VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. ROAD, V.V PURAM, BANGALORE – 560 004

DEPARTMENT OF
INFORMATION SCIENCE AND ENGINEERING

SUBJECT CODE: BCSL606

MACHINE LEARNING LABORATORY MANUAL

As per Choice Based Credit System Scheme (CBCS)

FOR VI SEMESTER CSE/ISE AS PRESCRIBED BY VTU

Effective from the Academic year 2024-2025

Prepared By:

Dr. H Roopa	Dr. Jayasheela C S
Professor	Associate Professor
Dept. of ISE, BIT	Dept. of ISE, BIT

Prerequisites:

1. Computer Concepts
2. Probability and Statistics
3. Operating Systems
4. Design and Analysis of Algorithm

Course Objectives:

This course will enable students to:

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

Course Outcomes:

VI-SEM:	**Course Name: Machine Learning Lab**
Course Code: BCSL606	**Year of Study: 2024-25**
At the end of the course the students should be able to:	
CO1	**Apply** Machine learning algorithms like PCA, Find-S, k-Nearest Neighbor, Locally Weighted Regression, Linear Regression and Polynomial Regression, decision tree, Naive Bayesian classifier and k-**means** clustering for any given data sets for prediction.
CO2	**Analyze** various machine learning models on datasets to gain the knowledge of performance of each model.

COs and POs Mapping													**CO-PSO Mapping**	
VI-SEM: **Course Code: BCSL606**							**Course Name: Machine Learning Lab** **Year of Study: 2024-25**							
COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
CO1	3	2											**2**	**2**
CO2	3	2											**2**	**2**
AVG	**3**	**2**											**2**	**2**

RESOURSES REQUIRED:

- Hardware resources
  - Desktop PC
  - Windows / Linux operating system
- Software resources
  - Python
  - Anaconda IDE
- Datasets from standard repositories (Ex: https://archive.ics.uci.edu/ml/datasets.php)

REFERENCE:

1. S Sridhar and M Vijayalakshmi, "Machine Learning", Oxford University Press, 2021.
2. M N Murty and Ananthanarayana V S, "Machine Learning: Theory and Practice", Universities Press (India) Pvt. Limited, 2024.

WEB LINKS AND VIDEO LECTURES (E-RESOURCES):

1. https://www.drssridhar.com/?page_id=1053
2. https://www.universitiespress.com/resources?id=9789393330697
3. https://onlinecourses.nptel.ac.in/noc23_cs18/preview

C. EVALUATION SCHEME

For CBCS 2022s scheme:

Conduction of Practical Examination:
1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution: Procedure + Conduction + Viva:**20 + 60 +20 (100)**
Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

MACHINE LEARNING LABORATORY **(Effective from the academic year 2024 -2025)** **SEMESTER – VI**			
Course Code	BCSL606		
Teaching Hours/Week (L:T:P: S)	0:0:2:0	**CIE Marks**	50
Credits	01	**SEE Marks**	50
Total Number of Contact Hours	24	**Exam Hours**	03
Examination type (SEE)	Practical		

Course Objectives:

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

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. 1. Label the first 50 points {x1,……,x50} as follows: if (xi ≤ 0.5), then xi ε Class1, else xi ε Class1 2. 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 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.

Laboratory Outcomes: The student should be able to:

- Implement and demonstrate ML algorithms.
- Evaluate different algorithms.

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.

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedules mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted

jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.

The minimum duration of SEE is 02 hours.

Rubrics of Machine Learning Lab

1. Daily conduction

Sl. No	Name of the Experiment	Date of Execution	Date of Submission	Marks(30)				Total 30 Marks	Staff Signature
				Write-Up & Implementation 6 Marks	Analysis & Execution 8 Marks	Results & Tabulation 6 Marks	Record 10 Marks		

2. Lab Internal Marks

	Write-up & Implementation (10Marks)	**Analysis & Execution** (20Marks)	**Results & Tabulation** (10Marks)	**Viva** (10Marks)	**Total 50 Marks**
Test-1					
Test-2					
	T1+T2 (100 Marks scaled down 20 Marks)			**20**	

3. Final Lab Marks

	Max Marks	**Marks scored**
Daily Conduction	30	
Lab Internals	20	
TOTAL	**50**	
Signature of the Faculty		

1. EXPERIMENT NO: 1

2. TITLE: The California housing dataset

3. LEARNING OBJECTIVES:

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

5. THEORY

California Housing dataset

**Data Set Characteristics: **

Number of Instances: 20640

Number of Attributes: 8 numeric, predictive attributes and the target

1. Longitude: A measure of how far west a house is; a higher value is farther west
2. Latitude: A measure of how far north a house is; a higher value is farther north
3. Housing Median Age: Median age of a house within a block; a lower number is a newer building
4. Total Rooms: Total number of rooms within a block
5. Total Bedrooms: Total number of bedrooms within a block
6. Population: Total number of people residing within a block
7. Households: Total number of households, a group of people residing within a home unit, for a block
8. Median Income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
9. Median House Value: Median house value for households within a block (measured in US Dollars)
10. Ocean Proximity: Location of the house w.r.t ocean/sea

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surprisingly large values for block groups with few households and many empty houses, such as vacation resorts.

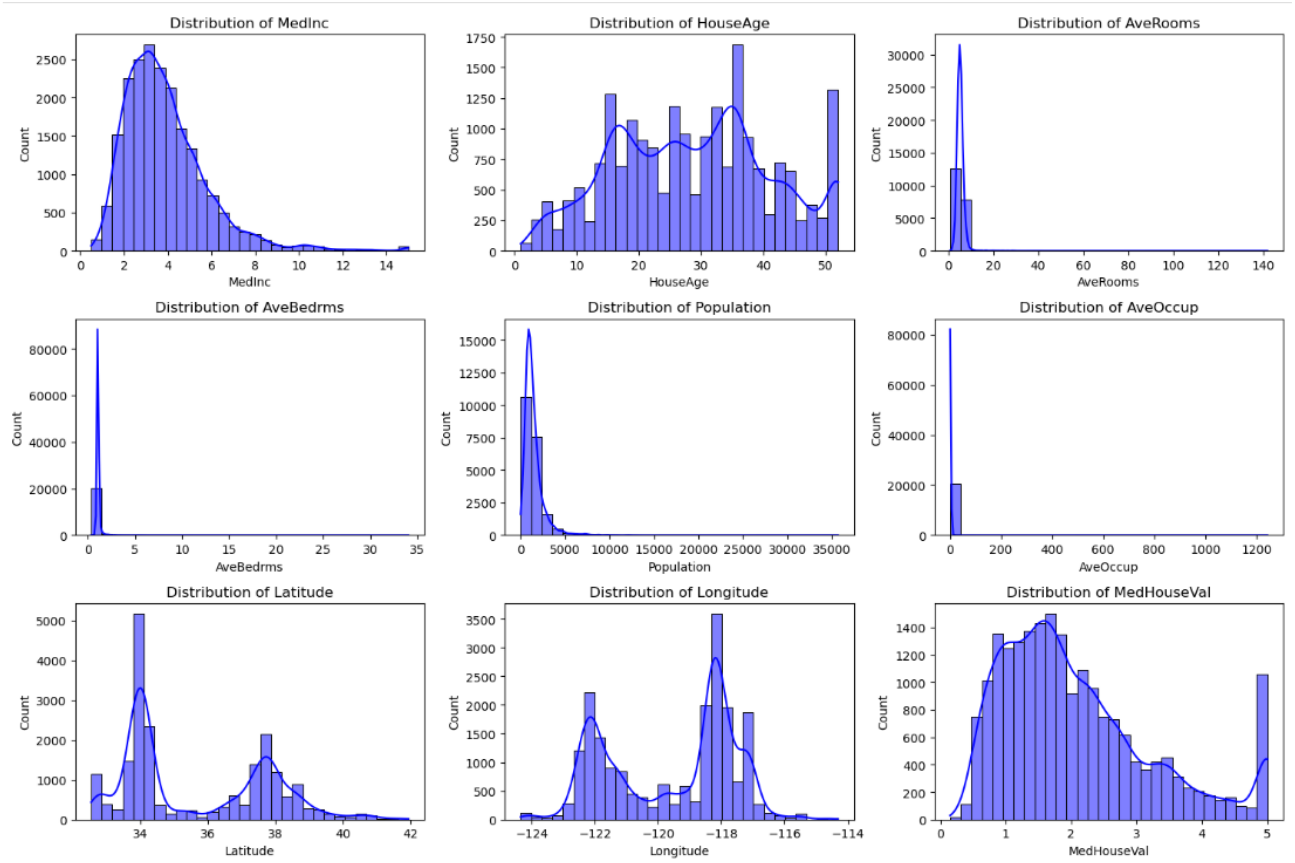
It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

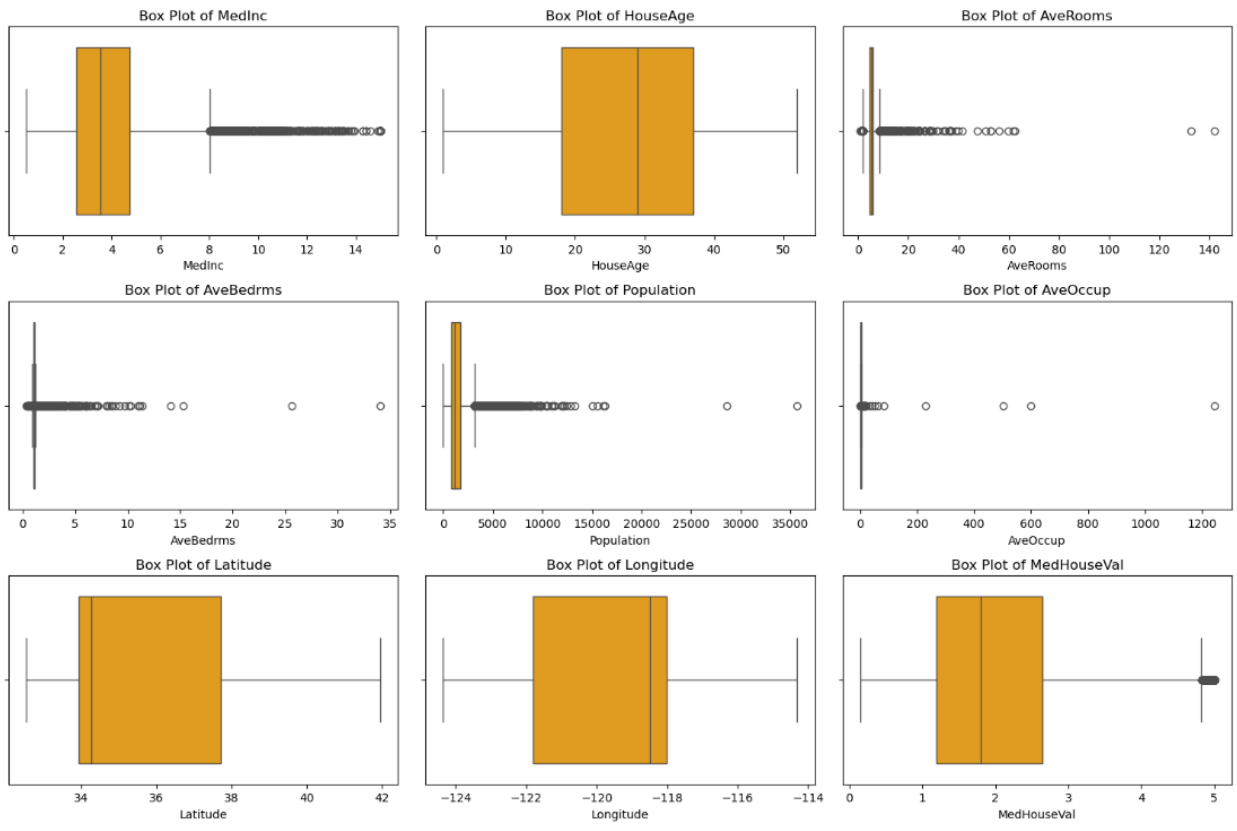
6. PROCEDURE / PROGRAMME:

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

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

# Step 2: Create histograms for numerical features
numerical_features =
housing_df.select_dtypes(include=[np.number]).columns

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

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

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

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

OUTPUT:


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


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
```

1. EXPERIMENT NO: 2

2. TITLE: correlation matrix to understand the relationships between pairs of features

3. LEARNING OBJECTIVES:

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

5. THEORY

California Housing dataset

**Data Set Characteristics: **

Number of Instances: 20640

Number of Attributes: 8 numeric, predictive attributes and the target

1. Longitude: A measure of how far west a house is; a higher value is farther west
2. Latitude: A measure of how far north a house is; a higher value is farther north
3. Housing Median Age: Median age of a house within a block; a lower number is a newer building
4. Total Rooms: Total number of rooms within a block
5. Total Bedrooms: Total number of bedrooms within a block
6. Population: Total number of people residing within a block
7. Households: Total number of households, a group of people residing within a home unit, for a block
8. Median Income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
9. Median House Value: Median house value for households within a block (measured in US Dollars)
10. Ocean Proximity: Location of the house w.r.t ocean/sea

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

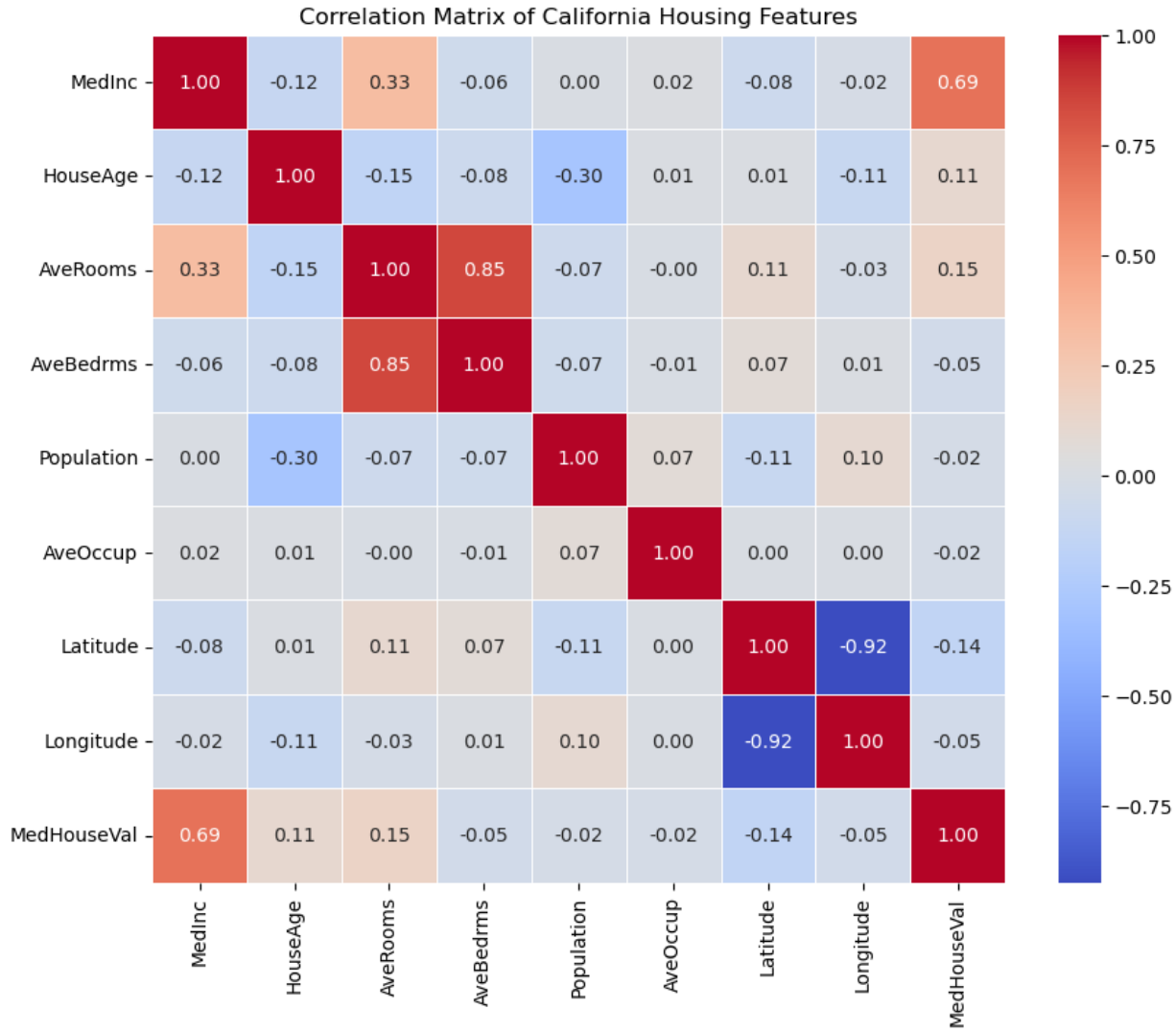
A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surprisingly large values for block groups with few households and many empty houses, such as vacation resorts.

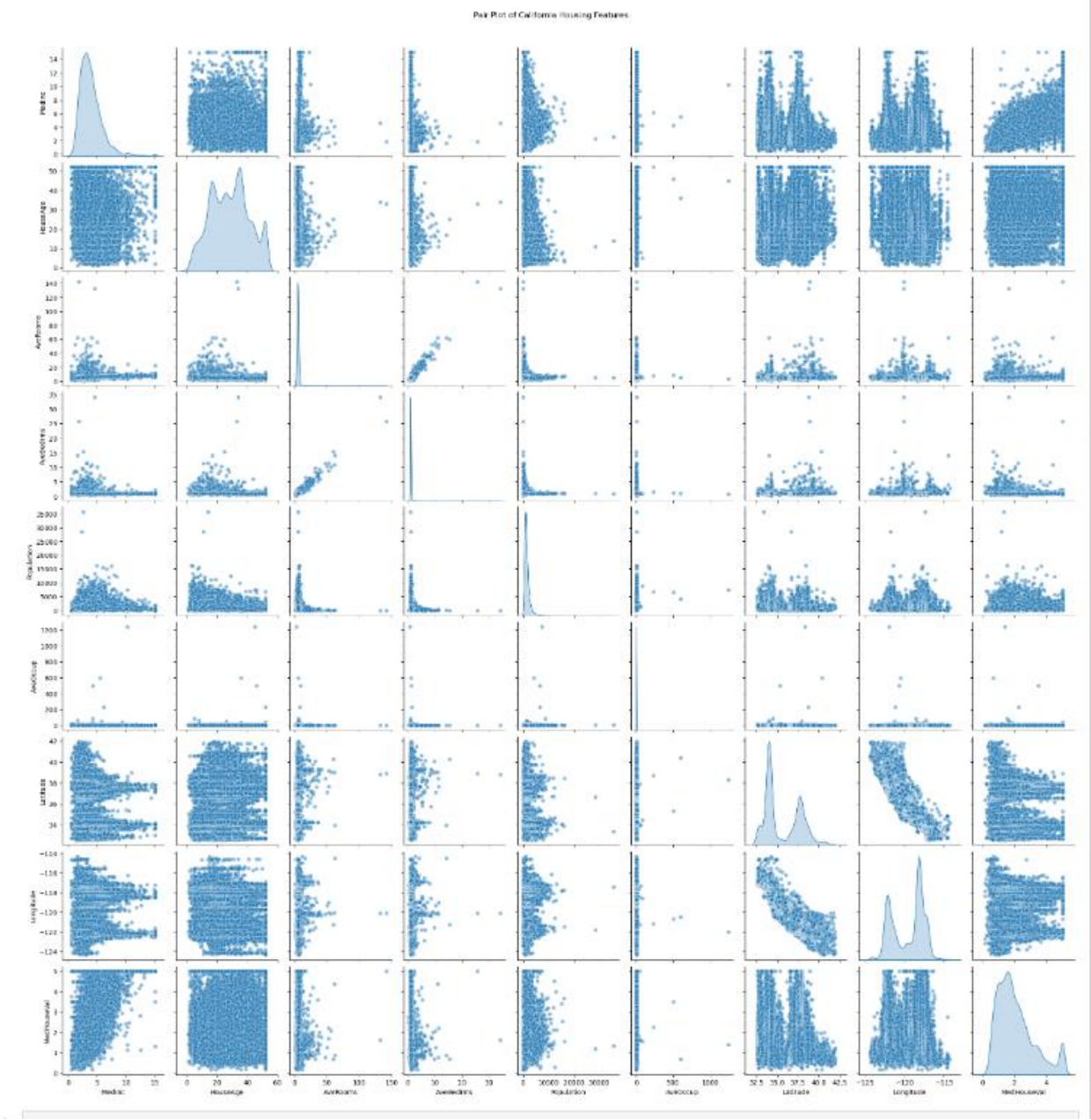
It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

6. PROCEDURE / PROGRAMME:

```
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


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

1. EXPERIMENT NO: 3

2. TITLE: Reduce Data Dimensionality using PCA – Python

3. LEARNING OBJECTIVES:

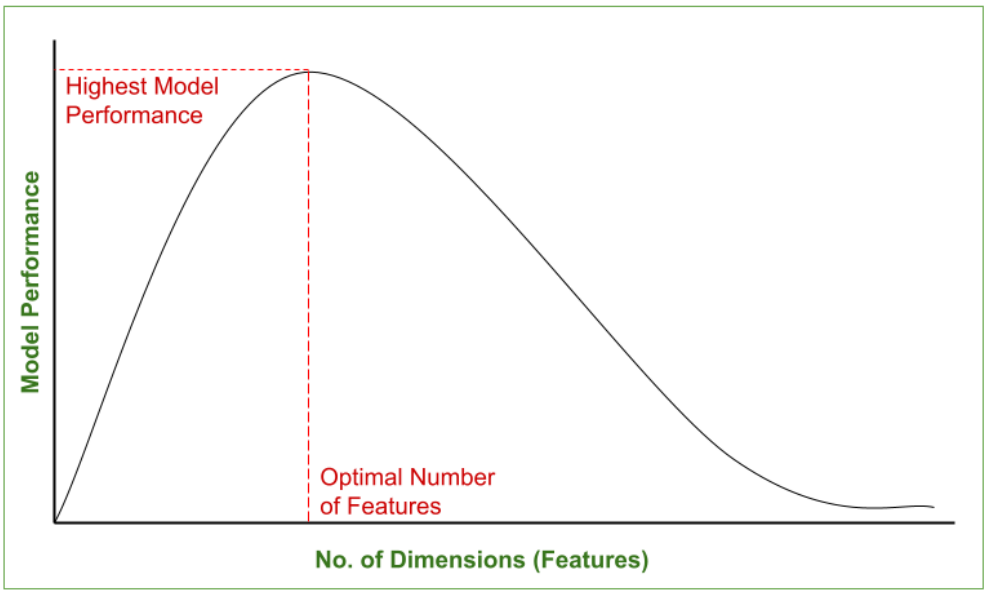
4. AIM: Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.

5. THEORY

Reduce Data Dimensionality using PCA – Python

The advancements in Data Science and Machine Learning have made it possible for us to solve several complex regression and classification problems. However, the performance of all these ML models depends on the data fed to them. Thus, it is imperative that we provide our ML models with an optimal dataset. Now, one might think that the more data we provide to our model, the better it becomes – however, it is not the case. If we feed our model with an excessively large dataset (with a large no. of features/columns), it gives rise to the problem of overfitting, wherein the model starts getting influenced by outlier values and noise. This is called the Curse of Dimensionality.

The following graph represents the change in model performance with the increase in the number of dimensions of the dataset. It can be observed that the model performance is best only at an option dimension, beyond which it starts decreasing.



