EXP.NO:1	IRIS FLOWER CLASSIFICATION USING LOGISTIC REGRESSION
DATE:	

Problem Statement:

The Iris dataset is used to classify different species of the Iris flower based on several features. This experiment aims to build a logistic regression model to classify Iris flowers into three species based on the petal and sepal dimensions.

Objective:

1. Explore the Iris dataset and its structure to identify key patterns.
2. Preprocess the data by encoding the target variables.
3. Develop a logistic regression model to classify the species of Iris flowers.
4. Evaluate the model's performance using metrics such as accuracy, confusion matrix, and classification report.
5. Visualize decision boundaries for better interpretation.

Dataset Description:

The Iris dataset contains the following features:

- Sepal Length: Length of the sepal in centimeters.
- Sepal Width: Width of the sepal in centimeters.
- Petal Length: Length of the petal in centimeters.
- Petal Width: Width of the petal in centimeters.
- Species: The target variable indicating the species of the Iris flower (Setosa, Versicolor, Virginica)

Implementation:

1. **Importing Necessary Libraries**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import numpy as np
import seaborn as sns
```

2. **Load the Dataset**

```
file_path = '/content/IRIS.csv'
data = pd.read_csv(file_path)
```

3. Exploratory Data Analysis

```
print("First 5 rows of the dataset:")
print(data.head())
print("\nMissing values in each column:")
print(data.isnull().sum())
print("\nStatistical summary of the dataset:")
print(data.describe())
print("\nDistribution of species:")
print(data['species'].value_counts())
```

4. **Boxplot to visualize feature distributions by species**

```
sns.boxplot(x='species', y='sepal_width', data=data)
plt.title('Sepal Width Distribution by Species')
plt.show()

sns.boxplot(x='species', y='petal_length', data=data)
plt.title('Petal Length Distribution by Species')
plt.show()
```

5. **Visualize the Pairplot**

```
sns.pairplot(data, hue='species')
plt.suptitle("Pairplot of Iris Dataset", y=1.02)
plt.show()
```

6. **Calculating Means of Columns**

```
X = data.drop('species', axis=1)
y = data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

7. **Logistic Regression**

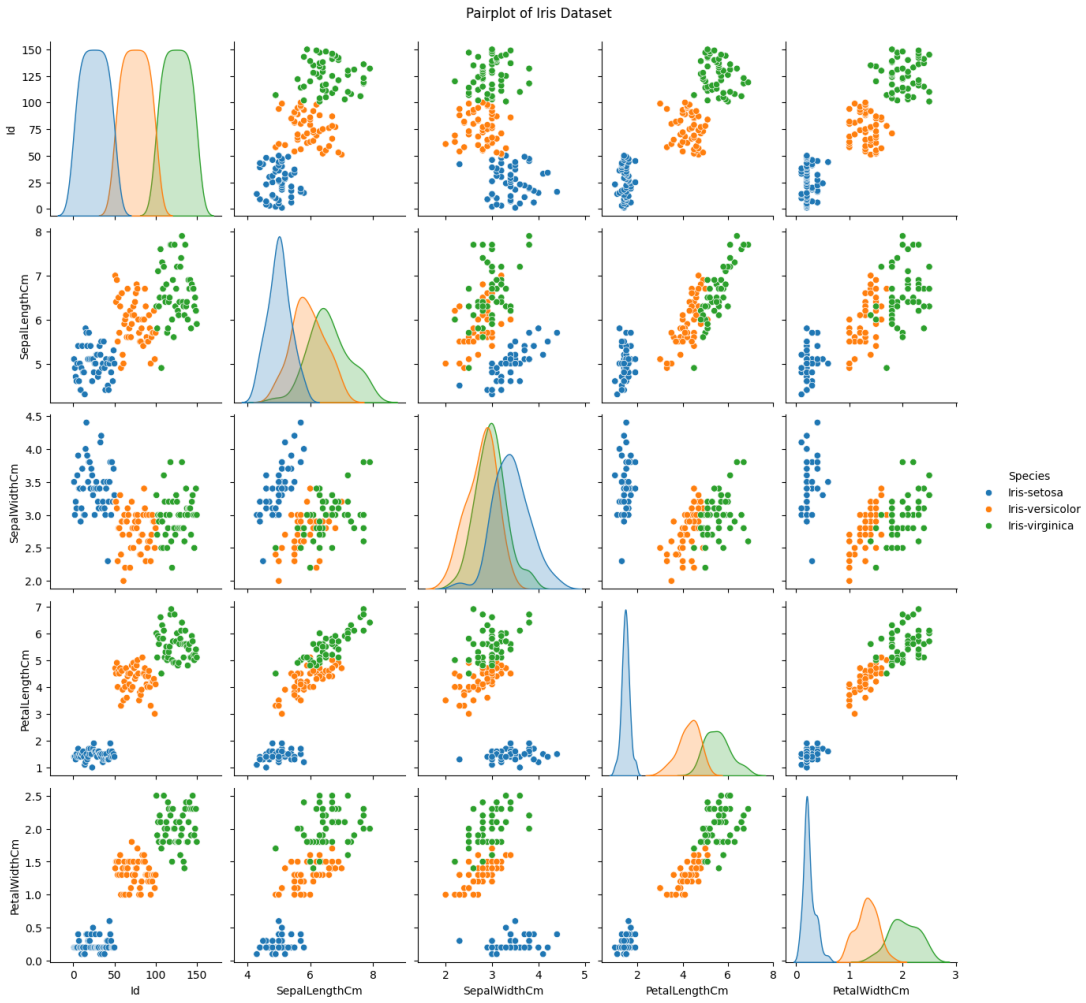
```
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

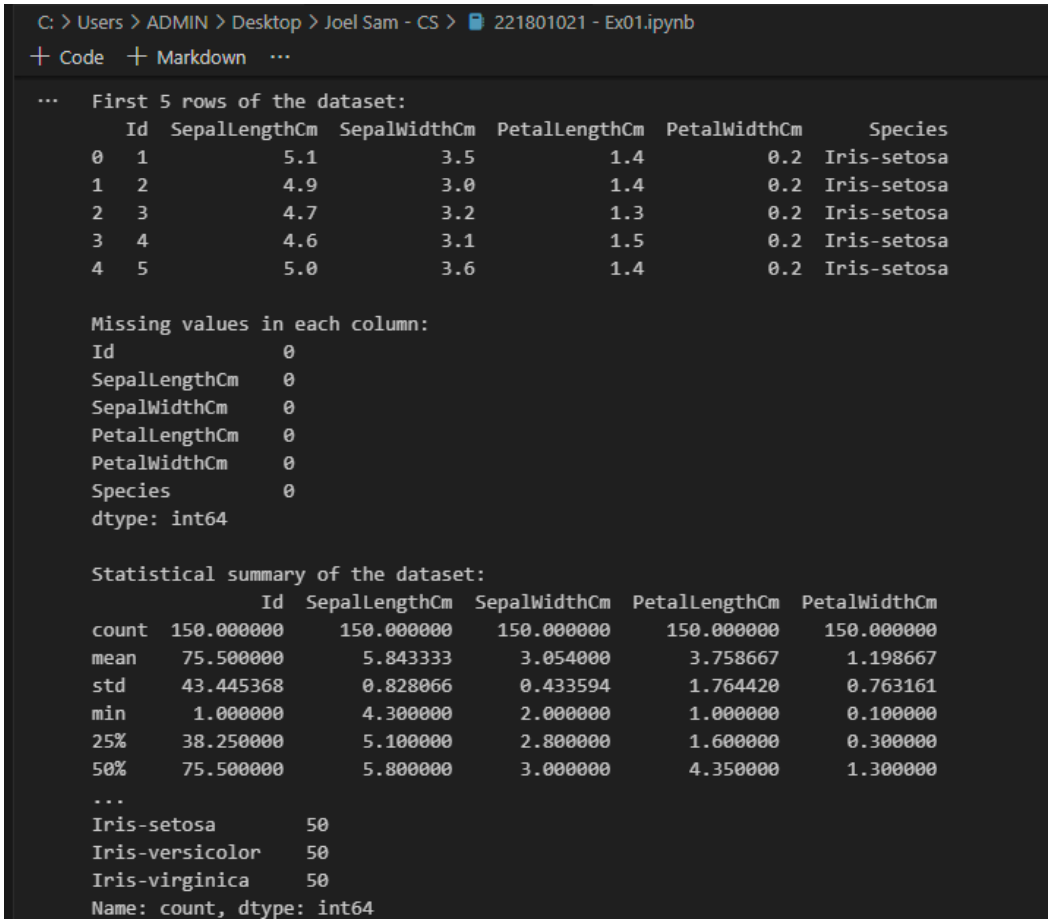
```
print(f'Accuracy: {accuracy * 100:.2f}%')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)
```

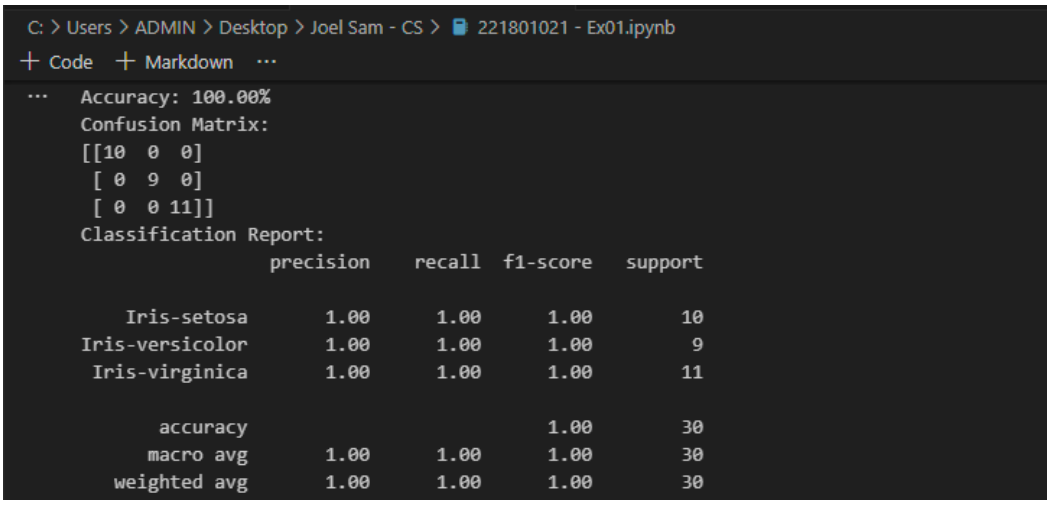
8. **Confusion Matrix**

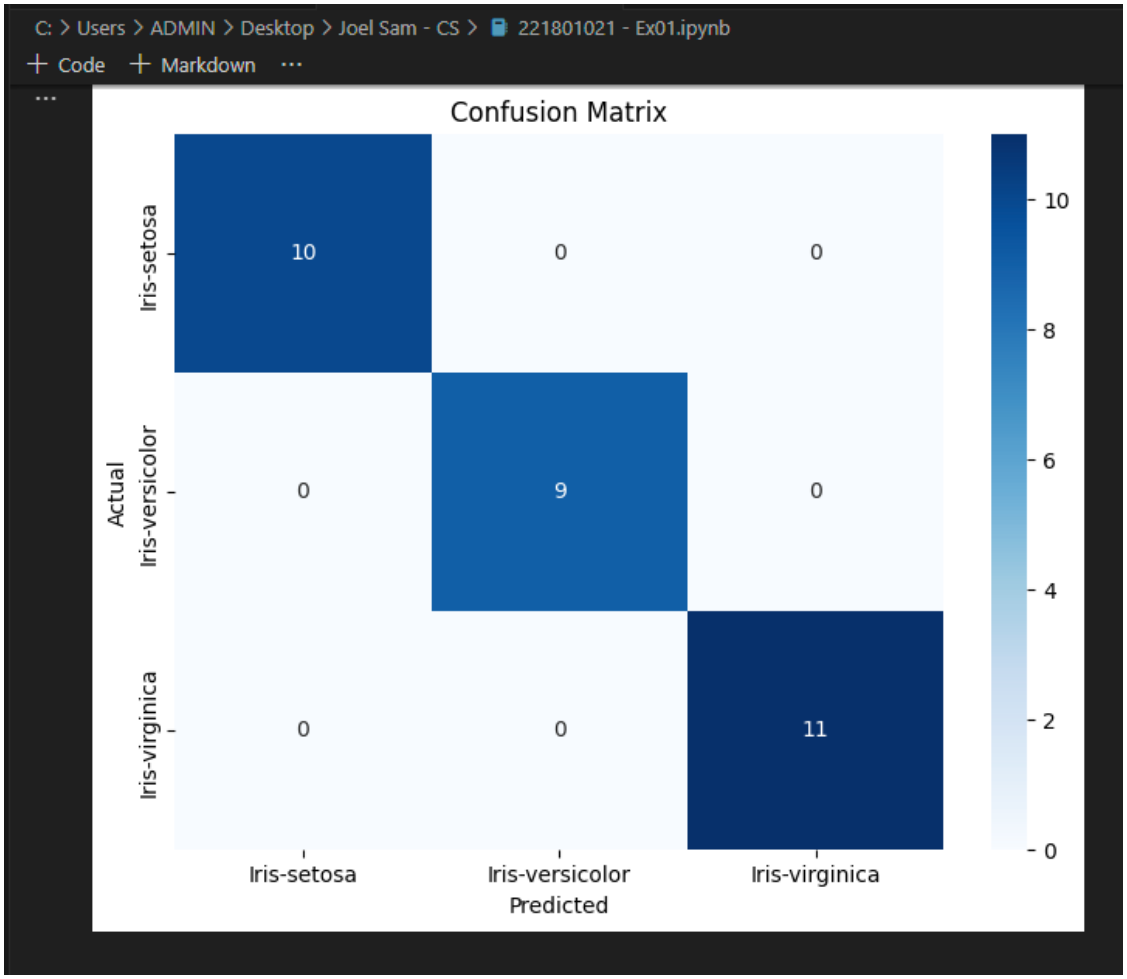
```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                                    xticklabels=model.classes_,yticklabels=model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output:




C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex01.ipynb
+ Code + Markdown ···
··· First 5 rows of the dataset:
Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species
0 1 5.1 3.5 1.4 0.2 Iris-setosa
1 2 4.9 3.0 1.4 0.2 Iris-setosa
2 3 4.7 3.2 1.3 0.2 Iris-setosa
3 4 4.6 3.1 1.5 0.2 Iris-setosa
4 5 5.0 3.6 1.4 0.2 Iris-setosa
Missing values in each column:
Id 0
SepalLengthCm 0
SepalWidthCm 0
PetalLengthCm 0
PetalWidthCm 0
Species 0
dtype: int64
Statistical summary of the dataset:
Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm
count 150.000000 150.000000 150.000000 150.000000 150.000000
mean 75.500000 5.843333 3.054000 3.758667 1.198667
std 43.445368 0.828066 0.433594 1.764420 0.763161
min 1.000000 4.300000 2.000000 1.000000 0.100000
25% 38.250000 5.100000 2.800000 1.600000 0.300000
50% 75.500000 5.800000 3.000000 4.350000 1.300000
...
Iris-setosa 50
Iris-versicolor 50
Iris-virginica 50
Name: count, dtype: int64


C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex01.ipynb
+ Code + Markdown ···
··· Accuracy: 100.00%
Confusion Matrix:
[[10 0 0]
[0 9 0]
[0 0 11]]
Classification Report:
precision recall f1-score support
Iris-setosa 1.00 1.00 1.00 10
Iris-versicolor 1.00 1.00 1.00 9
Iris-virginica 1.00 1.00 1.00 11
accuracy 1.00 30
macro avg 1.00 1.00 1.00 30
weighted avg 1.00 1.00 1.00 30



Result:

Thus the Logistic Regression model with exploratory data analysis achieved 100% accuracy in classifying the Iris species. The confusion matrix showed minimal misclassifications, mostly between Versicolor and Virginica. The classification report indicated high precision, recall, and F1-scores for all species. The pairplot and confusion matrix visually confirmed strong feature separation and model performance.

EXP.NO:2	HOUSE PRICE PREDICTION USING LINEAR REGRESSION
DATE:	

Problem Statement:

The real estate market is characterized by its dynamic nature, influenced by various factors such as location, property features, and economic conditions. Accurately predicting housing prices is crucial for buyers, sellers, and real estate professionals to make informed decisions.This experiment aims to develop a predictive model using linear regression to estimate housing prices based on various features.

