INTRODUCTION TO PYTHON AND COMPUTER PROGRAMMING

BY

AYUSHI SINGH

INDEX

Ass. No.	PROLEM STATEMENT	PAGES	REMARKS
1	Implement k-nearest neighbours classification using python		
2	Extract the data from database using python		
3	The probability that it is Friday and that a student is absent is 3 %. Since there are 5 school days in a week, the probability that it is Friday is 20 %. What is the probability that a student is absent given that today is Friday? Apply Baye’s rule in python to get the result		
4	Predict Canada's per capita income in year 2020.		
5	Employee retention dataset		
6	Predict a classification for a case where VAR1=0.906 and VAR2=0.606		
7	Predict if a person would buy Insurance or not using Logistic Regression		
8	Implement linear regression using python.		

9	**Implement Naïve Bayes theorem to classify the English text.**		
10	**Use wine dataset from sklearn.datasets to classify wines into 3 categories.**		
11	**Heart disease dataset.**		
12	**Write a python program to import and export data using Pandas library functions.**		
13	**Using Python implement Dimensionality reduction using Principle Component Analysis (PCA) method.**		
14	**Using Python implement Simple and Multiple Linear Regression Models**		
15	**Using Python develop Logistic Regression Model for a given dataset.**		
16	**Using Python develop Decision Tree Classification model for a given dataset and use it to classify a new sample.**		

Assignment 1

Ass. 1: Implement k-nearest neighbours classification using python

In [ ]:
```
# Question 1. Write a python program to implement KNN using sklear
n.
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')

# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
=0.2, random_state=42)

# Create a KNN classifier with k=3 (you can change this value as ne
eded)
knn_classifier = KNeighborsClassifier(n_neighbors=3)

# Fit the model to the training data
knn_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn_classifier.predict(X_test)

# Calculate and print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
<ipython-input-1-76c9024a1781>:2: DeprecationWarning: `set_matplotl
ib_formats` is deprecated since IPython 7.23, directly use `matplot
lib_inline.backend_inline.set_matplotlib_formats()`
  set_matplotlib_formats('pdf', 'svg')

Accuracy: 1.0
```

Assignment 2

Ass. 2: Extract the data from database using python

In [1]:
```python
import sqlite3

# Connect to a database (this will create a new file named 'exampl
e.db' if it doesn't exist)
connection = sqlite3.connect('example.db')

# Create a cursor object to execute SQL queries
cursor = connection.cursor()

# Create a table
cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY,
        name TEXT,
        age INTEGER
    )
''')

# Insert some data
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('Joh
n Doe', 25))
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ('Jan
e Smith', 30))

# Commit the changes and close the connection
connection.commit()
connection.close()

import sqlite3

# Connect to the database
connection = sqlite3.connect('example.db')
cursor = connection.cursor()

# Execute a SELECT query
cursor.execute("SELECT * FROM users")

# Fetch all the rows
rows = cursor.fetchall()

# Display the data
for row in rows:
    print(row)

# Close the connection
connection.close()
```

```
(1, 'John Doe', 25)
(2, 'Jane Smith', 30)
```

Assignment 3

Ass. 3: The probability that it is Friday and that a student is absent is 3 %. Since there are 5 school days in a week, the probability that it is Friday is 20 %. What is the probability that a student is absent given that today is Friday? Apply Baye's rule in python to get the result

In [ ]:
```
#Question 3. Probability of friday and student is absent..

p_A_given_B = 0.03
p_B =0.2

#p_not_A= 1-p_A

result= (p_A_given_B/p_B)
print("Answer is : ", result)
```

```
Answer is :  0.15
```

Assignment 4

Ass 4: Predict Canada's per capita income in year 2020. There is a data folder here on Kaggle, download that and you will find canada_per_capita_income.csv file. Using this build a regression model and predict the per capita income for Canadian citizens in year 2020. Link for csv file: https://www.kaggle.com/datasets/gurdit559/canada-per-capita-income-single-variable-data-se (https://www.kaggle.com/datasets/gurdit559/canada-per-capita-income-single-variable-data-se)

In [ ]:
```
#Question 4. Canada's per capita income
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, confusion_matrix

data = pd.read_csv("/content/Canada_per_capita_income (1).csv",sep
=",")
data
print(data.head())
x=data[['year']]
y=data[['income']]
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.3,
random_state=42)

model = LinearRegression()
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
prediction_2020=model.predict([[2020]])
print(f'income:{prediction_2020[0]}')
```

```
   year       income
0  1970  3399.299037
1  1971  3768.297935
2  1972  4251.175484
3  1973  4804.463248
4  1974  5576.514583
income:[40993.56532482]

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWa
rning: X does not have valid feature names, but LinearRegression wa
s fitted with feature names
  warnings.warn(
```

Assignment 5

Ass 5: Download employee retention dataset from here: [https://www.kaggle.com/giripujar/hr analytics (https://www.kaggle.com/giripujar/hr analytics)](https://www.kaggle.com/giripujar/hr-analytics).

1. Now do some exploratory data analysis to figure out which variables have direct and clear impact on employee retention (i.e., whether they leave the company or continue to work)
2. Plot bar charts showing impact of employee salaries on retention
3. Plot bar charts showing correlation between department and employee retention
4. Now build logistic regression model using variables that were narrowed down in step 1
5. Measure the accuracy of the model

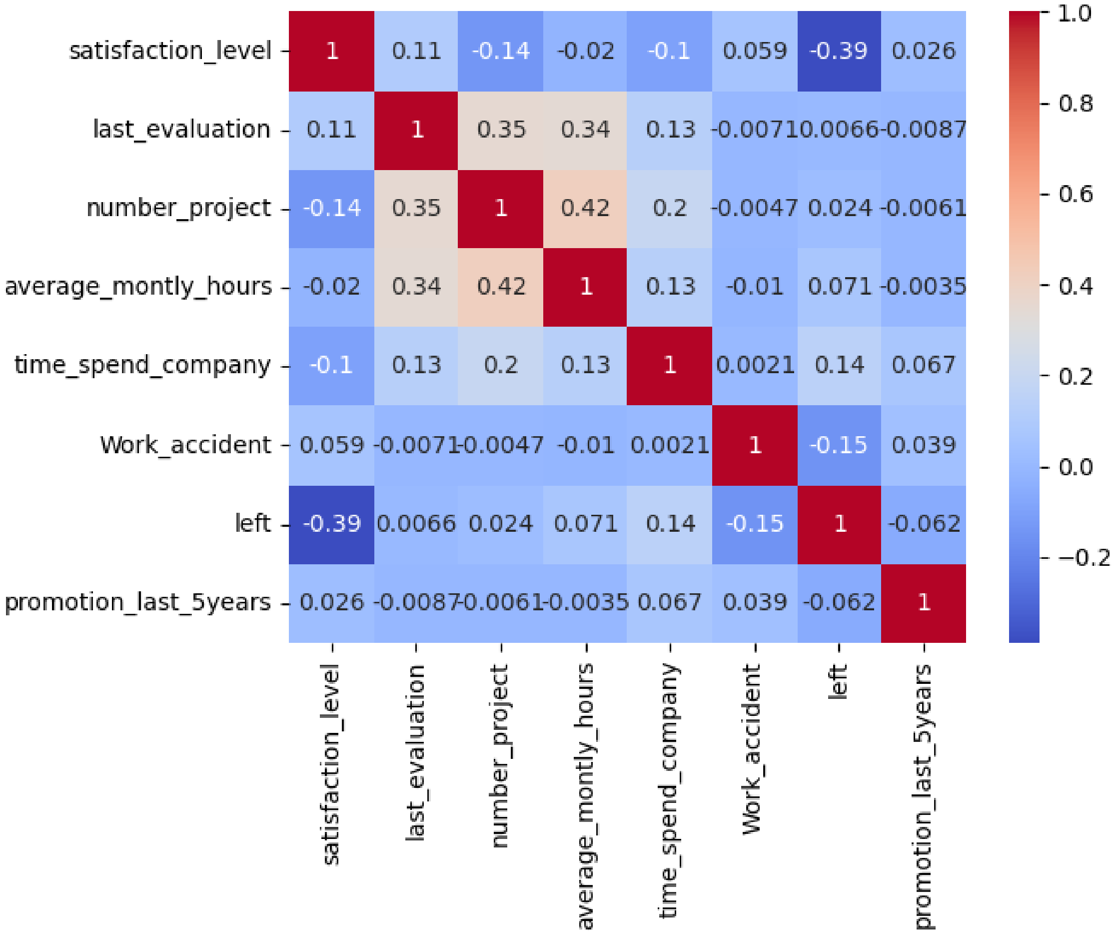
In [ ]:

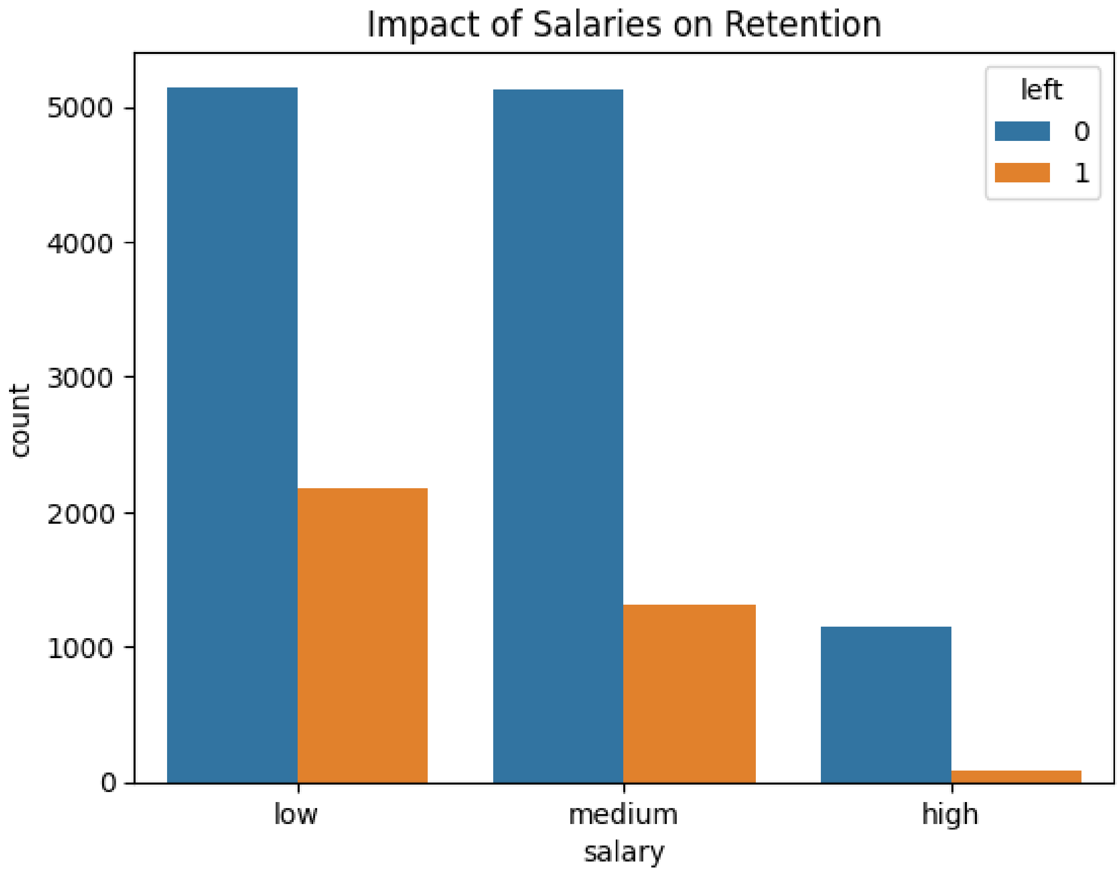
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Load the dataset
data = pd.read_csv('/content/HR_comma_sep.csv')

