1. PREDICTING HOUSE PRICES

EX.N0 : 1	**Predicting House Prices**
DATE :	

PROBLEM STATEMENT: Build a regression model to predict house prices based on features like location, size, and amenities.

PYTHON CONCEPTS: Functions, classes, numeric types, sequences.

VISUALIZATION: Plotting regression line, residual plots.

MULTIVARIATE ANALYSIS: Multiple regression.

DATASET: Kaggle House Prices

ALGORITHM:

Step 1: Start the program.

Step 2: Import necessary libraries.

Step 3: Load the dataset.

Step 4: Encode categorical variable, define feature & testing set.

Step 5: Split the dataset into training & testing set, create trained model.

Step 6: Print equal metric & test the cell.

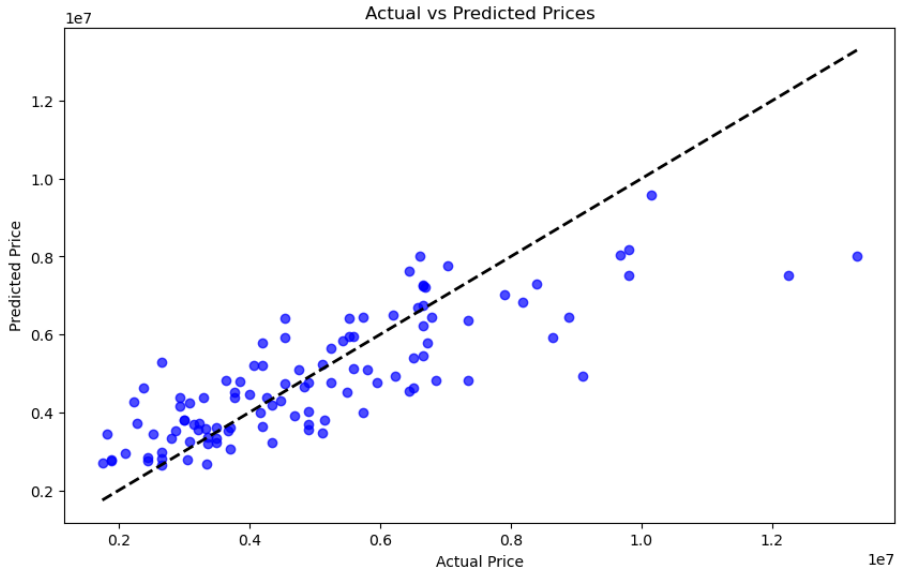
PROGRAM:

```
import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score, mean_absolute_error

import matplotlib.pyplot as plt

file_path = 'C:/Users/APPU/Downloads/Housing.csv'

housing_data = pd.read_csv(file_path)

categorical_features = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
'prefarea', 'furnishingstatus']

le = LabelEncoder()

for feature in categorical_features:

housing_data[feature] = le.fit_transform(housing_data[feature])

X = housing_data.drop('price', axis=1)y = housing_data['price']

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)

r2 = r2_score(y_test, y_pred)

mae = mean_absolute_error(y_test, y_pred)

plt.figure(figsize=(10, 6))

plt.scatter(y_test, y_pred, alpha=0.7, color='b')

plt.plot([y_test.min(), y_test.max()],

[y_test.min(), y_test.max()], 'k--', lw=2)

plt.xlabel('Actual Price')

plt.ylabel('Predicted Price')

plt.title('Actual vs Predicted Prices')

plt.show()
```

print(f'R-squared (R^2): {r2}')

print(f'Mean Absolute Error (MAE): {mae}')




```
import numpy as np
test=np.array([ 7420,4,2,3,1,0,0,0,1,2,1,0]).reshape(-12,12)
model.predict(test)
```

```
array([8004072.41154001])
```

RESULT:

Thus, the program for house price prediction is executed successfully.

2. CUSTOMER SEGMENTATION FOR AN E-COMMERCE COMPANY

EX.N0 : 2	**Customer Segmentation for an E-commerce Company**
DATE :	

PROBLEM STATEMENT: Perform cluster analysis to segment customers based on purchasing behaviour.

PYTHON CONCEPTS: Data structures, file reading/writing.

VISUALIZATION: Cluster plots.

MULTIVARIATE ANALYSIS: Cluster analysis with k-means, hierarchical clustering.

DATASET: Online Retail Dataset

ALGORITHM:

Step 1: Start the program.

Step 2: Import necessary libraries.

Step 3: Load the dataset.

Step 4: Encode categorical variable, define feature & testing set.