Objectives:

1. Explore the housing dataset to understand its patterns and distributions, and identify missing values and outliers.
2. Preprocess the data by converting categorical variables into numerical formats suitable for modeling and select relevant features that significantly impact housing prices.
3. Develop a linear regression model to predict housing prices based on the selected features.
4. Evaluate the regression model's performance using Mean Squared Error (MSE) and R-squared, and visualize the relationship between actual and predicted prices.
5. Provide insights into the factors that most significantly influence housing prices and assess the model's predictive capabilities.

Dataset Description:

The Housing dataset is designed to facilitate the prediction of housing prices based on several key features. It typically includes the following attributes:

- **Area:** The total area of the house, often measured in square feet.
- **Price:** The target variable representing the housing price.
- **Bedrooms:** The number of bedrooms in the house.
- **Bathrooms:** The number of bathrooms in the house.
- **Stories:** The number of stories (floors) in the house.
- **Parking:** The availability of parking spaces.
- **Mainroad:** A binary variable indicating proximity to a main road.
- **Guestroom:** A binary variable indicating the presence of a guest room.
- **Basement:** A binary variable indicating whether the house has a basement.

- **Hotwaterheating:** A binary variable indicating the presence of hot water heating.
- **Airconditioning:** A binary variable indicating whether the house has air conditioning.
- **Prefarea:** A binary variable indicating whether the house is located in a preferred area.

Implementation:

1. **Importing Libraries**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

2. **Loading the Data**

```
df = pd.read_csv("/content/Housing.csv")
```

3. **Exploratory Data Analysis (EDA)**

```
df.head()
df.info()
df.describe()
```

4. **Checking for Missing Values**

```
print(df.isnull().sum())
```

5. **Outlier Analysis**

```
fig, axs = plt.subplots(2,3, figsize = (10,5))
plt1 = sns.boxplot(df['price'], ax = axs[0,0])
plt2 = sns.boxplot(df['area'], ax = axs[0,1])
plt3 = sns.boxplot(df['bedrooms'], ax = axs[0,2])
plt1 = sns.boxplot(df['bathrooms'], ax = axs[1,0])
plt2 = sns.boxplot(df['stories'], ax = axs[1,1])
plt3 = sns.boxplot(df['parking'], ax = axs[1,2])
plt.tight_layout()
```

6. **Pair Plot**

```
sns.pairplot(df)
plt.show()
```

7. **Data Preprocessing**

Encoding Categorical Variables

```
features = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']
df_encoded = pd.get_dummies(df[features], drop_first=True)
```

Adding Target Variable

```
df_encoded['price'] = df['price']
```

8. **Splitting the Data**

```
X = df_encoded.drop('price', axis=1)
y = df_encoded['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

9. **Building the Regression Model**

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

10. **Making Predictions**

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

11. **Evaluating the Model**

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

12. **Visualization of Actual vs. Predicted Prices**

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)
plt.title('Actual vs Predicted Prices')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.show()
```

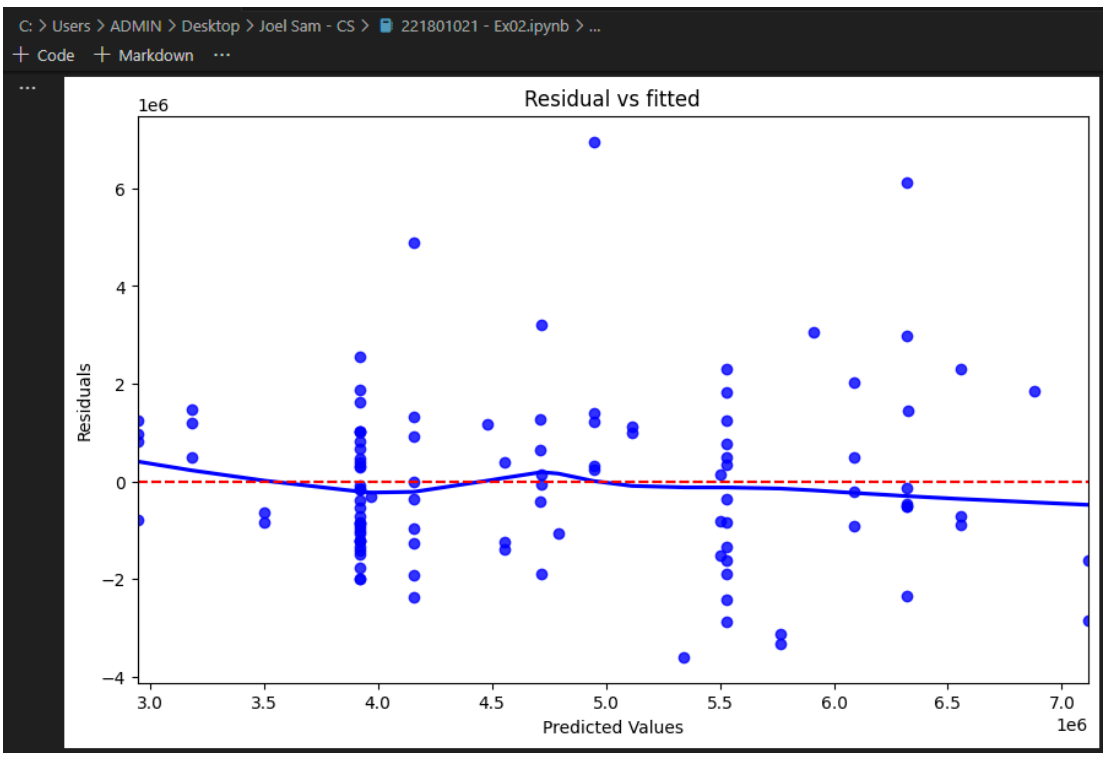
13. **Residual Plot**

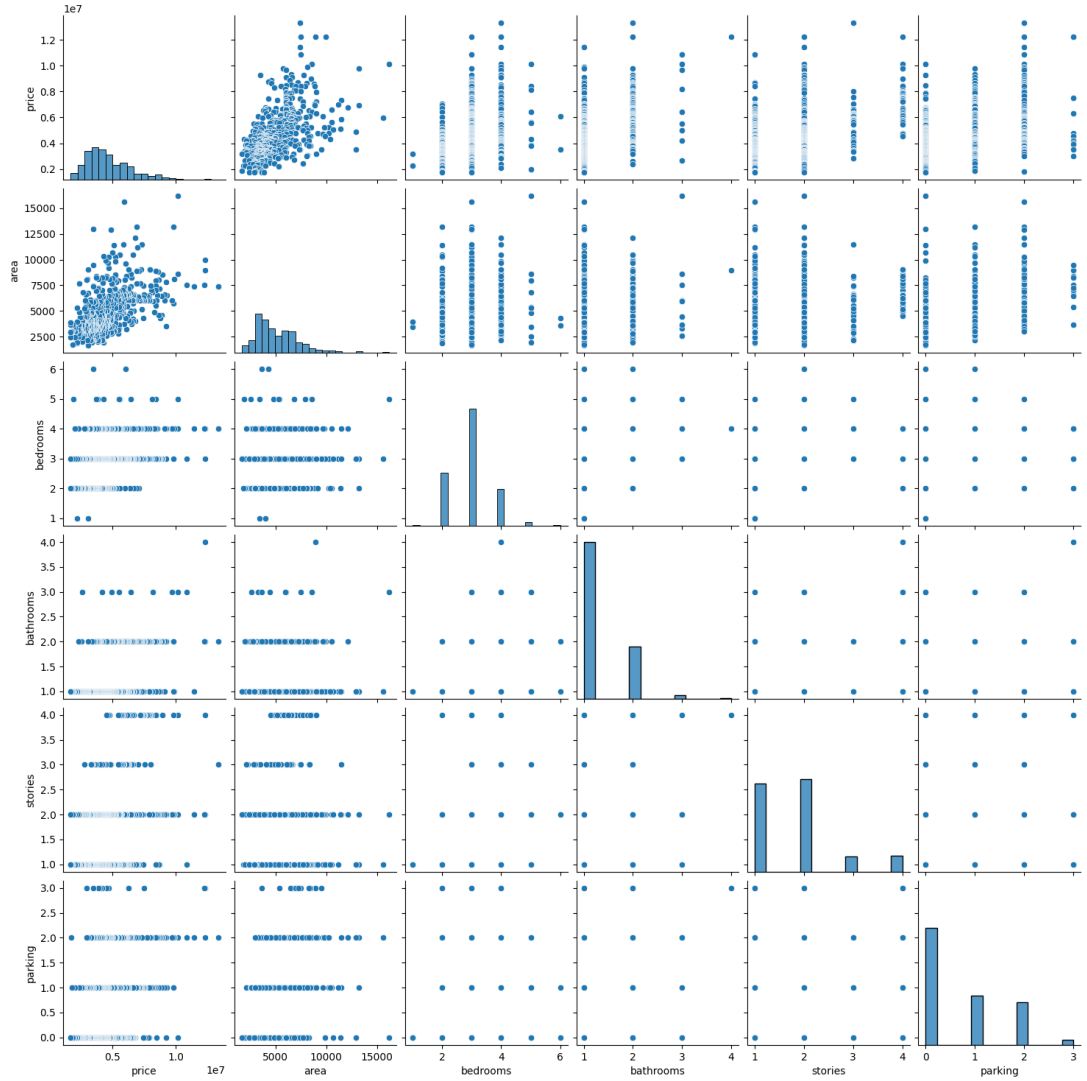
```
plt.figure(figsize=(10, 6))
sns.residplot(x=y_pred, y=y_test - y_pred, lowess=True, color='g')
plt.title('Residuals vs Fitted')
plt.xlabel('Fitted values')
```

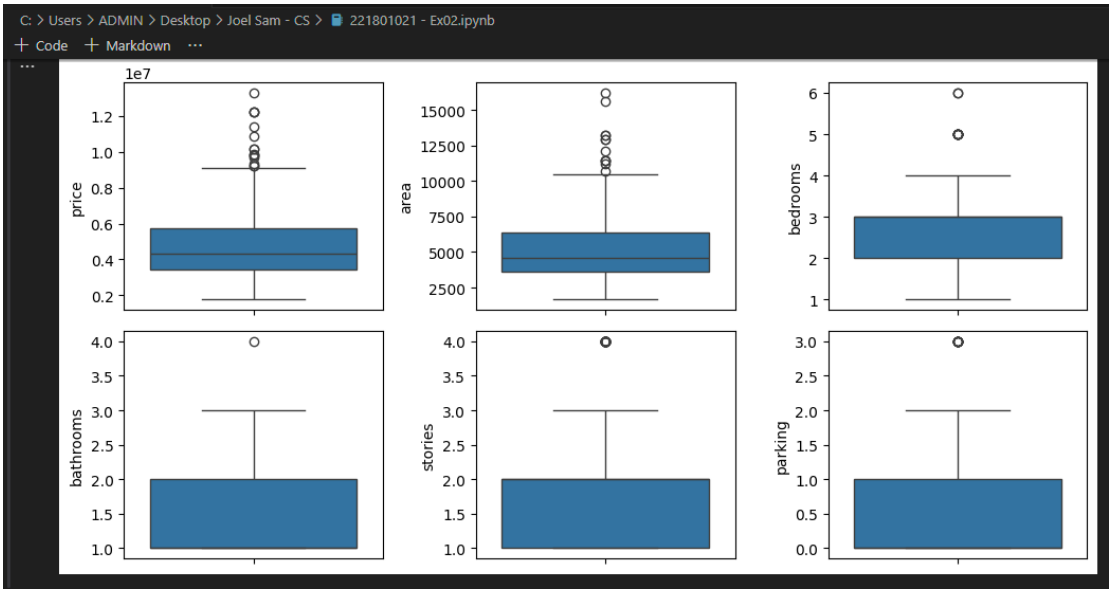
```
plt.ylabel('Residuals')
plt.axhline(0, linestyle='--', color='red')
plt.show()
```

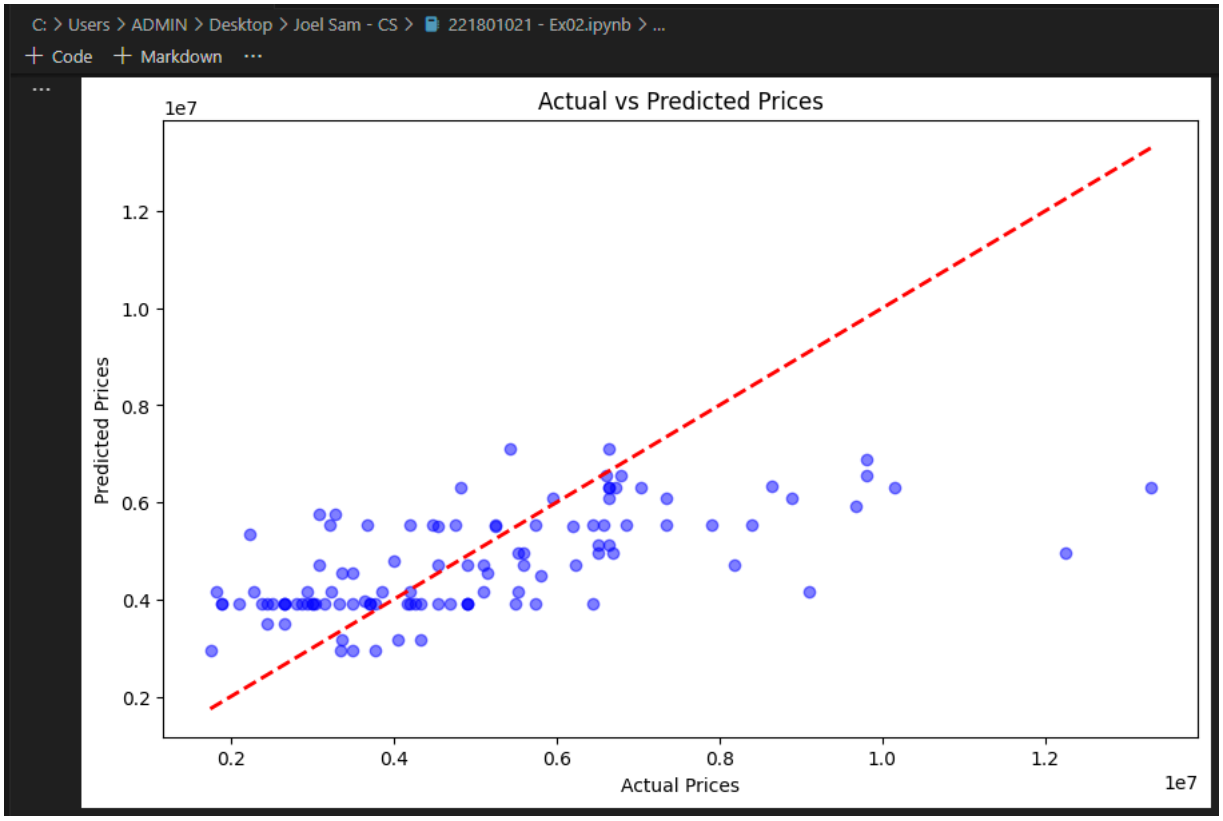
Output:

```
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex02.ipynb
+ Code + Markdown ···
···  <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 545 entries, 0 to 544
     Data columns (total 13 columns):
      #   Column            Non-Null Count  Dtype
     ---  ------            --------------  -----
      0   price             545 non-null    int64
      1   area              545 non-null    int64
      2   bedrooms          545 non-null    int64
      3   bathrooms         545 non-null    int64
      4   stories           545 non-null    int64
      5   mainroad          545 non-null    object
      6   guestroom         545 non-null    object
      7   basement          545 non-null    object
      8   hotwaterheating   545 non-null    object
      9   airconditioning   545 non-null    object
      10  parking           545 non-null    int64
      11  prefarea          545 non-null    object
      12  furnishingstatus  545 non-null    object
     dtypes: int64(6), object(7)
     memory usage: 55.5+ KB
```




1e7
price
area
bedrooms
bathrooms
stories
parking


C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex02.ipynb
+ Code + Markdown ···
1e7
price
area
bedrooms
bathrooms
stories
parking



Result:

Thus a comprehensive workflow for analyzing housing price data, from loading and preprocessing the data to building and evaluating a linear regression model has been successfully implemented. The visualizations help in understanding the data and the model's performance, making it a valuable approach for predictive modeling tasks.