# 1. Exploratory Data Analysis (EDA)
# Explore the dataset to identify variables impacting retention
# Use methods like describe(), info(), and correlation analysis
# For example:
print(data.describe())
print(data.info())
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()

# 2. Impact of salaries on retention
sns.countplot(x='salary', hue='left', data=data)
plt.title('Impact of Salaries on Retention')
plt.show()

# 3. Correlation between department and retention
sns.countplot(x='Department', hue='left', data=data)
plt.title('Correlation between Department and Retention')
plt.xticks(rotation=45)
plt.show()

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['salary']=le.fit_transform(data['salary'])
data['Department']=le.fit_transform(data['Department'])
# 4. Build Logistic Regression Model
X=data.iloc[:,:-1]
y=data.iloc[:,-1]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
=0.2, random_state=42)

# Initialize the logistic regression model
model = LogisticRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# 5. Measure the accuracy of the model
# Predict the values using the test set
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy of the model: {accuracy}')

# Optionally, you can also print the confusion matrix
```

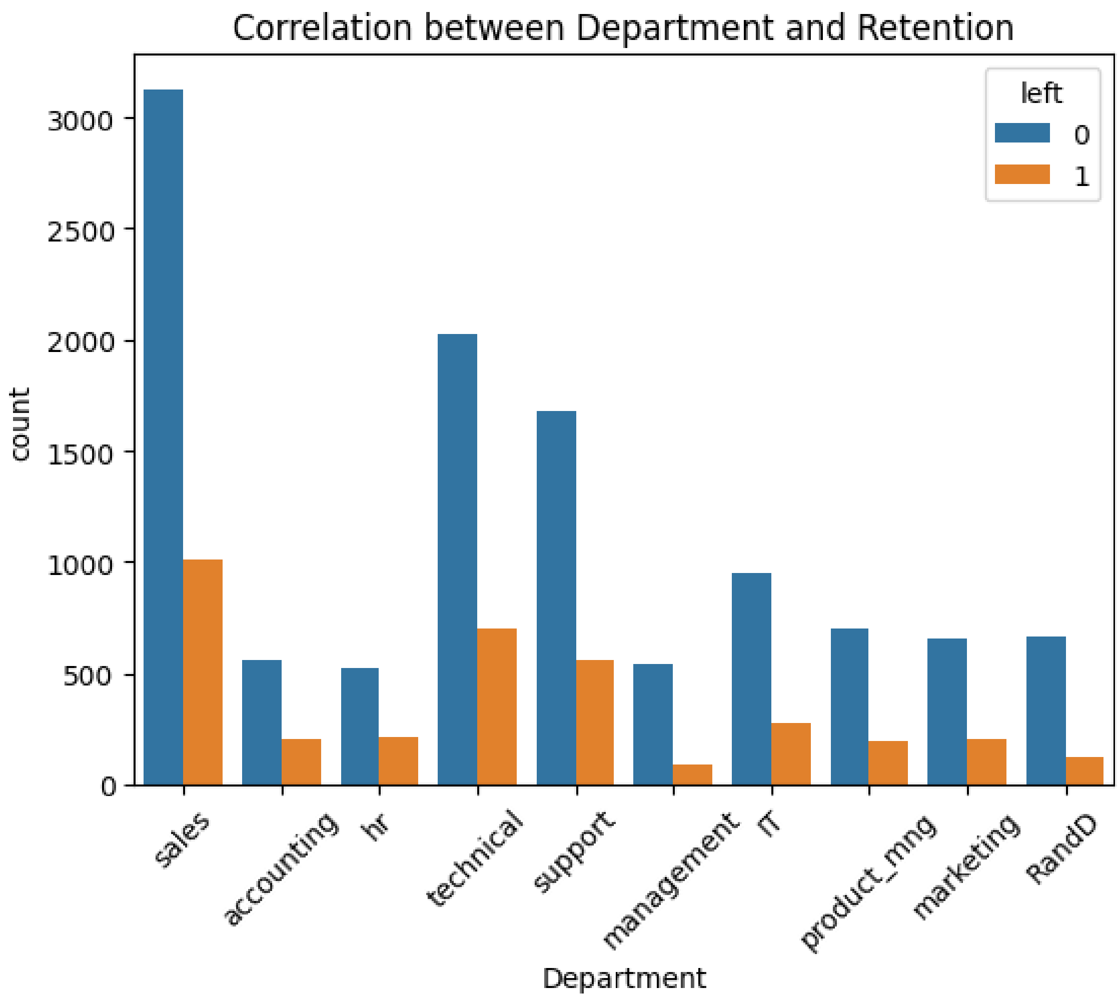
```
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n{conf_matrix}')
```

```
       satisfaction_level  last_evaluation  number_project  \
count        14999.000000     14999.000000    14999.000000
mean             0.612834         0.716102        3.803054
std              0.248631         0.171169        1.232592
min              0.090000         0.360000        2.000000
25%              0.440000         0.560000        3.000000
50%              0.640000         0.720000        4.000000
75%              0.820000         0.870000        5.000000
max              1.000000         1.000000        7.000000

       average_montly_hours  time_spend_company  Work_accident
left  \
count          14999.000000        14999.000000   14999.000000  149
99.000000
mean             201.050337            3.498233       0.144610
0.238083
std               49.943099            1.460136       0.351719
0.425924
min               96.000000            2.000000       0.000000
0.000000
25%              156.000000            3.000000       0.000000
0.000000
50%              200.000000            3.000000       0.000000
0.000000
75%              245.000000            4.000000       0.000000
0.000000
max              310.000000           10.000000       1.000000
1.000000

       promotion_last_5years
count           14999.000000
mean                0.021268
std                 0.144281
min                 0.000000
25%                 0.000000
50%                 0.000000
75%                 0.000000
max                 1.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
None
```

```
<ipython-input-8-2e65f0a87cc2>:19: FutureWarning: The default value
of numeric_only in DataFrame.corr is deprecated. In a future versio
n, it will default to False. Select only valid columns or specify t
he value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```




Impact of Salaries on Retention
count
5000
4000
3000
2000
1000
0
low
medium
high
salary
left
0
1


Correlation between Department and Retention
count
3000
2500
2000
1500
1000
500
0
sales
accounting
hr
technical
support
management
IT
product_mng
marketing
RandD
Department
left
0
1

```
Accuracy of the model: 0.5043333333333333
Confusion Matrix:
[[  0 123 130]
 [  0 988 486]
 [  0 748 525]]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logis
tic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as s
hown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver optio
ns:
    https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression
  n_iter_i = _check_optimize_result(
```

Assignment 6

Given the following data, which specify classifications for nine combinations of VAR1 and VAR2 predict a classification for a case where VAR1=0.906 and VAR2=0.606, using the result of kmeans clustering with 3 means (i.e., 3 centroids) VAR1 VAR2 CLASS 1.713 1.586 0 0.180 1.786 1 0.353 1.240 1 0.940 1.566 0 1.486 0.759 1 1.266 1.106 0 1.540 0.419 1 0.459 1.799 1 0.773 0.186 1

```
In [ ]: #Ass 6
from sklearn.cluster import KMeans
import numpy as np
data = np.array([
    [1.713, 1.586, 0],
    [0.180, 1.786, 1],
    [0.353, 1.240, 1],
    [0.940, 1.566, 0],
    [1.486, 0.759, 1],
    [1.266, 1.106, 0],
    [1.540, 0.419, 1],
    [0.459, 1.799, 1],
    [0.773, 0.186, 1]
])

X = data[:, :2]
new_case = np.array([[0.906, 0.606]])
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
predicted_cluster = kmeans.predict(new_case)
predicted_class = int(data[predicted_cluster, 2])

print(f"The predicted class for VAR1=0.906 and VAR2=0.606 is: {pred
icted_class}")
```

```
The predicted class for VAR1=0.906 and VAR2=0.606 is: 1

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:
870: FutureWarning: The default value of `n_init` will change from
10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre
ss the warning
  warnings.warn(
```

Assignment 7

Ass. 7: Predict if a person would buy Insurance or not using Logistic Regression the insurance-data.csv file existing on the link given below. https://www.kaggle.com/datasets/adepvenugopal/insurance-data (https://www.kaggle.com/datasets/adepvenugopal/insurance-data)

In [ ]:
```
#Question 7.
import pandas as pd
import numpy as np
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

data = pd.read_csv("/content/insurance_data.csv",sep=",")
data
print(data.head())
x=data[['age']]
y=data[['bought_insurance']]
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.3,
random_state=42)

model = LogisticRegression()
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
predict_yes_no=model.predict([[30]])
print(f'Yes_Not:{predict_yes_no[0]}')
```

```
   age  bought_insurance
0   22                 0
1   25                 0
2   47                 1
3   52                 0
4   46                 1
Yes_Not:0

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.p
y:1143: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWa
rning: X does not have valid feature names, but LogisticRegression
was fitted with feature names
  warnings.warn(
```

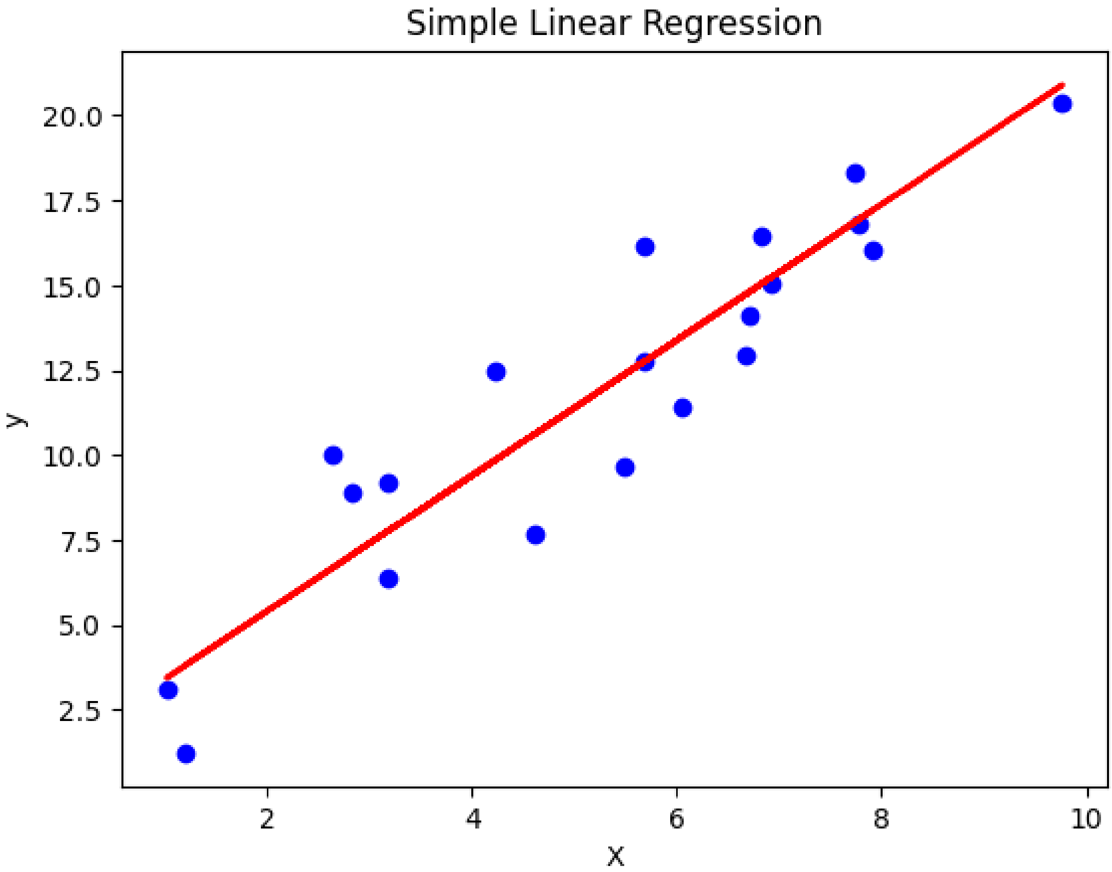
Assignment 8

Ass. 8: Implement linear regression using python.

