# 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:

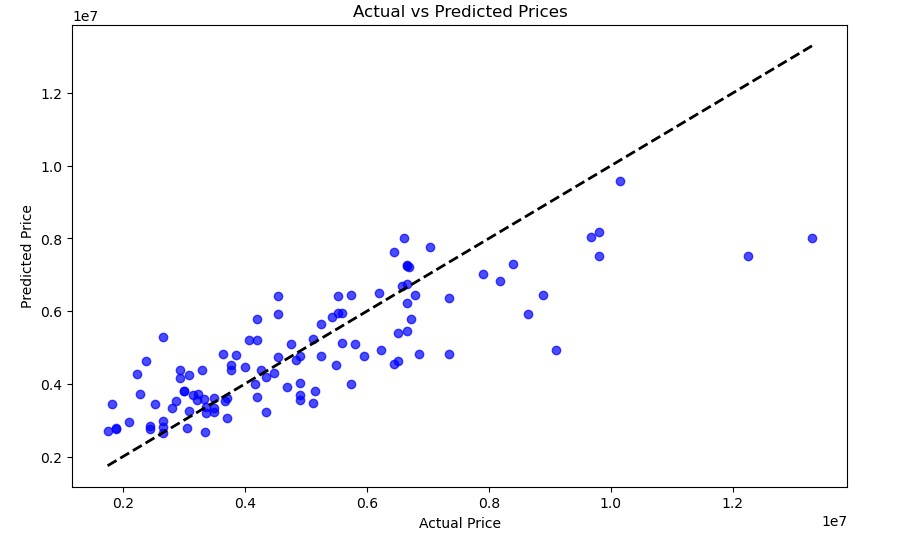
housing\_data[feature] = le.fit\_transform(housing\_data[feature])

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 (R²): {r2}') print(f'Mean Absolute Error (MAE): {mae}')







**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],

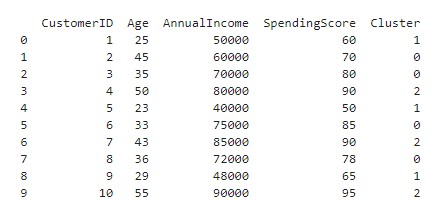
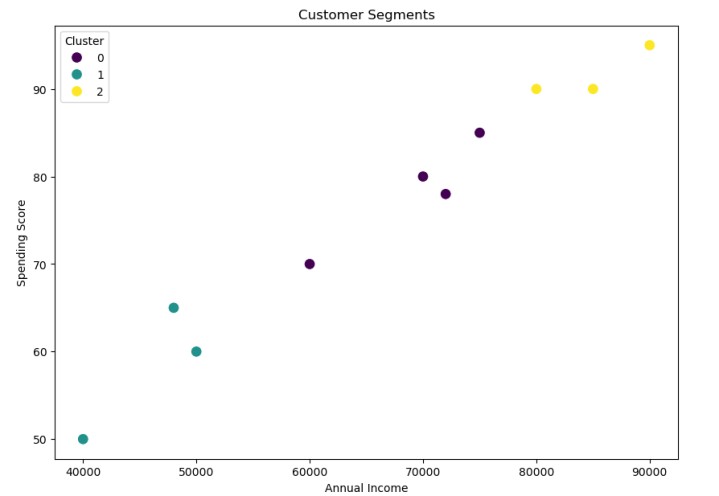
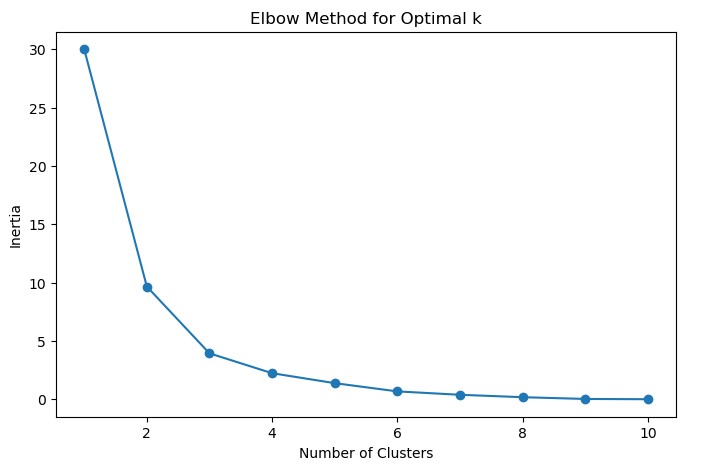
'AnnualIncome': [50000, 60000, 70000, 80000, 40000, 75000, 85000, 72000, 48000, 90000],

