ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS INSTITUTION
Approved By AICTE & Affiliated To Anna University
NBA Accredited for BE (ECE, EEE, MECH) | Accredited by NAAC with A+ Grade
Anjugramam-Kanyakumari Main Road, Palkulam, Variyoor P.O. - 629 401, Kanyakumari District.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

M.E.COMMUNICATION SYSTEMS

SUBJECT CODE: 24CP204

SUBJECT TITLE: MACHINE LEARNING LABORATORY

ROHINI COLLEGE OF ENGINEERING & TECHNOLOGY

AUTONOMOUS INSTITUTION

Approved By AICTE & Affiliated To Anna University

NBA Accredited for BE (ECE, EEE, MECH) | Accredited by NAAC with A+ Grade

Bonafide Certificate

Certified that this is a Bonafide Record of work done by Ms / Mr. ……..... of the 01 / 02 Year / Semester communication systems department of this college, in the 24CP204 MACHINE LEARNING Laboratory in the partial fulfillment of the requirement of the M.E Degree of the Anna University.

STAFF-IN-CHARGE **HOD**

University register number : …….

University Examination held on : …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Ex. No :	01	IMPLEMENT A LINEAR REGRESSION WITH A REAL DATASET
Date:		

AIM:

Implement a Linear Regression with a Real Dataset. Experiment with different features in building a model. Tune the model's hyperparameters.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Load the dataset and check for missing values.
2. Explore the dataset by displaying the first few rows.
3. Analyze the structure and data types of each column.
4. Generate visualizations to understand the distribution of numerical features.
5. Identify patterns, trends, or anomalies in the dataset.
6. Use the insights gained for further data analysis or modeling.

PROCEDURE:

1. Open the Jupyter Notebook and Create a new file
2. To upload the CSV file in Jupyter Notebook.
3. Type the below code and save it.
4. Run the Program and Check Bugs and Fix it.
5. Finally Observe the Output and Note it.

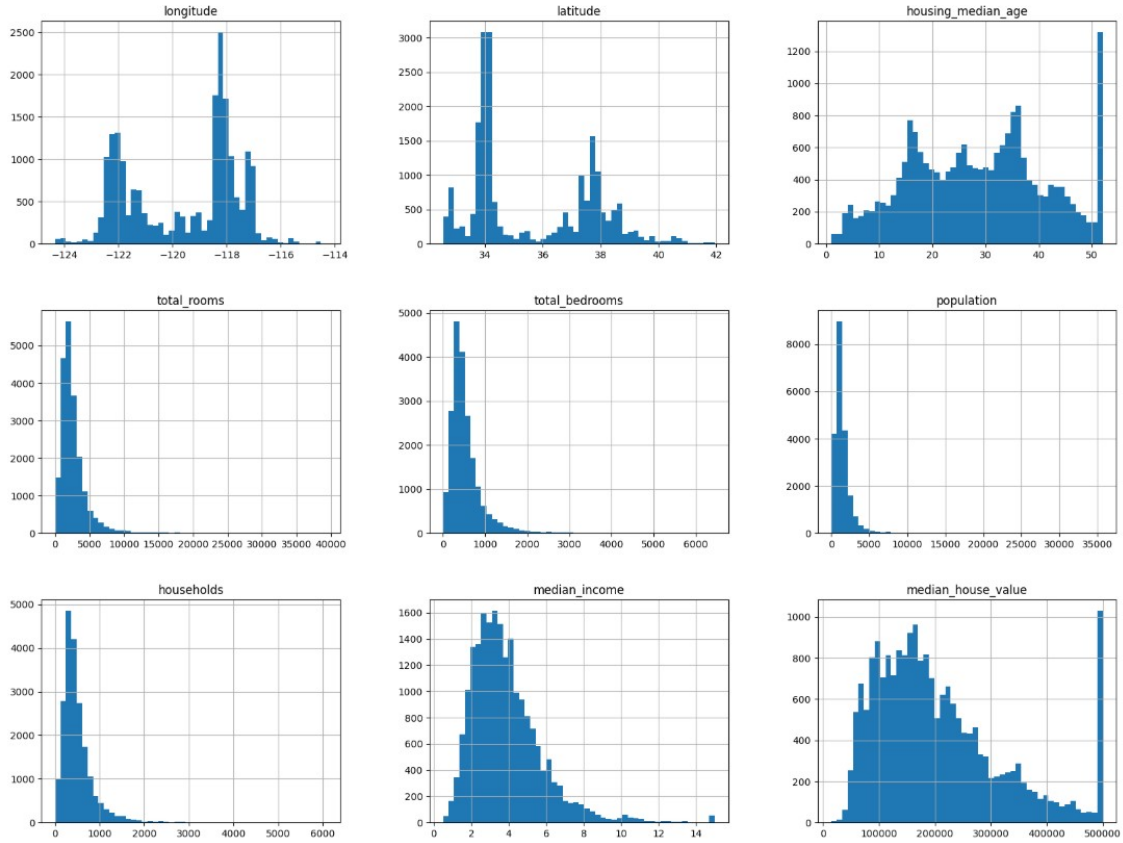
PROGRAM:

```
import numpy as np  # Linear algebra
import pandas as pd  # Data processing, CSV file I/O
import matplotlib.pyplot as plt  # Data visualization
import os

# List files in the input directory
input_dir = "input"
```

```
for dirname, _, filenames in os.walk(input_dir):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Load the housing dataset
housing = pd.read_csv("housing.csv")

# Display the first 5 rows
print(housing.head(5))

# Show dataset information
print(housing.info())

# Generate histograms for numerical features
housing.hist(bins=50, figsize=(20, 15))
plt.show()
```