In [ ]:

```
# Importing the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Generate some sample data
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1) * 2 # Adding some random no
ise

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

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

# Calculate the mean squared error and R-squared score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# Print the results
print("Coefficients:", coefficients)
print("Intercept:", intercept)
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)

# Plot the regression line
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.xlabel("X")
plt.ylabel("y")
plt.title("Simple Linear Regression")
plt.show()
```

```
Coefficients: [[1.99610364]]
Intercept: [1.41268038]
Mean Squared Error: 3.671012987885715
R-squared Score: 0.8453207776609701
```



Assignment 9

Ass. 9: Implement Naïve Bayes theorem to classify the English text.

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Sample data for demonstration
# You should replace this with your own dataset
texts = ["This is a positive sentence.", "Negative sentiment her
e.", "Another positive example.", "Negative vibes."]

# Labels for the corresponding texts
labels = ["positive", "negative", "positive", "negative"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(texts, labels,
test_size=0.2, random_state=42)

# Create a CountVectorizer to convert text data into a bag-of-words
representation
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Create and train a Multinomial Naive Bayes classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train_vec, y_train)

# Make predictions on the test set
y_pred = nb_classifier.predict(X_test_vec)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Display classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```

Assignment. 10

Ass. 10: Use wine dataset from sklearn.datasets to classify wines into 3 categories. Load the dataset and split it into test and train. After that train the model using Gaussian and Multinominal classifier and post which model performs better. Use the trained model to perform some predictions on test data

In [ ]:

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Load the Wine dataset
wine = load_wine()
X = wine.data
y = wine.target

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

# Train Gaussian Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions on the test set
y_pred_gaussian = gnb.predict(X_test)

# Evaluate Gaussian Naive Bayes classifier
accuracy_gaussian = accuracy_score(y_test, y_pred_gaussian)
print("Gaussian Naive Bayes Classifier:")
print(f"Accuracy: {accuracy_gaussian:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred_gaussian))

# Train Multinomial Naive Bayes classifier
mnb = MultinomialNB()
mnb.fit(X_train, y_train)

# Make predictions on the test set
y_pred_multinomial = mnb.predict(X_test)

# Evaluate Multinomial Naive Bayes classifier
accuracy_multinomial = accuracy_score(y_test, y_pred_multinomial)
print("\nMultinomial Naive Bayes Classifier:")
print(f"Accuracy: {accuracy_multinomial:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred_multinomial))
```

```
Gaussian Naive Bayes Classifier:
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36


Multinomial Naive Bayes Classifier:
Accuracy: 0.89
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.93        14
           1       0.93      0.93      0.93        14
           2       0.83      0.62      0.71         8

    accuracy                           0.89        36
   macro avg       0.88      0.85      0.86        36
weighted avg       0.89      0.89      0.88        36
```

Assignment. 11

Ass. 11: Download heart disease dataset heart.csv and do following, (credits of dataset: https://www.kaggle.com/fedesoriano/heart-failure-prediction (https://www.kaggle.com/fedesoriano/heart-failure-prediction))

1. Load heart disease dataset in pandas dataframe
2. Convert text columns to numbers using label encoding and one hot encoding
3. Apply scaling
4. Build a classification model using various methods (SVM, logistic regression, random forest) and check which model gives you the best accuracy
5. Now use PCA to reduce dimensions, retrain your model and see what impact it has on your model in terms of accuracy. Keep in mind that many times doing PCA reduces the accuracy but computation is much lighter and that's the trade-off you need to consider while building models in real life

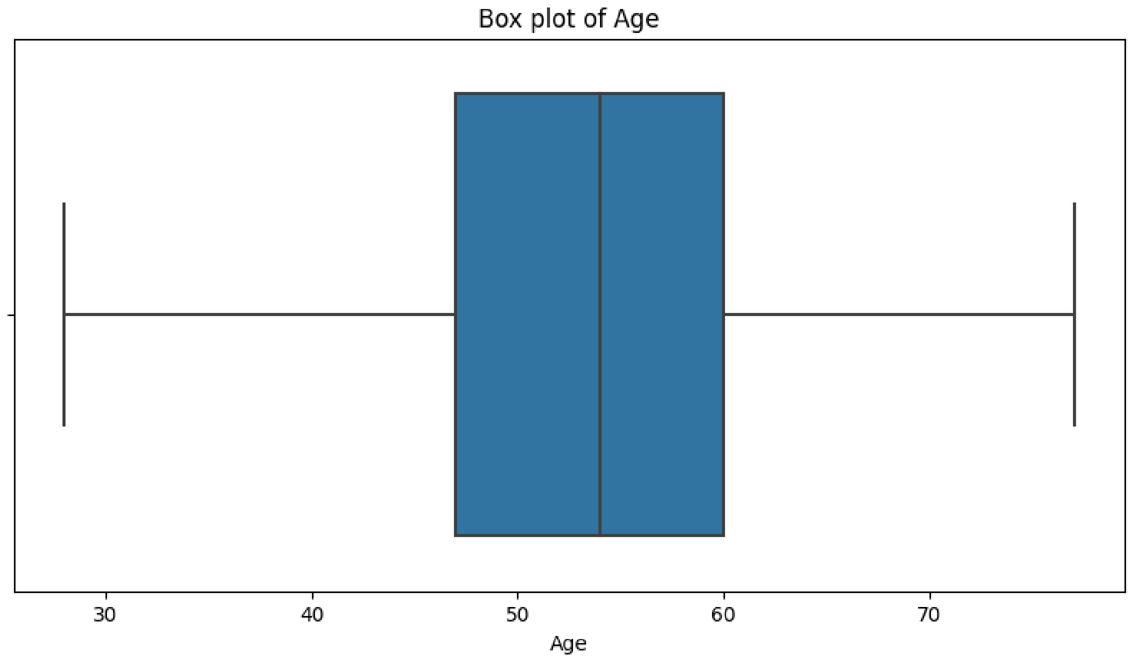
In [ ]:

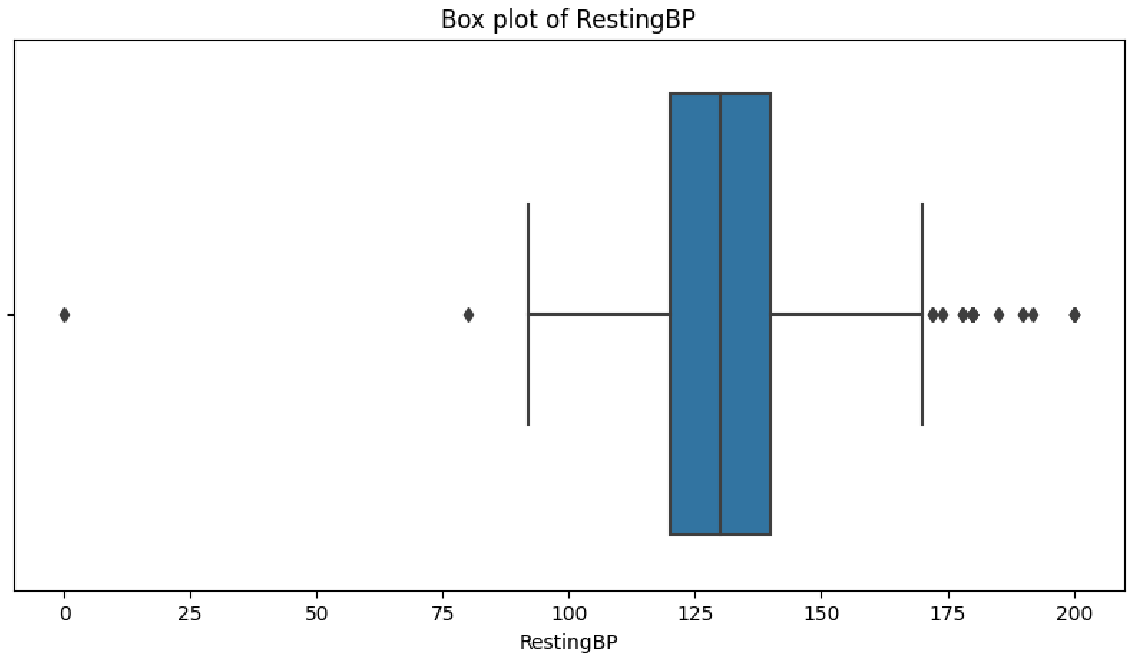
```
import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('/content/heart.csv')
data.head()
rows = data.shape[0]
cols = data.shape[1]
print(f'Rows : {rows}\nColumns : {cols}')
data.info()
data.describe()
data['HeartDisease'].value_counts()
continuous_columns = ['Age', 'RestingBP', 'Cholesterol', 'FastingB
S', 'MaxHR', 'Oldpeak']

# Plotting box plots for each continuous column
for column in continuous_columns:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=data[column])
    plt.title(f'Box plot of {column}')
    plt.show()
    gender_heart = pd.crosstab(data['Sex'], data['HeartDisease'],no
rmalize='index')
print(gender_heart)
gender_heart.plot(kind='bar')
plt.title('HeartDisease by Sex')
plt.show()
chest_heart = pd.crosstab(data['ChestPainType'], data['HeartDiseas
e'],normalize='index')
print(chest_heart)
chest_heart.plot(kind='bar')
plt.title('HeartDisease by ChestPainType')
plt.show()
rest_heart = pd.crosstab(data['RestingECG'], data['HeartDisease'],n
ormalize='index')
print(rest_heart)
rest_heart.plot(kind='bar')
plt.title('HeartDisease by rest type')
plt.show()
angina_heart = pd.crosstab(data['ExerciseAngina'], data['HeartDisea
se'],normalize='index')
print(angina_heart)
angina_heart.plot(kind='bar')
plt.title('HeartDisease by Angina')
plt.show()
st_heart = pd.crosstab(data['ST_Slope'], data['HeartDisease'],norma
lize='index')
print(st_heart)
st_heart.plot(kind='bar')
plt.title('HeartDisease by ST')
plt.show()
fasting_heart = pd.crosstab(data['FastingBS'], data['HeartDiseas
e'],normalize='index')
print(fasting_heart)
st_heart.plot(kind='bar')
plt.title('HeartDisease by fasting')
plt.show()
for column in continuous_columns:
  plt.figure(figsize=(10, 5))  # Set the figure size as desired
  plt.plot(data['HeartDisease'],data[column],linestyle='none', mark
```

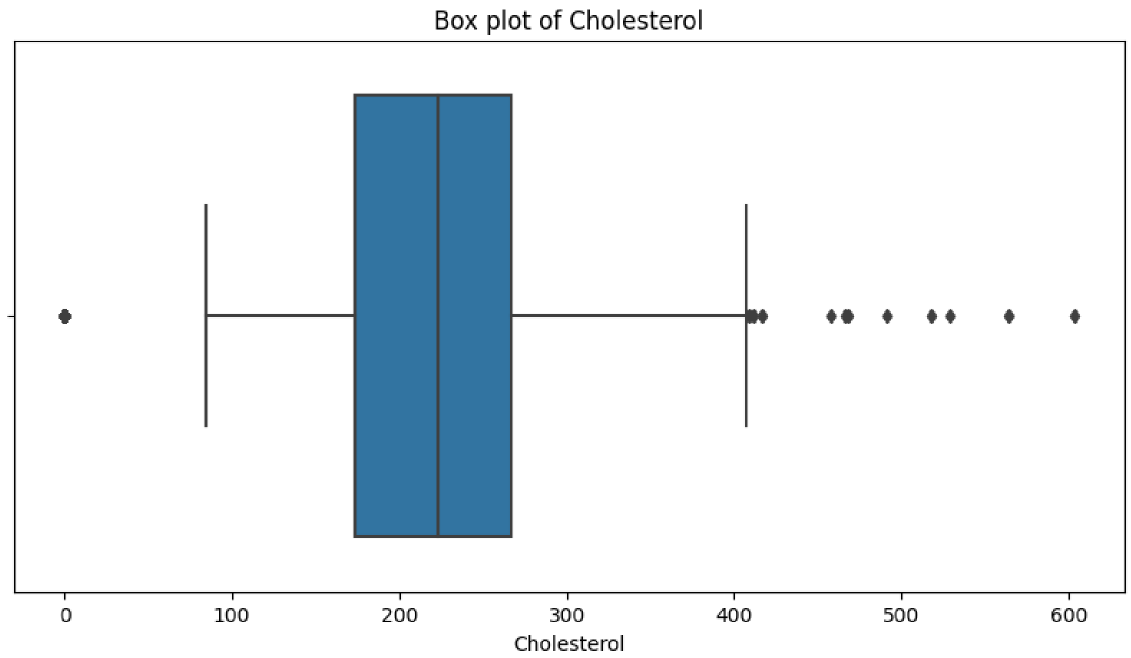
```
er='o')
  plt.title(f'plot of {column}')
  plt.show()
  from sklearn.preprocessing import LabelEncoder
  le = LabelEncoder()
data['Sex'] = le.fit_transform(data['Sex'])
data['ChestPainType'] = le.fit_transform(data['ChestPainType'])
data['RestingECG'] = le.fit_transform(data['RestingECG'])
data['ExerciseAngina'] = le.fit_transform(data['ExerciseAngina'])
data['ST_Slope'] = le.fit_transform(data['ST_Slope'])
data = data / data.max()
from sklearn.model_selection import train_test_split
X = data.drop('HeartDisease', axis=1)
y = data['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=data['FastingB
S']
)
from sklearn.ensemble import RandomForestClassifier, StackingClassi
fier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall
_score, f1_score