'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:**



**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):

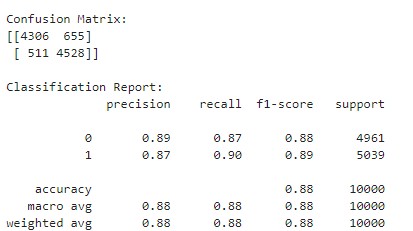
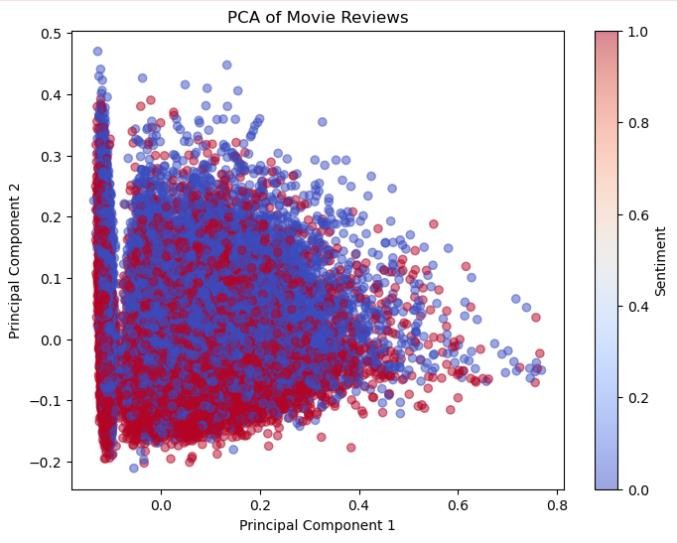
tokens = word\_tokenize(text.lower()) tokens = [stemmer.stem(word) for word in tokens if word.isalpha() and word not in stop\_words] return ' '.join(tokens) df['cleaned\_review'] = df['review'].apply(preprocess\_text) vectorizer = TfidfVectorizer(max\_features=5000)

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()            COMPUTATIONAL STATISTICS | 221501025 |

**OUTPUT:**



**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.

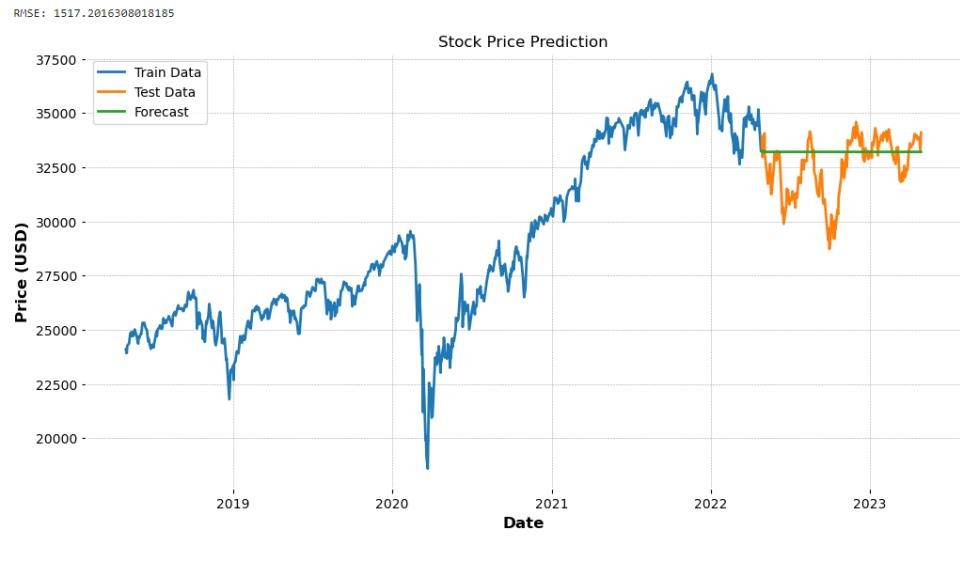
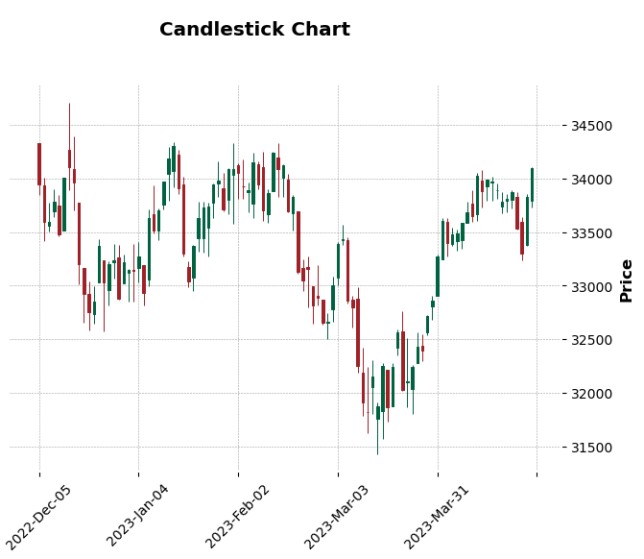
**PROGRAM:**