Step 5: Split the dataset into training & testing set, create trained model.

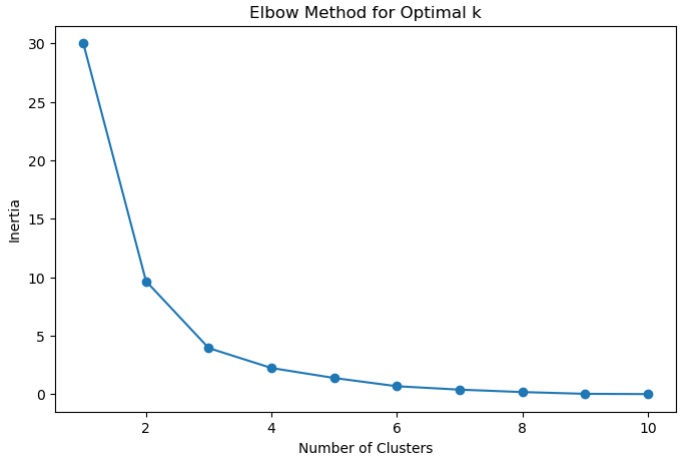
Step 6: Print equal metric & test the cell.

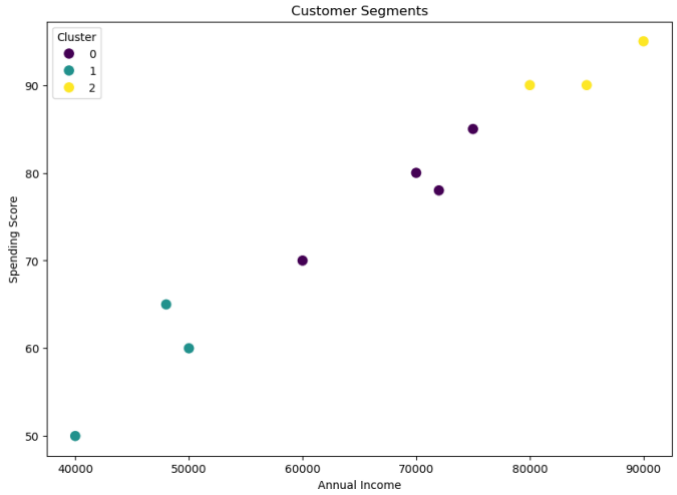
PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

```
os.environ['OMP_NUM_THREADS'] = '1'
data = {'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
'Age': [25, 45, 35, 50, 23, 33, 43, 36, 29, 55],
'AnnualIncome': [50000, 60000, 70000, 80000, 40000, 75000, 85000, 72000, 48000, 90000],
'SpendingScore': [60, 70, 80, 90, 50, 85, 90, 78, 65, 95] }
df = pd.DataFrame(data)
features = df[['Age', 'AnnualIncome', 'SpendingScore']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features) inertia = []
k_range = range(1, 11) for k in k_range:
kmeans = KMeans(n_clusters=k, n_init=10, random_state=0)
kmeans.fit(scaled_features)
inertia.append(kmeans.inertia_) plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters') plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k') plt.show() optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, n_init=10, random_state=0)
df['Cluster'] = kmeans.fit_predict(scaled_features)
plt.figure(figsize=(10, 7))
sns.scatterplot(data=df, x='AnnualIncome', y='SpendingScore', hue='Cluster', palette='viridis',
s=100)
plt.title('Customer Segments')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend(title='Cluster')
plt.show()
print(df)
```

OUTPUT:





	CustomerID	Age	AnnualIncome	SpendingScore	Cluster
0	1	25	50000	60	1
1	2	45	60000	70	0
2	3	35	70000	80	0
3	4	50	80000	90	2
4	5	23	40000	50	1
5	6	33	75000	85	0
6	7	43	85000	90	2
7	8	36	72000	78	0
8	9	29	48000	65	1
9	10	55	90000	95	2

RESULT:

Thus, the program for Customer Segmentation for an E-commerce Company is executed successfully.

3. SENTIMENT ANALYSIS OF MOVIE REVIEWS

EX.N0 : 3	**SENTIMENT ANALYSIS OF MOVIE REVIEWS**
DATE :	

PROBLEM STATEMENT: Classify movie reviews as positive or negative using text Data.

PYTHON CONCEPTS: Text files, sequences, flow controls.

VISUALIZATION: Word cloud, bar plots.

MULTIVARIATE ANALYSIS: PCA for text data, logistic regression.