OUTPUT :

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.5+ MB
None
```



RESULT:

Thus the Implemented a Linear Regression with a Real Dataset. Experiment with different features in building a model. Tune the model's hyper parameters was executed successfully.

Ex. No :	02	IMPLEMENT A BINARY CLASSIFICATION MODEL
Date:		

AIM:

To implement a binary classification model from given the dataset.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Load the dataset and define the target variable (above_median_price) based on the median house value.
2. Handle missing values in total_bedrooms using median imputation.
3. Convert categorical data (ocean_proximity) into numerical labels.
4. Split the dataset into training (80%) and testing (20%) sets, then normalize numerical features.
5. Train a Logistic Regression model and predict probabilities on the test set.
6. Apply thresholds (0.5 and 0.6) to classify test data and evaluate using Accuracy, Precision, Recall, and F1-score.
7. Plot the ROC Curve and calculate the AUC (Area Under Curve) for performance measurement.

PROCEDURE:

1. Open the Jupyter Notebook and Create a new file
2. To upload the CSV file in Jupyter Notebook.
3. Type the below code and save it.
4. Run the Program and Check Bugs and Fix it.
5. Finally Observe the Output and Note it.

PROGRAM:

```
import pandas as pd

import numpy as np
```

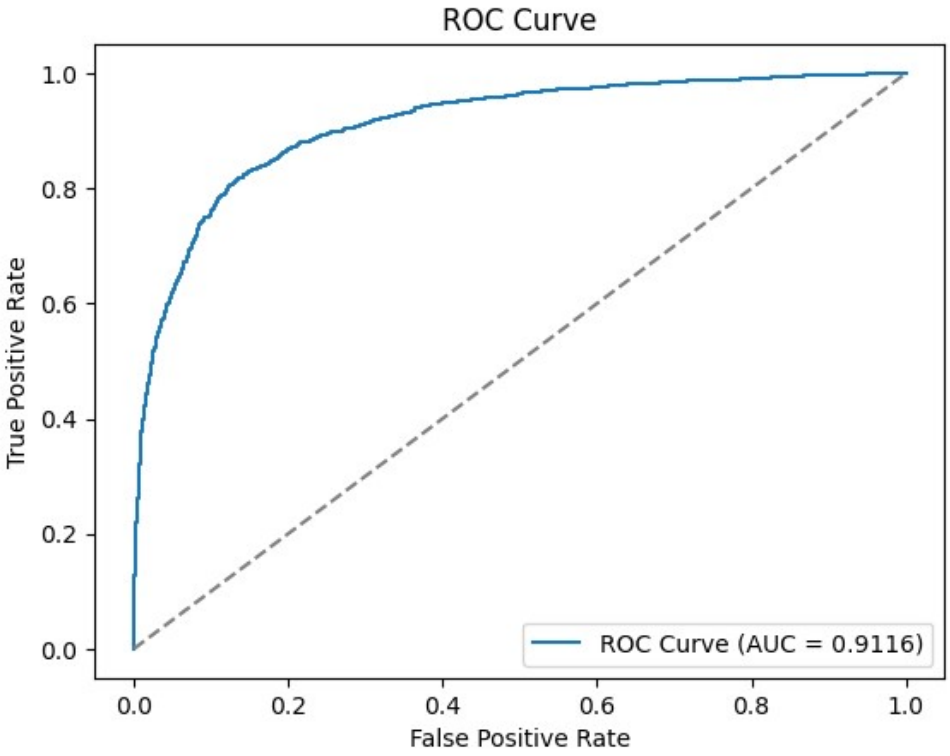
```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.impute import SimpleImputer

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score, roc_curve

import matplotlib.pyplot as plt

# Load and preprocess dataset

df = pd.read_csv("housing.csv")

df["above_median_price"] = (df["median_house_value"] >
df["median_house_value"].median()).astype(int)

df.drop(columns=["median_house_value"], inplace=True)

df["total_bedrooms"] =
SimpleImputer(strategy="median").fit_transform(df[["total_bedrooms"]])

df["ocean_proximity"] = LabelEncoder().fit_transform(df["ocean_proximity"])

X, y = df.drop(columns=["above_median_price"]), df["above_median_price"]

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