# Adjusting the parameters of the Random Forest
model_rf= RandomForestClassifier(n_estimators=50,  # number of esti
mators
                                 random_state=42)
model_rf.fit(X_train, y_train)

# Adjusting parameters for XGBoost
model_xgb = XGBClassifier(n_estimators=100,
                          max_depth=3,  # limiting depth of
trees
                          learning_rate=0.1,  # potentially
adding regularization via learning rate
                          subsample=0.8,  # using a subsampl
e of data to prevent overfitting
                          colsample_bytree=0.7,  # using a s
ubsample of features for each tree
                          eval_metric='logloss',
                          random_state=42)

model_xgb.fit(X_train, y_train)

# Recreate the stacked model with base models
model_stack = StackingClassifier(estimators=[('rf', model_rf),
                                             ('xgb', model_x
gb)],
                                 final_estimator=LogisticReg
ression(),
                                 stack_method='auto',
                                 n_jobs=-1)
model_stack.fit(X_train, y_train)

# Evaluate the pruned model
y_pred = model_stack.predict(X_test)
metrics = {
    'accuracy': accuracy_score(y_test, y_pred),
    'precision': precision_score(y_test, y_pred),
```

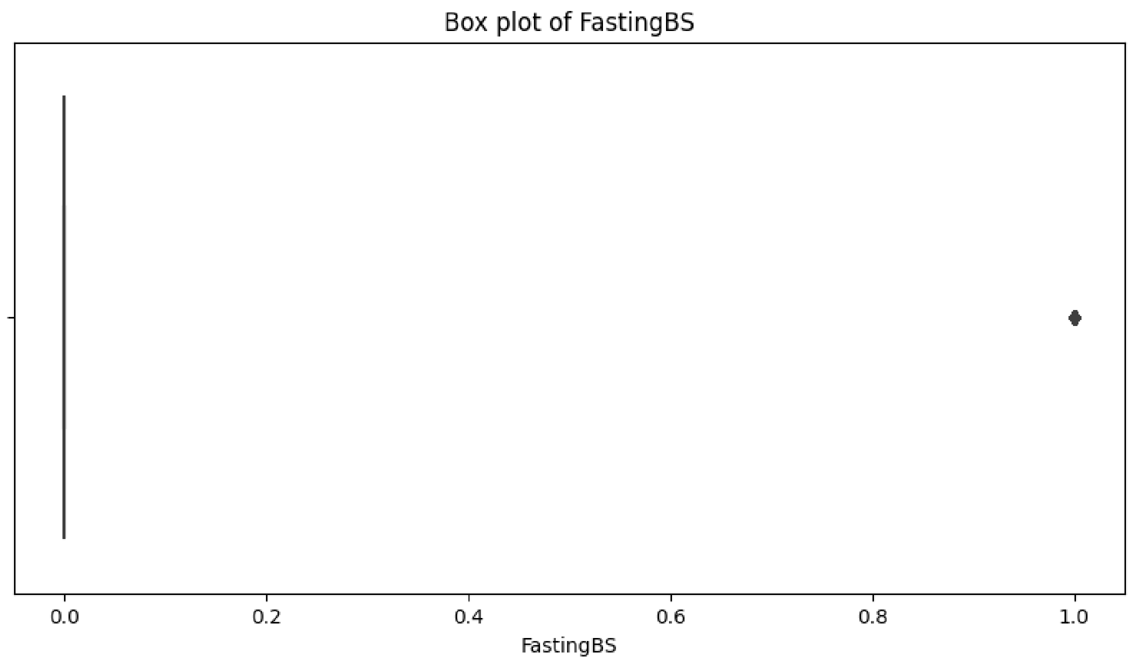
```
    'recall': recall_score(y_test, y_pred),
    'f1': f1_score(y_test, y_pred),
}

print(metrics)
importances_rf = model_rf.feature_importances_
features = X_train.columns
importances_rf_dict = dict(zip(features, importances_rf))
sorted_importances_rf = sorted(importances_rf_dict.items(), key=lam
bda x: x[1], reverse=True)

print("Feature importances from Random Forest:")
for feature, importance in sorted_importances_rf:
    print(f"{feature}: {importance}")

# Get feature importances from XGBoost
importances_xgb = model_xgb.feature_importances_
importances_xgb_dict = dict(zip(features, importances_xgb))
sorted_importances_xgb = sorted(importances_xgb_dict.items(), key=l
ambda x: x[1], reverse=True)

print("\nFeature importances from XGBoost:")
for feature, importance in sorted_importances_xgb:
    print(f"{feature}: {importance}")
```

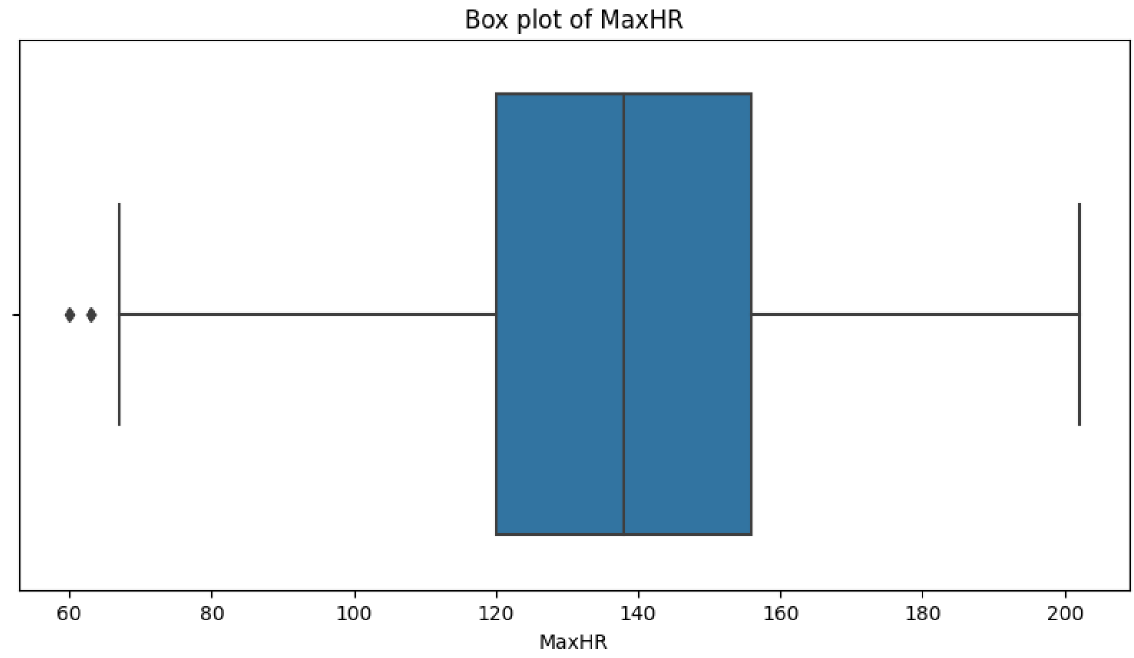
```
Rows : 918
Columns : 12
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