DATASET: IMDB Movie Reviews.

ALGORITHM:

Step 1: Start the program.

Step 2: Import necessary libraries.

Step 3: Load the dataset.

Step 4: Encode categorical variable, define feature & testing set.

Step 5: Split the dataset into training & testing set, create trained model.

Step 6: Print equal metric & test the cell.

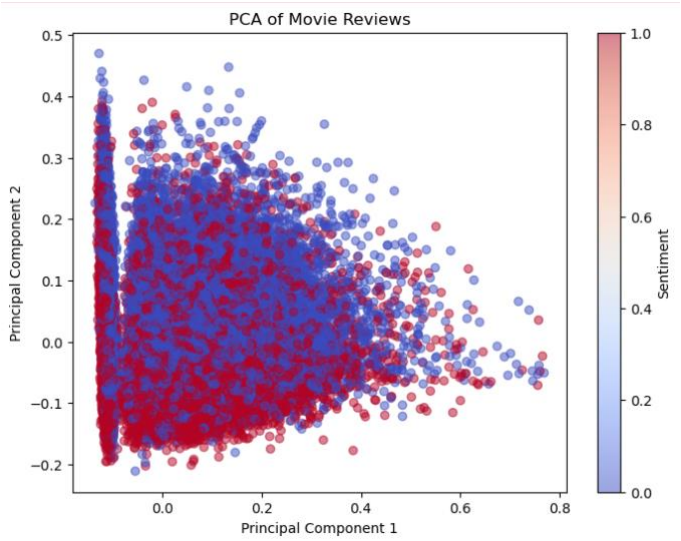
PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import seaborn as sns
nltk.download('punkt')
nltk.download('stopwords')
df = pd.read_csv('C:/Users/AI_LAB/Downloads/IMDB Dataset.csv')
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
def preprocess_text(text):
tokens = word_tokenize(text.lower())
tokens = [stemmer.stem(word) for word in tokens if word.isalpha() and word not in stop_words]
return ' '.join(tokens)
df['cleaned_review'] = df['review'].apply(preprocess_text)
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['cleaned_review']).toarray()
encoder = LabelEncoder()
y = encoder.fit_transform(df['sentiment'])
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='coolwarm', alpha=0.5)
plt.title('PCA of Movie Reviews')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Sentiment')
plt.show()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
positive_reviews = ' '.join(df[df['sentiment'] == 1]['cleaned_review'])
negative_reviews = ' '.join(df[df['sentiment'] == 0]['cleaned_review'])
plt.figure(figsize=(12, 6))
if len(positive_reviews.strip()) > 0:
plt.subplot(1, 2, 1)
plt.imshow(WordCloud(width=800, height=400,
background_color='white').generate(positive_reviews), interpolation='bilinear')
plt.title('Positive Reviews')
plt.axis('off')
else: print("No content available for positive reviews.")
if len(negative_reviews.strip()) > 0:
plt.subplot(1, 2, 2)
plt.imshow(WordCloud(width=800, height=400,
background_color='white').generate(negative_reviews), interpolation='bilinear')
plt.title('Negative Reviews')
plt.axis('off') else:
print("No content available for negative reviews.")
plt.show()
sns.countplot(x='sentiment', data=df)
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

