooms
AveRooms
Box Plot of AveBedrms
AveBedrms
Box Plot of Population
Population
Box Plot of AveOccup
AveOccup
Box Plot of Latitude
Latitude
Box Plot of Longitude
Longitude
Box Plot of MedHouseVal
MedHouseVal

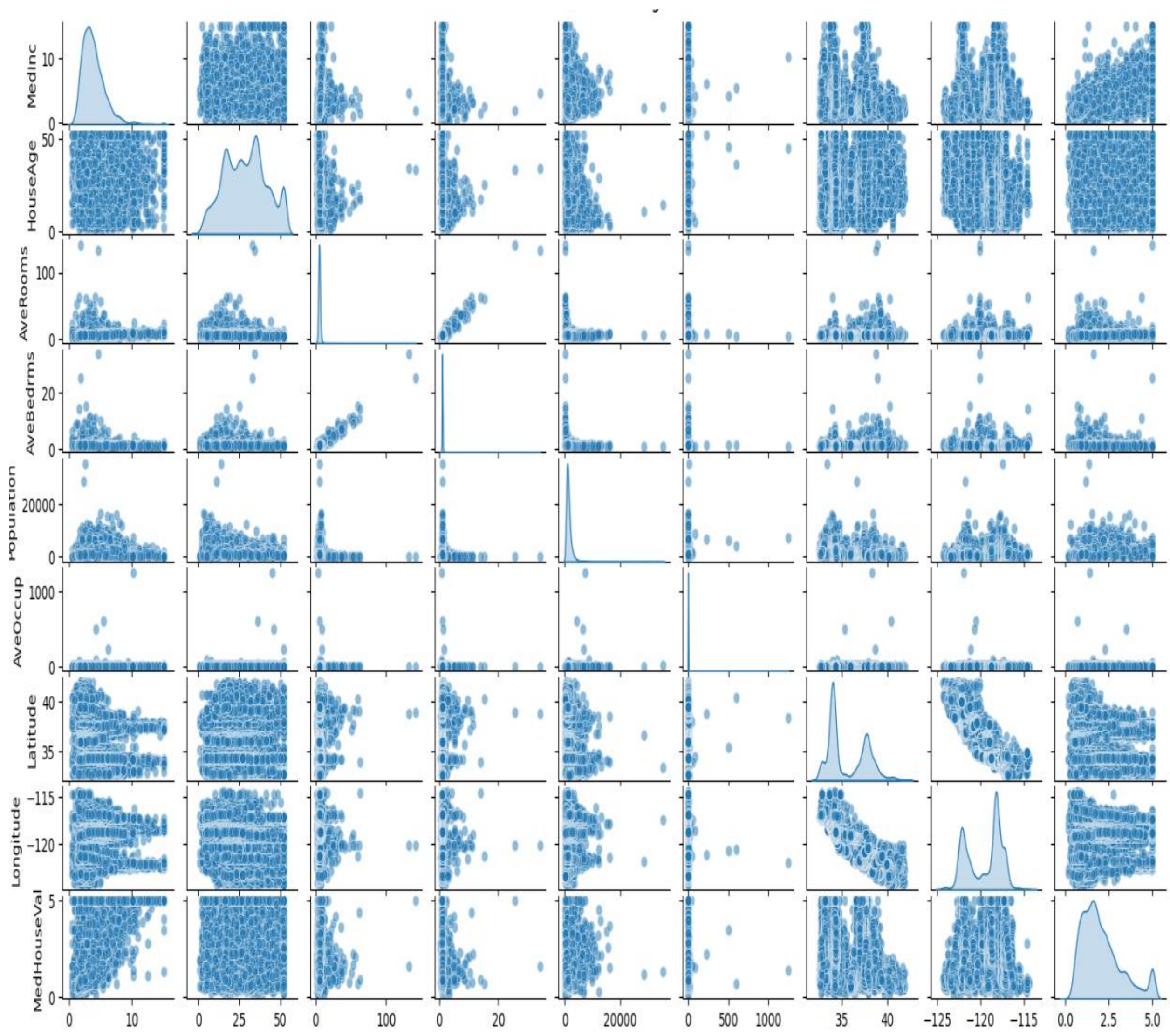
```
Outliers Detection:
MedInc: 681 outliers
HouseAge: 0 outliers
AveRooms: 511 outliers
AveBedrms: 1424 outliers
Population: 1196 outliers
AveOccup: 711 outliers
Latitude: 0 outliers
Longitude: 0 outliers
MedHouseVal: 1071 outliers

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
```

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()

# 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


Correlation Matrix of California Housing Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal
1.00 -0.12 0.33 -0.06 0.00 0.02 -0.08 -0.02 0.69
-0.12 1.00 -0.15 -0.08 -0.30 0.01 0.01 -0.11 0.11
0.33 -0.15 1.00 0.85 -0.07 -0.00 0.11 -0.03 0.15
-0.06 -0.08 0.85 1.00 -0.07 -0.01 0.07 0.01 -0.05
0.00 -0.30 -0.07 -0.07 1.00 0.07 -0.11 0.10 -0.02
0.02 0.01 -0.00 -0.01 0.07 1.00 0.00 0.00 -0.02
-0.08 0.01 0.11 0.07 -0.11 0.00 1.00 -0.92 -0.14
-0.02 -0.11 -0.03 0.01 0.10 0.00 -0.92 1.00 -0.05
0.69 0.11 0.15 -0.05 -0.02 -0.02 -0.14 -0.05 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