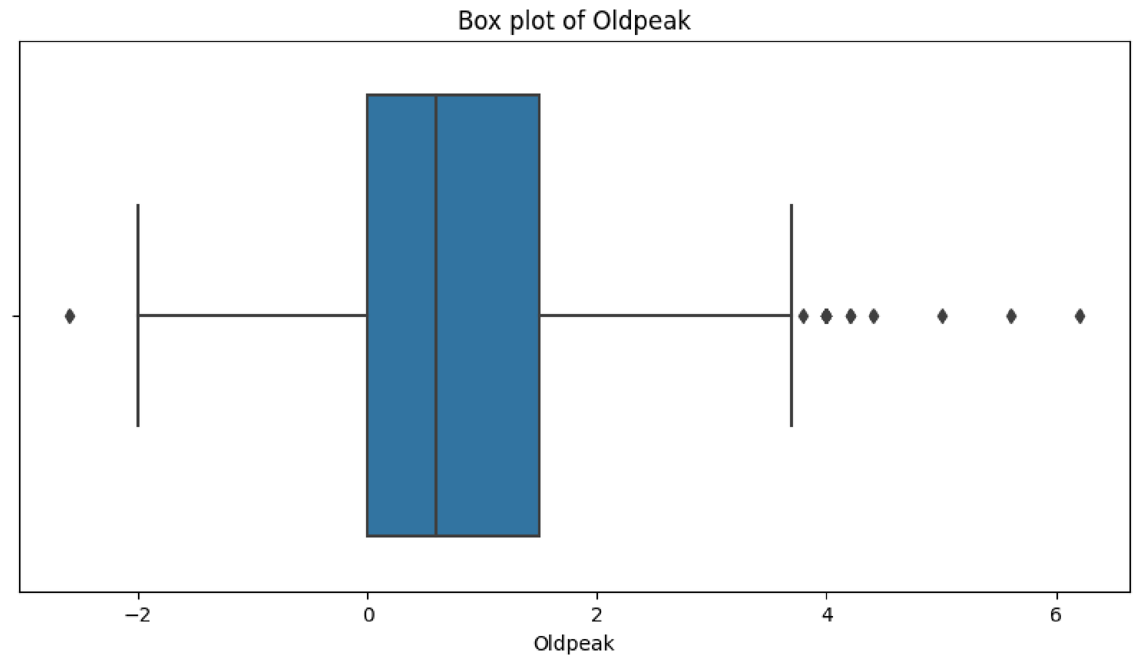



Box plot of RestingBP
0
25
50
75
100
125
150
175
200
RestingBP

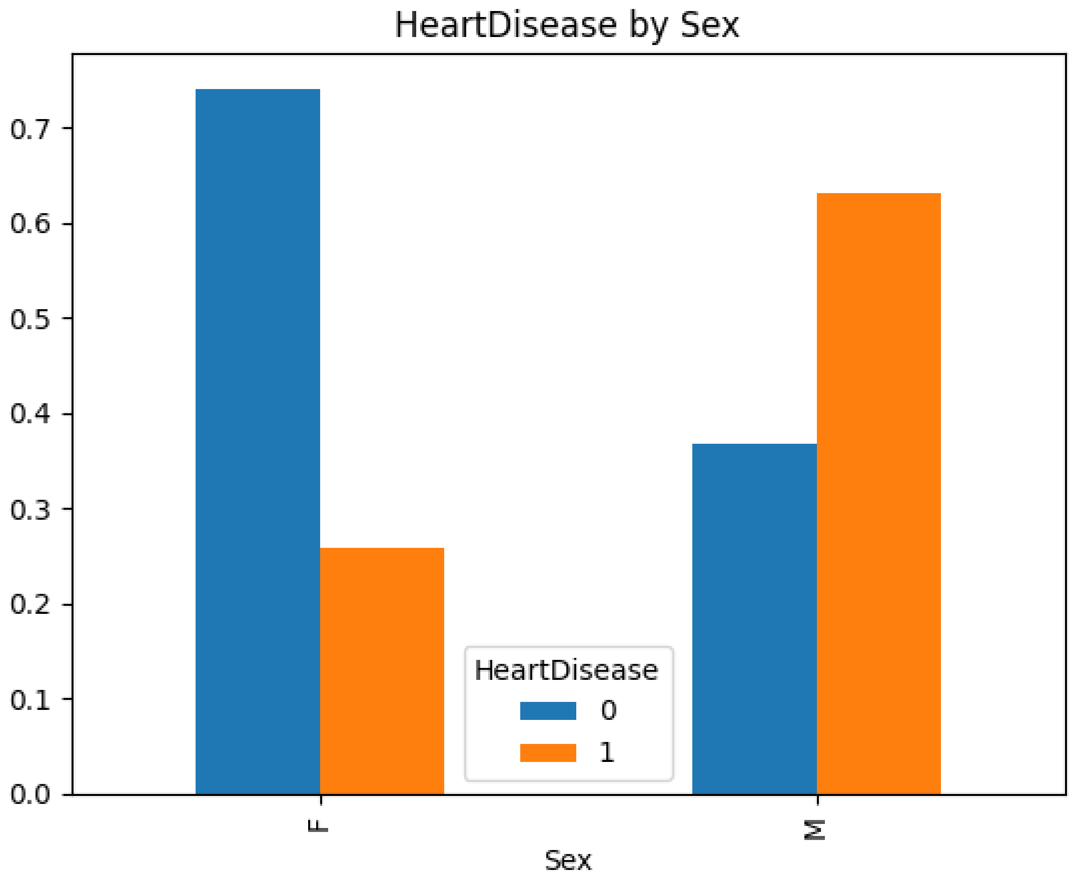

Box plot of Cholesterol
0
100
200
300
400
500
600
Cholesterol


Box plot of FastingBS
0.0
0.2
0.4
0.6
0.8
1.0
FastingBS

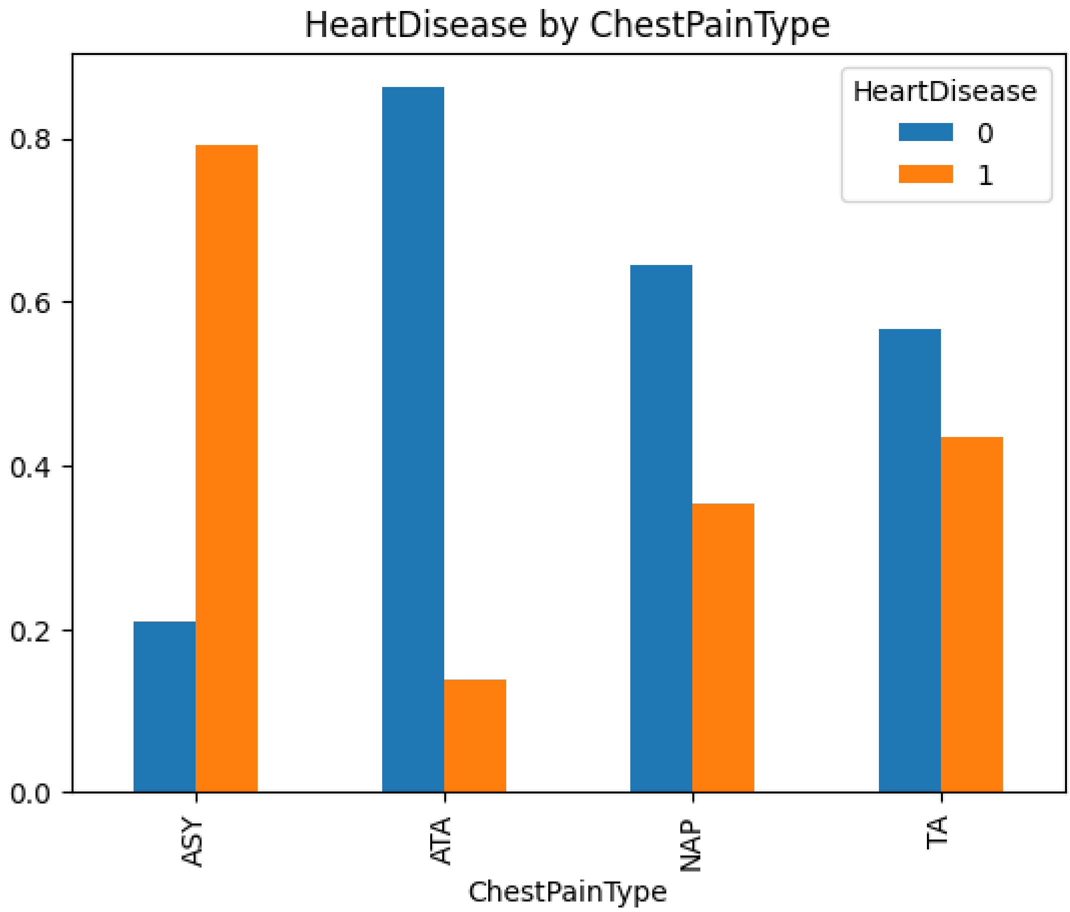




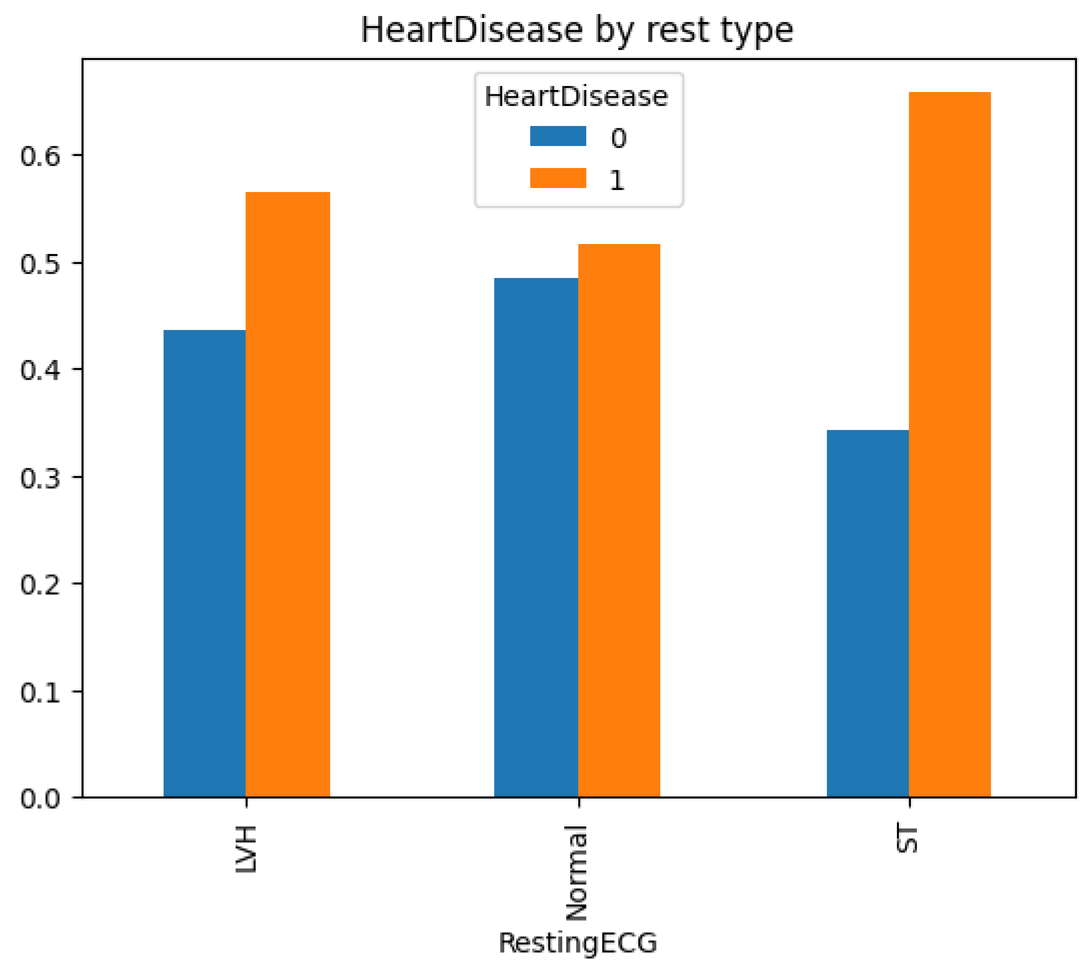

```
HeartDisease         0         1
Sex
F             0.740933  0.259067
M             0.368276  0.631724
```



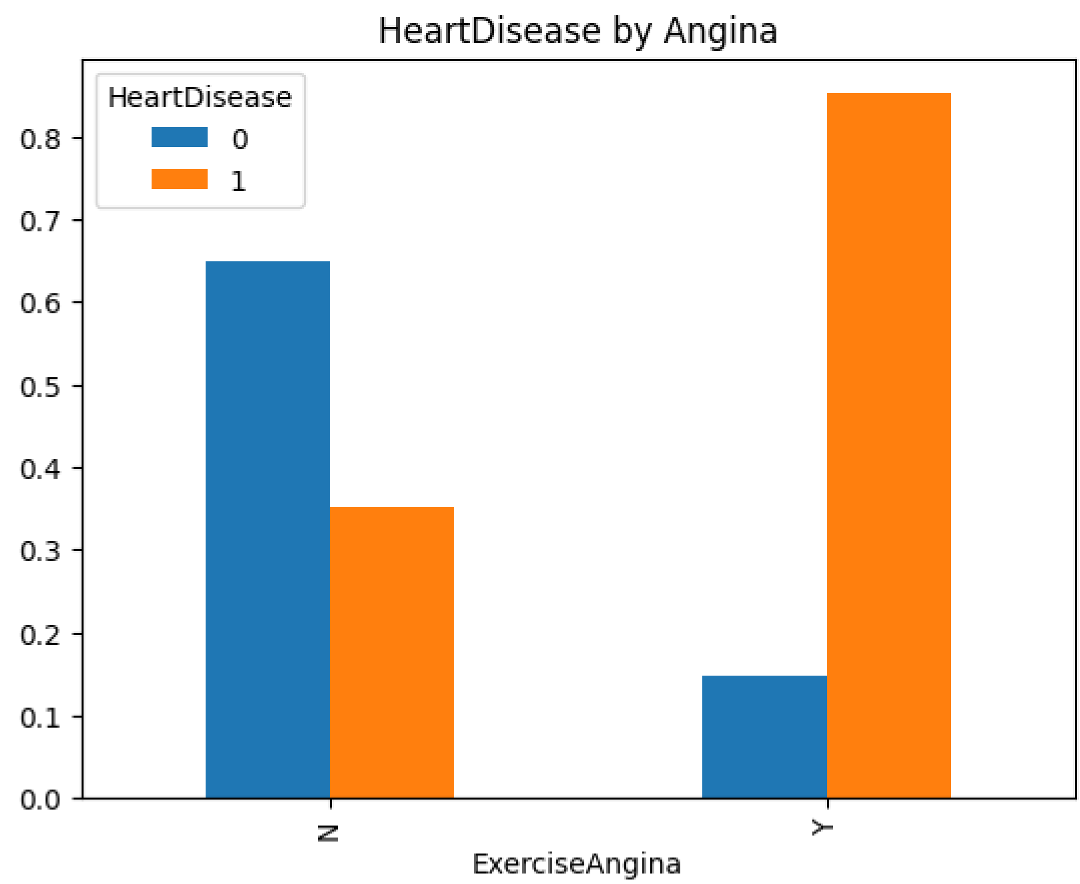

```
HeartDisease          0         1
ChestPainType
ASY            0.209677  0.790323
ATA            0.861272  0.138728
NAP            0.645320  0.354680
TA             0.565217  0.434783
```