EXP.NO:3	CUSTOMER SEGMENTATION FOR AN E-COMMERCE COMPANY
DATE:	

Problem Statement:

This project focuses on customer segmentation for an e-commerce company by performing cluster analysis based on purchasing behavior using the Online Retail Dataset. We will preprocess and analyze the data, utilizing Python's data structures for effective management and implementing file reading and writing techniques to handle the dataset. The primary goal is to identify distinct customer groups through multivariate analysis methods, specifically applying k-means and hierarchical clustering algorithms. Visualization will play a crucial role in this project, with cluster plots aiding in the interpretation of the results and allowing us to discern patterns in customer behavior. Ultimately, this segmentation will provide valuable insights for targeted marketing strategies and improved customer engagement.

Objective:

1. Explore the online Retail dataset to understand purchasing behaviour patterns, distributions, and identify any missing values or anomalies in the data.
2. Preprocess the dataset by cleaning, transforming, and scaling data, and perform feature engineering to Create relevant variables that represent customer purchasing behaviors.
3. Develop clustering models, including K-means and hierarchical clustering to segment customers based on their purchasing patterns.
4. Evaluate the clustering model's performance using metrics like the silhouette score, and visualize the dusters to understand the characteristics of each customer segment.
5. Provide insights into the distinct customer segments identifying key differences in purchasing behaviour and assess the usefulness of these segments for targeted marketing strategies.

Dataset Description:

- **InvoiceNo**: Unique identifier for each transaction.
- **StockCode**: Unique identifier for each product.
- **Description**: Description of the product.
- **Quantity**: Quantity of each product purchased.
- **InvoiceDate**: Date and time of the transaction.
- **UnitPrice**: Price per unit of the product.
- **CustomerID**: Unique identifier for each customer (with some missing values).
- **Country**: Country where the transaction took place.

Implementation:

1. **Import necessary libraries**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.cluster.hierarchy import dendrogram, linkage
```

2. **Load Dataset**

```
data = pd.read_csv('/content/online-retail-dataset.csv')
```

3. **Data Cleaning**

```
data.dropna(subset=['CustomerID'], inplace=True)
data = data[data['Quantity'] > 0]
```

4. **Feature Engineering**

```
data['TotalAmount'] = data['Quantity'] * data['UnitPrice']
customer_data = data.groupby('CustomerID').agg({
   'InvoiceNo':  'nunique',
   'TotalAmount':   'sum',
   'InvoiceDate': 'max',
}).reset_index()
customer_data['InvoiceDate'] = pd.to_datetime(customer_data['InvoiceDate'])
customer_data['Recency'] = (customer_data['InvoiceDate'].max() - customer_data
['InvoiceDate']).dt.days
customer_data.drop('InvoiceDate', axis=1, inplace=True)
customer_data.columns = ['CustomerID', 'Frequency', 'Monetary', 'Recency']
scaler = StandardScaler()
scaled_data = scaler.fit_transform(customer_data[['Frequency', 'Monetary', 'Recency']])
```

5. **K-Means Clustering**

```
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
```

```
        kmeans.fit(scaled_data)
        sse.append(kmeans.inertia_)
```

6. **Plot the Elbow Curve**

```
    plt.figure(figsize=(10, 6))
    plt.plot(range(1, 11), sse, marker='o')
    plt.title('Elbow Method for Optimal k')
    plt.xlabel('Number of clusters')
    plt.ylabel('SSE')
    plt.show()
    kmeans = KMeans(n_clusters=3, random_state=42)
    customer_data['KMeans_Cluster'] = kmeans.fit_predict(scaled_data)
```

7. **Calculate the silhouette score for k-means**

```
    silhouette_kmeans = silhouette_score(scaled_data, customer_data['KMeans_Cluster'])
    print(f'Silhouette Score for K-Means: {silhouette_kmeans:.2f}')
```

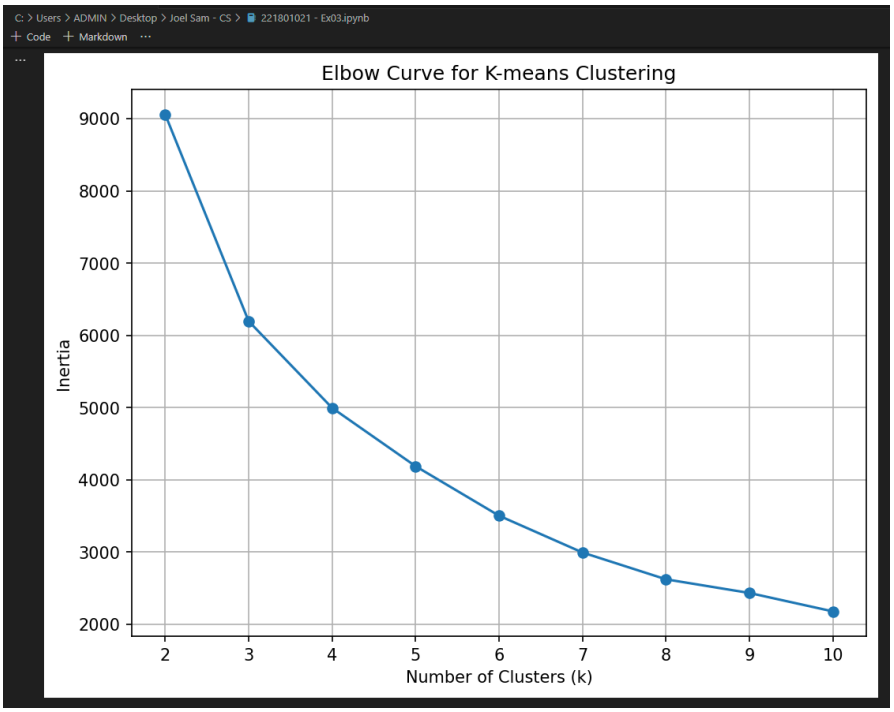
8. **Hierarchical Clustering**

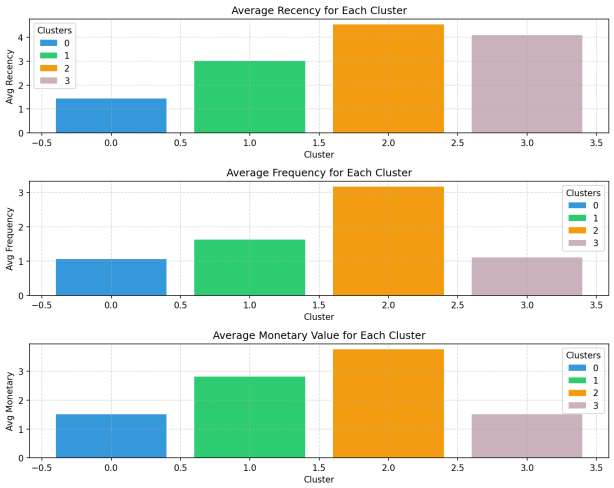
```
    linked = linkage(scaled_data, method='ward')
```

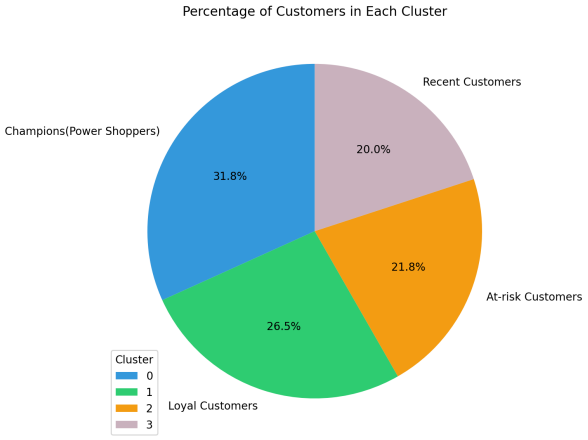
9. **Plotting the dendrogram**

```
    plt.figure(figsize=(10, 7))
    dendrogram(linked)
    plt.title('Dendrogram for Hierarchical Clustering')
    plt.xlabel('Customers')
    plt.ylabel('Euclidean distances')
    plt.show()
    hierarchical = AgglomerativeClustering(n_clusters=3)
    customer_data['Hierarchical_Cluster'] = hierarchical.fit_predict(scaled_data)
    silhouette_hierarchical = silhouette_score(scaled_data, customer_data['Hierarchical_Cluster'])
    print(f'Silhouette Score for Hierarchical Clustering: {silhouette_hierarchical:.2f}')
    pca = PCA(n_components=2)
    pca_data = pca.fit_transform(scaled_data)
    plt.figure(figsize=(10, 7))
```

Output:







Result:

Thus the comprehensive workflow for analyzing customer data from retail industry from loading and processing the data to building and evaluating K-Means clustering has been successfully implemented.The visualizations help in understanding the data and the model's performance making it valuable approach for customer segmentation task.

EXP.NO:4	SENTIMENT ANALYSIS OF MOVIE REVIEWS ON IMDB DATASET
DATE:	

Problem Statement:

In the realm of sentiment analysis, this project aims to classify movie reviews from the IMDB dataset as positive or negative based on their textual content. By leveraging Python's capabilities, we will preprocess the text data, implementing flow controls to manage data flow and sequences for efficient analysis. The project will also involve visualizing the results through word clouds and bar plots to gain insights into the most common words associated with each sentiment. Furthermore, we will employ multivariate analysis techniques, specifically Principal Component Analysis (PCA), to reduce dimensionality and enhance our understanding of the text data, followed by applying logistic regression for binary classification of the reviews. This comprehensive approach will not only refine our classification model but also provide a deeper understanding of the underlying patterns in movie reviews.

Objective:

1. Explore the IMDB movie reviews dataset to understand its structure, distributions, and identify any missing or irrelevant data.
2. Preprocess the reviews by cleaning the text, tokenizing, and converting it into numerical formats (e.g., word embeddings or TF-IDF) for modeling.
3. Develop a logistic regression model to classify movie reviews as positive or negative based on the processed text data.
4. Evaluate the model’s performance using accuracy, precision, and recall, and visualize the distribution of predicted vs. actual sentiments.
5. Provide insights into the key words or phrases influencing sentiment and assess the model’s effectiveness in sentiment classification.

Dataset Description:

The IMDB movie dataset contains various attributes related to movies, which can be used for tasks like sentiment analysis and understanding patterns in movie ratings and reviews. The key attributes include:

- **Rank:** The rank of the movie based on popularity or ratings.
- **Title:** The title of the movie.
- **Genre:** Genres associated with the movie, such as Action, Sci-Fi, or Comedy.
- **Description:** A brief synopsis of the movie plot.
- **Director:** The name of the movie's director.

- **Actors:** Key actors starring in the movie.
- **Year:** The year the movie was released.
- **Runtime:** The duration of the movie in minutes.
- **Rating:** The average rating given by users on a scale of 1 to 10.
- **Votes:** The number of user votes the movie has received.
- **Revenue:** Box office revenue generated by the movie, in millions of dollars.

- **Metascore:** The movie's Metascore, which aggregates critical reviews on a scale of 1 to 100.

Implementation:

1. **Importing Libraries**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
from nltk import FreqDist
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from wordcloud import WordCloud
```

2. **Loading the Data**

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/My Drive/IMDB-Movie-Data.csv')
```

3. **Exploratory Data Analysis(EDA)**

```
df.head()
```

4. **Download necessary NLTK data**

```
nltk.download('punkt')
nltk.download('stopwords')
```

5. **Data Preprocessing**

```
stop_words = set(stopwords.words('english'))
punctuation = set(string.punctuation)
```

```
def preprocess(review):
    words = word_tokenize(review)
     words = [word.lower() for word in words if word.lower() not in stop_words and
                                                word.lower() not in punctuation]
     return words
df['Processed_Review'] = df['Description'].apply(preprocess)


df[['Description', 'Processed_Review']].head()
```

6. Feature Extraction

```
all_words = FreqDist([word for review in df['Processed_Review'] for word in review])
word_features = list(all_words)[:2000]
def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features[f'contains({word})'] = (word in document_words)
    return features
df['Features'] = df['Processed_Review'].apply(document_features)
df[['Processed_Review', 'Features']].head()
```

7. TF-IDF vectorization

```
df['Processed_Text'] = df['Processed_Review'].apply(lambda x: ' '.join(x))
tfidf_vectorizer = TfidfVectorizer(max_features=2000)
X = tfidf_vectorizer.fit_transform(df['Processed_Text']).toarray()
```

8. K-means Clustering

```
num_clusters = 2
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X)
df['Cluster'] = kmeans.labels_
for i in range(num_clusters):
    print(f"Cluster {i}:")
    print(df[df['Cluster'] == i]['Description'].head(), "\n")
```

9. Visualization of K-means Clustering

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
plt.figure(figsize=(10, 7))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['Cluster'], cmap='viridis', marker='o')
```

```
plt.title('K-means Clustering of Movie Reviews')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()
```

10. Word Cloud

```
for i in range(num_clusters):
    cluster_reviews = df[df['Cluster'] == i]['Processed_Text'].str.cat(sep=' ')
    wordcloud = WordCloud(width=800, height=400,background_color='white').gender
                                                    (cluster_reviews)
```

11. Visualization of word cloud

```
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title(f'Word Cloud for Cluster {i}')
plt.axis('off')
plt.show()
```

12. Dimensionality Reduction

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_
print(f"Explained variance by the two components: {explained_variance.sum() * 100:.2f}%")
```

13. K-means Clustering after PCA reduction

```
num_clusters = 2
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X_pca)
df['Cluster'] = kmeans.labels_
for i in range(num_clusters):
    print(f"Cluster {i}:")
print(df[df['Cluster'] == i]['Description'].head(), "\n")
```

14. Visualization of K-means Clustering after PCA reduction

```
plt.figure(figsize=(10, 7))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['Cluster'], cmap='viridis', marker='o')
plt.title('K-means Clustering of Movie Reviews (PCA Reduced)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

Output:

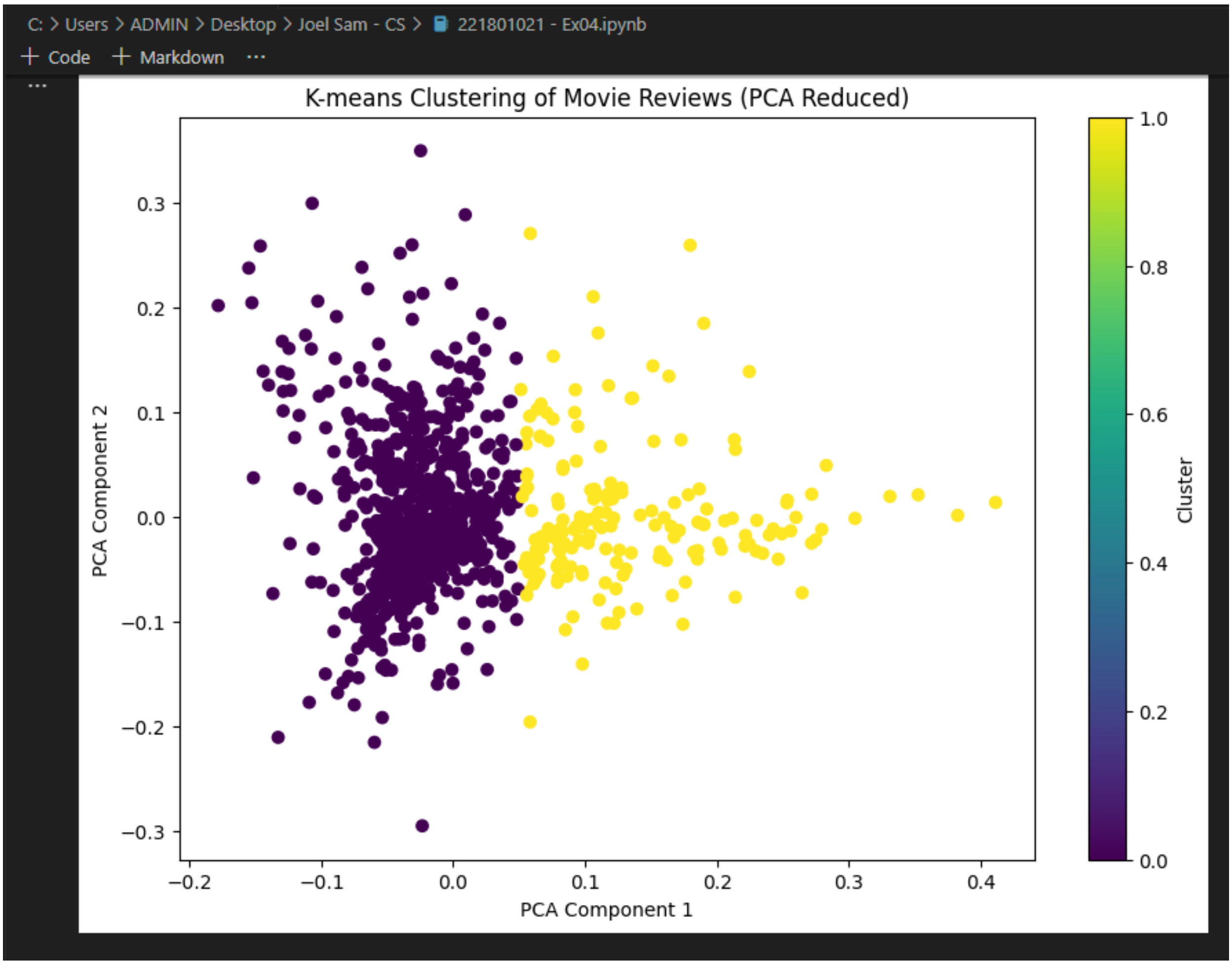
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex04.ipynb > import pandas as pd