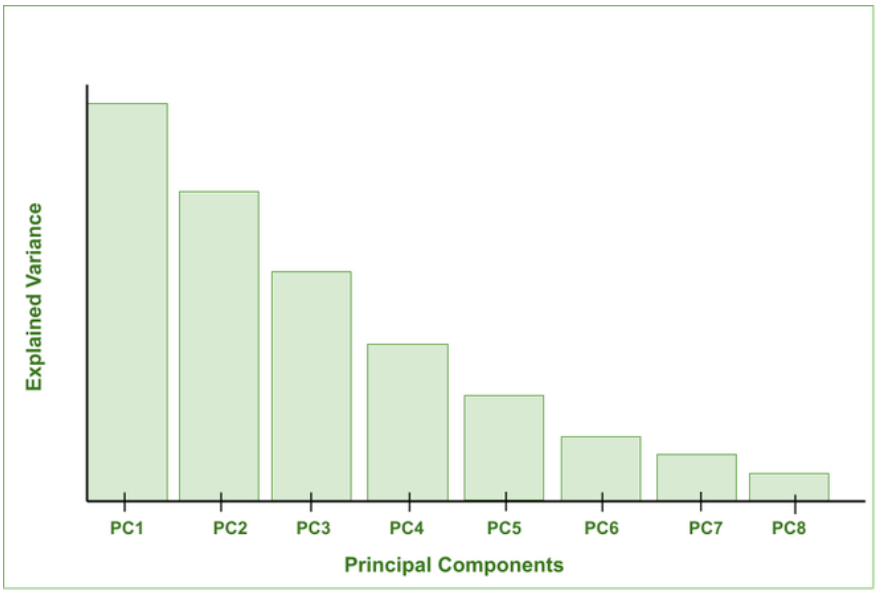
Dimensionality Reduction is a statistical/ML-based technique wherein we try to reduce the number of features in our dataset and obtain a dataset with an optimal number of dimensions.

One of the most common ways to accomplish Dimensionality Reduction is Feature Extraction, wherein we reduce the number of dimensions by mapping a higher dimensional feature space to a lower-dimensional feature space. The most popular technique of Feature Extraction is Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

As stated earlier, Principal Component Analysis is a technique of feature extraction that maps a higher dimensional feature space to a lower-dimensional feature space. While reducing the number of dimensions, PCA ensures that maximum information of the original dataset is retained in the dataset with the reduced no. of dimensions and the co-relation between the newly obtained Principal Components is minimum. The new features obtained after applying PCA are called Principal Components and are denoted as PCi (i=1,2,3…n). Here, (Principal Component-1) PC1 captures the maximum information of the original dataset, followed by PC2, then PC3 and so on.

The following bar graph depicts the amount of Explained Variance captured by various Principal Components. (The Explained Variance defines the amount of information captured by the Principal Components).

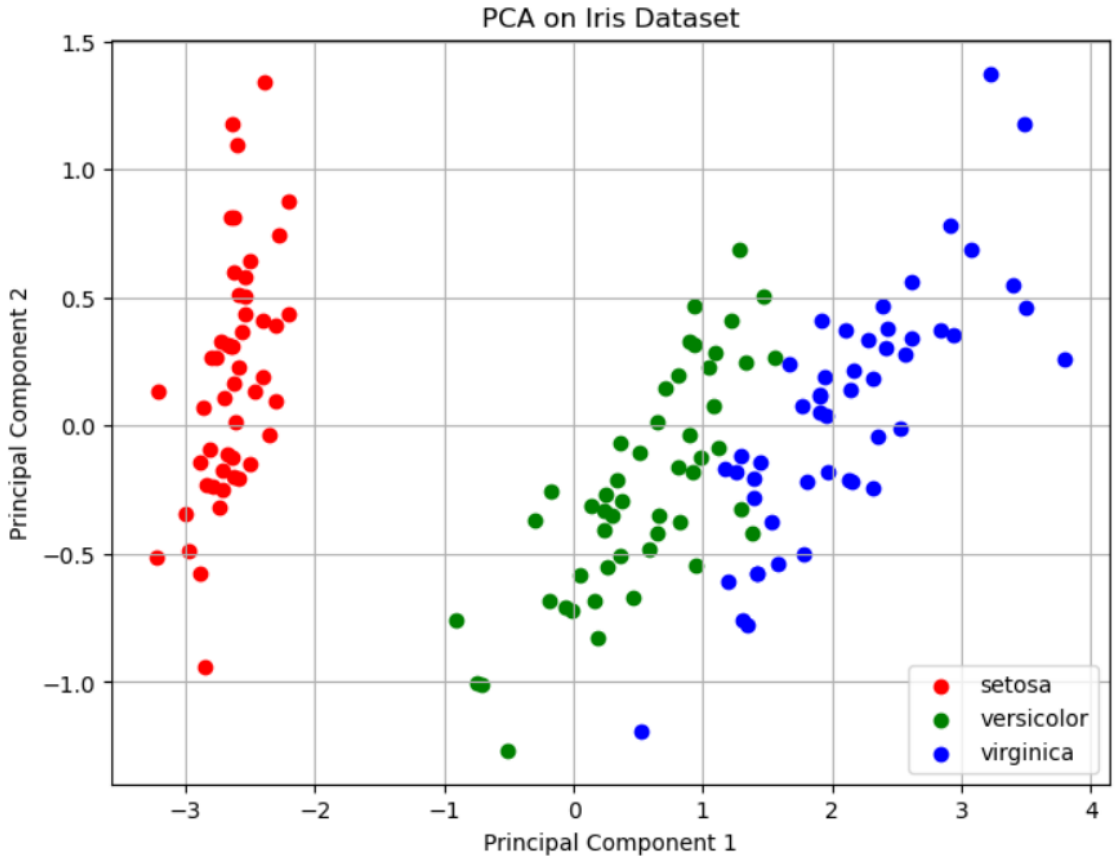


6. PROCEDURE / PROGRAMME :

```
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 2
Principal Component 1
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4
setosa
versicolor
virginica

1. EXPERIMENT NO: 4

2. TITLE: **FIND-S ALGORITHM**

3. LEARNING OBJECTIVES:
   a. Make use of Data sets in implementing the machine learning algorithms.
   b. Implement ML concepts and algorithms in Python

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

5. THEORY:

- The concept learning approach in machine learning, can be formulated as "Problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples".
- Find-S algorithm for concept learning is one of the most basic algorithms of machine learning.

Find-S Algorithm

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   For each attribute constraint a i in h :
   If the constraint a i in h is satisfied by x then do nothing
   Else replace a i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

- It is Guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.
- Also Notice that negative examples are ignored.

Limitations of the Find-S algorithm:

- No way to determine if the only final hypothesis (found by Find-S) is consistent with data or there are more hypothesis that is consistent with data.
- Inconsistent sets of training data can mislead the finds algorithm as it ignores negative data samples.
- A good concept learning algorithm should be able to backtrack the choice of hypothesis found so that the resulting hypothesis can be improved over time. Unfortunately, Find-S provide no such method.

DATA SETS

sky	airTemp	humidity	wind	water	forecast	enjoySport
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

6. PROCEDURE / PROGRAMME :

```
import numpy as np
import pandas as pd
data=pd.read_csv('finds.csv')
print('Data',data)
def train(concepts,target):
        specific_h=concepts[0]
        print('\nspecific1\n',specific_h)
        for i,h in enumerate(concepts):
            print('i',i)
            print('h',h)
            if target[i]=="Yes":
                for x in range(len(specific_h)):
                    print('x',x)
                    print('specific',specific_h)
                    if h[x]==specific_h[x]:
                        pass
                    else:
                        specific_h[x]="?"
        return specific_h
concepts=np.array(data.iloc[:,0:-1])
target=np.array(data.iloc[:,-1])
print('\nConcept\n',concepts)
print('Target',target)
print(train(concepts,target))
```

OUTPUT:

```
Data      sky airTemp humidity    wind water forecast enjoySport
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
Concept
 [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
```

```
Target ['Yes' 'Yes' 'No' 'Yes']

specific1
 ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
i 0
h ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 0
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 1
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 2
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 3
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 4
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 5
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
i 1
h ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
x 0
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 1
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 2
specific ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
x 3
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 4
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 5
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
i 2
h ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
i 3
h ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']
x 0
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 1
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 2
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 3
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 4
specific ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
x 5
specific ['Sunny' 'Warm' '?' 'Strong' '?' 'Same']
['Sunny' 'Warm' '?' 'Strong' '?' '?']
```

1. EXPERIMENT NO: 5

2. TITLE: **k-Nearest Neighbour algorithm**

3. LEARNING OBJECTIVES:

4. AIM: Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of x in the range of [0,1]. Perform the following based on dataset generated.
    1. Label the first 50 points {x1,……,x50} as follows: if (xi ≤ 0.5), then xi ε Class1, else xi ε Class1
    2. Classify the remaining points, x51,……,x100 using KNN. Perform this for k=1,2,3,4,5,20,30

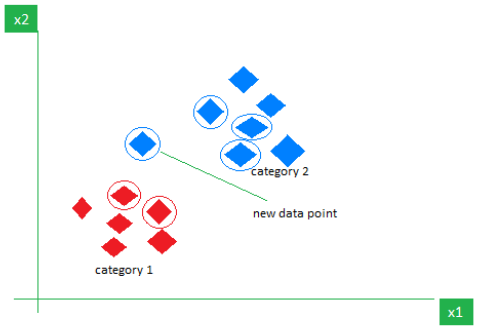
5. THEORY: K-Nearest Neighbor(KNN) Algorithm

K-Nearest Neighbors (KNN) is a simple way to classify things by looking at what's nearby. Imagine a streaming service wants to predict if a new user is likely to cancel their subscription (churn) based on their age. They checks the ages of its existing users and whether they churned or stayed. If most of the "K" closest users in age of new user canceled their subscription KNN will predict the new user might churn too. The key idea is that users with similar ages tend to have similar behaviors and KNN uses this closeness to make decisions.

Getting Started with K-Nearest Neighbors

K-Nearest Neighbors is also called as a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification it performs an action on the dataset.

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



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

The image shows how KNN predicts the category of a new data point based on its closest neighbours.

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

KNN works by using proximity and majority voting to make predictions.

What is 'K' in K Nearest Neighbour ?

In the k-Nearest Neighbours (k-NN) algorithm k is just a number that tells the algorithm how many nearby points (neighbours) to look at when it makes a decision.

Example:

Imagine you're deciding which fruit it is based on its shape and size. You compare it to fruits you already know.

- If k = 3, the algorithm looks at the 3 closest fruits to the new one.
- If 2 of those 3 fruits are apples and 1 is a banana, the algorithm says the new fruit is an apple because most of its neighbours are apples.

How to choose the value of k for KNN Algorithm?

The value of k is critical in KNN as it determines the number of neighbors to consider when making predictions. Selecting the optimal value of k depends on the characteristics of the input data. If the dataset has significant outliers or noise a higher k can help smooth out the predictions and reduce the influence of noisy data. However choosing very high value can lead to underfitting where the model becomes too simplistic.

Statistical Methods for Selecting k:

- Cross-Validation: A robust method for selecting the best k is to perform k-fold cross-validation. This involves splitting the data into k subsets training the model on some subsets and testing it on the remaining ones and repeating this for each subset. The value of k that results in the highest average validation accuracy is usually the best choice.
- Elbow Method: In the elbow method we plot the model's error rate or accuracy for different values of k. As we increase k the error usually decreases initially. However after a certain point the error rate starts to decrease more slowly. This point where the curve forms an "elbow" that point is considered as best k.
- Odd Values for k: It's also recommended to choose an odd value for k especially in classification tasks to avoid ties when deciding the majority class.

Distance Metrics Used in KNN Algorithm

KNN uses distance metrics to identify nearest neighbour, these neighbours are used for classification and regression task.

To identify nearest neighbour we use below distance metrics:

1. **Euclidean Distance**
   Euclidean distance is defined as the straight-line distance between two points in a plane or space. You can think of it like the shortest path you would walk if you were to go directly from one point to another.

   distance(x,Xi)=∑j=1d(xj–Xij)2]

2. **Manhattan Distance**
   This is the total distance you would travel if you could only move along horizontal and vertical lines (like a grid or city streets). It's also called "taxicab distance" because a taxi can only drive along the grid-like streets of a city.

   $d(x,y)=\sum i=1n|xi-yi|$

3. **Minkowski Distance**
   Minkowski distance is like a family of distances, which includes both Euclidean and Manhattan distances as special cases.

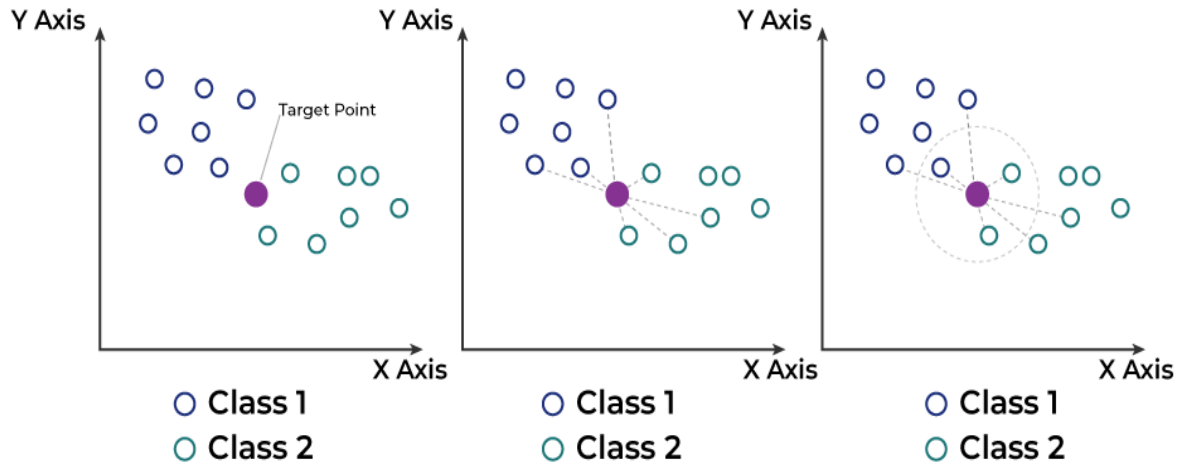
   $d(x,y)=(\sum i=1n(xi-yi)p)p1$

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

So, you can think of Minkowski as a flexible distance formula that can look like either Manhattan or Euclidean distance depending on the value of p

Working of KNN algorithm

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

Step 1: Selecting the optimal value of K
K represents the number of nearest neighbors that needs to be considered while making prediction.

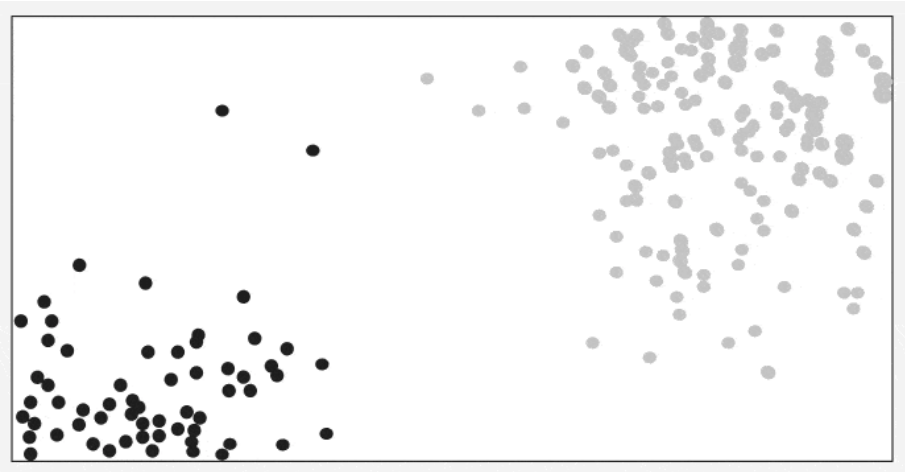
Step 2: Calculating distance
To measure the similarity between target and training data points Euclidean distance is used. Distance is calculated between data points in the dataset and target point.

Step 3: Finding Nearest Neighbors
The k data points with the smallest distances to the target point are nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

When you want to classify a data point into a category (like spam or not spam), the K-NN algorithm looks at the K closest points in the dataset. These closest points are called neighbors. The algorithm then looks at which category the neighbors belong to and picks the one that appears the most. This is called majority voting.

In regression, the algorithm still looks for the K closest points. But instead of voting for a class in classification, it takes the average of the values of those K neighbors. This average is the predicted value for the new point for the algorithm.



It shows how a test point is classified based on its nearest neighbors. As the test point moves the algorithm identifies the closest 'k' data points i.e 5 in this case and assigns test point the majority class label that is grey label class here.

6. PROCEDURE / PROGRAMME :