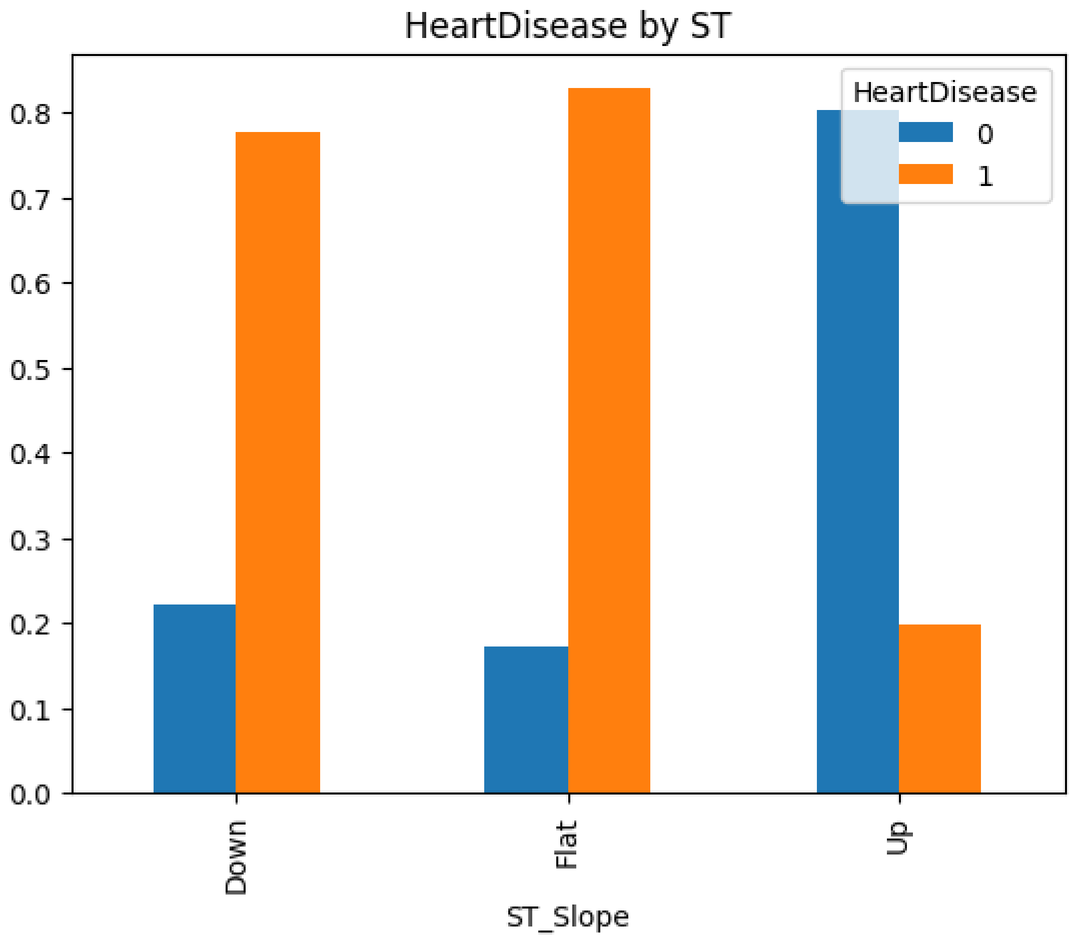

```
HeartDisease         0         1
RestingECG
LVH           0.436170  0.563830
Normal        0.483696  0.516304
ST            0.342697  0.657303
```



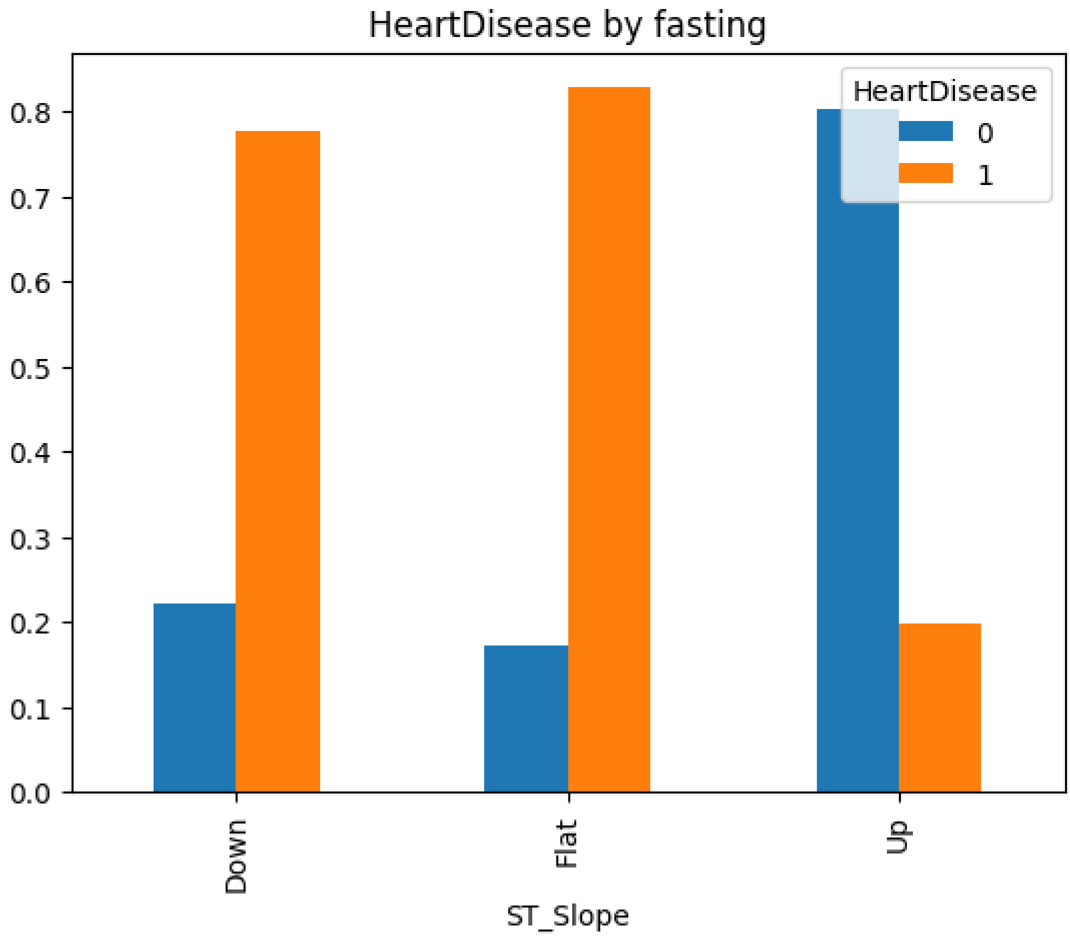

```
HeartDisease          0         1
ExerciseAngina
N              0.648995  0.351005
Y              0.148248  0.851752
```



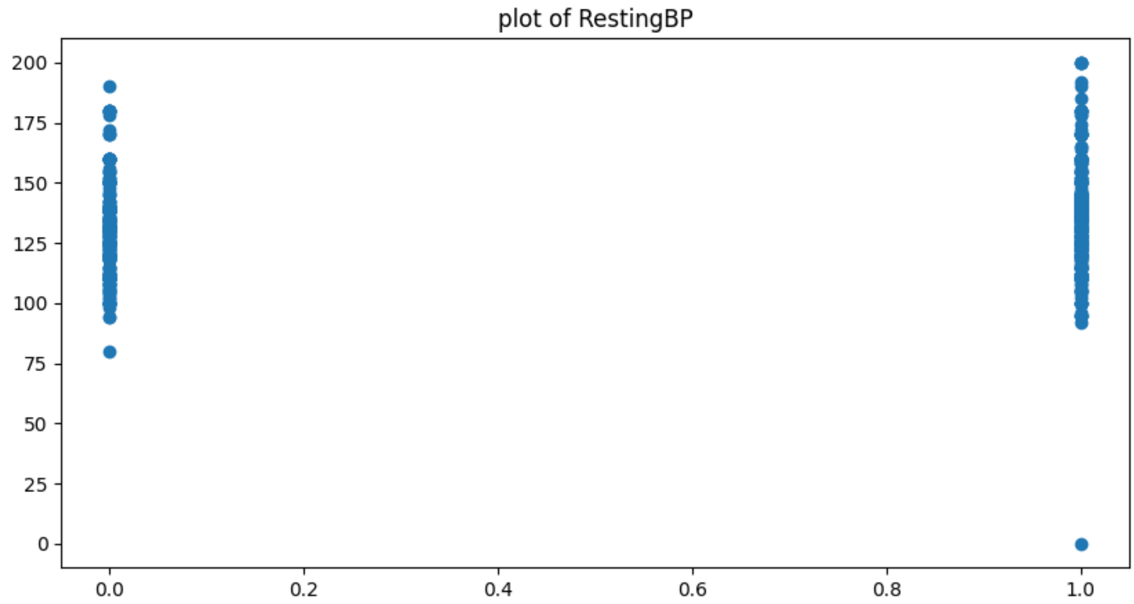
```
HeartDisease         0         1
ST_Slope
Down          0.222222  0.777778
Flat          0.171739  0.828261
Up            0.802532  0.197468
```