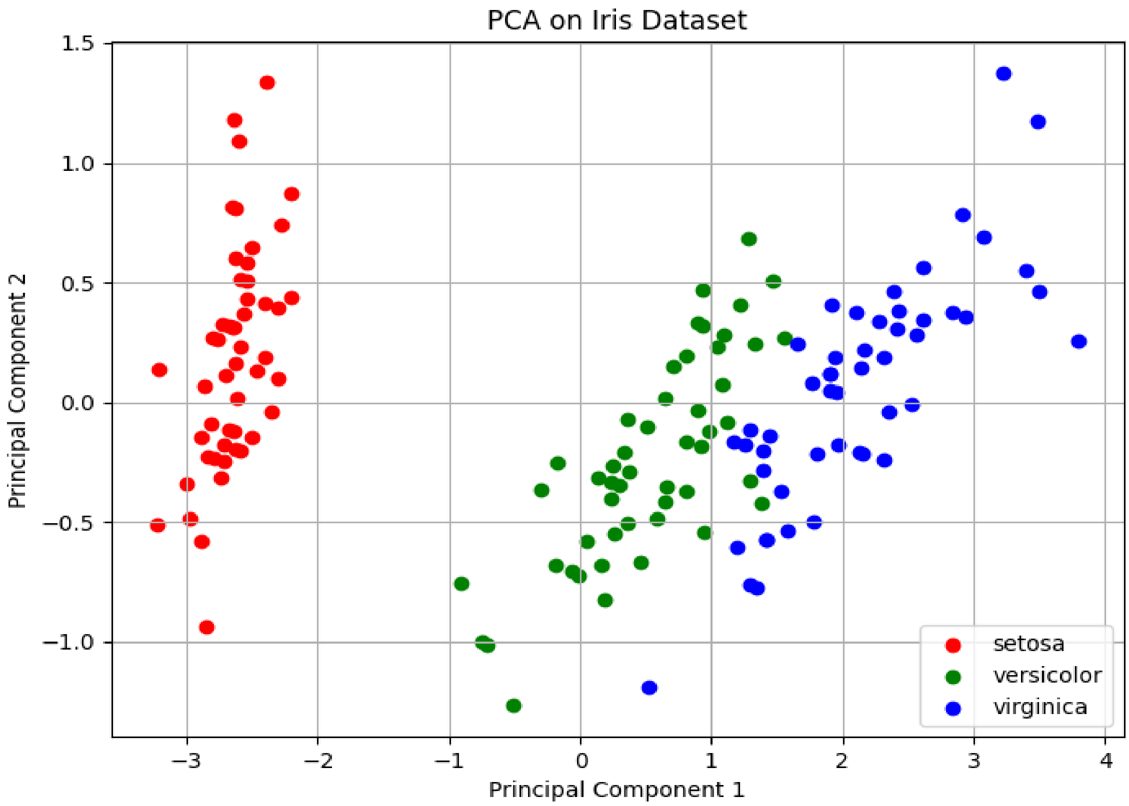

MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal

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()
```

OUTPUT:


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

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


def find_s_algorithm_from_csv(file_path):
    # Load CSV file and clean it
    df = pd.read_csv(file_path, skipinitialspace=True).dropna(axis=1, how="all")  # Remove
empty columns

    # Strip spaces from column names
    df.columns = df.columns.str.strip()

    # Rename target label column if necessary
    if "enjoy sport" in df.columns:
        df.rename(columns={"enjoy sport": "label"}, inplace=True)

    # Debugging Output
    print("\nDataset Preview:\n", df.head())
    print("\nColumn Names:", df.columns.tolist())
    print("\nLabels Found in Dataset:", set(df["label"]))

    # Extract attributes and labels
    attributes = df.iloc[:, :-1].values  # Features
    labels = df.iloc[:, -1].values       # Target labels

    hypothesis = None

    for i in range(len(labels)):
        if str(labels[i]).strip().lower() == "yes":  # Convert label to lowercase for consistency
            if hypothesis is None:
```

```
            hypothesis = list(attributes[i])  # First positive example initializes hypothesis
        else:
            for j in range(len(hypothesis)):
                if hypothesis[j] != attributes[i][j]:
                    hypothesis[j] = "?"  # Generalize differing attributes

    if hypothesis is None:
        print("\n No positive examples ('Yes') found in dataset! Check CSV formatting.")
    return hypothesis

# Run Find-S Algorithm
file_path = "C:/Users/radha/Downloads/data.csv"  # Ensure the correct file path
hypothesis = find_s_algorithm_from_csv(file_path)

print("\nFinal Hypothesis:", hypothesis)
```

OUTPUT:

DATASET

Temperature	air temp	humidity	wind	water	forecast	enjoy sport
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Dataset Preview:

```
  temperature air temp humidity    wind water forecast label
0       sunny     warm   normal  strong  warm     same   yes
1       sunny     warm     high  strong  warm     same   yes
2       rainy     cold     high  strong  warm   change    no
3       sunny     warm     high  strong  cool   change   yes
```

Column Names: ['temperature', 'air temp', 'humidity', 'wind', 'water', 'forecast', 'label']

Labels Found in Dataset: {'no', 'yes'}

Final Hypothesis: ['sunny', 'warm', '?', 'strong', '?', '?']

[ ]:

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 ϵ Class1, else xi ϵ 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

# Generate random data
data = np.random.rand(100)

# Assign labels to the first 50 points based on a threshold
labels = ["Class1" if x <= 0.5 else "Class2" for x in data[:50]]

# Function to calculate Euclidean distance
def euclidean_distance(x1, x2):
    return abs(x1 - x2)

# k-NN classification function
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]

# Split data into training and testing sets
train_data = data[:50]
train_labels = labels
```

```
test_data = data[50:]

# Define k values to test
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")

# Dictionary to store results
results = {}

# Perform classification for different k values
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")

# Visualization of classification results
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: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class1

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class1

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class1

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

Point x99 (value: 0.2242) is classified as Class1

Point x100 (value: 0.9074) is classified as Class2

Results for k = 2:

Point x51 (value: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class1

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class1

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class1

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

Point x99 (value: 0.2242) is classified as Class1

Point x100 (value: 0.9074) is classified as Class2

Results for k = 3:

Point x51 (value: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class1

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class1

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class1

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

Point x99 (value: 0.2242) is classified as Class1

Point x100 (value: 0.9074) is classified as Class2

Results for k = 4:

Point x51 (value: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class1

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class1

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class1

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

Point x99 (value: 0.2242) is classified as Class1

Point x100 (value: 0.9074) is classified as Class2

Results for k = 5:

Point x51 (value: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class1

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class1

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class1

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

Point x99 (value: 0.2242) is classified as Class1

Point x100 (value: 0.9074) is classified as Class2

Results for k = 20:

Point x51 (value: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class2

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class2

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class2

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

Point x99 (value: 0.2242) is classified as Class1

Point x100 (value: 0.9074) is classified as Class2

Results for k = 30:

Point x51 (value: 0.2861) is classified as Class1

Point x52 (value: 0.6656) is classified as Class2

Point x53 (value: 0.7032) is classified as Class2

Point x54 (value: 0.1030) is classified as Class1

Point x55 (value: 0.8310) is classified as Class2

Point x56 (value: 0.1928) is classified as Class1

Point x57 (value: 0.0589) is classified as Class1

Point x58 (value: 0.1024) is classified as Class1

Point x59 (value: 0.0162) is classified as Class1

Point x60 (value: 0.2494) is classified as Class1

Point x61 (value: 0.9699) is classified as Class2

Point x62 (value: 0.2614) is classified as Class1

Point x63 (value: 0.0175) is classified as Class1

Point x64 (value: 0.8446) is classified as Class2

Point x65 (value: 0.5356) is classified as Class2

Point x66 (value: 0.2169) is classified as Class1

Point x67 (value: 0.5320) is classified as Class2

Point x68 (value: 0.0512) is classified as Class1

Point x69 (value: 0.2760) is classified as Class1

Point x70 (value: 0.6664) is classified as Class2

Point x71 (value: 0.1888) is classified as Class1

Point x72 (value: 0.3048) is classified as Class1

Point x73 (value: 0.3410) is classified as Class1

Point x74 (value: 0.4311) is classified as Class2

Point x75 (value: 0.0585) is classified as Class1

Point x76 (value: 0.7908) is classified as Class2

Point x77 (value: 0.7435) is classified as Class2

Point x78 (value: 0.3186) is classified as Class1

Point x79 (value: 0.5324) is classified as Class2

Point x80 (value: 0.1231) is classified as Class1

Point x81 (value: 0.0651) is classified as Class1

Point x82 (value: 0.9023) is classified as Class2

Point x83 (value: 0.3121) is classified as Class1

Point x84 (value: 0.3951) is classified as Class1

Point x85 (value: 0.1904) is classified as Class1

Point x86 (value: 0.9808) is classified as Class2

Point x87 (value: 0.3424) is classified as Class1

Point x88 (value: 0.9225) is classified as Class2

Point x89 (value: 0.5381) is classified as Class2

Point x90 (value: 0.3974) is classified as Class1

Point x91 (value: 0.4856) is classified as Class2

Point x92 (value: 0.0586) is classified as Class1

Point x93 (value: 0.4983) is classified as Class2

Point x94 (value: 0.2392) is classified as Class1

Point x95 (value: 0.8406) is classified as Class2

Point x96 (value: 0.8007) is classified as Class2

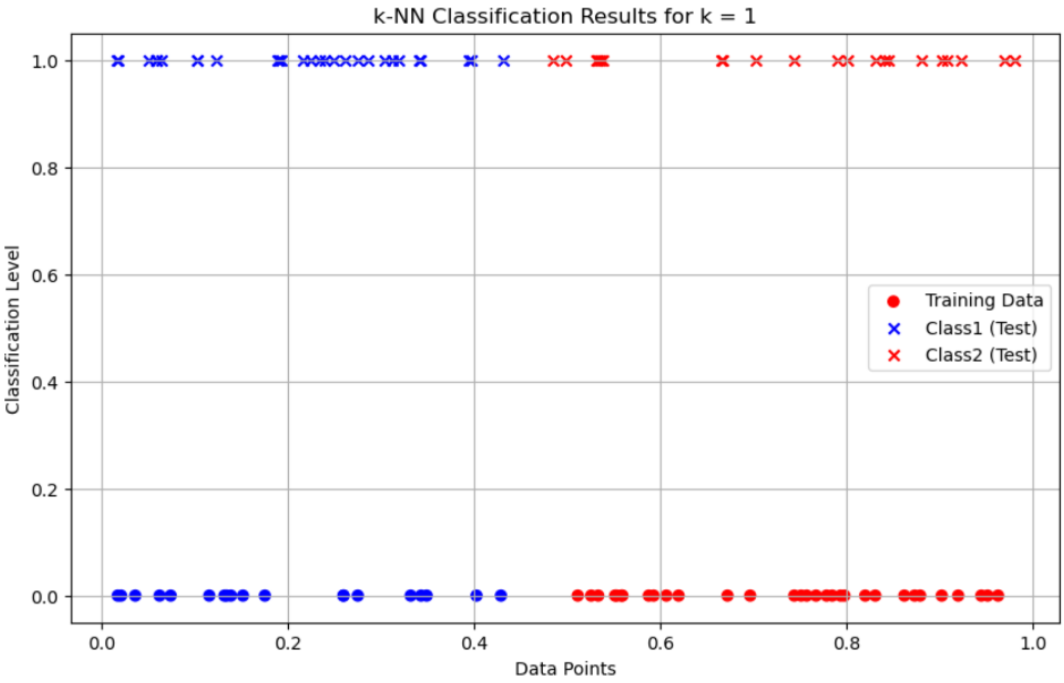
Point x97 (value: 0.2341) is classified as Class1

Point x98 (value: 0.8799) is classified as Class2

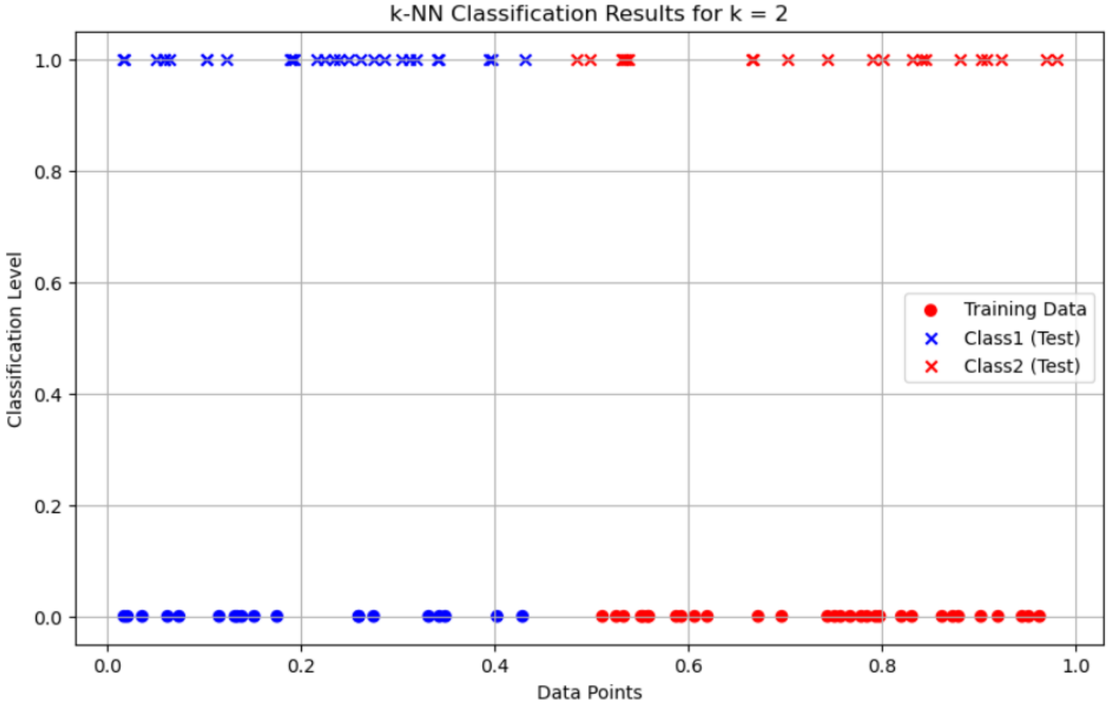
Point x99 (value: 0.2242) is classified as Class1

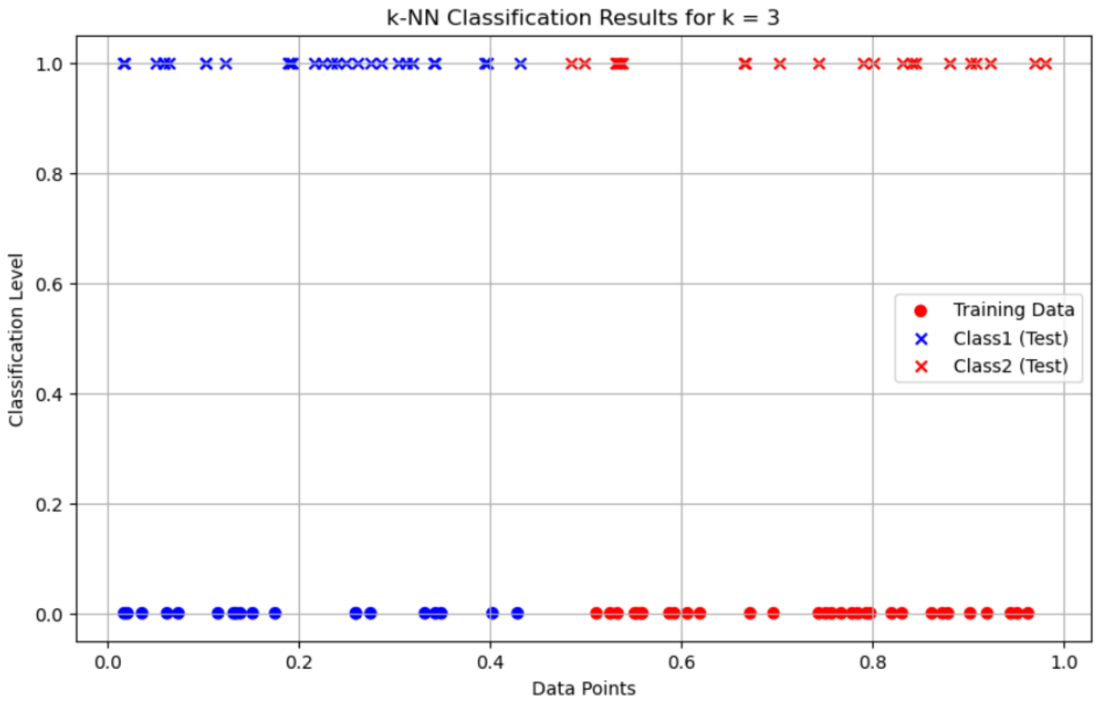
Point x100 (value: 0.9074) is classified as Class2