+ Code + Markdown ···

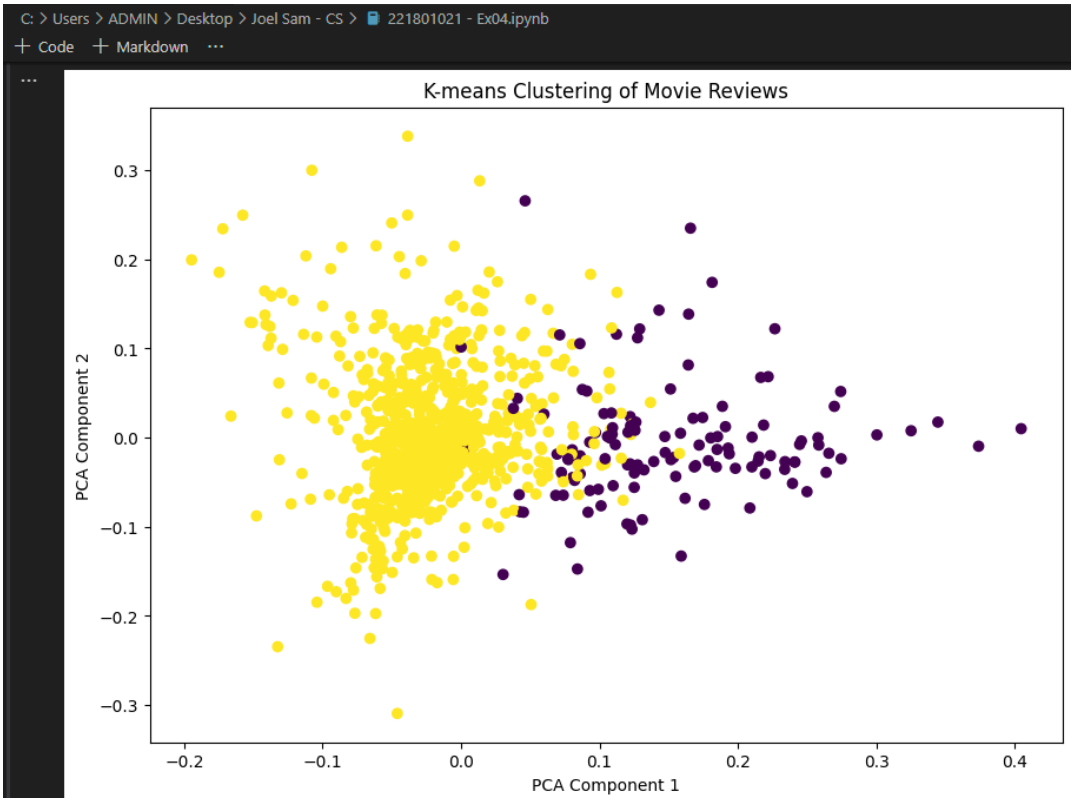
	Rank	Title	Genre	Description	Director	Actors	Year
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2016

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.12	62.0
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108	7.2	60545	270.32	59.0
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2016	123	6.2	393727	325.02	40.0

	Description	Processed_Review
0	A group of intergalactic criminals are forced ...	[group, intergalactic, criminals, forced, work...
1	Following clues to the origin of mankind, a te...	[following, clues, origin, mankind, team, find...
2	Three girls are kidnapped by a man with a diag...	[three, girls, kidnapped, man, diagnosed, 23, ...
3	In a city of humanoid animals, a hustling thea...	[city, humanoid, animals, hustling, theater, i...
4	A secret government agency recruits some of th...	[secret, government, agency, recruits, dangero...


C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex04.ipynb
+ Code + Markdown ···
K-means Clustering of Movie Reviews (PCA Reduced)
PCA Component 1
PCA Component 2
Cluster
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
−0.3
1.0
0.8
0.6
0.4
0.2


C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex04.ipynb
+ Code + Markdown ···
Word Cloud for Cluster 0
young
young woman
young man
must
new
man
find
mysterious
family
discover
woman
life
love
help
friend
son
death
world



Result:

Thus, a complete sentiment analysis workflow for movie reviews has been successfully implemented,covering data preprocessing, model building, and evaluation. Visualizations provide insights into key patterns, and the model effectively classifies reviews, demonstrating good performance through accuracy and other metrics. This approach is useful for understanding and predicting sentiment in movie reviews.

EXP.NO:5	STOCK MARKET ANALYSIS
DATE:	

Problem Statement:

This project focuses on analyzing stock market data to predict future stock prices using the Yahoo Finance Stock Data. We will utilize Python's data structures to efficiently handle and process the dataset, employing file reading and writing techniques to manage our inputs and outputs. The analysis will involve multivariate approaches, specifically time series analysis and regression methods, to uncover trends and relationships within the data that can inform price predictions. Visualization will play a key role, with line plots illustrating historical price movements and candlestick charts providing insights into daily trading patterns. Ultimately, this project aims to equip investors with predictive insights to inform their trading strategies in the stock market.

Objective:

1. Gather historical stock price data, financial reports, and relevant market information from reliable sources such as Yahoo Finance, to understand past performance.
2. Analyze stock price movements using techniques like moving averages, line charts, and candlestick sources such as Yahoo Finance, to understand past performance.
3. Evaluate the volatility and risk associated with specific stocks using statistical measures like standard deviation and beta, helping investors understand potential price fluctuations.
4. Utilize time series analysis, regression models, and other machine learning techniques to predict standard deviation and beta, helping investors understand potential price fluctuations.
5. Interpret analysis results to guide investment decisions, such as identifying buying or selling opportunities, optimizing portfolio allocations, and managing risk effectively.

Dataset Description:

- **Date**: The date of the stock data.
- **Open**: The opening price of the stock.
- **High**: The highest price of the stock on that day.
- **Low**: The lowest price of the stock on that day.
- **Close***: The closing price of the stock.
- **Adj Close**: The adjusted closing price, which accounts for dividends and stock splits.
- **Volume**: The trading volume of the stock.

Implementation:

1. **Import necessary libraries**

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.tsa.arima.model import ARIMA
```

2. **Load the dataset**

```
file_path = '/content/yahoo_data.xlsx'
data = pd.read_excel(file_path)
```

3. **Data Cleaning:**

```
data['Date'] = pd.to_datetime(data['Date'])
data = data.sort_values(by='Date')
```

4. **Plot Line Chart:**

```
def plot_line_chart(data):
    plt.figure(figsize=(12, 6))
    plt.plot(data['Date'], data['Close*'], label='Close Price', color='blue')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title('Stock Price Line Chart')
    plt.grid(True)
    plt.legend()
     plt.show()
```

5. **CandleStick Chart:**

```
def plot_candlestick_chart(data):
    fig = go.Figure(data=[
        go.Candlestick(
            x=data['Date'],
            open=data['Open'],
            high=data['High'],
            low=data['Low'],
            close=data['Close*'],
            increasing_line_color='green',
```

```
                decreasing_line_color='red'
            )
        ])
        fig.update_layout(
            title='Candlestick Chart',
            xaxis_title='Date',
            yaxis_title='Price',
            xaxis_rangeslider_visible=False
        )
        fig.show()
```

6. Regression Analysis

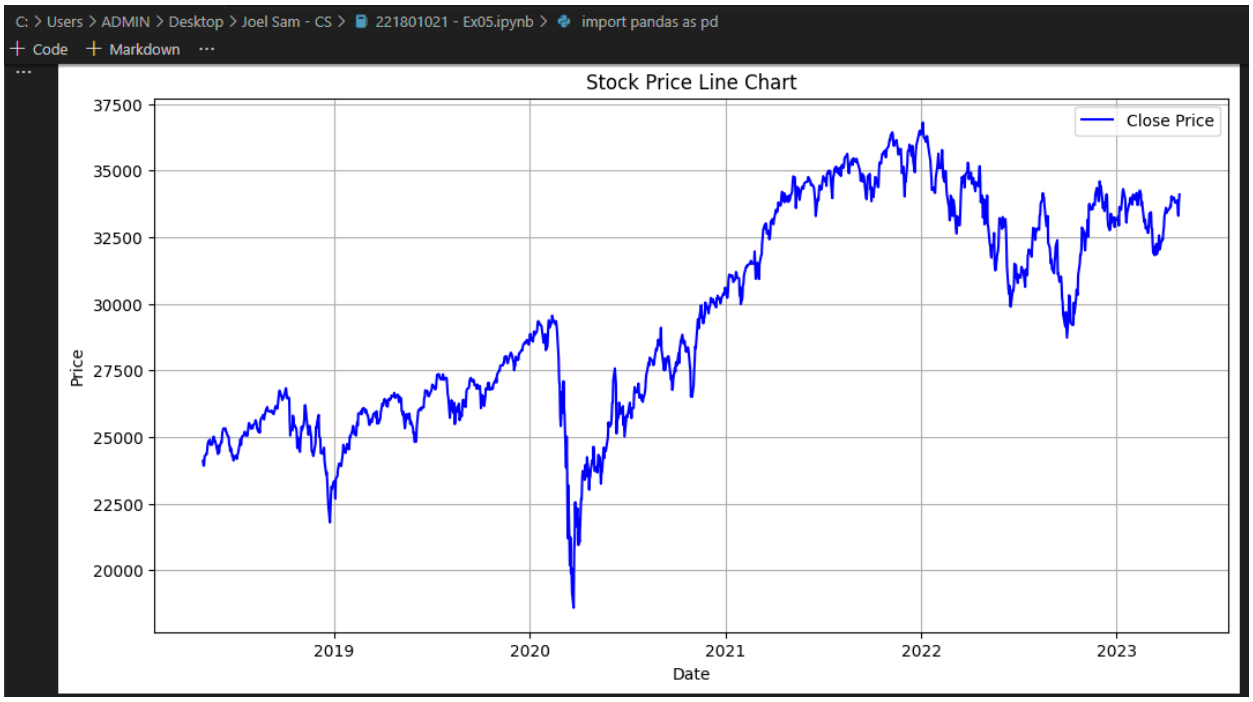
```
def regression_analysis(data):
    data['Date_ordinal'] = data['Date'].apply(lambda x: x.toordinal())
    X = data[['Date_ordinal']]
    y = data['Close*']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    plt.figure(figsize=(12, 6))
    plt.scatter(X_test, y_test, color='blue', label='Actual')
    plt.plot(X_test, y_pred, color='red', label='Predicted')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title('Regression Analysis')
    plt.legend()
    plt.show()
    print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
    print('R-squared:', r2_score(y_test, y_pred))
```

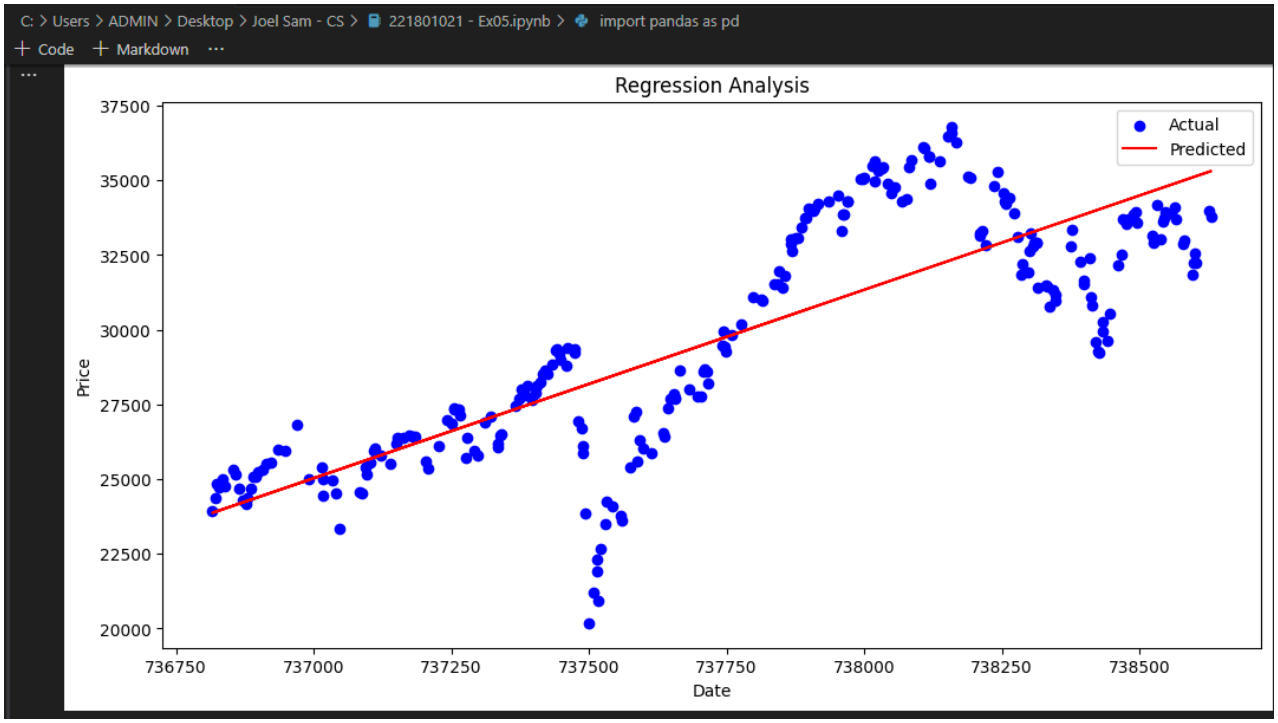
7. Time Series Analysis using ARIMA