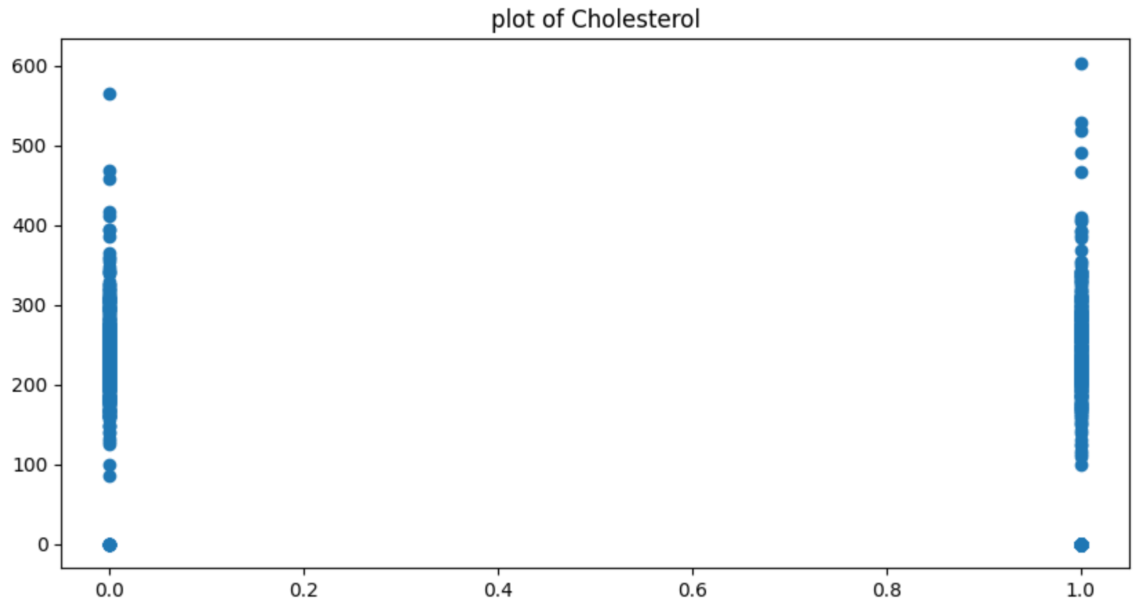


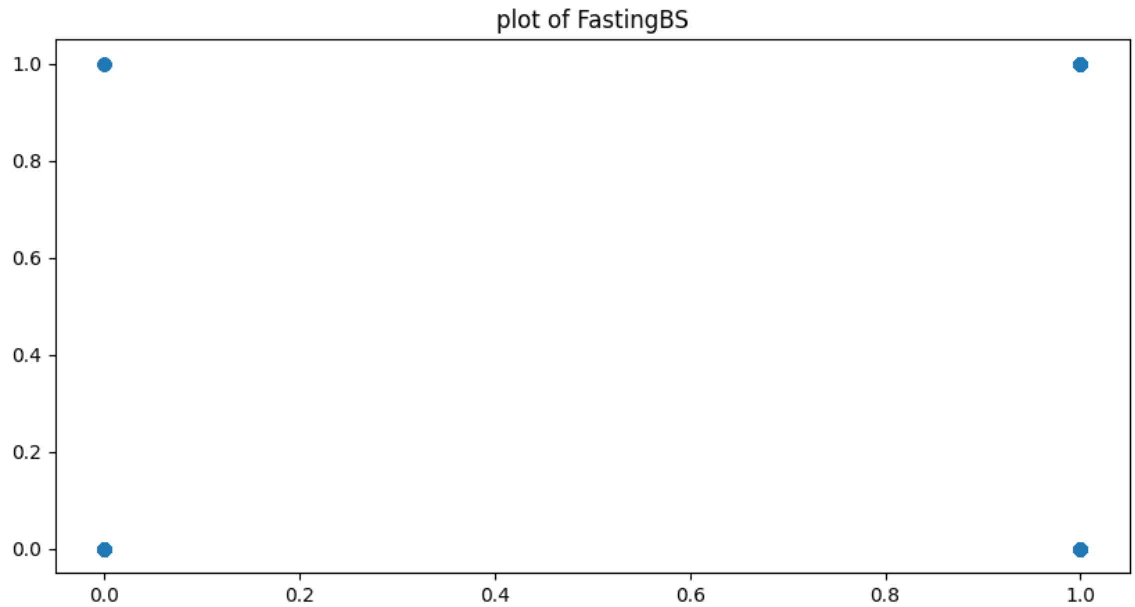

```
HeartDisease         0         1
FastingBS
0             0.519886  0.480114
1             0.205607  0.794393
```

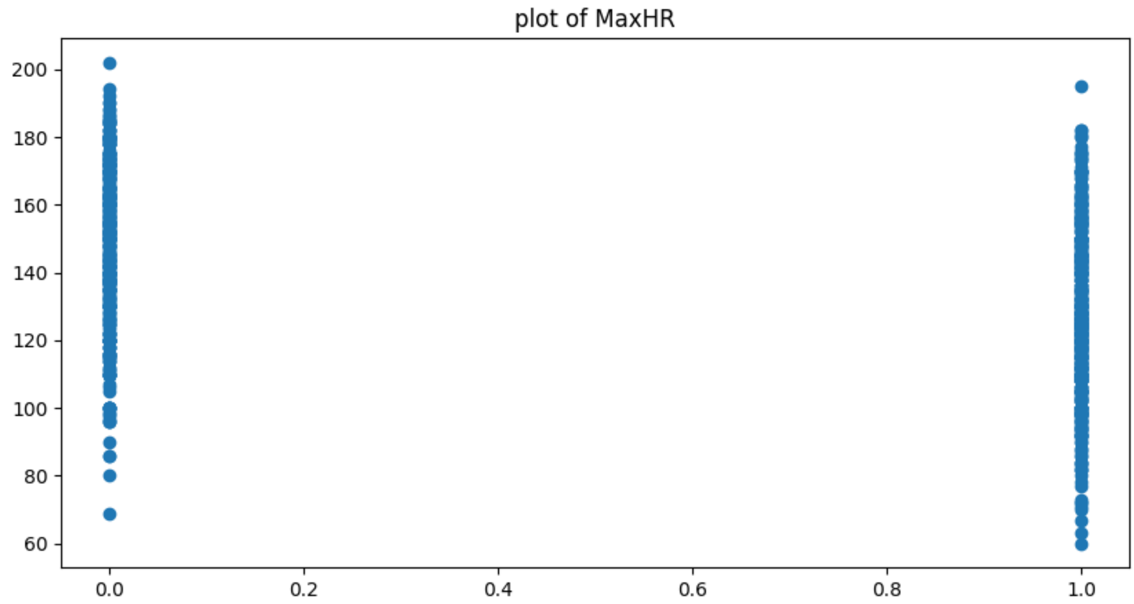

HeartDisease by fasting
HeartDisease
0
1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Down
Flat
Up
ST_Slope

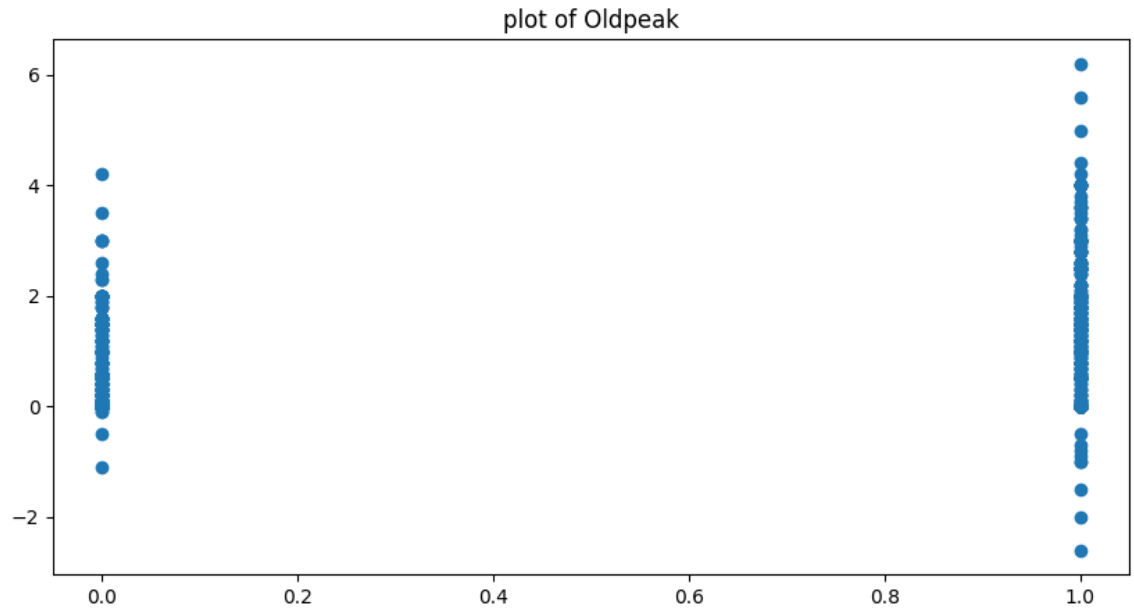

plot of Age
30
40
50
60
70
0.0
0.2
0.4
0.6
0.8
1.0


plot of RestingBP
200
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0


plot of Cholesterol
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0


plot of FastingBS
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


plot of MaxHR
200
180
160
140
120
100
80
60
0.0
0.2
0.4
0.6
0.8
1.0


plot of Oldpeak
6
4
2
0
−2
0.0
0.2
0.4
0.6
0.8
1.0

```
{'accuracy': 0.907608695652174, 'precision': 0.9107142857142857, 'r
ecall': 0.9357798165137615, 'f1': 0.923076923076923 1}
Feature importances from Random Forest:
ST_Slope: 0.21900350843265542
Cholesterol: 0.11614969583831752
ChestPainType: 0.11473519381230739
MaxHR: 0.11081418028186905
Oldpeak: 0.10948878239475818
ExerciseAngina: 0.09407294873441 56
Age: 0.08022767477386052
RestingBP: 0.07858468508468472
RestingECG: 0.02812574282011 9818
Sex: 0.027884311091323537
FastingBS: 0.020913276735688314

Feature importances from XGBoost:
ST_Slope: 0.31917333602905273
ChestPainType: 0.16173650324344635
ExerciseAngina: 0.14603181183338165
FastingBS: 0.06605455279350281
Sex: 0.05703895911574364
Oldpeak: 0.05463773384690285
Cholesterol: 0.04760991781949997
MaxHR: 0.04412927478551 8646
RestingECG: 0.04399832338094711
Age: 0.03075908310711 3838
RestingBP: 0.02883044071495533
```

Assignment. 12

Ass. 12:Write a python program to import and export data using Pandas library functions.

In [ ]:

```python
import pandas as pd
data = {'Name': ['shiva', 'parvati', 'ganesh'],
        'Age': [18, 14, 12],
        'City': ['rohtak', 'karnal', 'sonipat'],
        'Father name ':['shiv kumar','hawa singh ', 'ram ji'],
        }

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
print("\nImported DataFrame from CSV:")
print(df)
```

```
Original DataFrame:
      Name  Age     City Father name
0    shiva   18   rohtak   shiv kumar
1  parvati   14   karnal  hawa singh
2   ganesh   12  sonipat       ram ji

Imported DataFrame from CSV:
      Name  Age     City Father name
0    shiva   18   rohtak   shiv kumar
1  parvati   14   karnal  hawa singh
2   ganesh   12  sonipat       ram ji
```