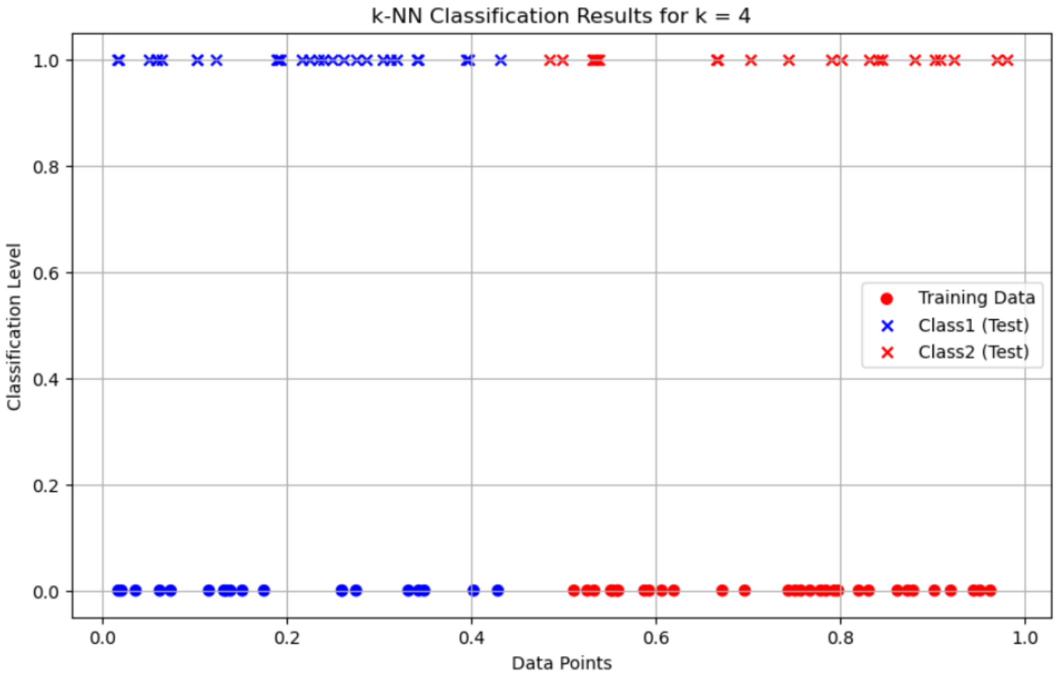
Classification complete.