# Normalize, train model, and make predictions

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)  # Fit and transform the training data

X_test = scaler.transform(X_test)  # Transform the test data using the already
fitted scaler

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

y_prob = model.predict_proba(X_test)[:, 1]

# Define a function for metrics calculation
```

```
def calculate_metrics(y_pred):
    return {
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1 Score": f1_score(y_test, y_pred),
    }
# Experiment with thresholds and print metrics
for threshold in [0.5, 0.6]:
    y_pred = (y_prob > threshold).astype(int)
    print(f"Classification Metrics (Threshold = {threshold}):")
    metrics = calculate_metrics(y_pred)
    for metric, value in metrics.items():
        print(f"{metric}: {value:.4f}")
    print()
# Plot ROC curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc_score(y_test,
y_prob):.4f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

OUTPUT:

Classification Metrics (Threshold = 0.5):
Accuracy: 0.8387
Precision: 0.8437
Recall: 0.8289
F1 Score: 0.8362

Classification Metrics (Threshold = 0.6):
Accuracy: 0.8326
Precision: 0.8778
Recall: 0.7704
F1 Score: 0.8206



RESULT:

Thus the implemented a binary classification model was executed successfully.

Ex. No :	03	IMPLEMENT A KNN CLASSIFER ALGORITHM
Date:		

AIM:

To implement a KNN classifier Algorithm using California Housing Dataset.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Load the data and create a target column (price_category) based on house price.
2. Handle missing values in total_bedrooms by filling with the median.
3. Convert ocean_proximity to numbers using one-hot encoding.
4. Define the features (X) and target (y).
5. Standardize the features using StandardScaler.
6. Split the data into training and testing sets.
7. Train the KNN classifier with n_neighbors=5.
8. Predict on the test set and calculate accuracy.

PROCEDURE:

1. Open the Jupyter Notebook and Create a new file
2. To upload the CSV file in Jupyter Notebook.
3. Type the below code and save it.
4. Run the Program and Check Bugs and Fix it.
5. Finally Observe the Output and Note it.