```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

data = np.random.rand(100)

labels = ["Class1" if x <= 0.5 else "Class2" for x in data[:50]]



def euclidean_distance(x1, x2):
    return abs(x1 - x2)

def knn_classifier(train_data, train_labels, test_point, k):
```

```
    distances = [(euclidean_distance(test_point, train_data[i]),
train_labels[i]) for i in range(len(train_data))]

    distances.sort(key=lambda x: x[0])
    k_nearest_neighbors = distances[:k]

    k_nearest_labels = [label for _, label in k_nearest_neighbors]

    return Counter(k_nearest_labels).most_common(1)[0][0]

train_data = data[:50]
train_labels = labels

test_data = data[50:]

k_values = [1, 2, 3, 4, 5, 20, 30]
print("--- k-Nearest Neighbors Classification ---")
print("Training dataset: First 50 points labeled based on the rule
(x <= 0.5 -> Class1, x > 0.5 -> Class2)")
print("Testing dataset: Remaining 50 points to be classified\n")

results = {}

for k in k_values:
    print(f"Results for k = {k}:")
    classified_labels = [knn_classifier(train_data, train_labels,
test_point, k) for test_point in test_data]
    results[k] = classified_labels

    for i, label in enumerate(classified_labels, start=51):
        print(f"Point x{i} (value: {test_data[i - 51]:.4f}) is
classified as {label}")
    print("\n")

print("Classification complete.\n")
```

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

    plt.figure(figsize=(10, 6))
    plt.scatter(train_data, [0] * len(train_data), c=["blue" if
label == "Class1" else "red" for label in train_labels],
                label="Training Data", marker="o")
    plt.scatter(class1_points, [1] * len(class1_points), c="blue",
label="Class1 (Test)", marker="x")
    plt.scatter(class2_points, [1] * len(class2_points), c="red",
label="Class2 (Test)", marker="x")

    plt.title(f"k-NN Classification Results for k = {k}")
    plt.xlabel("Data Points")
    plt.ylabel("Classification Level")
    plt.legend()
    plt.grid(True)
    plt.show()
```

OUTPUT:

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

Results for k = 1:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
```

```
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class1
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1
```

```
Results for k = 2:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class1
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
```

```
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1

Results for k = 3:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class1
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
```

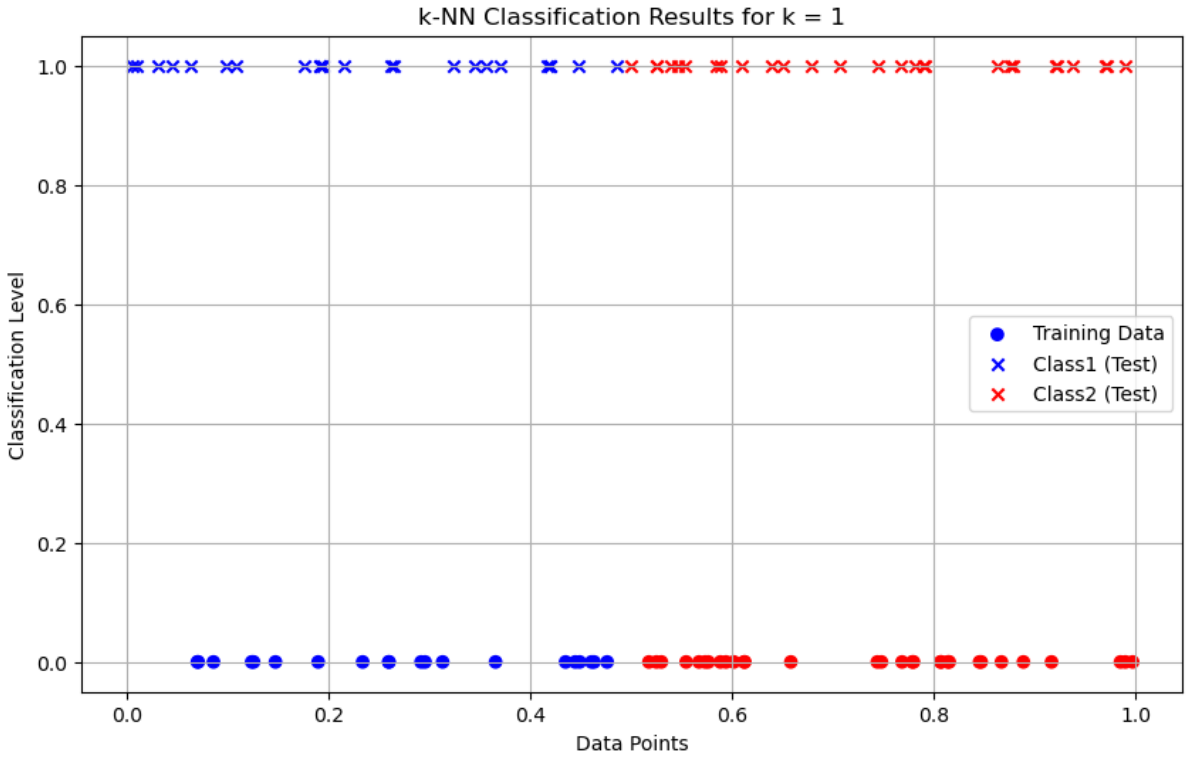
```
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1

Results for k = 4:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class1
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
```

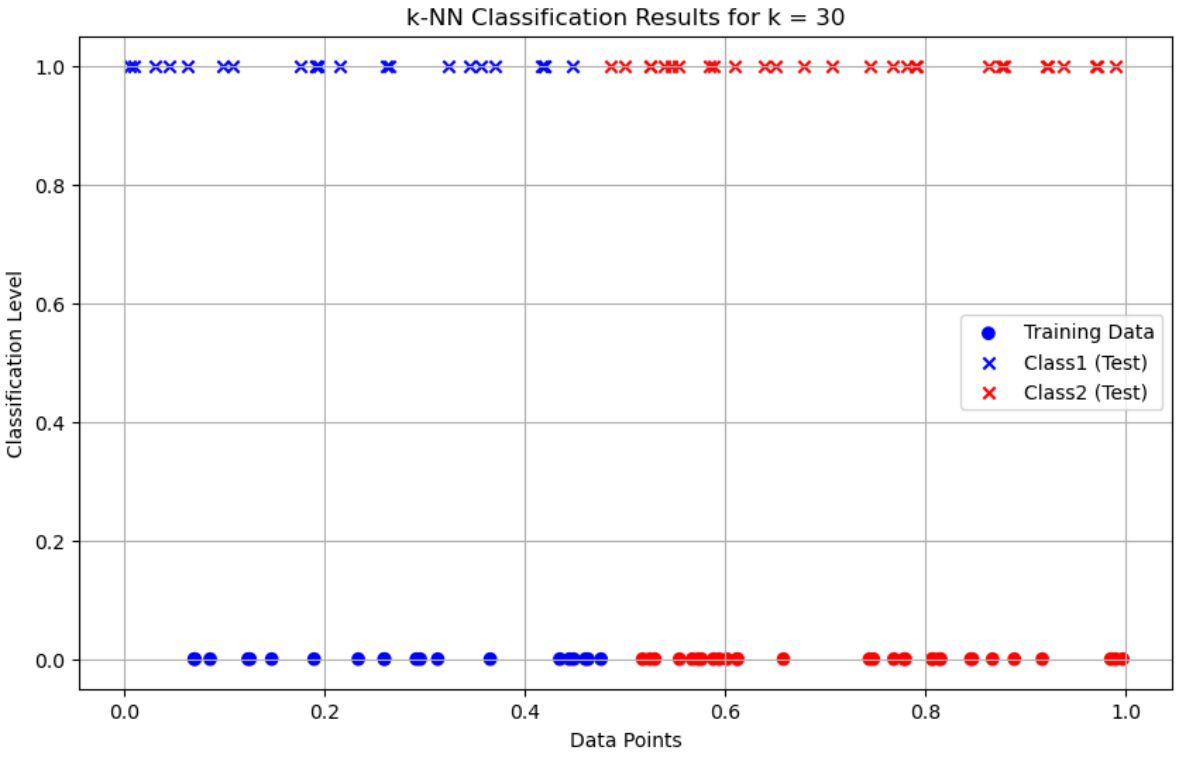
```
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1

Results for k = 5:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
```

```
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class1
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1

Results for k = 20:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
```

```
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class2
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1