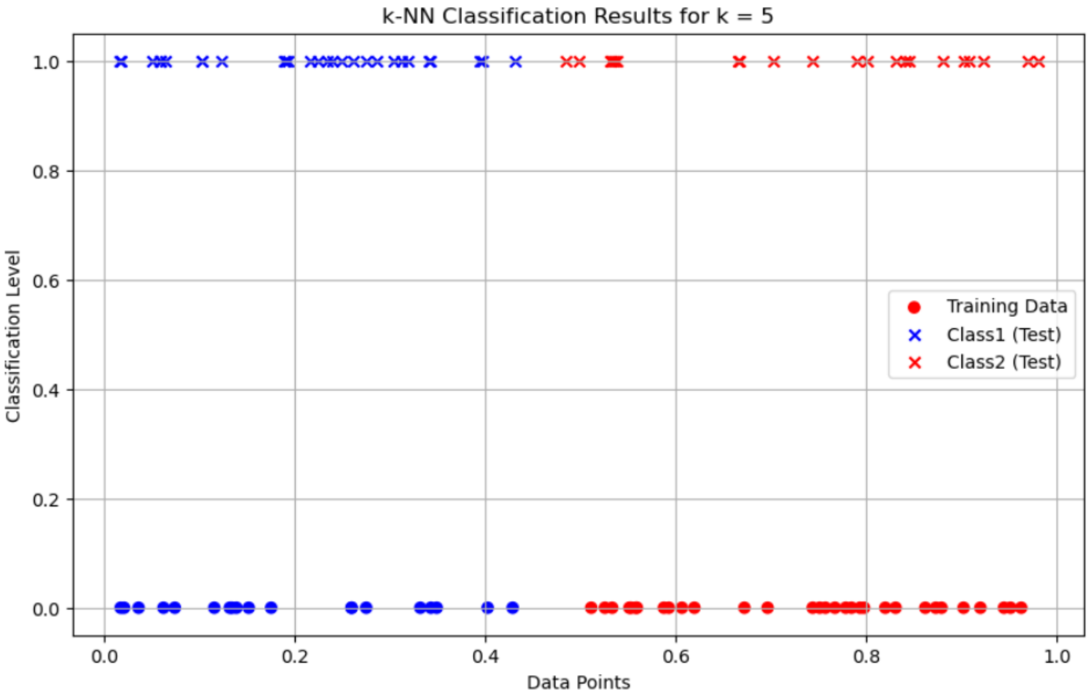


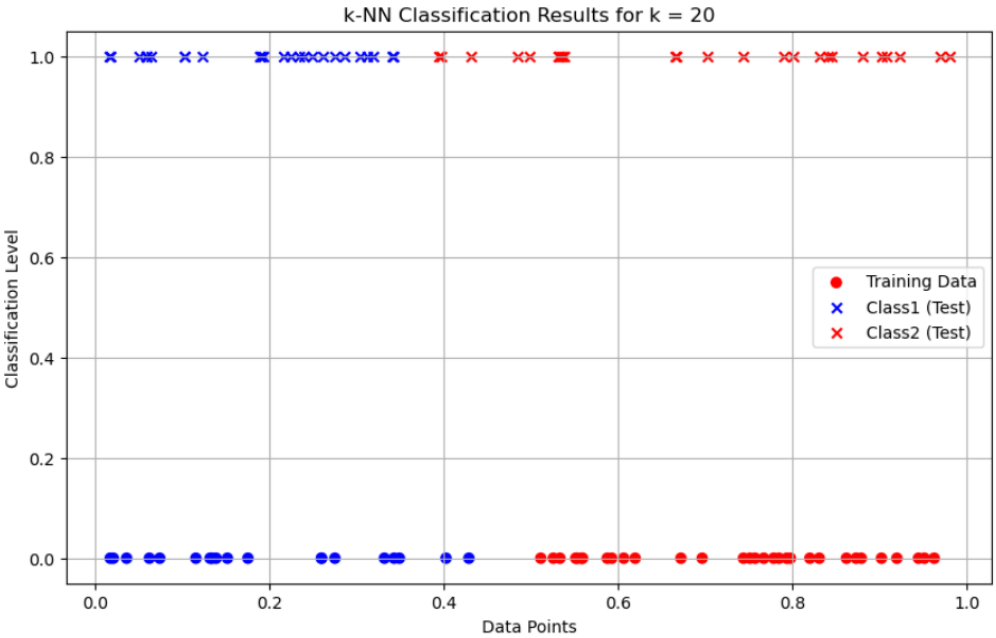
,

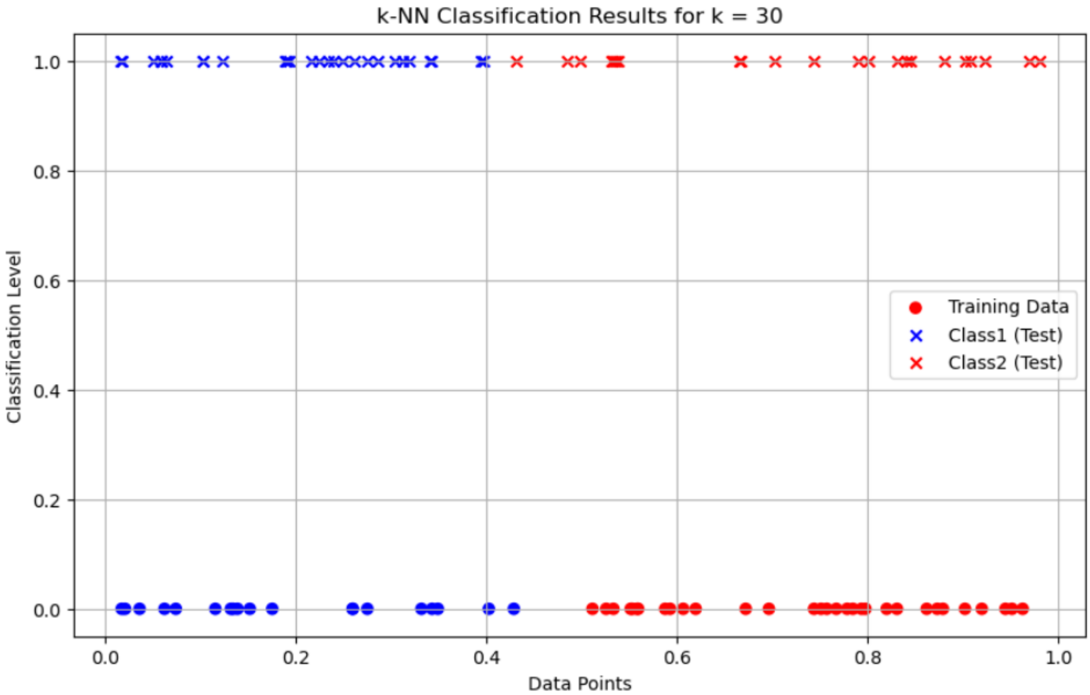

k-NN Classification Results for k = 2
1.0
0.8
0.6
0.4
0.2
0.0
Classification Level
0.0
0.2
0.4
0.6
0.8
1.0
Data Points
Training Data
Class1 (Test)
Class2 (Test)


k-NN Classification Results for k = 3
1.0
0.8
0.6
0.4
0.2
0.0
Classification Level
0.0
0.2
0.4
0.6
0.8
1.0
Data Points
Training Data
Class1 (Test)
Class2 (Test)


k-NN Classification Results for k = 4
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 5
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 20
Training Data
Class1 (Test)
Class2 (Test)
Classification Level
Data Points
0.0
0.2
0.4
0.6
0.8
1.0

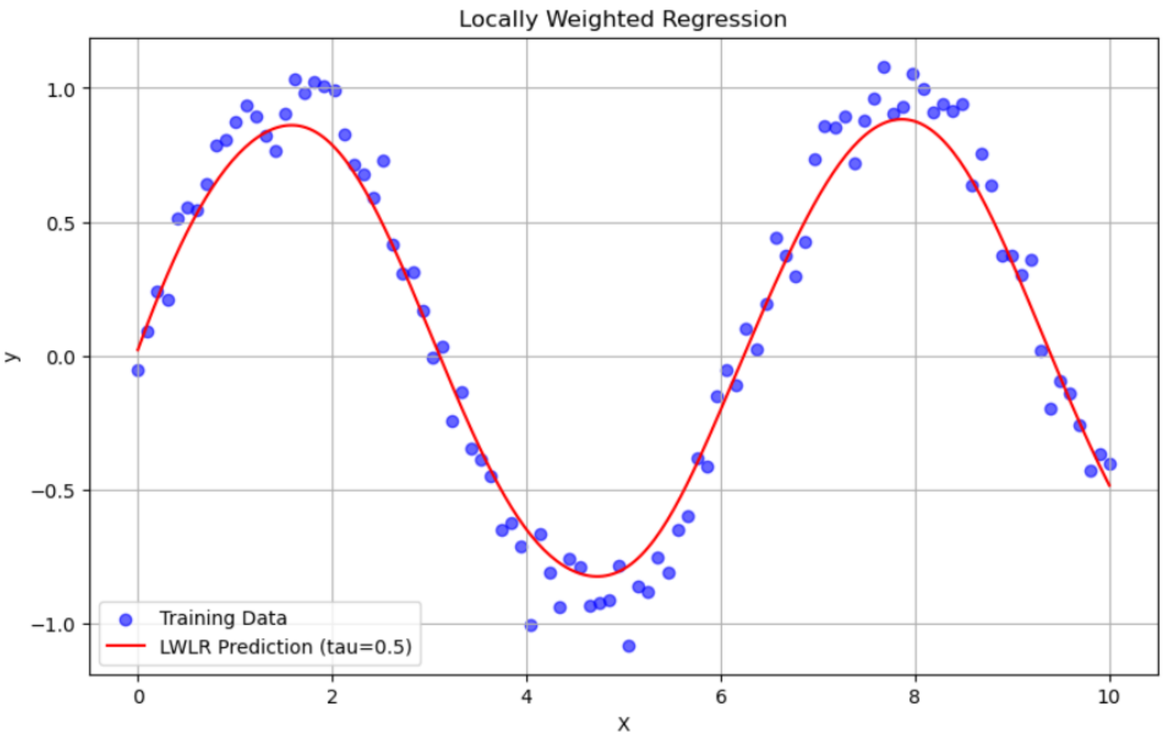

k-NN Classification Results for k = 30
Training Data
Class1 (Test)
Class2 (Test)
Classification Level
Data Points
0.0
0.2
0.4
0.6
0.8
1.0