PROGRAM:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
```

```
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load dataset
file_path = "housing.csv"  # Update this path if needed
df = pd.read_csv(file_path)

# Create a classification target (high price vs. low price)
median_value = df["median_house_value"].median()
df["price_category"] = (df["median_house_value"] >=
median_value).astype(int)

# Drop the original target column
df = df.drop(columns=["median_house_value"])

# Handle missing values in total_bedrooms
imputer = SimpleImputer(strategy="median")
df["total_bedrooms"] = imputer.fit_transform(df[["total_bedrooms"]])

# Encode categorical feature 'ocean_proximity' using one-hot encoding
df = pd.get_dummies(df, columns=["ocean_proximity"], drop_first=True)

# Define features (X) and target (y)
X = df.drop(columns=["price_category"])
```

```
y = df["price_category"]

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

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# Train KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
# Predict and evaluate
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"KNN Classification Accuracy: {accuracy:.4f}")
```

OUTPUT:

```
KNN Classification Accuracy: 0.8450
```

RESULT:

Thus the implementation for a KNN classifier Algorithm using California Housing Dataset was executed successfully**.**

Ex. No :	04	IMPLEMENT A TRAINING SET AND VALIDATION SET RESULTS
Date:		

AIM:

To analyze and comparison of Training Set and Validation Set from the given dataset.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Load the dataset into a pandas DataFrame.
2. Remove rows with missing values in the total_bedrooms column.
3. Convert the ocean_proximity column to numeric values using one-hot encoding.
4. Split the data into three sets: 70% for training, 15% for validation, and 15% for testing.
5. Separate the target variable (median_house_value) from the features in each set.
6. Train a linear regression model using the training data.
7. Make predictions using the trained model and calculate the R^2 score for training, validation, and test sets to measure how well the model works.

PROCEDURE:

1. Open the Jupyter Notebook and Create a new file
2. To upload the CSV file in Jupyter Notebook.
3. Type the below code and save it.
4. Run the Program and Check Bugs and Fix it.
5. Finally Observe the Output and Note it.

PROGRAM:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import r2_score
# Load and prepare the dataset
df = pd.read_csv("housing.csv").dropna(subset=['total_bedrooms'])
df = pd.get_dummies(df, columns=['ocean_proximity'], drop_first=True)

# Split the data
train_set, temp_set = train_test_split(df, test_size=0.3, random_state=42)
val_set, test_set = train_test_split(temp_set, test_size=0.5, random_state=42)

# Separate features and target
X_train, y_train = train_set.drop(columns=['median_house_value']),
train_set['median_house_value']
X_val, y_val = val_set.drop(columns=['median_house_value']),
val_set['median_house_value']
X_test, y_test = test_set.drop(columns=['median_house_value']),
test_set['median_house_value']
# Train and evaluate the model
model = LinearRegression().fit(X_train, y_train)

# Get R² scores
train_r2 = r2_score(y_train, model.predict(X_train))
val_r2 = r2_score(y_val, model.predict(X_val))
test_r2 = r2_score(y_test, model.predict(X_test))
# Print R² results
print(f"Training Accuracy : {train_r2:.2f}")
print(f"Validation Accuracy : {val_r2:.2f}")
print(f"Test Accuracy : {test_r2:.2f}")
```

OUTPUT:

```
Training Accuracy : 0.64
Validation Accuracy : 0.65
Test Accuracy : 0.66
```

RESULT:

Thus the analyze and comparison of Training set and Validation set was executed successfully.

Ex. No :	05	IMPLEMENT THE K-MEANS ALGORITHM
Date:		

AIM:

To Implement the K-Means Algorithm from the given dataset.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Load the dataset into a table or an array.
2. Ensure that the features are numeric and handle any missing values by filling them with zeros or other appropriate values.
3. Take a small sample of the data to make the calculation faster (e.g., 2000 rows).
4. For each sample, calculate the distances between each data point and the centroids of potential clusters.
5. Try different numbers of clusters (from 2 to 10) and calculate how well the data fits into the clusters (inertia) and how well the clusters are separated (silhouette score).
6. Plot the inertia and silhouette score to help determine the best number of clusters.
7. After selecting the best number of clusters, apply the K-Means algorithm and assign each data point to its respective cluster.