Results for k = 30:
Point x51 (value: 0.4186) is classified as Class1
Point x52 (value: 0.1913) is classified as Class1
Point x53 (value: 0.9719) is classified as Class2
Point x54 (value: 0.6504) is classified as Class2
Point x55 (value: 0.2149) is classified as Class1
Point x56 (value: 0.0625) is classified as Class1
Point x57 (value: 0.8785) is classified as Class2
Point x58 (value: 0.7059) is classified as Class2
Point x59 (value: 0.6395) is classified as Class2
Point x60 (value: 0.3241) is classified as Class1
Point x61 (value: 0.0987) is classified as Class1
Point x62 (value: 0.1907) is classified as Class1
Point x63 (value: 0.1081) is classified as Class1
Point x64 (value: 0.5463) is classified as Class2
Point x65 (value: 0.5245) is classified as Class2
```

```
Point x66 (value: 0.0095) is classified as Class1
Point x67 (value: 0.1940) is classified as Class1
Point x68 (value: 0.7450) is classified as Class2
Point x69 (value: 0.0305) is classified as Class1
Point x70 (value: 0.0046) is classified as Class1
Point x71 (value: 0.4473) is classified as Class1
Point x72 (value: 0.0449) is classified as Class1
Point x73 (value: 0.5532) is classified as Class2
Point x74 (value: 0.7819) is classified as Class2
Point x75 (value: 0.7890) is classified as Class2
Point x76 (value: 0.8762) is classified as Class2
Point x77 (value: 0.8628) is classified as Class2
Point x78 (value: 0.9900) is classified as Class2
Point x79 (value: 0.7665) is classified as Class2
Point x80 (value: 0.4851) is classified as Class2
Point x81 (value: 0.5881) is classified as Class2
Point x82 (value: 0.9204) is classified as Class2
Point x83 (value: 0.4165) is classified as Class1
Point x84 (value: 0.4188) is classified as Class1
Point x85 (value: 0.9696) is classified as Class2
Point x86 (value: 0.1754) is classified as Class1
Point x87 (value: 0.2621) is classified as Class1
Point x88 (value: 0.3443) is classified as Class1
Point x89 (value: 0.5252) is classified as Class2
Point x90 (value: 0.2649) is classified as Class1
Point x91 (value: 0.5842) is classified as Class2
Point x92 (value: 0.6777) is classified as Class2
Point x93 (value: 0.7905) is classified as Class2
Point x94 (value: 0.9382) is classified as Class2
Point x95 (value: 0.6094) is classified as Class2
Point x96 (value: 0.3705) is classified as Class1
Point x97 (value: 0.5000) is classified as Class2
Point x98 (value: 0.9226) is classified as Class2
Point x99 (value: 0.5391) is classified as Class2
Point x100 (value: 0.3558) is classified as Class1

Classification complete.
```


k-NN Classification Results for k = 1
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 = 30
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0

1. EXPERIMENT NO: 6

2. TITLE: **Locally Weighted Regression algorithm**

3. LEARNING OBJECTIVES:
    1. Make use of Data sets in implementing the machine learning algorithms.
    2. Implement ML concepts and algorithms in Python

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

5. THEORY:
    - Given a dataset X, y, we attempt to find a linear model h(x) that minimizes residual sum of squared errors. The solution is given by Normal equations.
    - Linear model can only fit a straight line, however, it can be empowered by polynomial features to get more powerful models. Still, we have to decide and fix the number and types of features ahead.
    - Alternate approach is given by locally weighted regression.
    - Given a dataset X, y, we attempt to find a model h(x) that minimizes residual sum of weighted squared errors.
    - The weights are given by a kernel function which can be chosen arbitrarily and in my case I chose a Gaussian kernel.
    - The solution is very similar to Normal equations, we only need to insert diagonal weight matrix W.

total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.5	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4
8.77	2	Male	No	Sun	Dinner	2
26.88	3.12	Male	No	Sun	Dinner	4
15.04	1.96	Male	No	Sun	Dinner	2
14.78	3.23	Male	No	Sun	Dinner	2
10.27	1.71	Male	No	Sun	Dinner	2
35.26	5	Female	No	Sun	Dinner	4
15.42	1.57	Male	No	Sun	Dinner	2
18.43	3	Male	No	Sun	Dinner	4
14.83	3.02	Female	No	Sun	Dinner	2
21.58	3.92	Male	No	Sun	Dinner	2

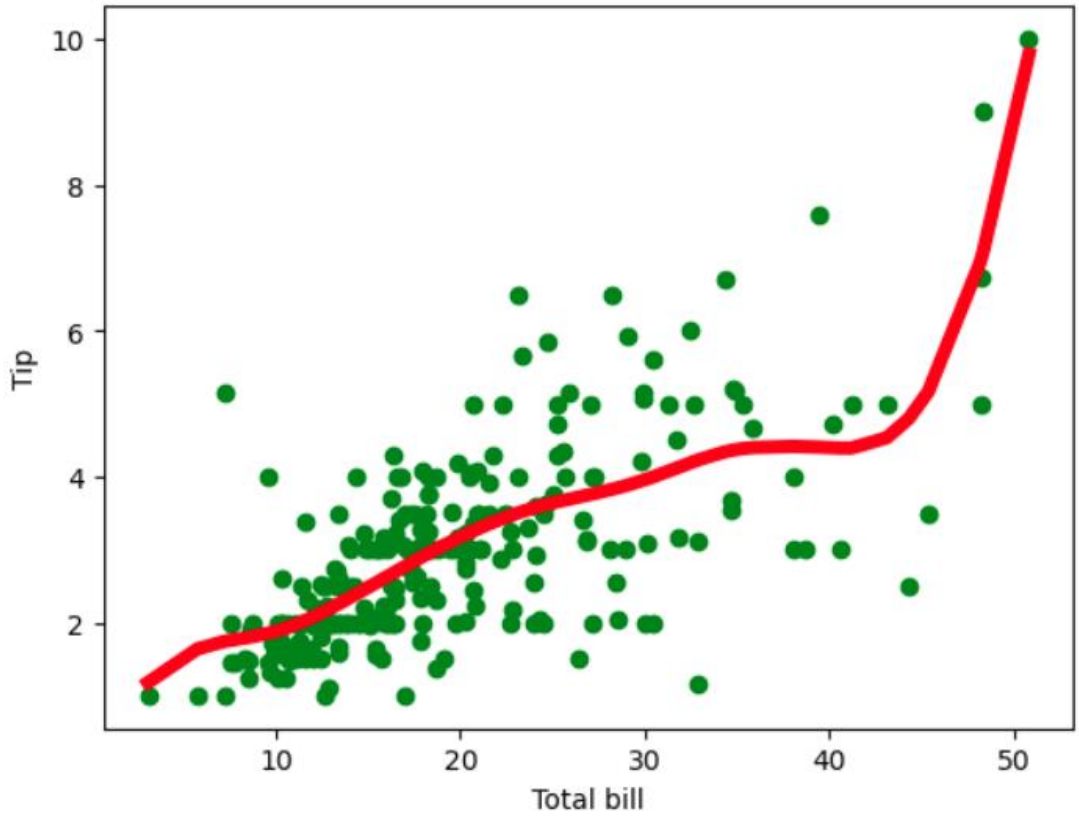
6. PROCEDURE / PROGRAMME

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

def kernel(point,xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m))) # eye - identity matrix
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
    return weights
def localWeight(point,xmat,ymat,k):
    wei = kernel(point,xmat,k)
    W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
    return W

def localWeightRegression(xmat,ymat,k):
    m,n = np.shape(xmat)
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred

def graphPlot(X,ypred):
    sortindex = X[:,1].argsort(0) #argsort - index of the smallest
    xsort = X[sortindex][:,0]
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.scatter(bill,tip, color='green')
    ax.plot(xsort[:,1],ypred[sortindex], color = 'red',
linewidth=5)
    plt.xlabel('Total bill')
    plt.ylabel('Tip')
    plt.show();
```

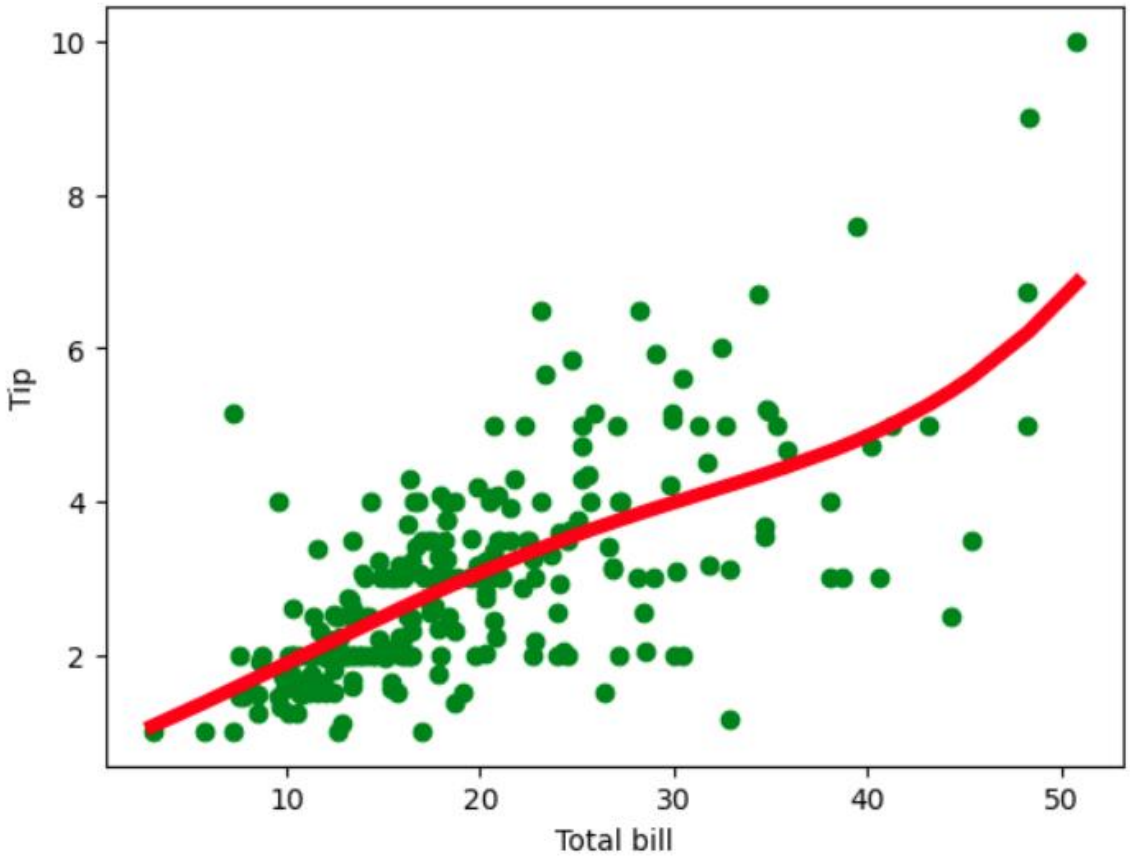
```
# load data points
   data = pd.read_csv('Program6_dataset_tips.csv')
   bill = np.array(data.total_bill) # We use only Bill amount and Tips data
   tip = np.array(data.tip)
   mbill = np.mat(bill) # .mat will convert nd array is converted in 2D array
   mtip = np.mat(tip)
   m= np.shape(mbill)[1]
   one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T)) # 244 rows, 2 cols increase k to get smooth curves
   ypred = localWeightRegression(X,mtip,9)
   graphPlot(X,ypred)
```

OUTPUT:

Regression with parameter k = 3



Regression with parameter k = 9



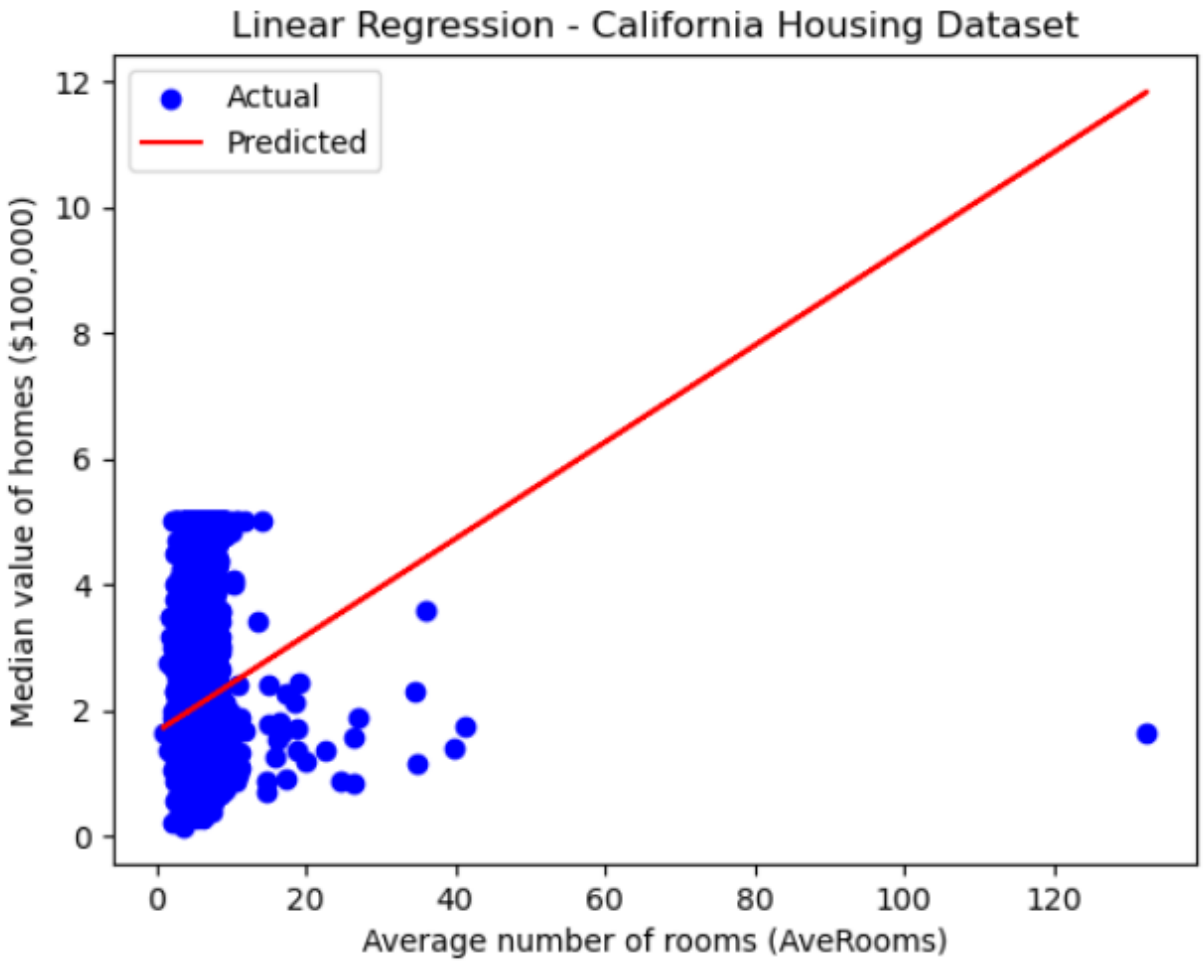
1. EXPERIMENT NO: 7
2. TITLE: **Linear Regression and Polynomial Regression**
3. LEARNING OBJECTIVES:
    1. Make use of Data sets in implementing the machine learning algorithms.
    2. Implement ML concepts and algorithms in Python
4. AIM: Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.
5. THEORY:
6. PROCEDURE / PROGRAMME

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

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

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

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

    y_pred = model.predict(X_test)
```

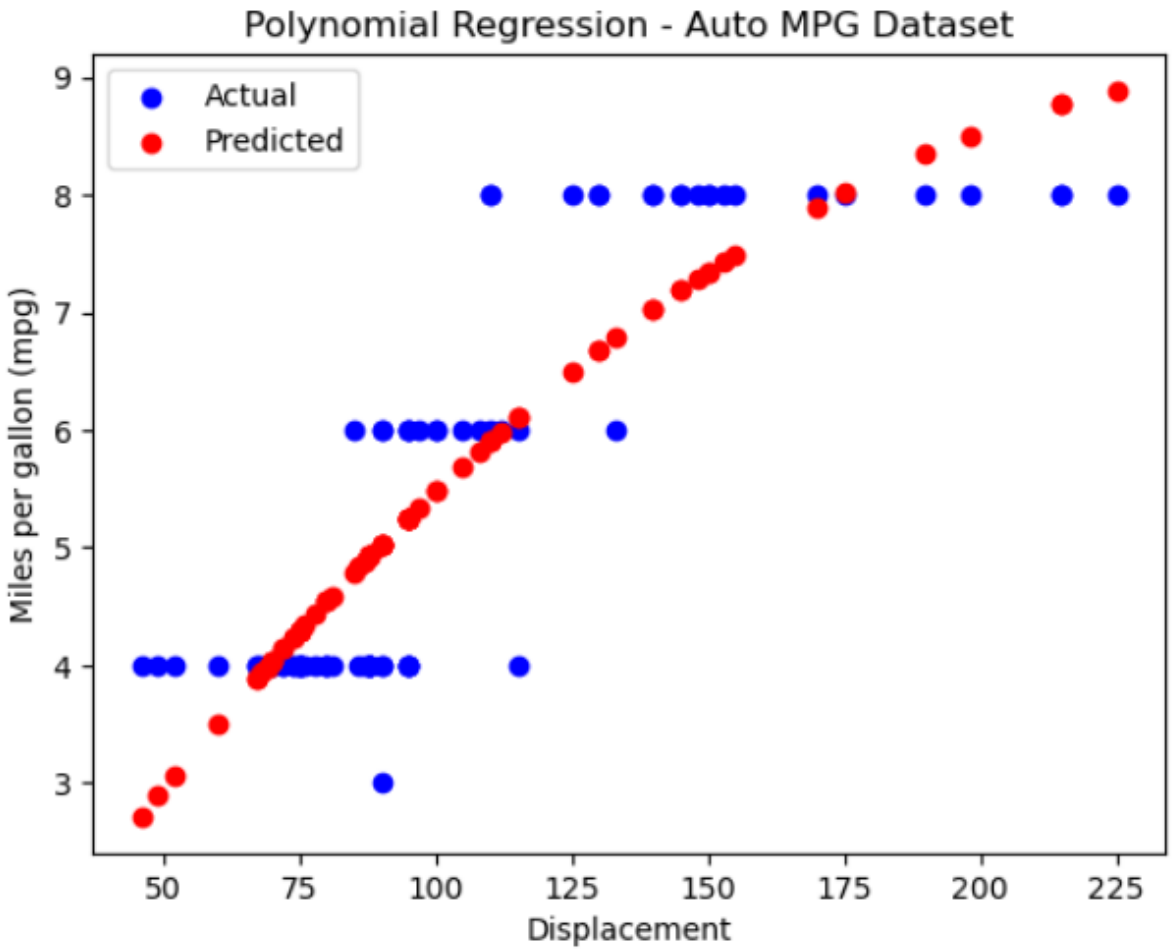
```
    plt.scatter(X_test, y_test, color="blue", label="Actual")
    plt.plot(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Average number of rooms (AveRooms)")
    plt.ylabel("Median value of homes ($100,000)")
    plt.title("Linear Regression - California Housing Dataset")
    plt.legend()
    plt.show()

    print("Linear Regression - California Housing Dataset")
    print("Mean Squared Error:", mean_squared_error(y_test,
y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))

def polynomial_regression_auto_mpg():
    url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/auto-mpg/auto-mpg.data"
    column_names = ["mpg", "cylinders", "displacement",
"horsepower", "weight", "acceleration", "model_year", "origin"]
    data = pd.read_csv(url, sep='\s+', names=column_names,
na_values="?")
    data = data.dropna()

    X = data["displacement"].values.reshape(-1, 1)
    y = data["mpg"].values

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

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

    y_pred = poly_model.predict(X_test)
    plt.scatter(X_test, y_test, color="blue", label="Actual")
    plt.scatter(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Displacement")
    plt.ylabel("Miles per gallon (mpg)")
```

```
    plt.title("Polynomial Regression - Auto MPG Dataset")
    plt.legend()
    plt.show()

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

OUTPUT



```
Linear Regression - California Housing Dataset
Mean Squared Error: 1.2923314440807296
R^2 Score: 0.013795337532285012
```




```
Polynomial Regression - Auto MPG Dataset
Mean Squared Error: 0.743149055720586
R^2 Score: 0.7505650609469627
```

1. EXPERIMENT NO: 8

2. TITLE: **Decision tree algorithm**

3. LEARNING OBJECTIVES:
    1. Make use of Data sets in implementing the machine learning algorithms.
    2. Implement ML concepts and algorithms in Python

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

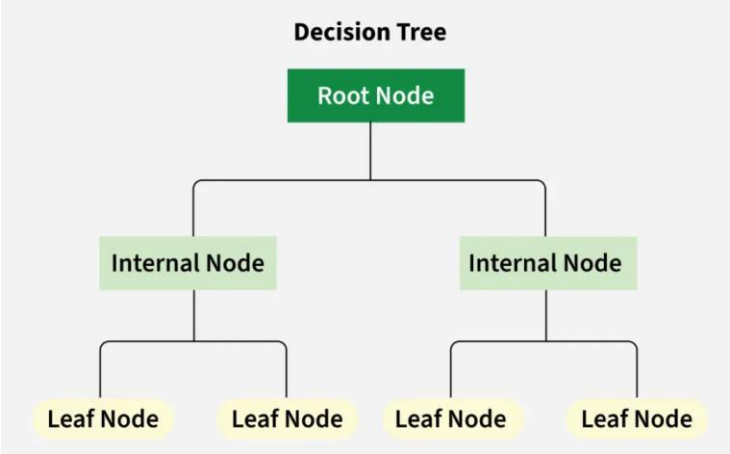
5. THEORY: Decision Tree

Decision tree is a simple diagram that shows different choices and their possible results helping you make decisions easily. This article is all about what decision trees are, how they work, their advantages and disadvantages and their applications.

Understanding Decision Tree

A decision tree is a graphical representation of different options for solving a problem and show how different factors are related. It has a hierarchical tree structure starts with one main question at the top called a node which further branches out into different possible outcomes where:

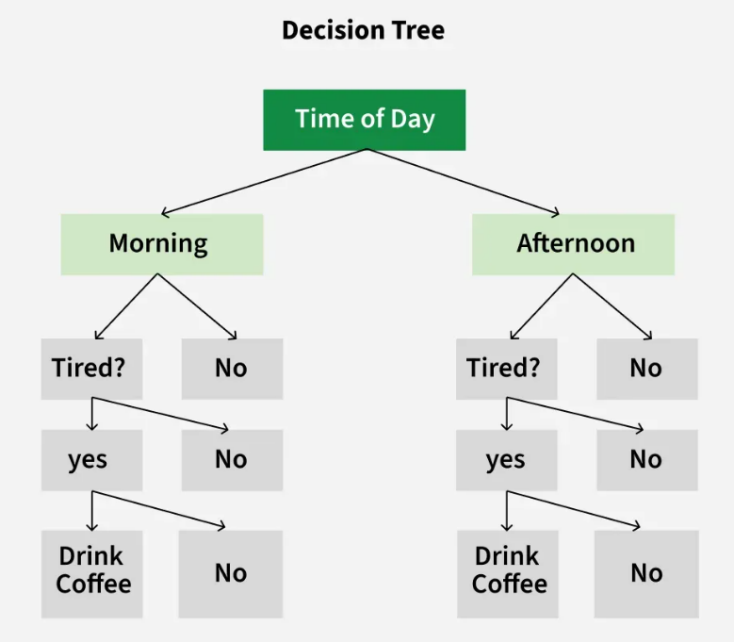
- **Root Node** is the starting point that represents the entire dataset.
- **Branches**: These are the lines that connect nodes. It shows the flow from one decision to another.
- **Internal Nodes** are Points where decisions are made based on the input features.
- **Leaf Nodes**: These are the terminal nodes at the end of branches that represent final outcomes or predictions



They also support decision-making by visualizing outcomes. You can quickly evaluate and compare the "branches" to determine which course of action is best for you.

Now, let's take an example to understand the decision tree. Imagine you want to decide whether to drink coffee based on the time of day and how tired you feel. First the tree checks the time of day—if it's morning it asks whether you are tired. If you're tired the tree suggests

drinking coffee if not it says there's no need. Similarly in the afternoon the tree again asks if you are tired. If you recommends drinking coffee if not it concludes no coffee is needed.