OUTPUT:




```
Confusion Matrix:
[[4306  655]
 [ 511 4528]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      4961
           1       0.87      0.90      0.89      5039

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

RESULT:

Thus, the program for sentiment analysis of movie reviews is executed successfully.

4. STOCK MARKET ANALYSIS

EX.N0 : 4	**STOCK MARKET ANALYSIS**
DATE :	

PROBLEM STATEMENT: Analyse stock market data to predict future stock prices.

PYTHON CONCEPTS: Data structures, file reading/writing, functions.

VISUALIZATION: Line plots, candlestick charts.

MULTIVARIATE ANALYSIS: Time series analysis, regression.

DATASET: Yahoo Finance Stock Data.

ALGORITHM:

Step 1: Start the program.

Step 2: Import necessary libraries.

Step 3: Load the dataset.

Step 4: Encode categorical variable, define feature & testing set.

Step 5: Split the dataset into training & testing set, create trained model.

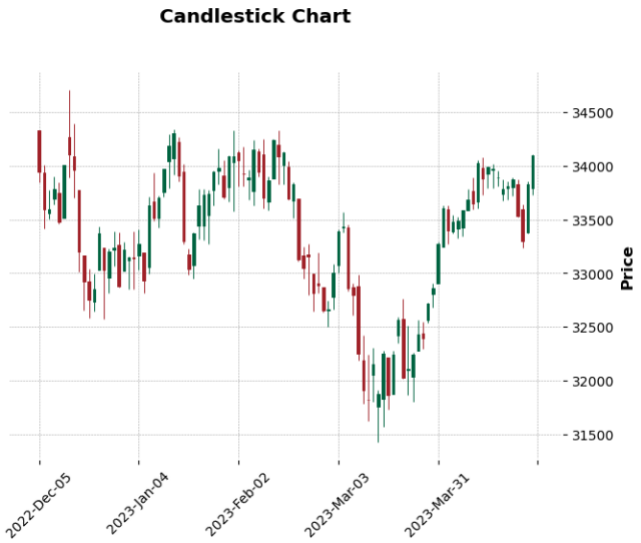
Step 6: Print equal metric & test the cell.

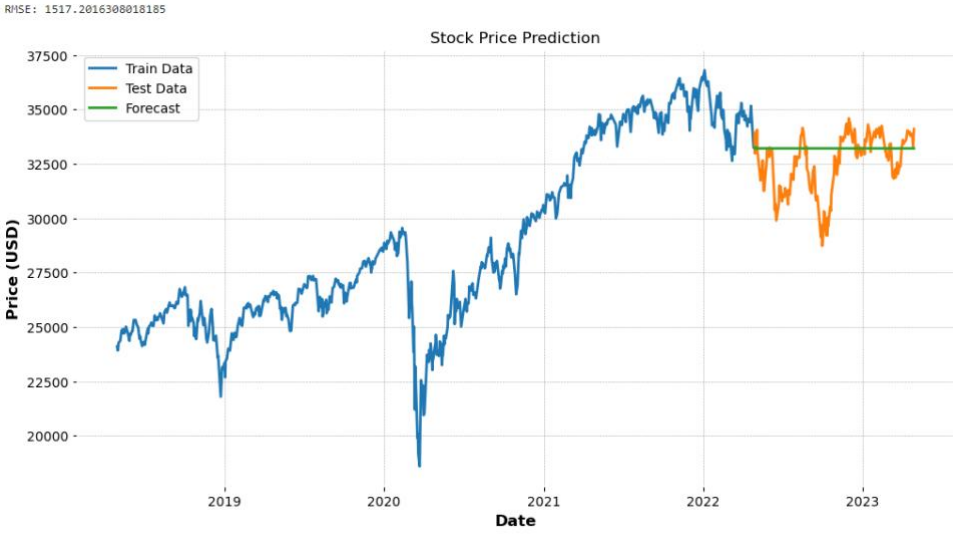
PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
import mplfinance as mpf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import numpy as np
```

```
file_path = r'C:\Users\APPU\Downloads\yahoo_data.xlsx'
data = pd.read_excel(file_path, index_col='Date', parse_dates=True)
data.rename(columns={'Close*': 'Close', 'Adj Close**': 'Adj Close'}, inplace=True)
data.sort_index(inplace=True)
data.ffill(inplace=True)
if 'Adj Close' in data.columns:
plt.figure(figsize=(12, 6))
plt.plot(data['Adj Close'], label='Adjusted Close Price')
plt.title('Adjusted Close Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()
reduced_data = data[-100:]  # Reduce data points for candlestick chart
mpf.plot(reduced_data, type='candle', style='charles', title='Candlestick Chart')
train_data, test_data = data['Adj Close'][:int(len(data)*0.8)], data['Adj Close'][int(len(data)*0.8):]
model = ARIMA(train_data, order=(5, 1, 0))
model_fit = model.fit()
forecast = model_fit.forecast(steps=len(test_data))
mse = mean_squared_error(test_data, forecast)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')
plt.figure(figsize=(12, 6))
plt.plot(train_data.index, train_data, label='Train Data')
plt.plot(test_data.index, test_data, label='Test Data')
plt.plot(test_data.index, forecast, label='Forecast')
plt.title('Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()
```

OUTPUT:





RESULT:

Thus, the program for stock market analysis is executed successfully.

5. LOAN DEFAULT PREDICTION

EX.N0 : 5	**LOAN DEFAULT PREDICTION**
DATE :	

PROBLEM STATEMENT: Predict loan default probability based on borrower information.

PYTHON CONCEPTS: Classes, functions, sequences.

VISUALIZATION: ROC curve, bar plots.

MULTIVARIATE ANALYSIS: Logistic regression, factor analysis.