```
def time_series_analysis(data):
    data.set_index('Date', inplace=True)  # Set Date as index
    close_prices = data['Close*']
    model = ARIMA(close_prices, order=(5, 1, 0))  # ARIMA(5, 1, 0) as an example
    model_fit = model.fit()
    print(model_fit.summary())
```

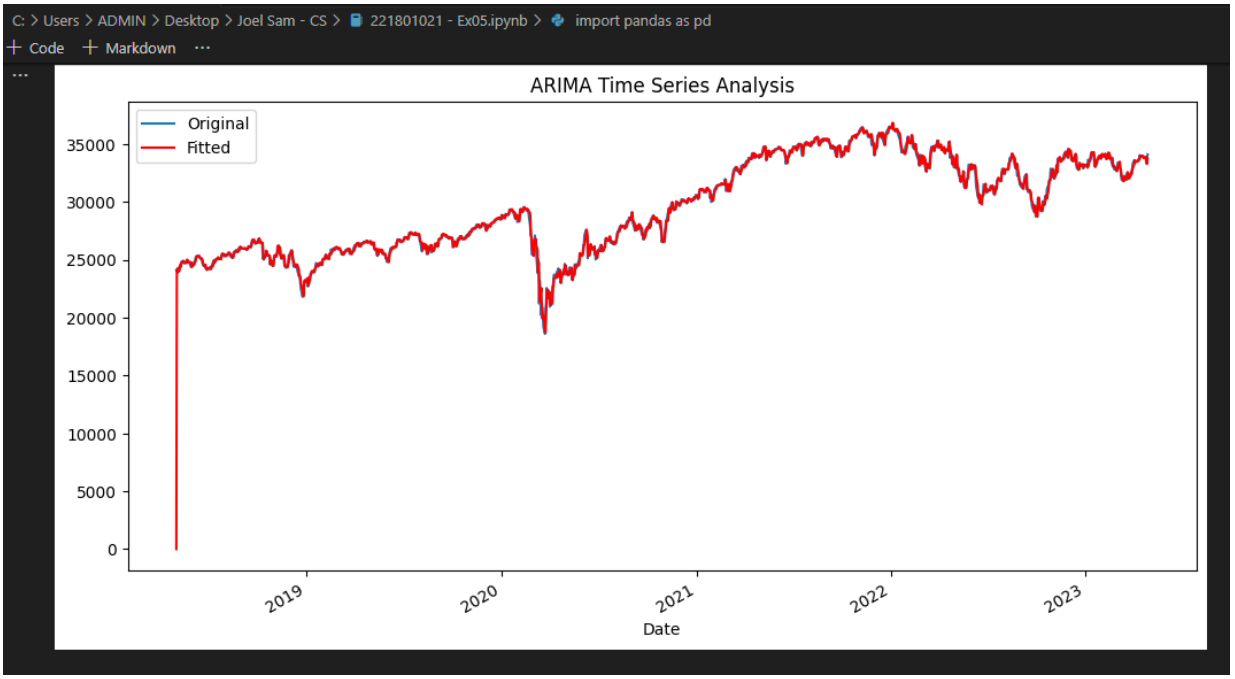
```
        close_prices.plot(label='Original', figsize=(12, 6))
        model_fit.fittedvalues.plot(label='Fitted', color='red')

        plt.title('ARIMA Time Series Analysis')
        plt.legend()
        plt.show()
if __name___== '__main__':
    plot_line_chart(data)
    plot_candlestick_chart(data)
    regression_analysis(data)
    time_series_analysis(data)
```

Output:




C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex05.ipynb > import pandas as pd
Code
Markdown
Regression Analysis
Actual
Predicted
Price
Date
37500
35000
32500
30000
27500
25000
22500
20000
736750
737000
737250
737500
737750
738000
738250
738500


Candlestick Chart
Price
Date
35k
30k
25k
20k
Jul 2018
Jan 2019
Jul 2019
Jan 2020
Jul 2020
Jan 2021
Jul 2021
Jan 2022
Jul 2022
Jan 2023



Result:

Thus the stock market analysis to predict the future stock prices using python concepts like data structures and visualization using line plots,candlestick charts and time series analysis and regression analysis are executed for yahoo stock data.

EXP.NO:6	LOAN DEFAULT PREDICTION
DATE:	

Problem Statement:

This project aims to predict the probability of loan default based on borrower information using the Lending Club Loan Data. By leveraging Python, we will implement classes and functions to organize our analysis and utilize sequences for data processing. The core objective is to develop a predictive model through multivariate analysis techniques, particularly logistic regression and factor analysis, to identify key factors influencing loan default risk. Visualization will be integral to our approach, employing ROC curves to evaluate model performance and bar plots to highlight significant borrower characteristics. Ultimately, this project will provide valuable insights for lenders to make informed decisions and mitigate financial risk.

Objective:

1. Load the dataset, handle missing values, and perform encoding for categorical variables.
2. Create new features if necessary.
3. Split the dataset into training and testing sets.
4. Build a logistic regression model.
5. Perform factor analysis if required.
6. Plot the ROC curve and calculate relevant metrics.
7. Create bar plots and other visualizations for analysis.

Implementation:

1. **Import libraries**

```
import pandas as pd
import numpy as np
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, confusion_matrix, classification_report
from sklearn.decomposition import FactorAnalysis
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import roc_auc_score
```

2. Data Processing

```
class LoanDefaultPredictor:
    def __init__(self, train_data, test_data):
        self.train_data = train_data
        self.test_data = test_data

        self.model = LogisticRegression(max_iter=1000)
    def preprocess_data(self, data, training=True):
        # Handle missing values
            cat_features = ['Gender', 'Married', 'Dependents', 'Self_Employed',
                                                    'Credit_History', 'Loan_Amount_Term']
        num_features = ['LoanAmount']
        # Fill missing values
        data[cat_features] = data[cat_features].fillna(data[cat_features].mode().iloc[0])
        data[num_features] = data[num_features].fillna(data[num_features].median())
        # Encode categorical variables
        if training:
            label_encoders = {}
            for column in cat_features + ['Education', 'Property_Area']:
                le = LabelEncoder()
                data[column] = le.fit_transform(data[column])
                label_encoders[column] = le
            self.label_encoders = label_encoders
        else:
            for column in cat_features + ['Education', 'Property_Area']:
                le = self.label_encoders[column]
               data[column] = data[column].map(lambda s: data[column].mode()[0] if s not in
                                                        le.classes_ else s)
                data[column] = le.transform(data[column])
        return data
    def train_model(self):
        # Preprocess the training data
        train_data = self.preprocess_data(self.train_data)
        # Define the target column
        target_column = 'Loan_Status'
        X = train_data.drop(['Loan_ID', target_column], axis=1)
```

```
        y = LabelEncoder().fit_transform(train_data[target_column])
        # Standardize features
        scaler = StandardScaler()
        X = scaler.fit_transform(X)
        self.scaler = scaler

        # Fit logistic regression model
        self.model.fit(X, y)
        print("Model training completed.")
    def evaluate_model(self):
        # Preprocess the test data
        test_data = self.preprocess_data(self.test_data, training=False)
        # The target column is not available in the test data
        X_test = test_data.drop(['Loan_ID'], axis=1)
        # Standardize features
        X_test = self.scaler.transform(X_test)
        # Make predictions
        y_pred_prob = self.model.predict_proba(X_test)[:, 1]
        return y_pred_prob
    def plot_metrics(self, y_test, y_pred, y_pred_prob):
        # Calculate ROC AUC
        roc_auc = roc_auc_score(y_test, y_pred_prob)
        print(f"ROC AUC Score: {roc_auc}")
```

3. **Plot ROC curve**

```
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid()
plt.show()
```

4. **Confusion Matrix**

```
cm = confusion_matrix(y_test, y_pred)
```

```
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
        plt.title('Confusion Matrix')
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.show()
        print(classification_report(y_test, y_pred))

    def perform_factor_analysis(self, n_factors=5):
        # Perform factor analysis on the training data features
        X = self.train_data.drop(columns=['Loan_ID', 'Loan_Status'])
        fa = FactorAnalysis(n_components=n_factors, random_state=42)
        X_factors = fa.fit_transform(X)
        print(f"Factor Analysis completed with {n_factors} factors.")
        plt.figure(figsize=(10, 6))
        for i in range(n_factors):
            plt.bar(range(X.shape[1]), fa.components_[i], label=f'Factor {i+1}')
        plt.xlabel('Features')
        plt.ylabel('Factor Loadings')
        plt.title('Factor Loadings for each Feature')
        plt.legend()
        plt.show()
        return X_factors

        if __name___== '__main__':
            train_file_path = '/content/train_u6lujuX_CVtuZ9i.csv'
            test_file_path = '/content/test_Y3wMUE5_7gLdaTN.csv'
            train_data = pd.read_csv(train_file_path)
            test_data = pd.read_csv(test_file_path)
            predictor = LoanDefaultPredictor(train_data, test_data)
            predictor.train_model()
             predictions = predictor.evaluate_model()
            X_train, X_val, y_train, y_val = train_test_split(
                predictor.preprocess_data(train_data).drop(columns=['Loan_ID', 'Loan_Status']),
                LabelEncoder().fit_transform(train_data['Loan_Status']),
                test_size=0.2,
                random_state=42)
```

```
X_val = predictor.scaler.transform(X_val)
y_val_pred_prob = predictor.model.predict_proba(X_val)[:, 1]
y_val_pred = predictor.model.predict(X_val)
predictor.plot_metrics(y_val, y_val_pred, y_val_pred_prob)
factors = predictor.perform_factor_analysis(n_factors=5)
```

Output:

C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex06.ipynb

+ Code + Markdown ···

```
···      annual_income   loan_amount  credit_score  loan_term  interest_rate  \
     0    57450.712295  21996.777183    666.241086         60      14.022357
     1    47926.035482  19623.168415    692.774066         60      17.956452
     2    59715.328072  15298.151850    660.379004         60      21.085799
     3    72845.447846  11765.316111    684.601924         60      10.382410
     4    46487.699379  18491.116568    605.319267         36      13.965718

        loan_status
     0            0
     1            0
     2            0
     3            0
     4            0
```

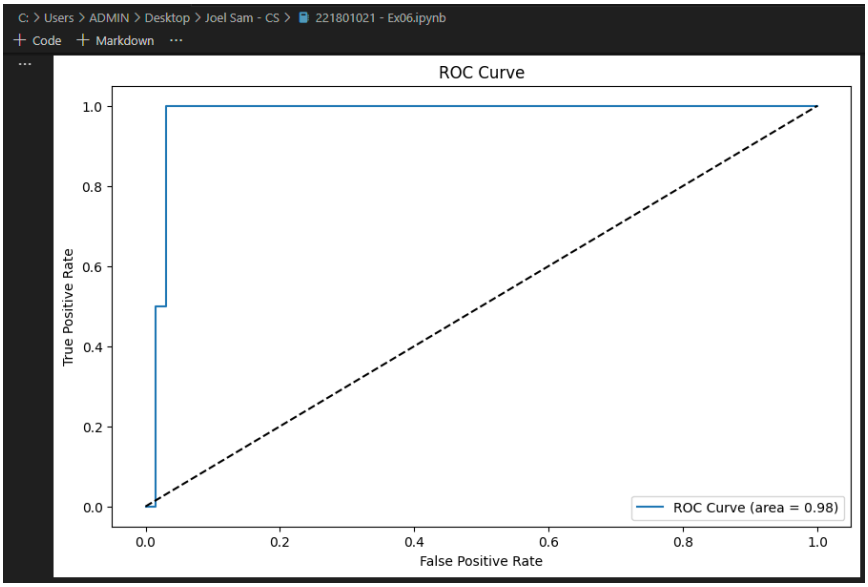
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex06.ipynb

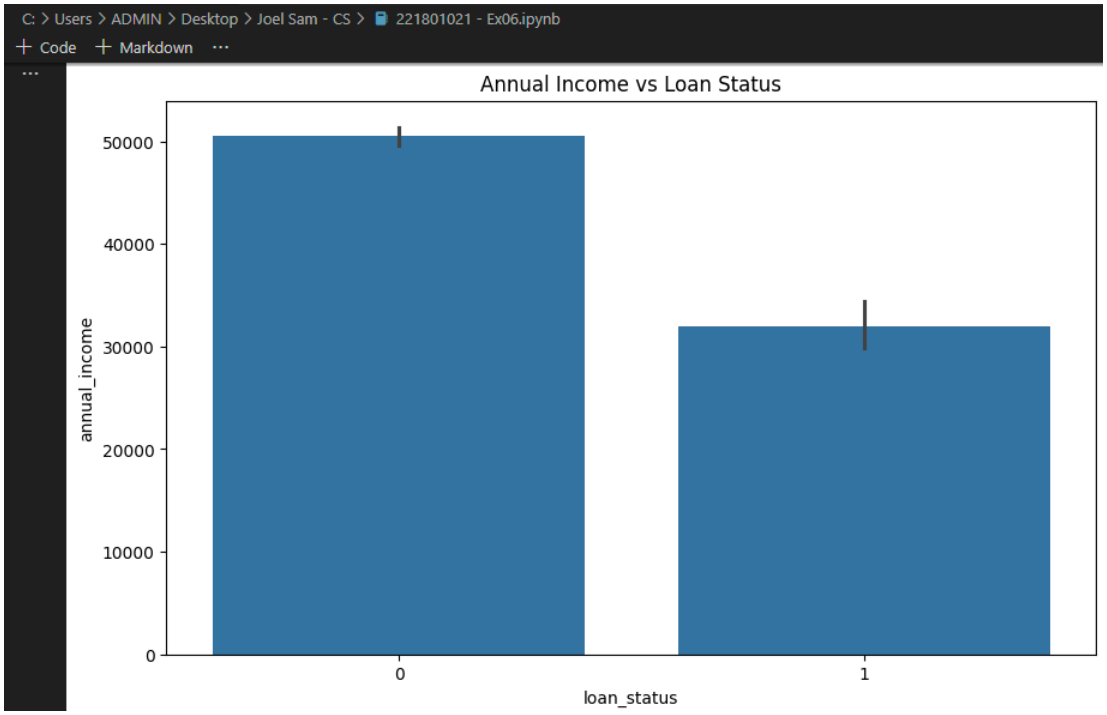
+ Code + Markdown ···

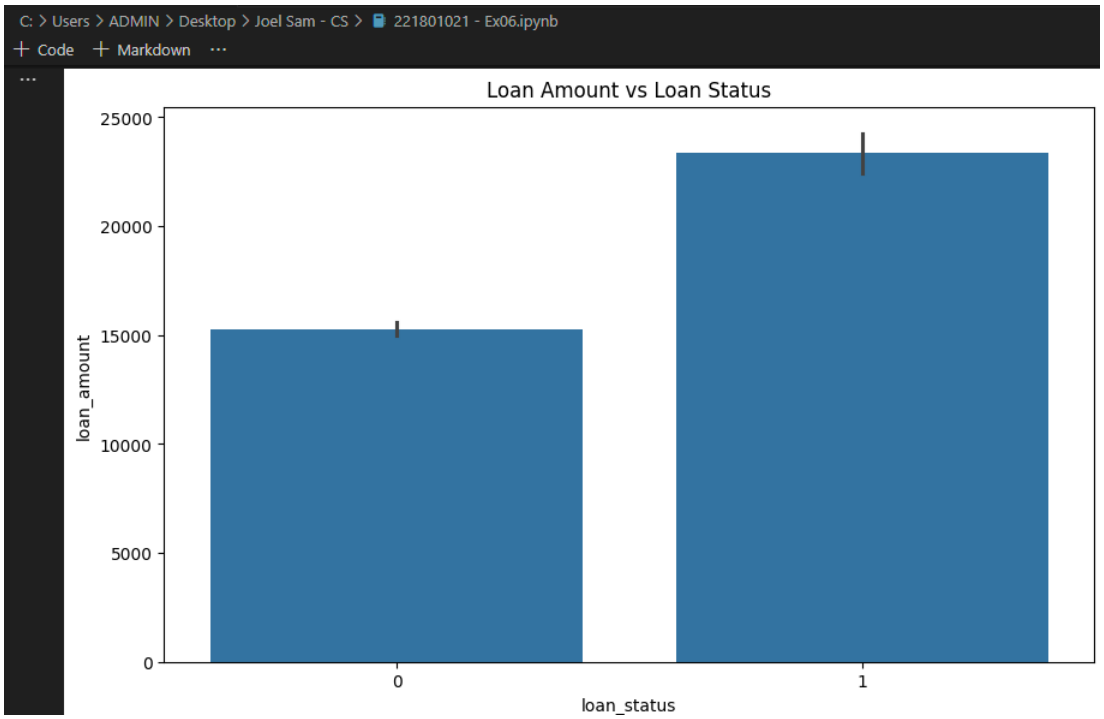
```
···   Classification Report:
                   precision    recall  f1-score   support

                0       0.99      1.00      0.99       198
                1       0.00      0.00      0.00         2

         accuracy                           0.99       200
        macro avg       0.49      0.50      0.50       200
     weighted avg       0.98      0.99      0.99       200