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

```
import numpy as np

import matplotlib.pyplot as plt


# Generate synthetic data

def generate_data():

    X = np.linspace(0, 10, 100)

    y = np.sin(X) + np.random.normal(0, 0.1, X.shape[0])

    return X.reshape(-1, 1), y


# Add bias term

def add_bias(X):

    return np.hstack((np.ones((X.shape[0], 1)), X))


# Compute weights for query point x0

def compute_weights(X, x0, tau):

    m = X.shape[0]

    W = np.eye(m)

    for i in range(m):

        xi = X[i]

        W[i, i] = np.exp(-np.sum((xi - x0) ** 2) / (2 * tau ** 2))

    return W


# Locally Weighted Linear Regression

def lwlr_predict(X, y, x0, tau):

    X_bias = add_bias(X)

    x0_bias = add_bias(x0.reshape(1, -1))

    W = compute_weights(X, x0, tau)

    theta = np.linalg.pinv(X_bias.T @ W @ X_bias) @ (X_bias.T @ W @ y)
```

```
    return x0_bias @ theta

# Fit and plot
def plot_lwlr(X, y, tau):
    x_query = np.linspace(X.min(), X.max(), 300).reshape(-1, 1)
    y_pred = np.array([lwlr_predict(X, y, x0, tau) for x0 in x_query])

    plt.figure(figsize=(10, 6))
    plt.scatter(X, y, label="Training Data", color='blue', alpha=0.6)
    plt.plot(x_query, y_pred, label=f'LWLR Prediction (tau={tau})', color='red')
    plt.title("Locally Weighted Regression")
    plt.xlabel("X")
    plt.ylabel("y")
    plt.legend()
    plt.grid(True)
    plt.show()

# Run everything
X, y = generate_data()
plot_lwlr(X, y, tau=0.5)  # Try tau = 0.1, 0.5, 1, 5 for comparison
```

OUTPUT:


Locally Weighted Regression
1.0
0.5
0.0
−0.5
−1.0
y
0
2
4
6
8
10
X
Training Data
LWLR Prediction (tau=0.5)

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

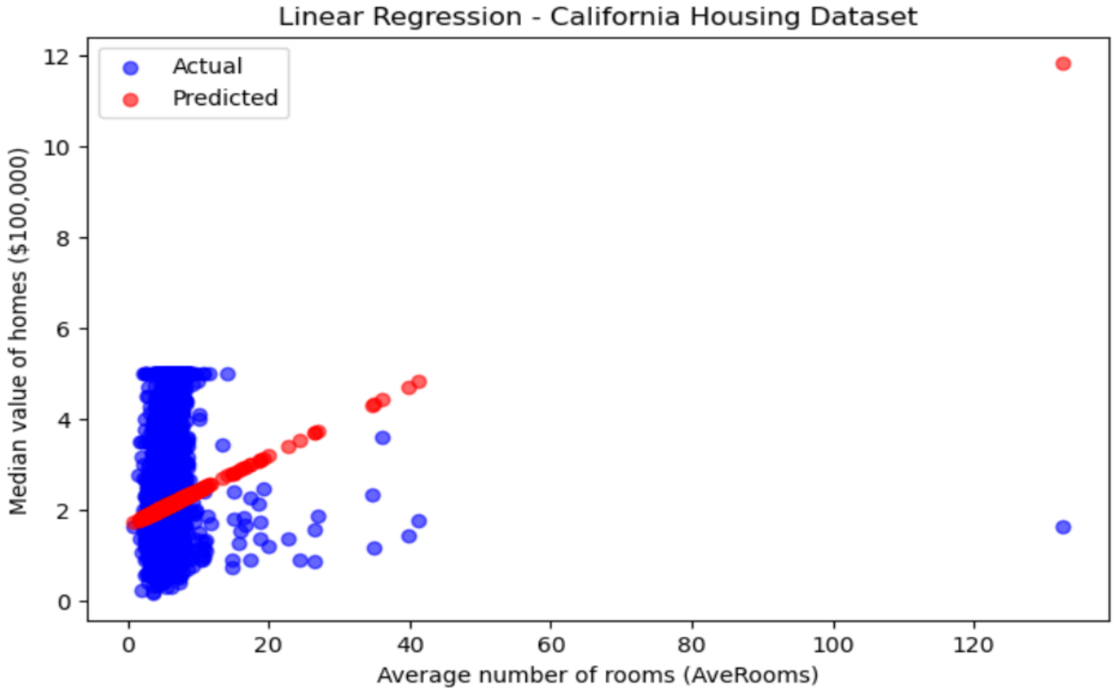
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score


def linear_regression_california():
    # Load California housing dataset
    housing = fetch_california_housing(as_frame=True)
    X = housing.data[["AveRooms"]]
    y = housing.target

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

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

```
    # Plot results
    plt.figure(figsize=(8, 5))
    plt.scatter(X_test, y_test, color="blue", label="Actual", alpha=0.6)
    plt.scatter(X_test, y_pred, color="red", label="Predicted", alpha=0.6)
    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 performance metrics
    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():
    # Load Auto MPG dataset
    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"]

    # Read dataset, handling missing values
    data = pd.read_csv(url, sep='\s+', names=column_names, na_values="?")
    data = data.dropna()

    # Prepare features and target
    X = data["displacement"].values.reshape(-1, 1)
    y = data["mpg"].values
```

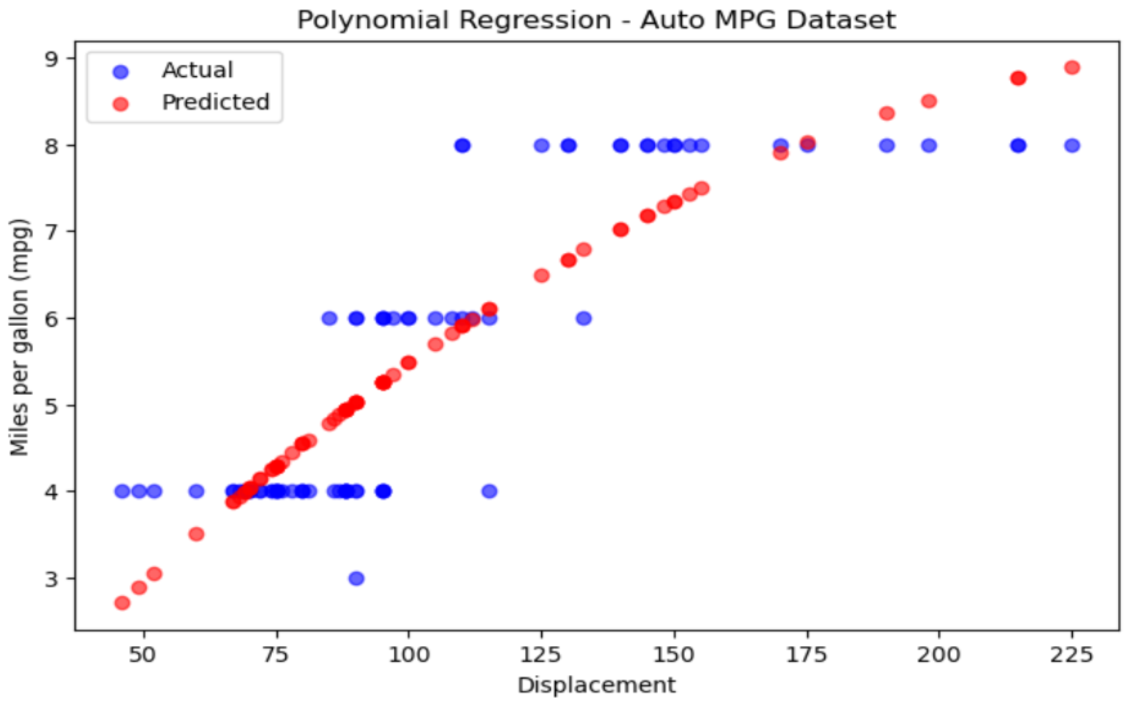
```
    # Split dataset
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train polynomial regression model
    poly_model = make_pipeline(PolynomialFeatures(degree=2), StandardScaler(), LinearRegression())
    poly_model.fit(X_train, y_train)

    # Predict
    y_pred = poly_model.predict(X_test)

    # Plot results
    plt.figure(figsize=(8, 5))
    plt.scatter(X_test, y_test, color="blue", label="Actual", alpha=0.6)
    plt.scatter(X_test, y_pred, color="red", label="Predicted", alpha=0.6)
    plt.xlabel("Displacement")
    plt.ylabel("Miles per gallon (mpg)")
    plt.title("Polynomial Regression - Auto MPG Dataset")
    plt.legend()
    plt.show()

    # Print performance metrics
    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
```