DATASET: Lending Club Loan Data

ALGORITHM:

Step 1: Start the program.

Step 2: Import necessary libraries.

Step 3: Load the dataset.

Step 4: Encode categorical variable, define feature & testing set.

Step 5: Split the dataset into training & testing set, create trained model.

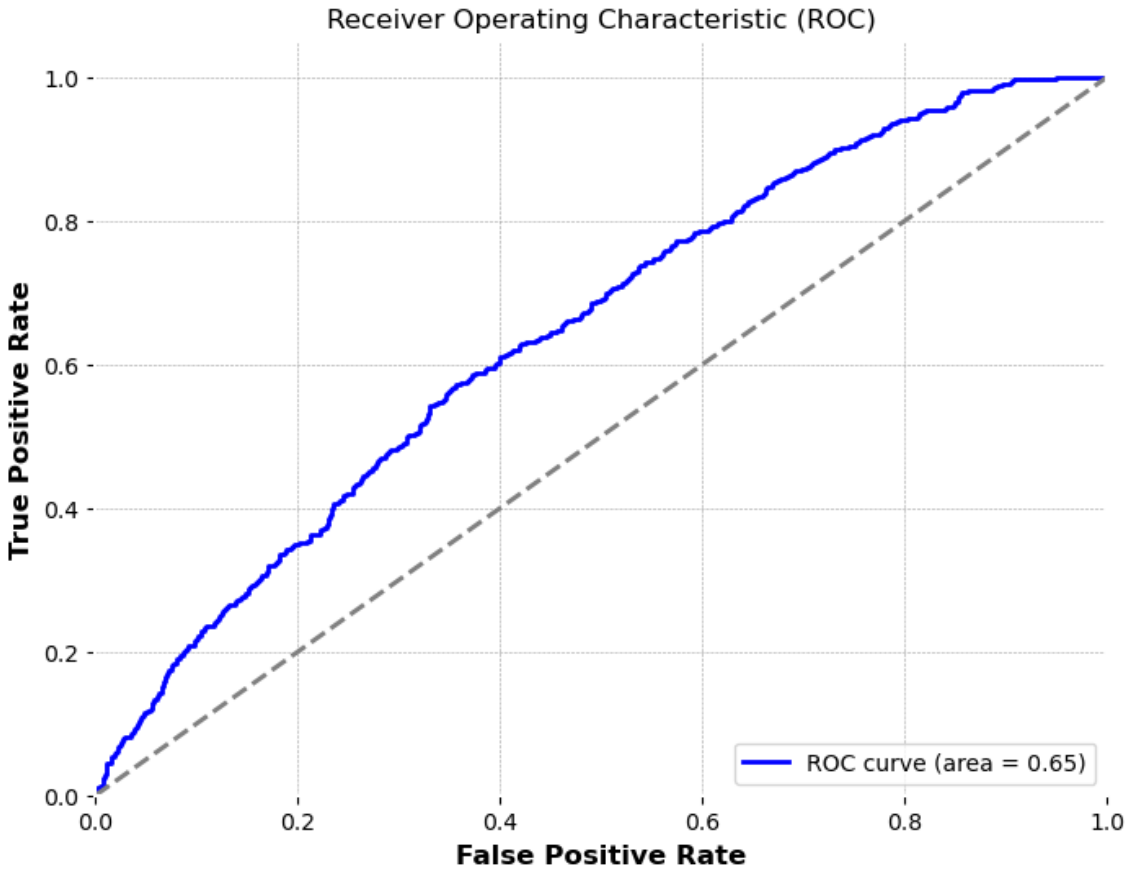
Step 6: Print equal metric & test the cell.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import os
```

```
file_path = 'C:/Users/APPU/Downloads/loan_data.csv'  # Update path accordingly
if os.path.exists(file_path):
df = pd.read_csv(file_path)
print("Data loaded successfully.") else:
print(f"File not found: {file_path}")
dummies = pd.get_dummies(df['purpose'], drop_first=True)
df = pd.concat([df, dummies], axis=1)
df.drop('purpose', inplace=True, axis=1)
X = df.drop(['not.fully.paid'], axis=1)
y = df['not.fully.paid']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.33, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred_prob = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```

OUTPUT:



RESULT:

Thus, the program for loan default prediction is executed successfully.

6. IMAGE CLASSIFICATION

EX.N0 : 6	**IMAGE CLASSIFICATION**
DATE :	

PROBLEM STATEMENT: Classify images into categories using various features.

PYTHON CONCEPTS: File handling, classes.