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:**

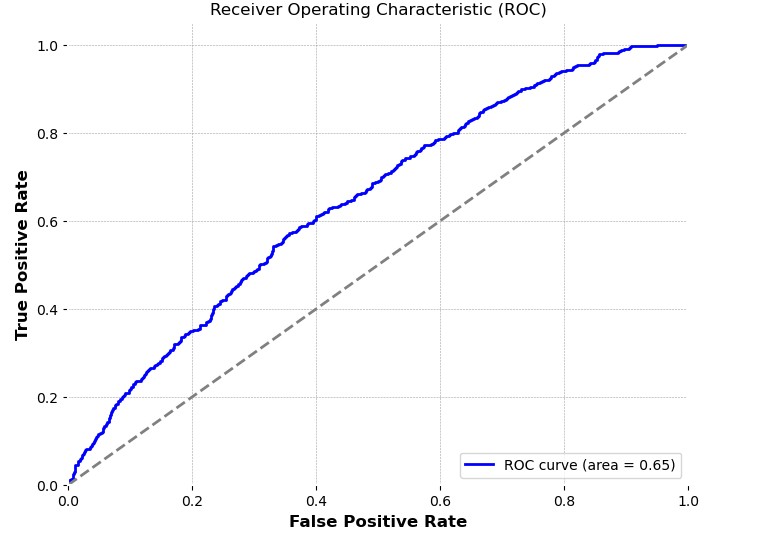
import pandas as pd import matplotlib.pyplot as plt import seaborn as sns from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LogisticRegression from sklearn.metrics import roc\_curve, auc from sklearn.preprocessing import StandardScaler from sklearn.decomposition 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()  COMPUTATIONAL STATISTICS | 221501025 |

COMPUTATIONAL STATISTICS

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**

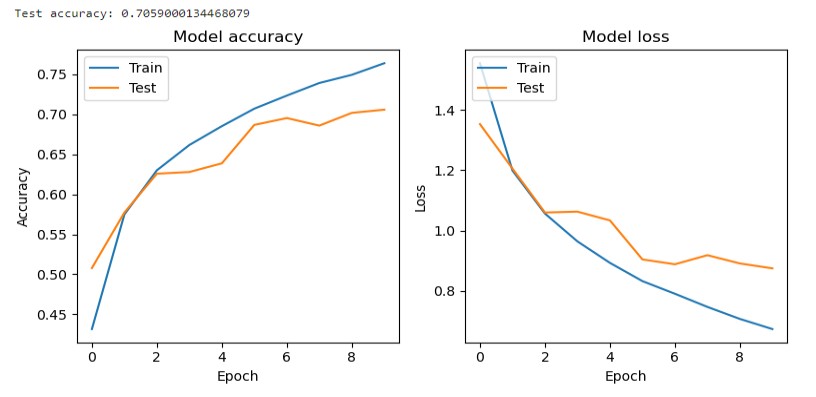
**:**

Thus,

the program for

Image Classification

is executed successfully.



221501025