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.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

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

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=42)
clf.fit(X_train, y_train)

# Evaluate model accuracy
accuracy = clf.score(X_test, y_test)
print(f"Model Accuracy: {accuracy:.2f}")

# Print the decision tree structure
tree_rules = export_text(clf, feature_names=data.feature_names)
print("\nDecision Tree Rules:\n", tree_rules)

# Classify a new sample (Example)
```

```
new_sample = np.array([X_test[0]])  # Take a sample from the test set
prediction = clf.predict(new_sample)
print("\nPredicted Class for the New Sample:", "Malignant" if prediction[0] == 0 else "Benign")
```

OUTPUT

```
Model Accuracy: 0.95

Decision Tree Rules:
 |--- mean concave points <= 0.05
|   |--- worst radius <= 16.83
|   |   |--- area error <= 48.70
|   |   |   |--- worst smoothness <= 0.18
|   |   |   |   |--- class: 1
|   |   |   |--- worst smoothness >  0.18
|   |   |   |   |--- class: 0
|   |   |--- area error >  48.70
|   |   |   |--- texture error <= 1.93
|   |   |   |   |--- class: 1
|   |   |   |--- texture error >  1.93
|   |   |   |   |--- class: 0
|   |--- worst radius >  16.83
|   |   |--- worst texture <= 19.91
|   |   |   |--- class: 1
|   |   |--- worst texture >  19.91
|   |   |   |--- concave points error <= 0.01
|   |   |   |   |--- class: 0
|   |   |   |--- concave points error >  0.01
|   |   |   |   |--- class: 1
|--- mean concave points >  0.05
|   |--- worst concave points <= 0.15
|   |   |--- worst perimeter <= 115.25
|   |   |   |--- mean texture <= 21.06
|   |   |   |   |--- class: 1
|   |   |   |--- mean texture >  21.06
|   |   |   |   |--- class: 0
|   |   |--- worst perimeter >  115.25
|   |   |   |--- class: 0
|   |--- worst concave points >  0.15
|   |   |--- concavity error <= 0.14
|   |   |   |--- class: 0
|   |   |--- concavity error >  0.14
|   |   |   |--- class: 1


Predicted Class for the New Sample: Benign
```

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
import matplotlib.pyplot as plt
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

# Load the Olivetti Faces dataset
data = fetch_olivetti_faces(shuffle=True, random_state=42)

# Extract features and labels
X = data.data
y = data.target

# Split dataset into training and testing sets (70% training, 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train a Gaussian Naïve Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions
y_pred = gnb.predict(X_test)

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

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

# Print confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

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

# Visualize sample images with predicted labels
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

```
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
```

```
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

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

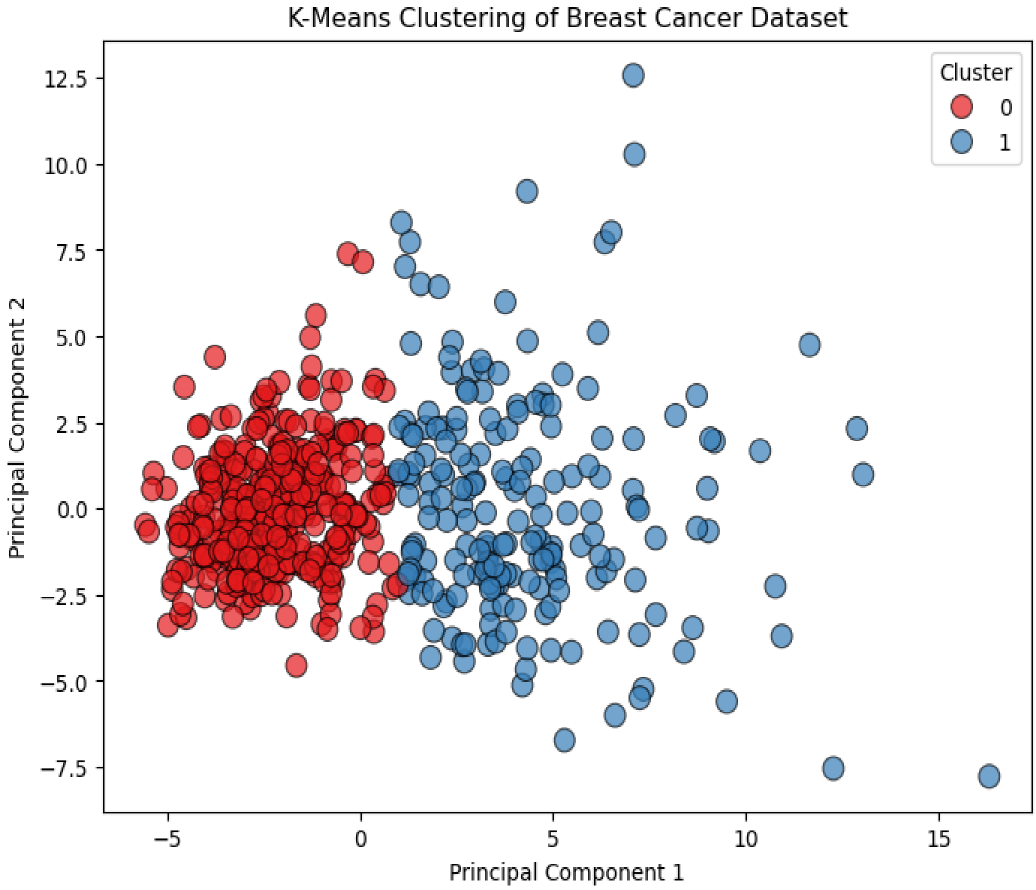
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

# Load the dataset
data = load_breast_cancer()

X = data.data
y = data.target

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
y_kmeans = kmeans.fit_predict(X_scaled)