VISUALIZATION: Image plots, feature importance plots.

MULTIVARIATE ANALYSIS: PCA, clustering.

DATASET: CIFAR-10 Dataset

ALGORITHM:

Step 1: Start the program.

Step 2: Import necessary libraries.

Step 3: Load the dataset.

Step 4: Encode categorical variable, define feature & testing set.

Step 5: Split the dataset into training & testing set, create trained model.

Step 6: Print equal metric & test the cell.

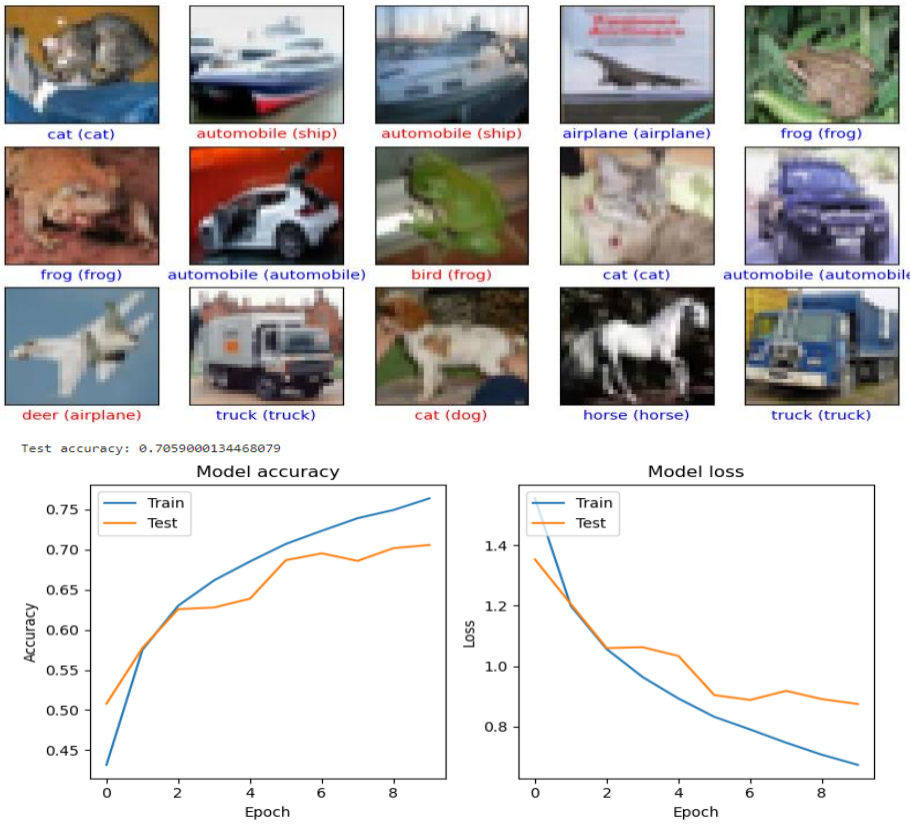
PROGRAM:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
```

```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.cifar10.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,10))
for i in range(25): plt.subplot(5,5,i+1)
plt.xticks([]) plt.yticks([]) plt.grid(False)
plt.imshow(X_train[i], cmap=plt.cm.binary)
plt.xlabel(class_names[y_train[i][0]])
plt.show() model = models.Sequential([
layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
layers.MaxPooling2D((2, 2)),
layers.Conv2D(64, (3, 3), activation='relu'),
layers.MaxPooling2D((2, 2)),
layers.Conv2D(64, (3, 3), activation='relu'),
layers.Flatten(), layers.Dense(64, activation='relu'),
layers.Dense(10)    ]) model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=10,
validation_data=(X_test, y_test))
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f"\nTest accuracy: {test_acc}")
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1) plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy') plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.subplot(1, 2, 2) plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss') plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.tight_layout() plt.show()
```

```
predictions = model.predict(X_test)
plt.figure(figsize=(10, 10))
for i in range(25): plt.subplot(5, 5, i+1)
plt.xticks([]) plt.yticks([]) plt.grid(False)
plt.imshow(X_test[i], cmap=plt.cm.binary)
predicted_label = np.argmax(predictions[i])
true_label = y_test[i][0]
color = 'blue' if predicted_label == true_label else 'red'
plt.xlabel(f"{class_names[predicted_label]} ({class_names[true_label]})", color=color)
plt.show()
```

OUTPUT:



RESULT:

Thus, the program for Image Classification is executed successfully.