Classification of Decision Tree

We have mainly two types of decision tree based on the nature of the target variable: classification trees and regression trees.

Classification trees: They are designed to predict categorical outcomes means they classify data into different classes. They can determine whether an email is "spam" or "not spam" based on various features of the email.
Regression trees: These are used when the target variable is continuous It predict numerical values rather than categories. For example a regression tree can estimate the price of a house based on its size, location, and other features.

How Decision Trees Work?

A decision tree working starts with a main question known as the root node. This question is derived from the features of the dataset and serves as the starting point for decision-making.

From the root node, the tree asks a series of yes/no questions. Each question is designed to split the data into subsets based on specific attributes. For example if the first question is "Is it raining?" the answer will determine which branch of the tree to follow. Depending on the response to each question you follow different branches. If your answer is "Yes," you might proceed down one path if "No," you will take another path.

This branching continues through a sequence of decisions. As you follow each branch, you get more questions that break the data into smaller groups. This step-by-step process continues until you have no more helpful questions.

You reach at the end of a branch where you find the final outcome or decision. It could be a classification (like "spam" or "not spam") or a prediction (such as estimated price).

Advantages of Decision Trees

- **Simplicity and Interpretability**: Decision trees are straightforward and easy to understand. You can visualize them like a flowchart which makes it simple to see how decisions are made.
- **Versatility**: It means they can be used for different types of tasks can work well for both classification and regression
- **No Need for Feature Scaling**: They don't require you to normalize or scale your data.
- **Handles Non-linear Relationships**: It is capable of capturing non-linear relationships between features and target variables.

Disadvantages of Decision Trees

- **Overfitting**: Overfitting occurs when a decision tree captures noise and details in the training data and it perform poorly on new data.
- **Instability**: instability means that the model can be unreliable slight variations in input can lead to significant differences in predictions.
- **Bias towards Features with More Levels**: Decision trees can become biased towards features with many categories focusing too much on them during decision-making. This can cause the model to miss out other important features led to less accurate predictions.

Applications of Decision Trees

- **Loan Approval in Banking**: A bank needs to decide whether to approve a loan application based on customer profiles.
  - Input features include income, credit score, employment status, and loan history.
  - The decision tree predicts loan approval or rejection, helping the bank make quick and reliable decisions
- **Medical Diagnosis**: A healthcare provider wants to predict whether a patient has diabetes based on clinical test results.
  - Features like glucose levels, BMI, and blood pressure are used to make a decision tree.
  - Tree classifies patients into diabetic or non-diabetic, assisting doctors in diagnosis.

- **De Predicting Exam Results in Education**: School wants to predict whether a student will pass or fail based on study habits.
  - Data includes attendance, time spent studying, and previous grades.
  - The decision tree identifies at-risk students, allowing teachers to provide additional support.

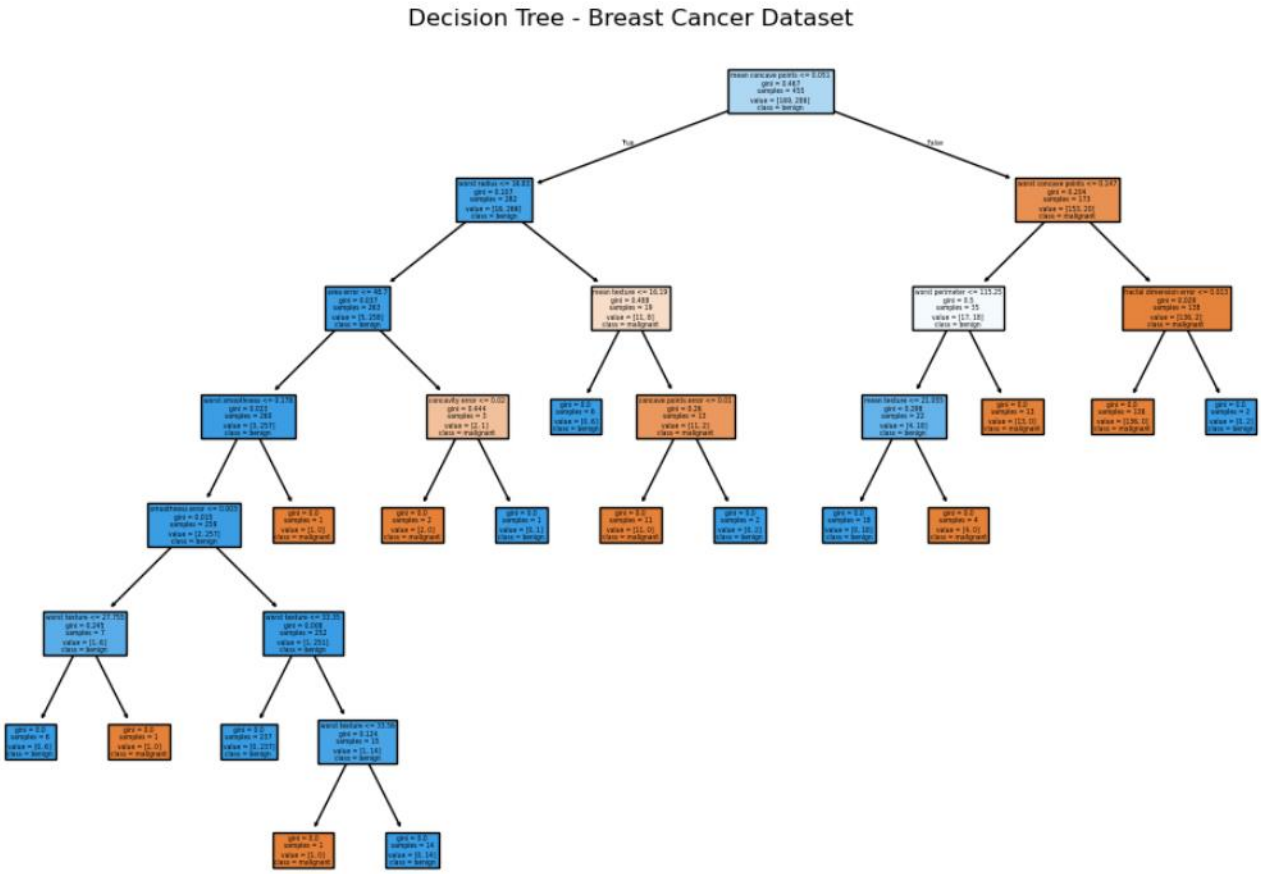
A decision tree can also be used to help build automated predictive models, which have applications in machine learning, data mining, and statistics

6. PROCEDURE / PROGRAMME

```
# 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)

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()
```

OUTPUT:

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



1. EXPERIMENT NO: 9

2. TITLE: **Naive Bayesian classifier**

3. LEARNING OBJECTIVES:
    1. Make use of Data sets in implementing the machine learning algorithms.
    2. Implement ML concepts and algorithms in Python

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

   THEORY:

5. THEORY: Naive Bayes Classifiers

   Naive Bayes classifiers are supervised machine learning algorithms used for classification tasks, based on Bayes' Theorem to find probabilities. This article will give you an overview as well as more advanced use and implementation of Naive Bayes in machine learning.

   Key Features of Naive Bayes Classifiers

   The main idea behind the Naive Bayes classifier is to use Bayes' Theorem to classify data based on the probabilities of different classes given the features of the data. It is used mostly in high-dimensional text classification

   - The Naive Bayes Classifier is a simple probabilistic classifier and it has very few number of parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.
   - It is a probabilistic classifier because it assumes that one feature in the model is independent of existence of another feature. In other words, each feature contributes to the predictions with no relation between each other.
   - Naïve Bayes Algorithm is used in spam filtration, Sentimental analysis, classifying articles and many more.

Why it is Called Naive Bayes?

It is named as "Naive" because it assumes the presence of one feature does not affect other features.

The "Bayes" part of the name refers to for the basis in Bayes' Theorem.

Assumption of Naive Bayes

The fundamental Naive Bayes assumption is that each feature makes an:

- **Feature independence**: This means that when we are trying to classify something, we assume that each feature (or piece of information) in the data does not affect any other feature.
- **Continuous features are normally distributed**: If a feature is continuous, then it is assumed to be normally distributed within each class.
- **Discrete features have multinomial distributions**: If a feature is discrete, then it is assumed to have a multinomial distribution within each class.

- **Features are equally important**: All features are assumed to contribute equally to the prediction of the class label.
- **No missing data**: The data should not contain any missing values.

The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice. Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Types of Naive Bayes Model

There are three types of Naive Bayes Model:

Gaussian Naive Bayes: In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution when plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

Multinomial Naive Bayes: Multinomial Naive Bayes is used when features represent the frequency of terms (such as word counts) in a document. It is commonly applied in text classification, where term frequencies are important.

Bernoulli Naive Bayes: Bernoulli Naive Bayes deals with binary features, where each feature indicates whether a word appears or not in a document. It is suited for scenarios where the presence or absence of terms is more relevant than their frequency. Both models are widely used in document classification tasks.

Advantages of Naive Bayes Classifier

- Easy to implement and computationally efficient.
- Effective in cases with a large number of features.
- Performs well even with limited training data.
- It performs well in the presence of categorical features.
- For numerical features data is assumed to come from normal distributions

Disadvantages of Naive Bayes Classifier

- Assumes that features are independent, which may not always hold in real-world data.
- Can be influenced by irrelevant attributes.
- May assign zero probability to unseen events, leading to poor generalization**.**

Applications of Naive Bayes Classifier

- Spam Email Filtering: Classifies emails as spam or non-spam based on features.
- Text Classification: Used in sentiment analysis, document categorization, and topic classification.
- Medical Diagnosis: Helps in predicting the likelihood of a disease based on symptoms.
- Credit Scoring: Evaluates creditworthiness of individuals for loan approval.
- Weather Prediction: Classifies weather conditions based on various factors.

6. PROCEDURE / PROGRAMME

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

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

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

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

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

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

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

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

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

plt.show()
```

OUTPUT:

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

Classification Report:

	precision	recall	f1-score	support
0	0.67	1.00	0.80	2
1	1.00	1.00	1.00	2
2	0.33	0.67	0.44	3
3	1.00	0.00	0.00	5
4	1.00	0.50	0.67	4
5	1.00	1.00	1.00	2
7	1.00	0.75	0.86	4
8	1.00	0.67	0.80	3
9	1.00	0.75	0.86	4
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	1
12	0.40	1.00	0.57	4
13	1.00	0.80	0.89	5
14	1.00	0.40	0.57	5
15	0.67	1.00	0.80	2
16	1.00	0.67	0.80	3
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	3
19	0.67	1.00	0.80	2
20	1.00	1.00	1.00	3
21	1.00	0.67	0.80	3
22	1.00	0.60	0.75	5
23	1.00	0.75	0.86	4
24	1.00	1.00	1.00	3
25	1.00	0.75	0.86	4
26	1.00	1.00	1.00	2
27	1.00	1.00	1.00	5
28	0.50	1.00	0.67	2
29	1.00	1.00	1.00	2
30	1.00	1.00	1.00	2
31	1.00	0.75	0.86	4
32	1.00	1.00	1.00	2
34	0.25	1.00	0.40	1
35	1.00	1.00	1.00	5
36	1.00	1.00	1.00	3
37	1.00	1.00	1.00	1
38	1.00	0.75	0.86	4
39	0.50	1.00	0.67	5
accuracy			0.81	120
macro avg	0.89	0.85	0.83	120
weighted avg	0.91	0.81	0.81	120

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

Cross-validation accuracy: 87.25%
```

1. EXPERIMENT NO: 10

2. TITLE: **k-means clustering**

3. LEARNING OBJECTIVES:
    1. Make use of Data sets in implementing the machine learning algorithms.
    2. Implement ML concepts and algorithms in Python

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

5. THEORY: **k-means clustering**

K-Means Clustering is an Unsupervised Machine Learning algorithm which groups the unlabeled dataset into different clusters. The article aims to explore the fundamentals and working of k means clustering along with its implementation.