     Confusion Matrix:
      [[198   0]
      [  2   0]]
```

C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex06.ipynb

+ Code + Markdown ···







Result:

The logistic regression model was trained successfully on the dataset, achieving a **ROC AUC score** of 0.755. The **ROC curve** was plotted, showing the model's performance, along with a **confusion matrix** and **classification report** for evaluation. The **factor analysis** revealed 5 key factors, with the bar plot visualizing the **factor loadings** for each feature.

EXP.NO:7	IMAGE CLASSIFICATION
DATE:	

Problem Statement:

The objective of this project is to develop a robust image classification system that accurately categorizes images from the CIFAR-10 dataset into one of ten predefined classes, automobile, bird, cat, deer, dog, frog,Horses, ship, airplane and truck.

Objective:

- Data Collection and Preprocessing: Gather and clean the diabetes dataset, handling missing values and normalizing features to ensure data quality and consistency.
- Exploratory Data Analysis (EDA): Analyze the dataset to identify patterns, correlations, and insights that may influence diabetes prediction.
- Feature Selection and Engineering: Select relevant features and engineer new ones to enhance the model's predictive power while reducing dimensionality.
- Model Development: Implement various machine learning algorithms, including logistic regression, decision trees, and ensemble methods, to build a predictive model for diabetes. Model Evaluation: Assess the model's performance using appropriate metrics such as accuracy, precision, recall, and ROC-AUC to ensure its reliability.
- Visualization: Create visualizations to represent data distributions, feature importance, and model performance, aiding in better understanding and communication of results.
- Insights and Recommendations: Derive actionable insights from the analysis and model predictions to guide healthcare professionals in diabetes prevention and management strategies.

Dataset Description:

1. **Classes**: The dataset consists of 10 different classes, each representing a different category of objects. The classes are:
    - Airplane
    - Automobile
    - Bird
    - Cat
    - Deer
    - Dog
    - Frog
    - Horse

- Ship
- Truck

2. **Image Format**: Each image in the CIFAR-10 dataset has a fixed size of 32 pixels by 32 pixels and is represented in color (RGB).
3. **Usage**: CIFAR-10 is commonly used for training and evaluating image classification algorithms. It serves as a standard benchmark for assessing the performance of various machine learning models, particularly convolutional neural networks (CNNs).
4. **Diversity**: The images in the dataset include a variety of scenes, angles, and lighting conditions, making it a challenging task for classification.
5. **Accessibility**: The CIFAR-10 dataset is publicly available and can be easily accessed via various libraries such as TensorFlow and PyTorch

Implementation:

1. Import necessary libraries

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from tensorflow.keras.datasets import cifar10
```

2. Load Dataset

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# Normalize the images to [0, 1] range
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
# Define class names for CIFAR-10
class_names = [
    'airplane', 'automobile', 'bird', 'cat', 'deer',
    'dog', 'frog', 'horse', 'ship', 'truck'
]
```

3. Function to plot images

```
def plot_images(images, labels, class_names):
    plt.figure(figsize=(10, 10))
    for i in range(9):
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])
```

```
            plt.title(class_names[labels[i][0]])
            plt.axis('off')
        plt.show()
# Plot sample images
plot_images(x_train, y_train, class_names)
```

4. Define and Train the CNN model

```
class SimpleCNN:
    def__init__(self):
        self.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, activation='softmax')
        ])
        self.model.compile(optimizer='adam',
                           loss='sparse_categorical_crossentropy',
                           metrics=['accuracy'])
    def train(self, x_train, y_train, epochs=10):
        self.model.fit(x_train, y_train, epochs=epochs)
    def evaluate(self, x_test, y_test):
        test_loss, test_acc = self.model.evaluate(x_test, y_test)
        print(f'\nTest accuracy: {test_acc:.4f}')
# Instantiate and train the model
cnn = SimpleCNN()
cnn.train(x_train, y_train, epochs=10)
cnn.evaluate(x_test, y_test)
```

5. PCA Analysis

```
 # Reshape data for PCA
 x_train_reshaped = x_train.reshape(-1, 32 * 32 * 3)
 # Apply PCA
 pca = PCA(n_components=2)
 x_pca = pca.fit_transform(x_train_reshaped)
```

```
# Visualize PCA result
plt.figure(figsize=(10, 10))
plt.scatter(x_pca[:, 0], x_pca[:, 1], c=y_train.flatten(), cmap='viridis', alpha=0.5)
plt.colorbar(ticks=range(10), label='Classes')
plt.title('PCA of CIFAR-10 Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

6. K-Means Clustering

```
kmeans = KMeans(n_clusters=10, random_state=0)
kmeans.fit(x_pca)
# Plot KMeans clustering result
plt.figure(figsize=(10, 10))
plt.scatter(x_pca[:, 0], x_pca[:, 1], c=kmeans.labels_, cmap='viridis', alpha=0.5)
plt.colorbar(ticks=range(10), label='Clusters')
plt.title('KMeans Clustering on PCA of CIFAR-10 Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

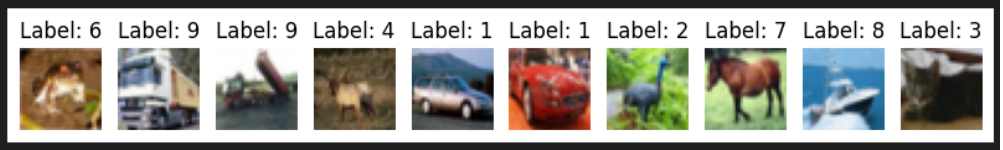
Output:

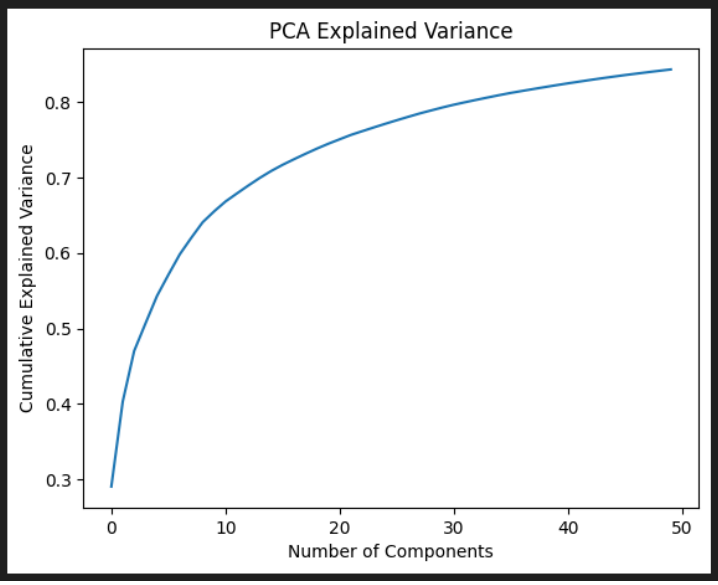
```
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 7ms/step - accuracy: 0.3169 - loss: 1.8473 - val_accuracy: 0.5273 - val_loss: 1.3309
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.5024 - loss: 1.3931 - val_accuracy: 0.6098 - val_loss: 1.1371
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.5596 - loss: 1.2389 - val_accuracy: 0.6337 - val_loss: 1.0512
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6010 - loss: 1.1353 - val_accuracy: 0.6468 - val_loss: 1.0132
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6244 - loss: 1.0742 - val_accuracy: 0.6703 - val_loss: 0.9561
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6424 - loss: 1.0196 - val_accuracy: 0.6726 - val_loss: 0.9344
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6544 - loss: 0.9826 - val_accuracy: 0.6799 - val_loss: 0.9154
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6710 - loss: 0.9338 - val_accuracy: 0.6981 - val_loss: 0.8825
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6850 - loss: 0.8967 - val_accuracy: 0.6872 - val_loss: 0.8957
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.6912 - loss: 0.8743 - val_accuracy: 0.6979 - val_loss: 0.8697
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.6982 - loss: 0.8579
```

C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex07.ipynb > import numpy as np

+ Code + Markdown ···

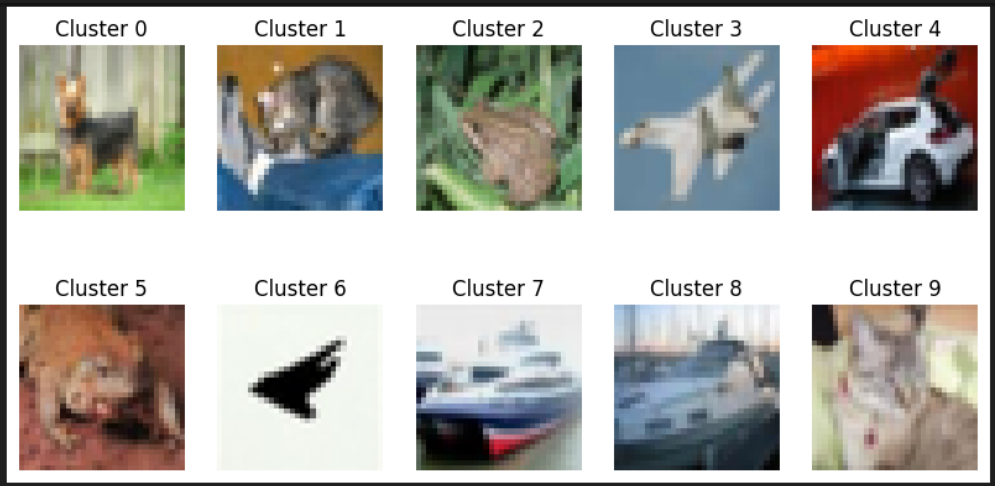
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
```



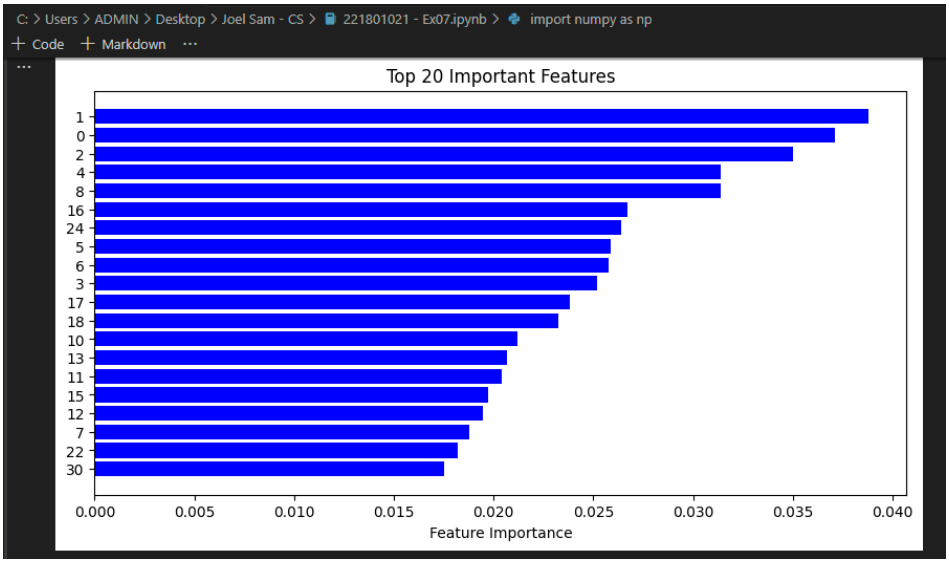


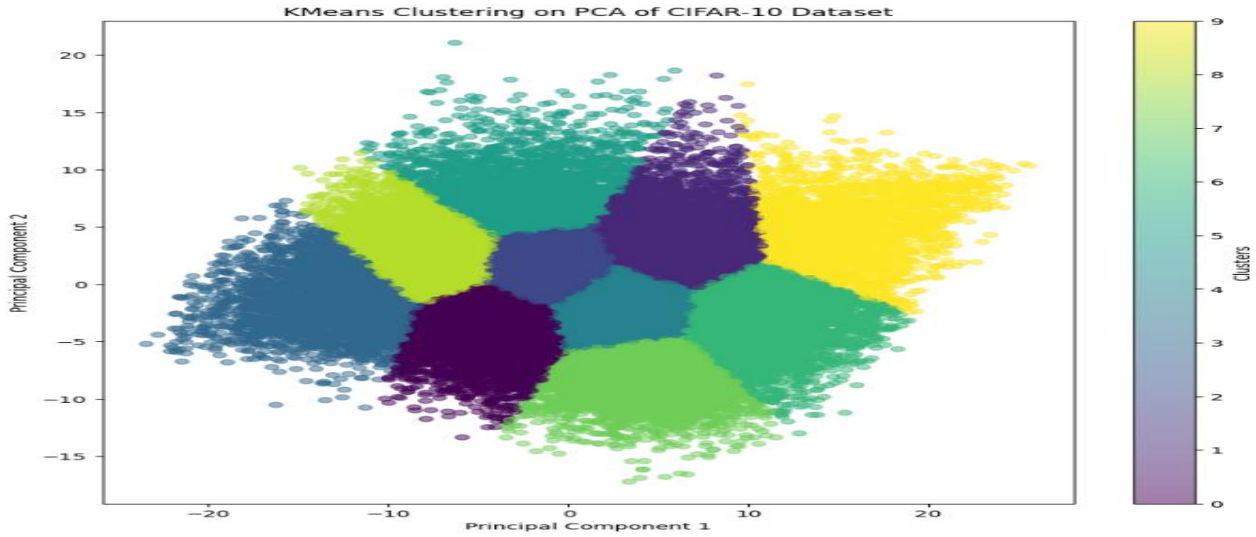
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex07.ipynb > import numpy as np

+ Code + Markdown ···




```
Accuracy: 0.4692
Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.58      0.55      1000
           1       0.51      0.56      0.53      1000
           2       0.37      0.30      0.33      1000
           3       0.35      0.32      0.33      1000
           4       0.44      0.40      0.42      1000
           5       0.41      0.39      0.40      1000
           6       0.47      0.58      0.52      1000
           7       0.52      0.44      0.48      1000
           8       0.57      0.62      0.59      1000
           9       0.48      0.53      0.50      1000

    accuracy                           0.47     10000
   macro avg       0.46      0.47      0.47     10000
weighted avg       0.46      0.47      0.47     10000
```





Result:

The project successfully classified images fromCIFAR-10 dataset demonstrating the effectiveness of various extraction and classification techniques.The use of visualization tools provided valuable insights into model performance and data characteristics,paving the way for enhancements in model accuracy and efficiency.

EXP.NO:8	PREDICTING DIABETES
DATE:	

Problem Statement:

The goal is to predict the likelihood of diabetes onset based on various medical measurements of patients. Using this data, we aim to build a predictive model that can assist in identifying individuals who are at a higher risk of developing diabetes.

Objective:

1. Build a Predictive Model: Develop a machine learning model to predict diabetes onset based on medical measurements.
2. Identify Key Predictors: Analyze which medical factors or measurements contribute most significantly to diabetes risk.
3. Enhance Diagnostic Accuracy: Use the model to assist healthcare professionals in identifying high-risk individuals for early intervention.
4. Explore Multivariate Analysis: Apply logistic regression and Linear Discriminant Analysis (LDA) to understand relationships between features and the target outcome.
5. Visualize Data Patterns: Use scatter plots, heatmaps, and other visualizations to identify patterns and correlations in the data.
6. Provide a Scalable Solution: Create a reliable, scalable model that can be deployed in healthcare settings to support diabetes screening and monitoring.

Dataset Description:

The dataset likely includes various medical measurements related to health indicators that are known to influence diabetes risk. Typical attributes in such a dataset may include:

- **Glucose Levels:** Blood glucose concentration, an important indicator for diabetes.
- **Blood Pressure:** Systolic or diastolic blood pressure measurements.
- **Body Mass Index (BMI):** An indicator of body fat based on height and weight.
- **Age:** Age of the individual, as diabetes risk can increase with age.
- **Insulin Levels:** Levels of insulin in the blood.
- **Skin Thickness:** Skin fold thickness as an indirect measure of body fat.
- **Pregnancies:** The number of times a person has been pregnant, relevant for gestational diabetes.
- **Outcome:** A binary variable indicating whether or not a patient has diabetes (1 for diabetes, 0 for no diabetes).