PROCEDURE:

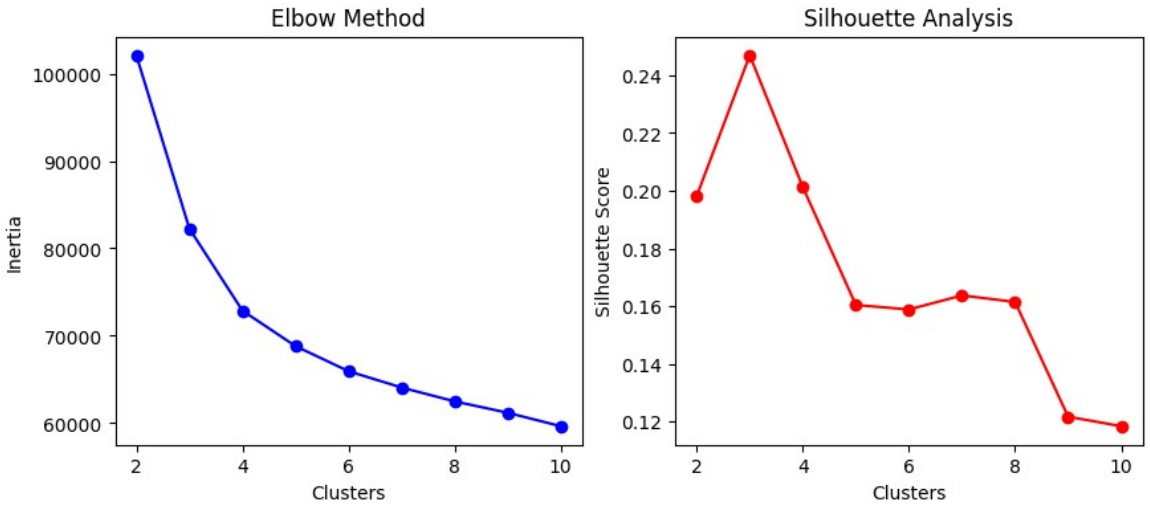
1. Open the Jupyter Notebook and Create a new file
2. To upload the CSV file in Jupyter Notebook.
3. Type the below code and save it.
4. Run the Program and Check Bugs and Fix it.
5. Finally Observe the Output and Note it.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
# Load dataset (Ensure correct file path)
file_path = "codon_usage.csv"
df = pd.read_csv(file_path, low_memory=False)
# Convert codon frequency columns to numeric, handle missing values
codon_columns = df.columns[5:]
df[codon_columns] = df[codon_columns].apply(pd.to_numeric,
errors='coerce').fillna(0)
# Sample 2000 rows for faster computation
df_sampled = df.sample(n=min(2000, len(df)), random_state=42)  # Avoid error
if dataset < 2000 rows
X_scaled = StandardScaler().fit_transform(df_sampled[codon_columns])
# Find optimal k using elbow method & silhouette score
k_range = range(2, 11)
inertia, silhouette_scores = [], []
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_scaled, labels))
# Plot elbow method & silhouette score
fig, ax = plt.subplots(1, 2, figsize=(10, 4))
ax[0].plot(k_range, inertia, 'bo-')
ax[0].set_xlabel('Clusters'); ax[0].set_ylabel('Inertia'); ax[0].set_title('Elbow
Method')
ax[1].plot(k_range, silhouette_scores, 'ro-')
ax[1].set_xlabel('Clusters'); ax[1].set_ylabel('Silhouette Score');
ax[1].set_title('Silhouette Analysis')
plt.show()
# Determine best k
optimal_k = k_range[silhouette_scores.index(max(silhouette_scores))]
```