# Evaluate clustering performance
print("Confusion Matrix:")
print(confusion_matrix(y, y_kmeans))
print("\nClassification Report:")
```

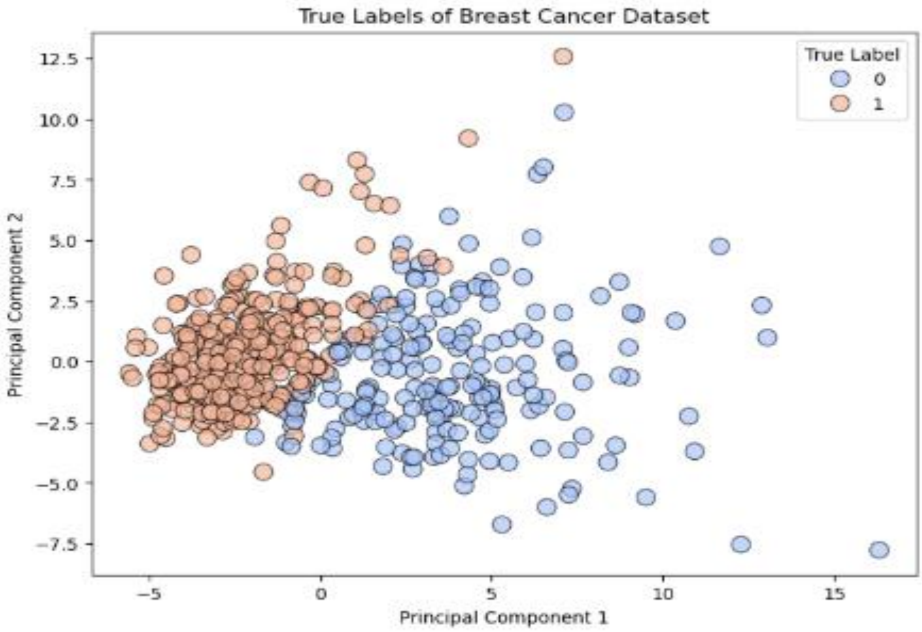
```
print(classification_report(y, y_kmeans))

# Apply PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

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

# Plot K-Means Clustering results
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()

# Plot the true labels
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()
```

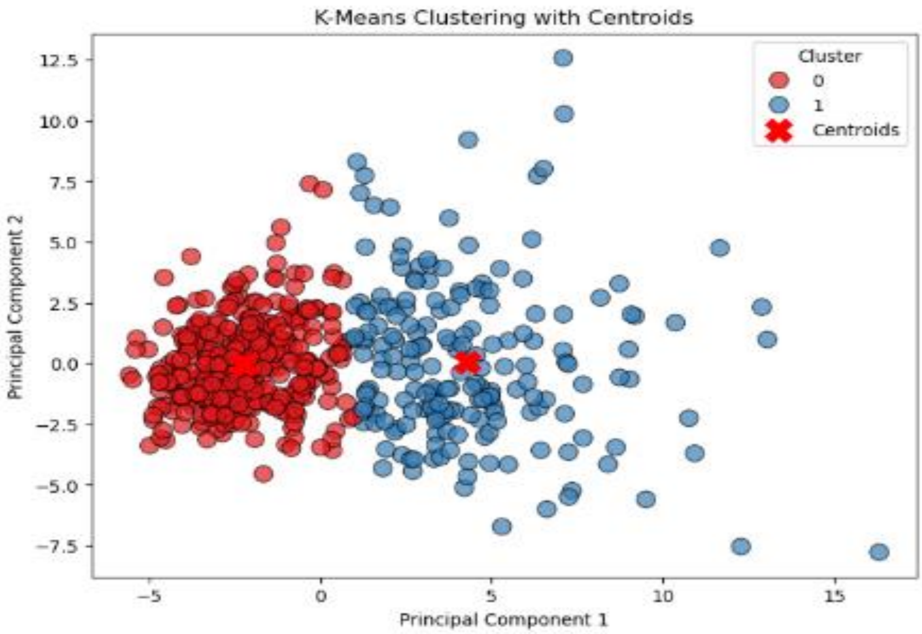
```
# Plot K-Means Clustering with Centroids
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100,
edgecolor='black', alpha=0.7)

# Project centroids onto PCA space
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?

Answer: Machine Learning is a subset of Artificial Intelligence that enables systems to learn from

data, identify patterns, and make decisions with minimal human intervention.

2. What are the types of Machine Learning?

Answer: 1. Supervised Learning

2. Unsupervised Learning

3. Reinforcement Learning

3. What is overfitting and underfitting?

Answer: Overfitting: Model performs well on training data but poorly on test data.

Underfitting: Model performs poorly on both training and test data.

4. What is the difference between classification and regression?

Answer: Classification: Predicts categorical labels.

Regression: Predicts continuous values.

5. What is the use of the train-test split?

Answer: It helps evaluate model performance by separating data into training and testing sets.

6. What is the role of feature scaling?

Answer: Ensures all features contribute equally to the model by normalizing or standardizing data.

7. What is confusion matrix?

Answer: A matrix used to evaluate classification models using TP, FP, TN, and FN.

8. Explain KNN Algorithm.

Answer: KNN classifies data based on the majority label of the K nearest data points using a

distance metric.

9. What is cross-validation?

Answer: A technique to evaluate model performance by testing it on multiple data subsets.

10. What is the difference between bagging and boosting?

Answer: Bagging: Trains multiple models independently.

Boosting: Trains models sequentially to correct previous errors.

11. What is regularization?

Answer: Adds a penalty to the loss function to reduce overfitting.

L1 (Lasso), L2 (Ridge) are common types.

12. What is precision, recall, and F1-score?

Answer: Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1-score = Harmonic mean of Precision and Recall.

13. What is dimensionality reduction?

Answer: It reduces the number of input variables using techniques like PCA.

14. Name a few common ML libraries in Python.

Answer: scikit-learn, Pandas, NumPy, TensorFlow, Keras, Matplotlib

15. What is the purpose of a cost/loss function?

Answer: It measures the difference between predicted and actual output. The goal is to minimize it.

16. Explain the working of Decision Tree algorithm.

Answer: It splits data into subsets based on feature values, forming a tree structure to make decisions.

17. What is entropy in decision trees?

Answer: Entropy is a measure of impurity or randomness used to decide data splits.

18. What is gradient descent?

Answer: An optimization algorithm used to minimize the cost function by updating weights iteratively.

19. What is the difference between parametric and non-parametric models?

Answer: Parametric models assume a fixed number of parameters. Non-parametric models do not.

20. Explain the Naïve Bayes algorithm.

Answer: A probabilistic classifier based on Bayes' theorem with an assumption of feature independence.

21. What is a ROC curve?

Answer: A graphical plot showing the diagnostic ability of a binary classifier system.

22. What is a support vector machine (SVM)?

Answer: A supervised ML algorithm that finds the optimal hyperplane to classify data.

23. What are hyperparameters?

Answer: Settings or configurations that are set before training the model (e.g., learning rate, depth).

24. What is model evaluation?

Answer: The process of assessing how well a model performs using metrics like accuracy, precision, recall.

25. What is the curse of dimensionality?

Answer: The problem of increased data sparsity and computation as the number of features increases