Implementation:

1. Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import seaborn as sns
```

2. Loading the Data

```
data = pd.read_csv('/content/diabetes.csv')
print(data.info())
print("Missing values:\n", data.isnull().sum())
print(data.describe())
```

3. Scatter Plot Matrix

```
sns.pairplot(data, hue="Outcome")
plt.suptitle("Scatter Plot Matrix", y=1.02)
plt.show()
```

4. **Correlation matrix and heatmap**

```
plt.figure(figsize=(10,8))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

5. Train test split

```
X = data.drop('Outcome', axis=1)
y = data['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

6. Logistic Regression Model

```
logreg = LogisticRegression(max_iter=200)
logreg.fit(X_train, y_train)
logreg_predictions = logreg.predict(X_test)
print("Logistic Regression Confusion Matrix:\n", confusion_matrix(y_test,
                                                    logreg_predictions))
```

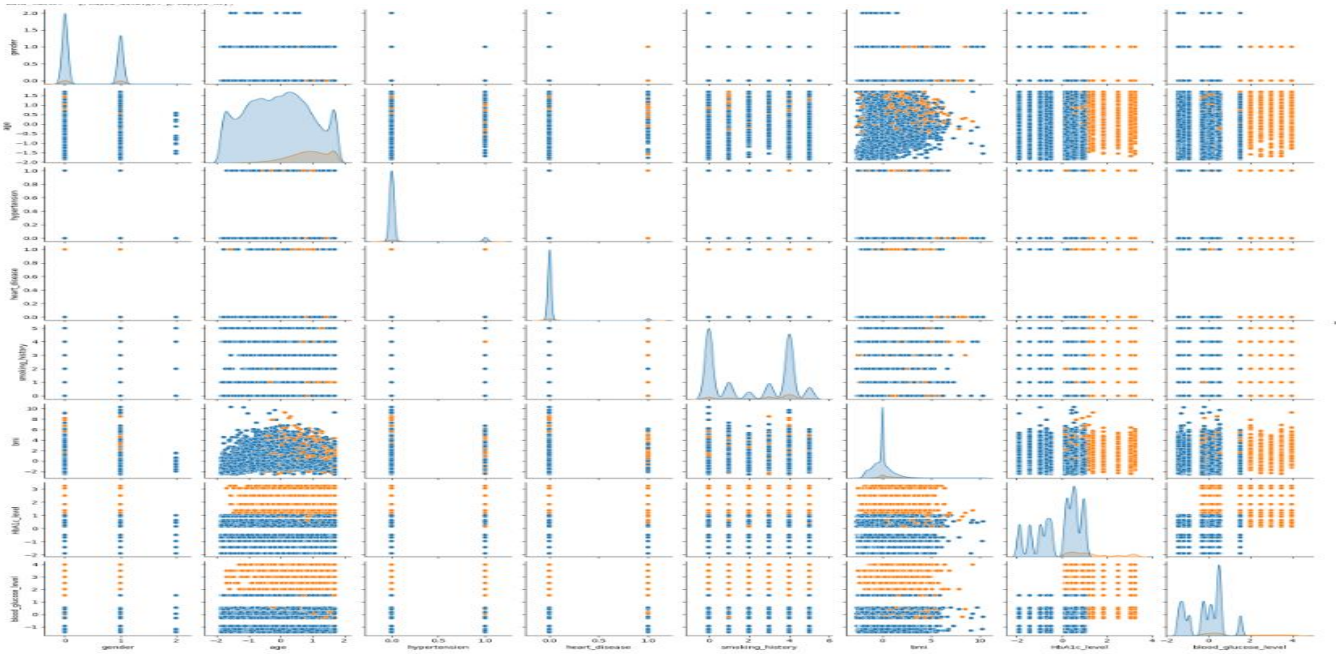
```
print("Logistic Regression Accuracy:", accuracy_score(y_test, logreg_predictions))
print("Logistic Regression Classification Report:\n", classification_report(y_test,
                                                                    logreg_predictions))
```

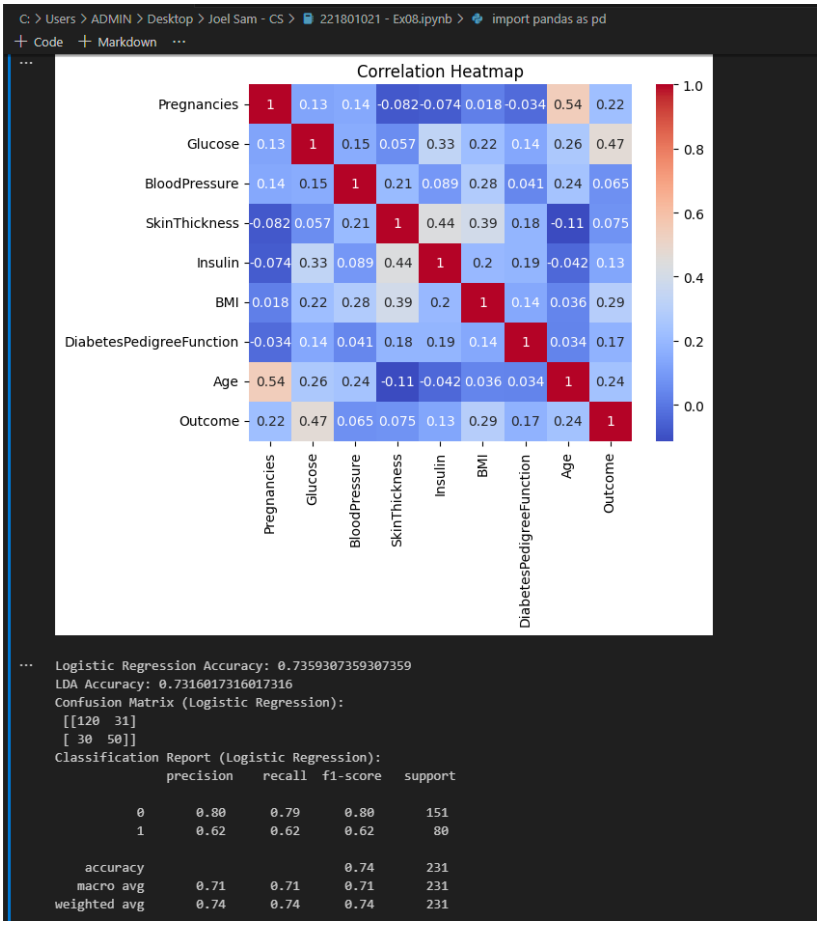
7. Linear Discriminant Analysis (LDA)

```
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
lda_predictions = lda.predict(X_test)
print("LDA Confusion Matrix:\n", confusion_matrix(y_test, lda_predictions))
print("LDA Accuracy:", accuracy_score(y_test, lda_predictions))
print("LDA Classification Report:\n", classification_report(y_test, lda_predictions))
```

Output:

```
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex08.ipynb > import pandas as pd
+ Code + Markdown ···
···     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
    0            6      148             72             35        0  33.6
    1            1       85             66             29        0  26.6
    2            8      183             64              0        0  23.3
    3            1       89             66             23       94  28.1
    4            0      137             40             35      168  43.1

       DiabetesPedigreeFunction  Age  Outcome
    0                     0.627   50        1
    1                     0.351   31        0
    2                     0.672   32        1
    3                     0.167   21        0
    4                     2.288   33        1
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 768 entries, 0 to 767
    Data columns (total 9 columns):
     #   Column                    Non-Null Count  Dtype
    ---  ------                    --------------  -----
     0   Pregnancies               768 non-null    int64
     1   Glucose                   768 non-null    int64
     2   BloodPressure             768 non-null    int64
     3   SkinThickness             768 non-null    int64
     4   Insulin                   768 non-null    int64
     5   BMI                       768 non-null    float64
     6   DiabetesPedigreeFunction  768 non-null    float64
    ...
     8   Outcome                   768 non-null    int64
    dtypes: float64(2), int64(7)
    memory usage: 54.1 KB
    None
```





Result:

The models demonstrated a reasonable ability to predict diabetes based on medical measurements, and further tuning, such as hyperparameter optimization, could enhance performance. These results emphasize the importance of feature selection and preprocessing in developing effective predictive models for healthcare applications.

EXP.NO:9	WINE QUALITY PREDICTION
DATE:	

Problem Statement:

The goal is to predict the quality of wine based on various chemical properties. By analyzing these properties, we aim to determine how they impact the overall quality rating of the wine, providing insights that could assist winemakers, distributors, and consumers in assessing wine quality without extensive taste testing.

Objective:

- Identifying significant chemical factors affecting wine quality.
- Building a regression model that uses these factors to predict wine quality.
- Visualizing the distribution and relationships of different chemical properties with wine quality.
- Conducting a factor analysis to uncover underlying factors influencing wine quality.

Dataset Description:

- **Fixed Acidity:** Non-volatile acids that contribute to the total acidity of the wine.
- **Volatile Acidity:** Acids that evaporate easily, contributing to a vinegar taste if in excess.
- **Citric Acid:** Found naturally in wine, contributes to freshness and flavor.
- **Residual Sugar:** Sugar remaining after fermentation, affecting sweetness.
- **Chlorides:** Salt content in wine.
- **Free Sulfur Dioxide:** Sulfur dioxide that is not chemically bound, acting as a preservative.
- **Total Sulfur Dioxide:** Sum of free and bound sulfur dioxide, affecting shelf life and stability.
- **Density:** Density of wine, influenced by sugar, alcohol, and other dissolved substances.
- **pH:** Measure of acidity/alkalinity; affects taste and preservation.
- **Sulphates:** Acts as an antimicrobial and antioxidant.
- **Alcohol:** Alcohol content by volume, a key determinant in wine character and quality.
- **Quality:** Target variable, often rated on a scale (e.g., 0 to 10), representing the wine's quality based on sensory evaluations.

Implementation:

1. **Import necessary libraries**

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.decomposition import FactorAnalysis
from sklearn.metrics import mean_squared_error, r2_score
```

2. **Class for handling the wine dataset and analysis**

```
class WineQualityPrediction:
    def __init__(self, file_path):
        self.data = pd.read_csv(file_path)
        self.X = self.data.drop("quality", axis=1)
        self.y = self.data["quality"]
    # Method to display dataset information
    def data_overview(self):
        print(self.data.info())
        print("\nMissing Values:\n", self.data.isnull().sum())
        print("\nSummary Statistics:\n", self.data.describe())
    # Method for visualizing the data
    def visualize_data(self):
        # Histograms
        self.data.hist(bins=15, figsize=(15, 10))
        plt.suptitle("Histograms of Wine Quality Dataset Features")
        plt.show()
        # Box plots
        plt.figure(figsize=(15, 8))
        sns.boxplot(data=self.data, orient="h")
        plt.title("Box Plot of Wine Quality Dataset Features")
        plt.show()
    # Method to perform multiple linear regression
    def multiple_regression(self):
        X_train, X_test, y_train, y_test = train_test_split(self.X, self.y, test_size=0.3,
```

```
                                                                    random_state=42)
                model = LinearRegression()
                model.fit(X_train,  y_train)
                # Predictions and evaluation
                predictions = model.predict(X_test)
                mse = mean_squared_error(y_test, predictions)
                r2 = r2_score(y_test, predictions)
                print("Multiple Regression MSE:", mse)
                print("Multiple Regression R-squared:", r2)
            # Method to perform factor analysis
            def factor_analysis(self, n_factors=5):
                fa = FactorAnalysis(n_components=n_factors, random_state=42)
                fa.fit(self.X)
                # Display factor loadings
                loadings = pd.DataFrame(fa.components_, columns=self.X.columns)
                print(f"Factor Loadings for {n_factors} Factors:\n", loadings)
        # Instantiate the class and use its methods
        file_path = '/content/winequality-dataset_updated.csv'
        wine_quality = WineQualityPrediction(file_path)
```

3. Display data overview

```
wine_quality.data_overview()
```

4. Visualize data

```
wine_quality.visualize_data()
```

5. Perform multiple regression

```
wine_quality.multiple_regression()
```

5. Perform factor analysis

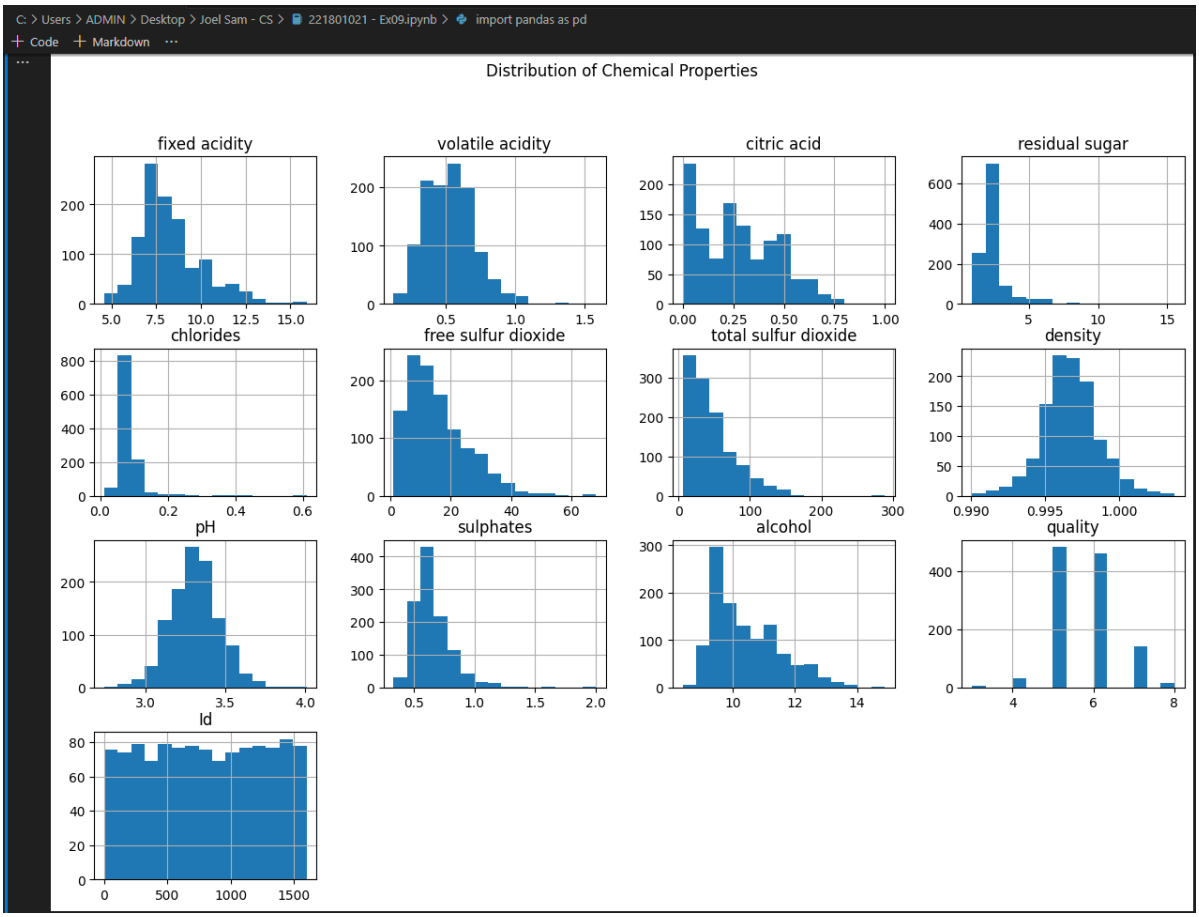
```
wine_quality.factor_analysis(n_factors=5)
```

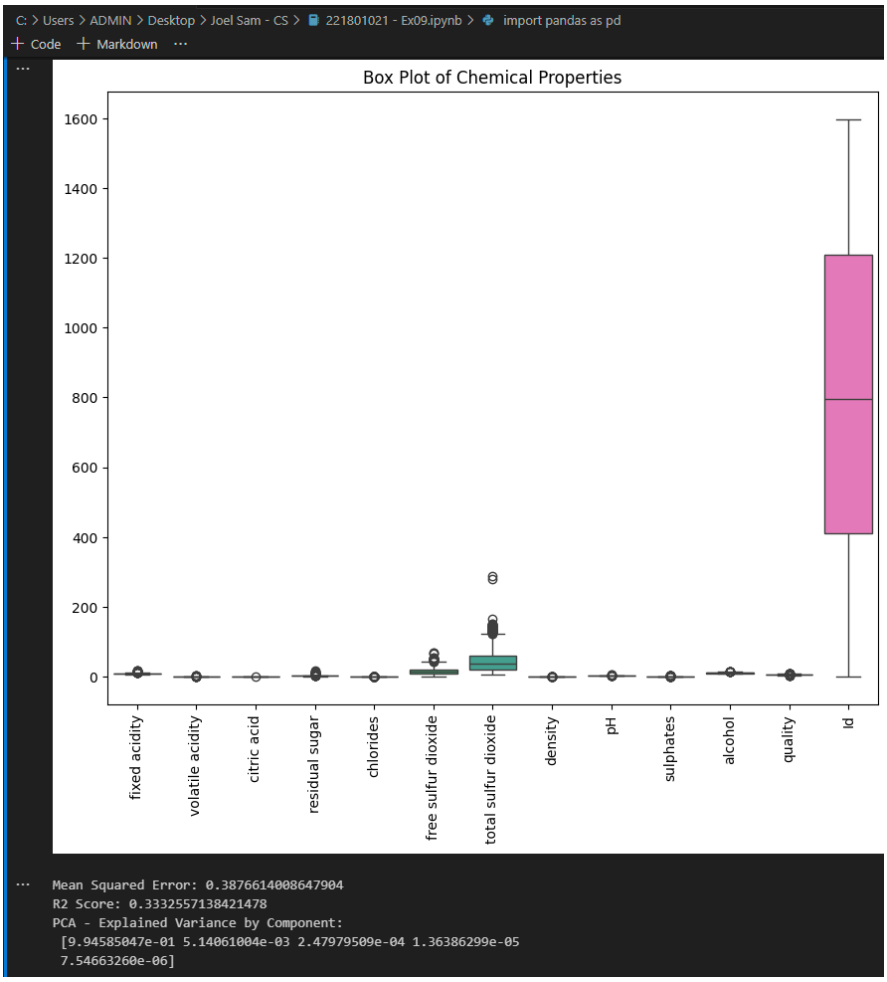
Output:

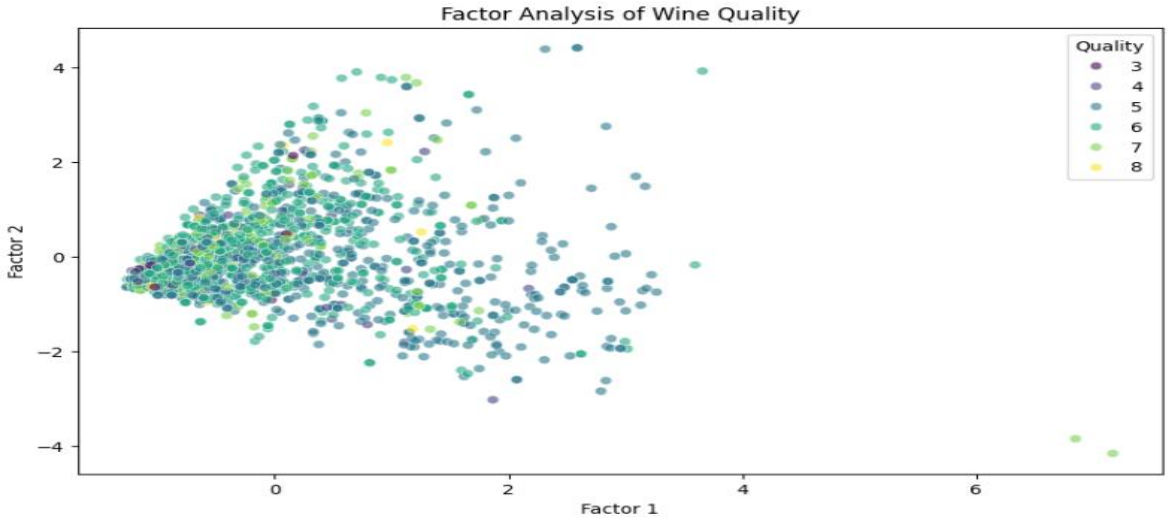
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex09.ipynb > import pandas as pd

+ Code + Markdown ···

```
···      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
    0             7.4              0.70         0.00             1.9      0.076
    1             7.8              0.88         0.00             2.6      0.098
    2             7.8              0.76         0.04             2.3      0.092
    3            11.2              0.28         0.56             1.9      0.075
    4             7.4              0.70         0.00             1.9      0.076

       free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
    0                 11.0                  34.0   0.9978  3.51       0.56
    1                 25.0                  67.0   0.9968  3.20       0.68
    2                 15.0                  54.0   0.9970  3.26       0.65
    3                 17.0                  60.0   0.9980  3.16       0.58
    4                 11.0                  34.0   0.9978  3.51       0.56

       alcohol  quality  Id
    0      9.4        5   0
    1      9.8        5   1
    2      9.8        5   2
    3      9.8        6   3
    4      9.4        5   4
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 1143 entries, 0 to 1142
    Data columns (total 13 columns):
     #   Column                Non-Null Count  Dtype
    ---  ------                --------------  -----
    ...
     12  Id                    1143 non-null   int64
    dtypes: float64(11), int64(2)
    memory usage: 116.2 KB
    None
```







Result:

The project effectively developed a predictive model for wine quality, providing insights into the distributions of chemical properties and their relationships with quality ratings. The findings can assist winemakers in understanding which factors most influence wine quality and inform better decision-making in wine production. Future work could explore more advanced models or feature engineering techniques to enhance prediction accuracy.

EXP.NO:10	HEART DISEASE PREDICTION
DATE:	

Problem Statement:

Heart disease is one of the leading causes of morbidity and mortality worldwide. Early and accurate diagnosis is crucial for effective treatment and management. However, diagnosing heart disease can be challenging due to the complex interplay of various clinical factors. The goal of this project is to leverage data science techniques to predict the presence of heart disease based on a set of clinical parameters, thereby assisting healthcare professionals in making informed decisions.

Objective:

1. Predictive Modeling: Develop a predictive model using logistic regression to classify individuals as having heart disease or not based on clinical data.
2. Feature Engineering: Analyze and preprocess clinical parameters, standardize data, and perform dimensionality reduction using Principal Component Analysis (PCA) to enhance model performance.
3. Visualization: Employ visualization techniques like pair plots to explore relationships between variables, and ROC curves to assess model performance.
4. Evaluation: Assess the model's accuracy and interpret its predictions through various metrics such as accuracy, AUC, and a classification report.

Dataset Description:

- age: Age of the patient in years.
- sex: Gender of the patient (1 = male, 0 = female).
- cp (Chest Pain Type): Type of chest pain experienced by the patient:
  1. Typical angina
  2. Atypical angina
  3. Non-anginal pain
  4. Asymptomatic
- trestbps (Resting Blood Pressure): Resting blood pressure in mm Hg on admission to the hospital.
- chol (Cholesterol): Serum cholesterol in mg/dL.
- fbs (Fasting Blood Sugar): Fasting blood sugar level (1 if > 120 mg/dL, 0 otherwise).
- restecg (Resting Electrocardiographic Results):
  1. Normal
  2. Having ST-T wave abnormality

3. Showing probable or definite left ventricular hypertrophy

- thalach (Maximum Heart Rate Achieved): The highest heart rate achieved during the test.
- exang (Exercise Induced Angina): Exercise-induced chest pain (1 = yes, 0 = no).
- oldpeak: ST depression induced by exercise relative to rest.
- slope: The slope of the peak exercise ST segment:
  1. Upsloping
  2. Flat
  3. Downsloping
- ca (Number of Major Vessels): Number of major vessels (0-3) colored by fluoroscopy.
- thal: A blood disorder type with values:
  1. Normal
  2. Fixed defect
  3. Reversible defect
- target: The target variable, indicating the presence of heart disease (1 = presence, 0 = absence).

Implementation:

1. Import necessary libraries

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

2. Load the dataset

```
data = pd.read_csv('/content/heart.csv')
```

3. Display the first few rows of the dataset

```
print(data.head())
```

4. Exploratory Data Analysis

```
# Display summary statistics
print(data.describe())
# Check for missing values
print(data.isnull().sum())
```

5. Data Preprocessing

```
# Separate features and target variable
X = data.drop('target', axis=1)    # Replace 'target' with the actual name if different
y = data['target']
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

6. Dimensionality Reduction using PCA

```
pca = PCA(n_components=2)    # Adjust n_components as needed
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

7. Logistic Regression Model

```
log_reg = LogisticRegression()
log_reg.fit(X_train_pca, y_train)
# Predictions
y_pred = log_reg.predict(X_test_pca)
y_pred_proba = log_reg.predict_proba(X_test_pca)[:, 1]
```

8. Model Evaluation

```
# Confusion Matrix and Classification Report
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
# Accuracy Score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

9. ROC Curve and AUC

```
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)
```

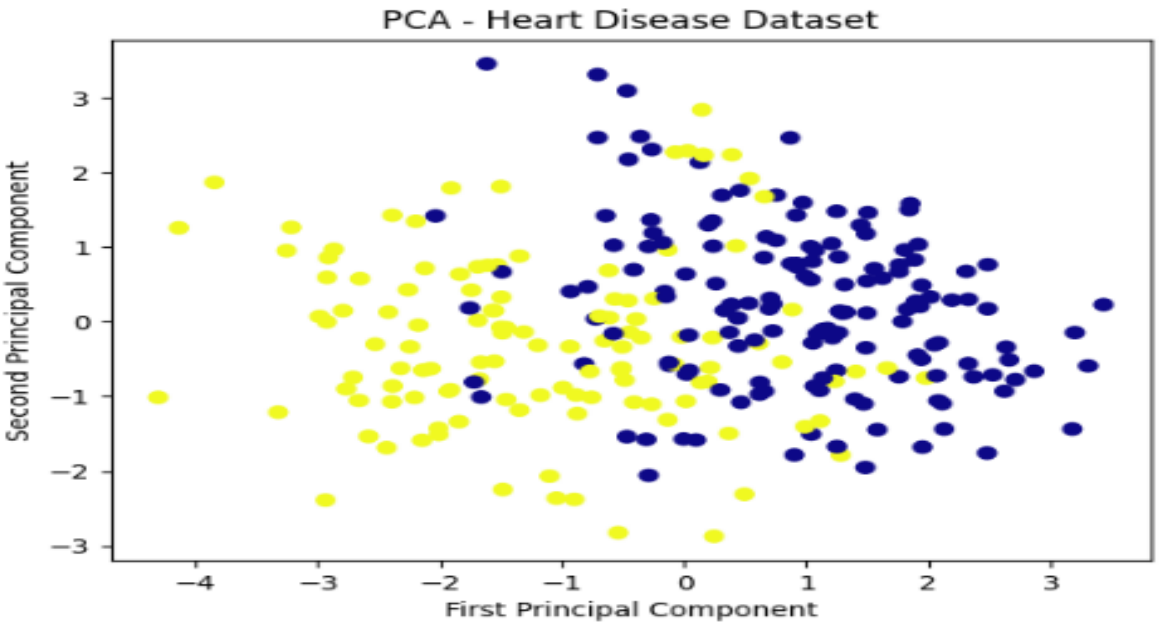
10. Plotting the ROC Curve

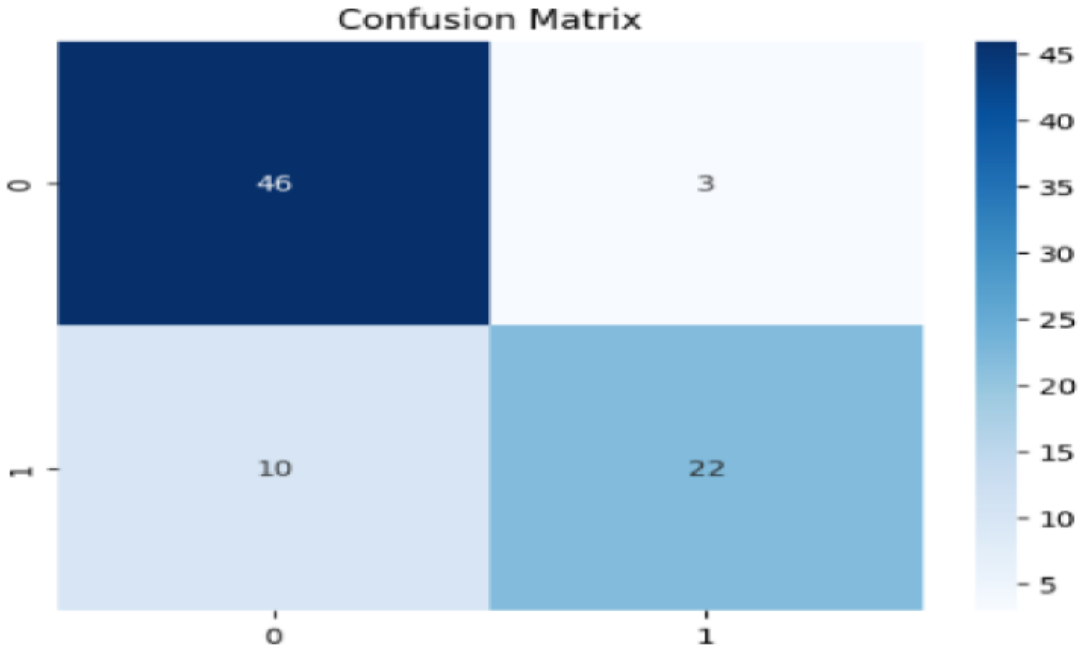
```
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'Logistic Regression (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
plt.title("ROC Curve")
plt.legend()
plt.show()
```

OUTPUT:

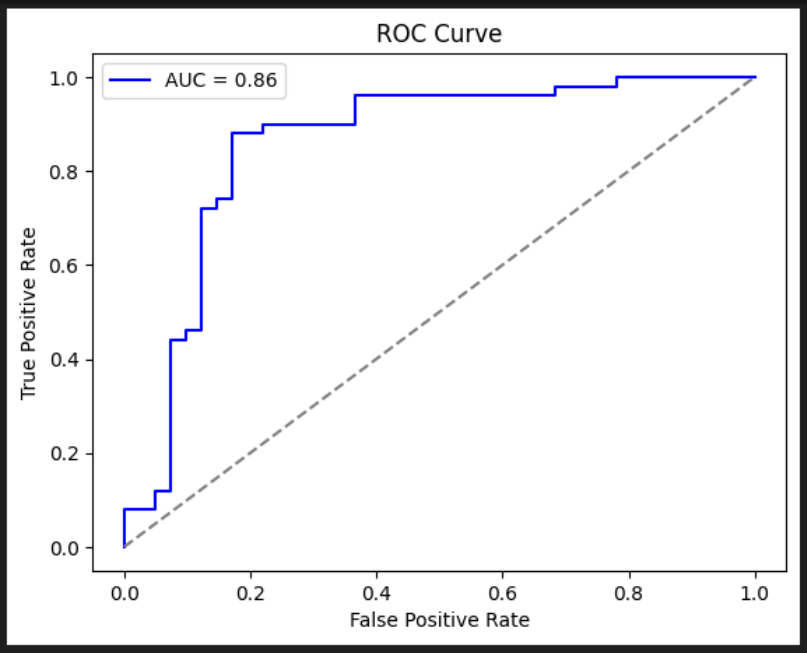
```
C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex10.ipynb > import pandas as pd
+ Code + Markdown ···

···      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
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 303 entries, 0 to 302
     Data columns (total 14 columns):
      #   Column    Non-Null Count  Dtype
     ---  ------    --------------  -----
      0   age       303 non-null    int64
      1   sex       303 non-null    int64
      2   cp        303 non-null    int64
      3   trestbps  303 non-null    int64
      4   chol      303 non-null    int64
      5   fbs       303 non-null    int64
      6   restecg   303 non-null    int64
     ...
      13  target    303 non-null    int64
     dtypes: float64(1), int64(13)
     memory usage: 33.3 KB
     None
```


PCA - Heart Disease Dataset
Second Principal Component
First Principal Component
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3


Confusion Matrix
46
3
10
22
0
1
0
1
45
40
35
30
25
20
15
10
5

C: > Users > ADMIN > Desktop > Joel Sam - CS > 221801021 - Ex10.ipynb > import pandas as pd

+ Code + Markdown ···




```
Accuracy: 0.8461538461538461
Confusion Matrix:
 [[34  7]
 [ 7 43]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83        41
           1       0.86      0.86      0.86        50

    accuracy                           0.85        91
   macro avg       0.84      0.84      0.84        91
weighted avg       0.85      0.85      0.85        91
```

RESULT:

The results demonstrate that clinical parameters can be effectively used to predict the likelihood of heart disease. The combination of exploratory data analysis, model training, and evaluation yielded a robust predictive model that can aid healthcare professionals in early detection and intervention strategies for heart disease. Further refinements, such as exploring advanced modeling techniques and optimizing hyperparameters, could enhance predictive accuracy and clinical applicability.