```
print(f"Optimal number of clusters: {optimal_k}")
# Apply K-Means with optimal k
df_sampled['Cluster'] = KMeans(n_clusters=optimal_k, random_state=42,
n_init=10).fit_predict(X_scaled)
# Display sample output
print(df_sampled[['SpeciesName', 'Cluster']].head(10))
```

OUTPUT:




```
Optimal number of clusters: 3
                                SpeciesName  Cluster
4649                             Pseudomonas        0
379                  Human coxsackievirus B1        1
5556                    Haloarcula hispanica        0
3693          Geobacillus stearothermophilus        0
6961               Chlamydomonas reinhardtii        0
9037                    Argopecten irradians        0
5419                 Pseudomonas alcaligenes        0
6756          chloroplast Cyanidium caldarium        1
828                    Human parechovirus 3        1
2227  Tomato leaf curl Bangalore virus - [India        1
```

RESULT:

Thus the implementation for the k-means algorithm was executed successfully.

Ex. No :	06	IMPLEMENT THE NAÏVE BAYES CLASSIFIER
Date:		

AIM:

To implement the Naïve Bayes Classifier from the given dataset.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Learn how to load a dataset from a .mat file using scipy.io.loadmat.
2. Understand how the program separates the features (inputs) and labels (outputs) from the dataset.
3. Study how missing values in the dataset are handled using SimpleImputer, which fills in missing values with the column's mean.
4. Learn how the dataset is split into training and testing sets using train_test_split.
5. Understand how the Naive Bayes classifier (GaussianNB) is used to train the model with the training data.
6. Study how the trained model is used to make predictions on the test data.
7. Learn how to evaluate the model's performance by calculating accuracy and generating a classification report.
8. Understand what the classification report shows, including precision, recall, and F1 score for each class.
9. Study how the program prints out the accuracy and the classification report to assess the model's performance.

PROCEDURE:

1. Open the Jupyter Notebook and Create a new file
2. To upload the CSV file in Jupyter Notebook.
3. Type the below code and save it.
4. Run the Program and Check Bugs and Fix it.
5. Finally Observe the Output and Note it.

PROGRAM:

```
import numpy as np
import scipy.io
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer
import os

# Load the .mat file
file_path = "PersonGaitDataSet.mat"
if not os.path.exists(file_path):
    print(f"Error: {file_path} not found.")
else:
    mat_contents = scipy.io.loadmat(file_path)
    X, Y = np.array(mat_contents["X"], dtype=np.float64),
np.array(mat_contents["Y"]).ravel()

    # Handle missing values and split data
    X = SimpleImputer(strategy="mean").fit_transform(X)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)

    # Train and predict
    nb_classifier = GaussianNB().fit(X_train, Y_train)
    Y_pred = nb_classifier.predict(X_test)

    # Evaluate and print results
    print(f"Accuracy: {accuracy_score(Y_test, Y_pred):.2f}")
    print(classification_report(Y_test, Y_pred, zero_division=1))
```

OUTPUT:

```
Accuracy: 0.10
              precision    recall  f1-score   support

           1       1.00      0.00      0.00         1
           3       0.00      1.00      0.00         0
           4       1.00      0.00      0.00         1
           5       0.00      1.00      0.00         0
           6       1.00      0.00      0.00         1
           8       1.00      0.00      0.00         3
           9       1.00      0.00      0.00         1
          12       1.00      1.00      1.00         1
          13       1.00      0.00      0.00         1
          14       1.00      0.00      0.00         1

    accuracy                           0.10        10
   macro avg       0.80      0.30      0.10        10
weighted avg       1.00      0.10      0.10        10
```

RESULT:

Thus the implementation for the Naïve Bayes Classifier was executed successfully.

Ex. No :	7	PREDICTIG HEART DISEASES USING MACHINE LEARNING (PROJECT)
Date:		

AIM:

The aim is to develop a machine learning model for predicting heart disease based on patient health data. This helps in early diagnosis and better decision-making for treatment.

APPARATUS REQUIRED:

- PC
- Jupyter Notebook

ALGORITHM:

1. Import the dataset into the program.
2. Check for missing values and clean the data.
3. Separate input features and target variable.
4. Split the dataset into training (80%) and testing (20%) sets.
5. Normalize the feature values using StandardScaler.
6. Train a Random Forest Classifier using the training data.
7. Use the trained model to predict heart disease on the test data.
8. Evaluate accuracy, classification report, and feature importance.

PROCEDURE:

6. Open the Jupyter Notebook and Create a new file
7. To upload the CSV file in Jupyter Notebook.
8. Type the below code and save it.
9. Run the Program and Check Bugs and Fix it.
10. Finally Observe the Output and Note it.

PROGRAM:

```
# Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

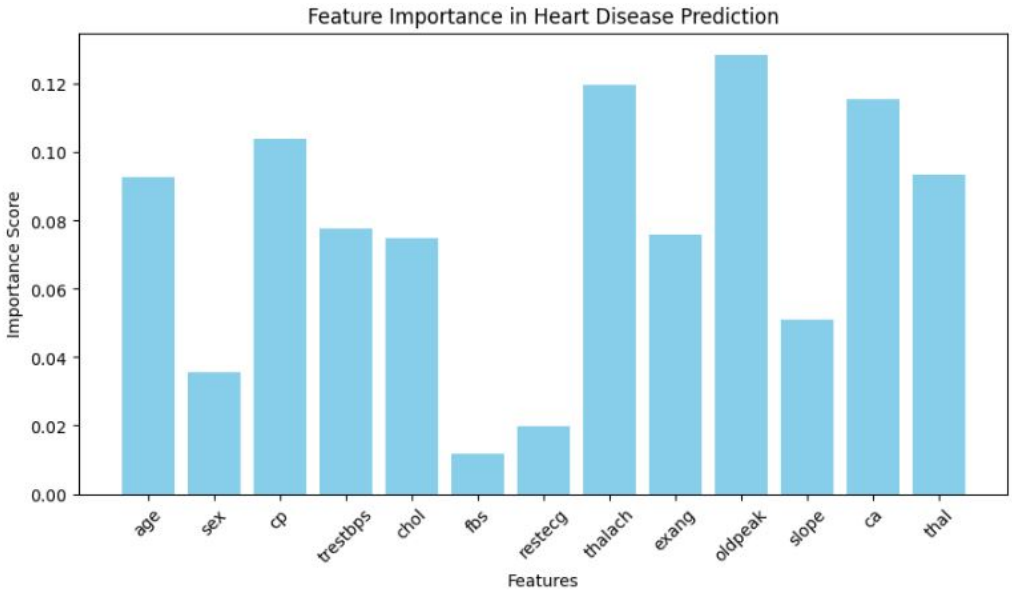
```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix


# Load the dataset

file_path = "heart_disease_dataset.csv"  # Update this if needed

df = pd.read_csv(file_path)


# Display basic dataset information

print("Dataset Shape:", df.shape)

print(df.head())


# Check for missing values

print("\nMissing Values:\n", df.isnull().sum())


# Define features and target

X = df.drop(columns=['target'])  # Assuming 'target' is the label column

y = df['target']


# Split the dataset (80% training, 20% testing)

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

```
# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
print("\nModel Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Plot feature importance
feature_importances = model.feature_importances_
plt.figure(figsize=(10, 5))
plt.bar(x=df.drop(columns=['target']).columns, height=feature_importances,
color='skyblue')
plt.xticks(rotation=45)
plt.xlabel("Features")
```

```
plt.ylabel("Importance Score")
plt.title("Feature Importance in Heart Disease Prediction")
plt.show()
```

OUTPUT:

```
Dataset Shape: (303, 14)
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1

Missing Values:
 age         0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64

 0.8360655737704918

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83        29
           1       0.84      0.84      0.84        32

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61


Confusion Matrix:
 [[24  5]
 [ 5 27]]
```



RESULT:

The heart disease prediction model was successfully implemented using a Random Forest Classifier. The model achieved good accuracy, providing reliable predictions based on the g