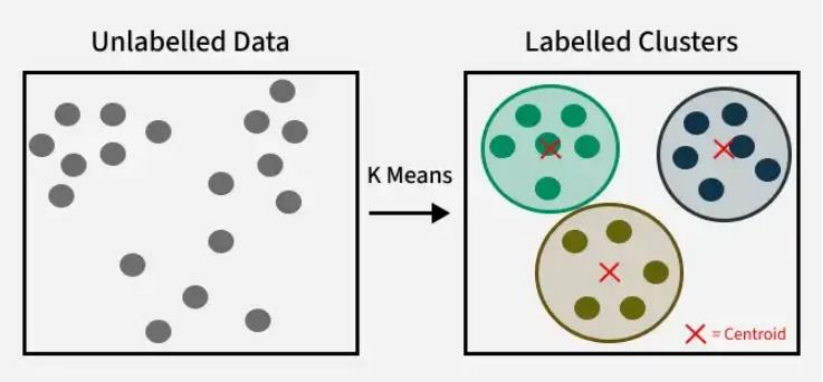
Understanding K-means Clustering

K-means clustering is a technique used to organize data into groups based on their similarity. For example online store uses K-Means to group customers based on purchase frequency and spending creating segments like Budget Shoppers, Frequent Buyers and Big Spenders for personalized marketing.

The algorithm works by first randomly picking some central points called centroids and each data point is then assigned to the closest centroid forming a cluster. After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster. This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group.

How k-means clustering works?

We are given a data set of items with certain features and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.



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

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean, and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The "points" mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set. For example for a feature x the items have values in [0,3] we will initialize the means with values for x at [0,3].

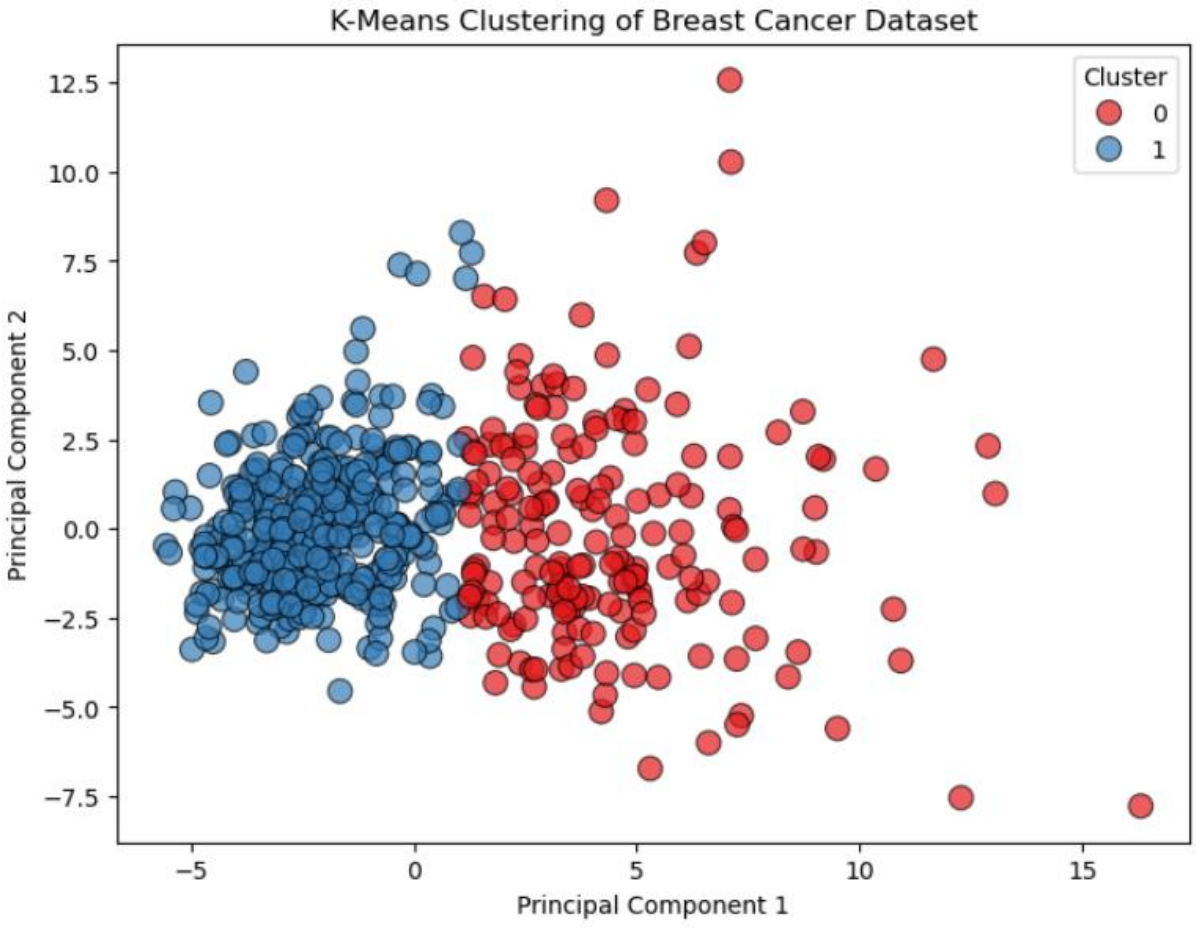
6. PROCEDURE / PROGRAMME

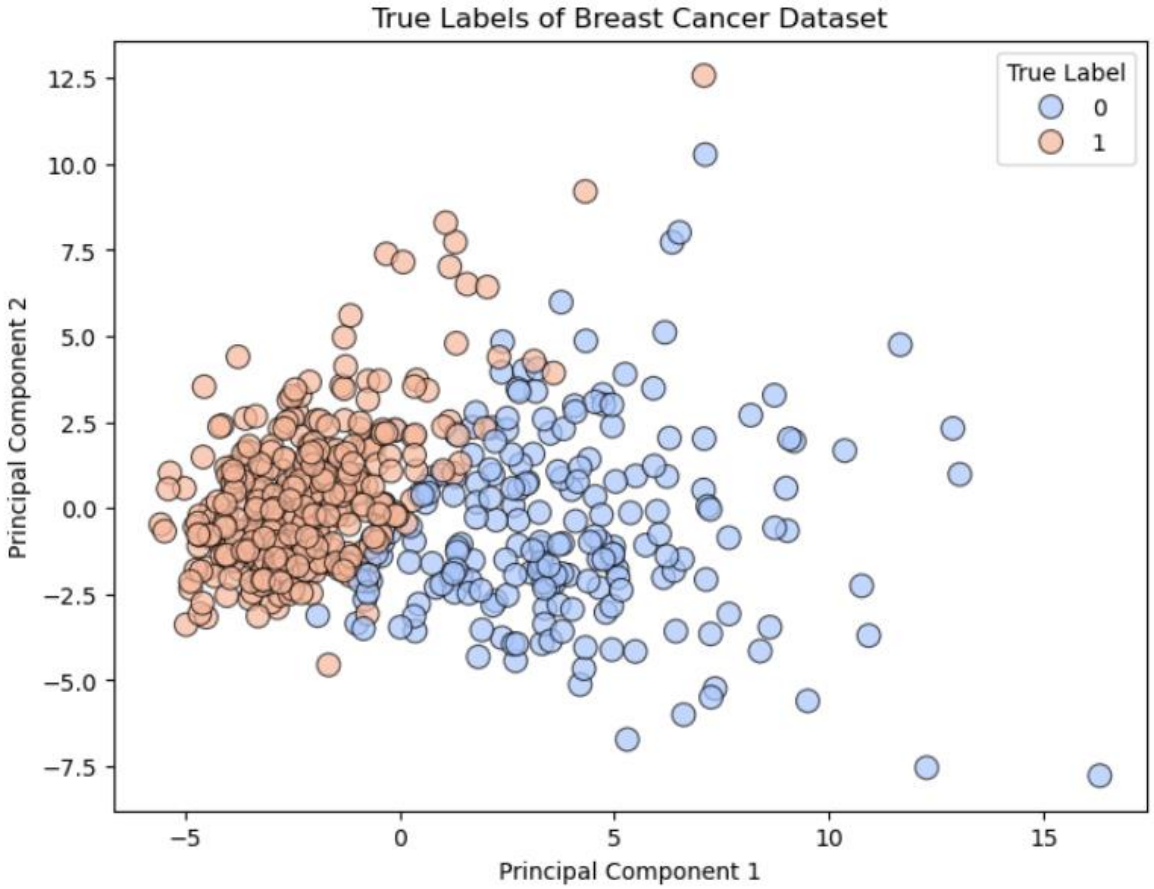
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix,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

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

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=2, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)

print("Confusion Matrix:")
print(confusion_matrix(y, y_kmeans))
print("\nClassification Report:")
print(classification_report(y, y_kmeans))
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

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

plt.figure(figsize=(8, 6))
```

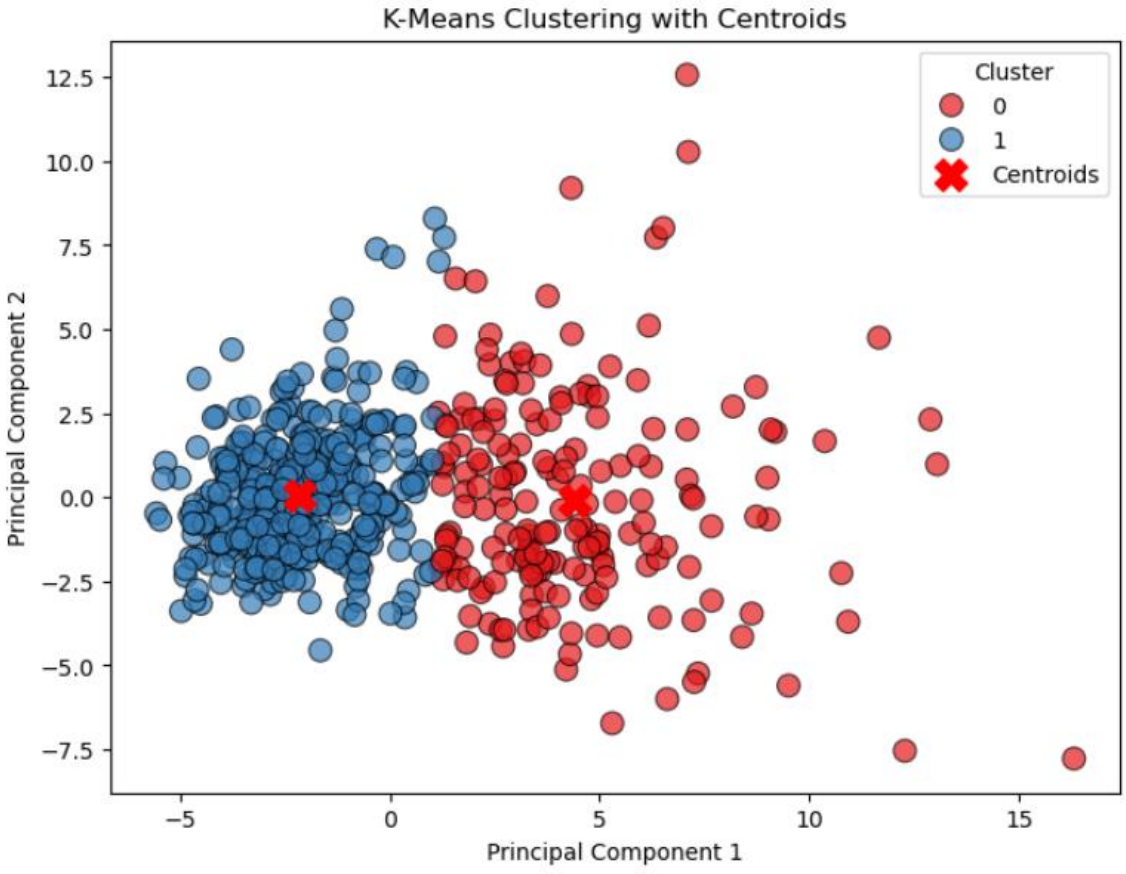
```
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster',
palette='Set1', s=100, edgecolor='black', alpha=0.7)
plt.title('K-Means Clustering of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()

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

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster',
palette='Set1', s=100, edgecolor='black', alpha=0.7)
centers = pca.transform(kmeans.cluster_centers_)
plt.scatter(centers[:, 0], centers[:, 1], s=200, c='red',
marker='X', label='Centroids')
plt.title('K-Means Clustering with Centroids')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()
```

OUTPUT:

```
Confusion Matrix:
[[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
```




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