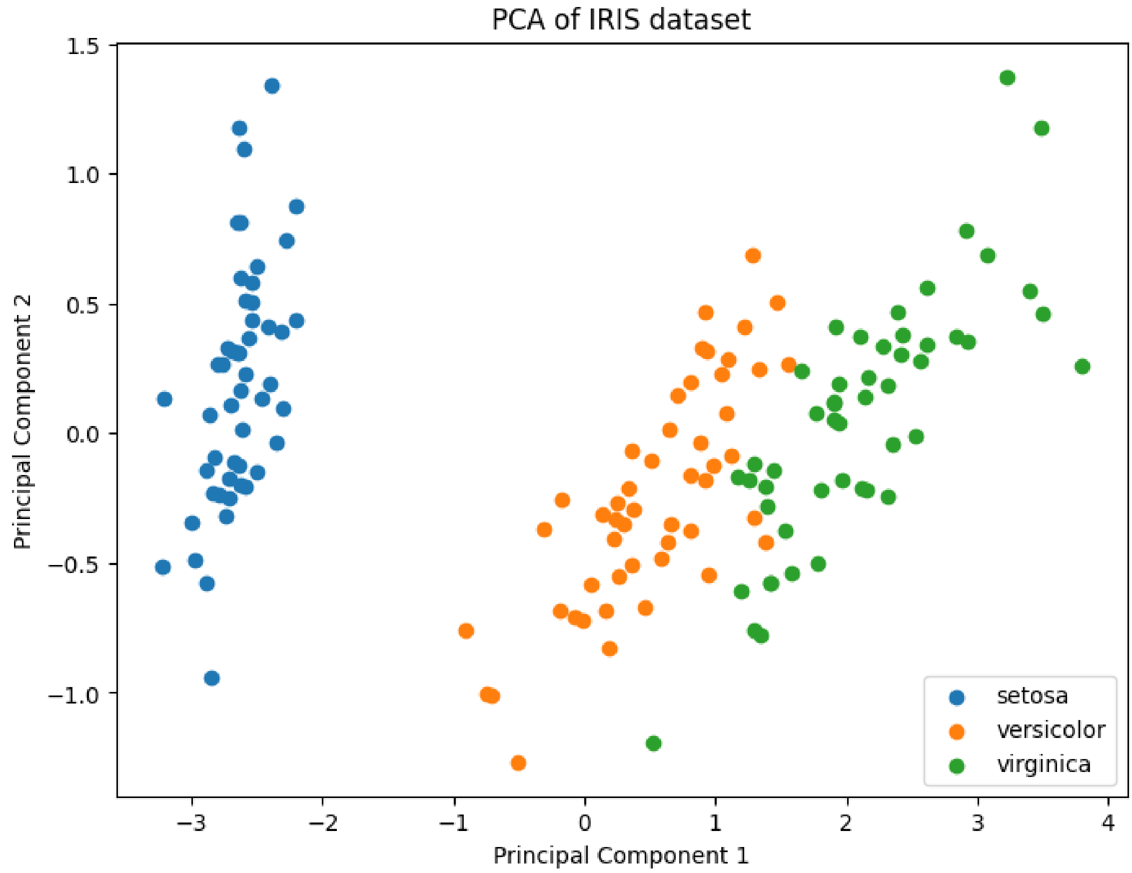
Assignment. 13

Ass. 13: Using Python implement Dimensionality reduction using Principle Component Analysis (PCA) method.

In [ ]:

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

# Load sample data (for demonstration)
data = load_iris()
X = data.data  # Features
y = data.target  # Target

# Initialize PCA and specify the number of components (dimensions)
pca = PCA(n_components=2)

# Fit the PCA model to the data
X_pca = pca.fit_transform(X)

# Percentage of variance explained by each of the selected componen
ts
explained_variance_ratio = pca.explained_variance_ratio_

print("Explained variance ratio:", explained_variance_ratio)

# Plotting the transformed data
plt.figure(figsize=(8, 6))

for i, target_name in enumerate(data.target_names):
    plt.scatter(
        X_pca[y == i, 0],
        X_pca[y == i, 1],
        label=target_name
    )

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of IRIS dataset')
plt.legend()
plt.show()
```

```
Explained variance ratio: [0.92461872 0.05306648]
```

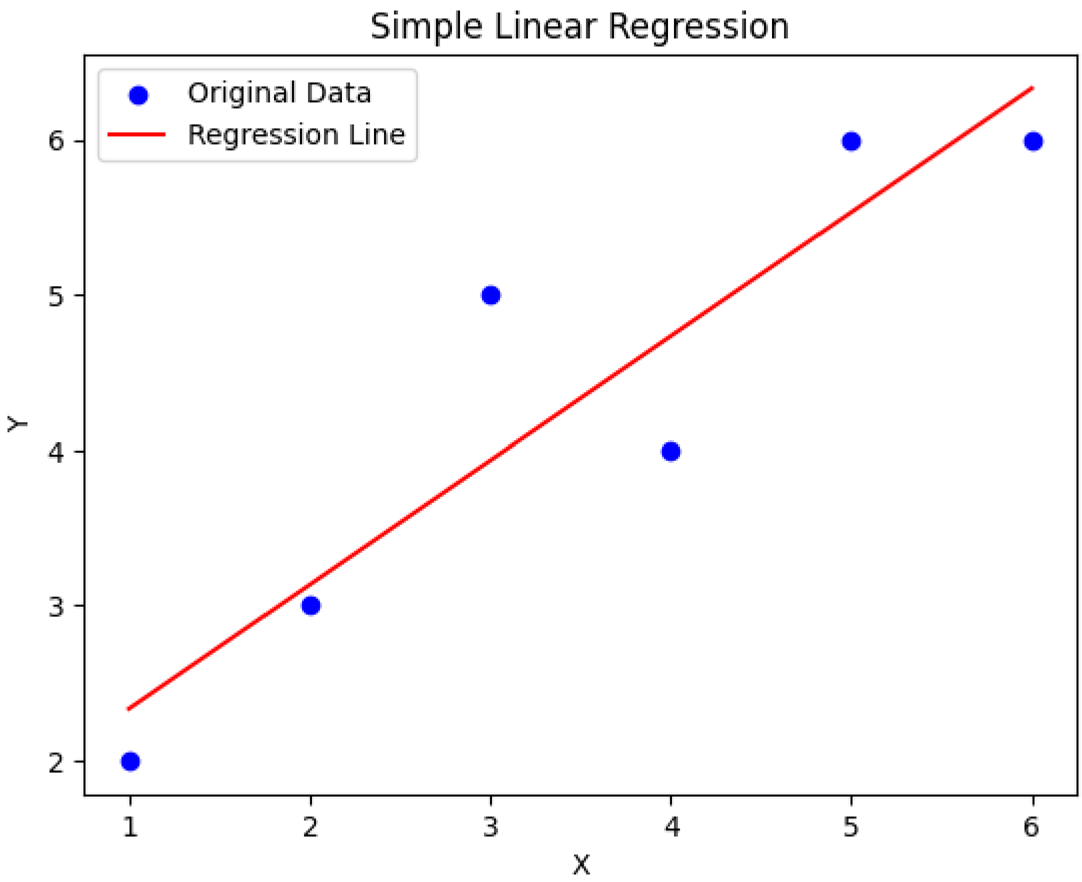


Assignment. 14

Ass. 14: Using Python implement Simple and Multiple Linear Regression Models

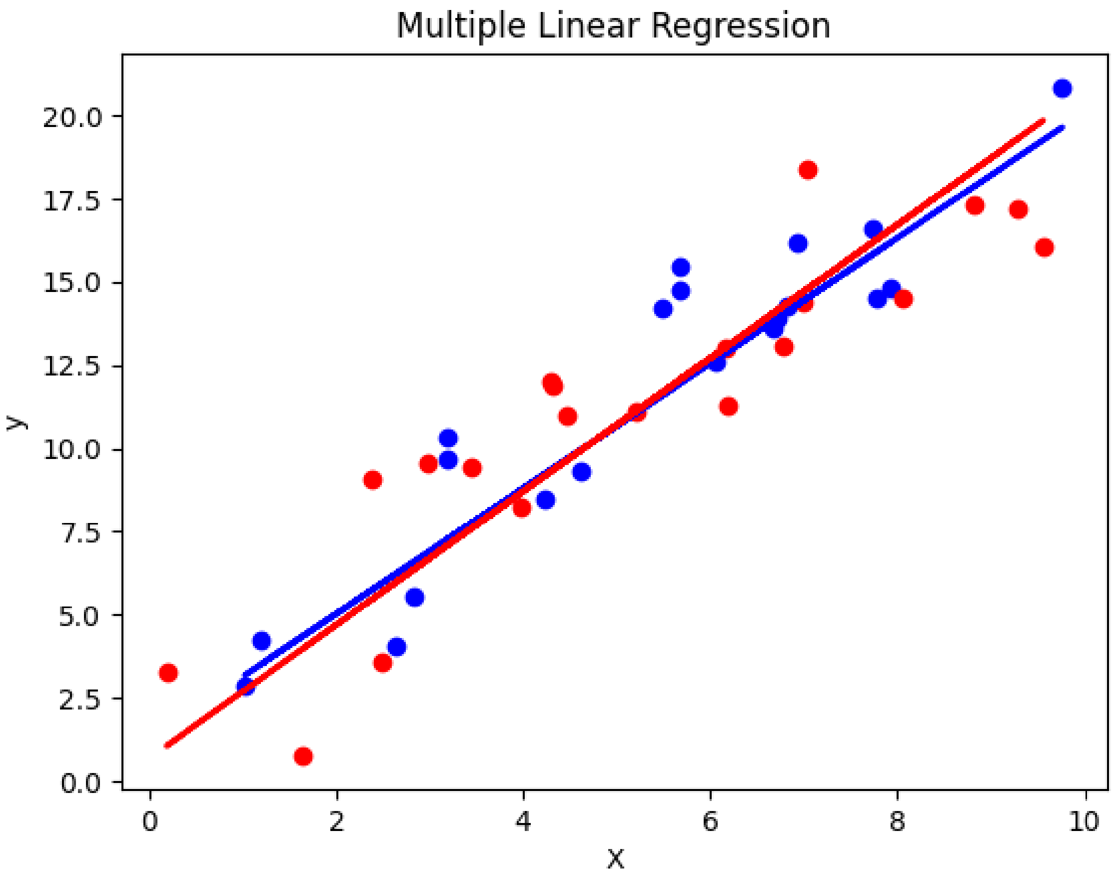
In [ ]:
```
# Question 14(Part A). Write a Python program to implement Simple L
inear Regression.

import numpy as np
import matplotlib.pyplot as plt

# Sample data
X = np.array([1, 2, 3, 4, 5, 6])
Y = np.array([2, 3, 5, 4, 6, 6])

# Calculate the mean of X and Y
mean_X = np.mean(X)
mean_Y = np.mean(Y)

# Calculate the total number of data points
n = len(X)

# Calculate the slope (m) and the y-intercept (b) using the least s
quares method
numerator = np.sum((X - mean_X) * (Y - mean_Y))
denominator = np.sum((X - mean_X) ** 2)
m = numerator / denominator
b = mean_Y - m * mean_X

# Print the slope and y-intercept
print("Slope (m):", m)
print("Y-Intercept (b):", b)

# Predict the values of Y based on the linear regression model
Y_pred = m * X + b

# Plot the original data and the regression line
plt.scatter(X, Y, label='Original Data', color='blue')
plt.plot(X, Y_pred, label='Regression Line', color='red')
plt.legend()
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Linear Regression')
plt.show()
```

```
Slope (m): 0.8
Y-Intercept (b): 1.5333333333333328
```



In [ ]:
```
# Multiple Linear Regression.
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score

np.random.seed(0)
X1=np.random.rand(100,1)*10
X2=np.random.rand(100,1)*10
y1=2 * X1 + 1 + np.random.randn(100,1)*2
y2=2 * X2 + 1 + np.random.randn(100,1)*2

X1_train,X1_test,y1_train,y1_test = train_test_split(X1,y1,test_siz
e=0.2,random_state=42)
X2_train,X2_test,y2_train,y2_test = train_test_split(X2,y2,test_siz
e=0.2,random_state=42)

model1 = LinearRegression()
model2 = LinearRegression()

model1.fit(X1_train,y1_train)
model2.fit(X2_train,y2_train)

y_pred1=model1.predict(X1_test)
y_pred2=model2.predict(X2_test)

coefficient1 = model1.coef_
coefficient2 = model2.coef_
intercept1 = model1.intercept_
intercept2 = model2.intercept_

mse1 = mean_squared_error(y1_test,y_pred1)
mse2 = mean_squared_error(y2_test,y_pred2)
r1 = r2_score(y1_test,y_pred1)
r2 = r2_score(y1_test,y_pred2)

print("Coefficients: ",coefficient1)
print("Coefficients: ",coefficient2)
print("Intercept: ",intercept1)
print("Intercept: ",intercept2)
print("Mean_Squared_Error: ",mse1)
print("Mean_Squared_Error: ",mse2)
print("R-squared Score: ",r1)
print("R-squared Score: ",r2)

plt.scatter(X1_test,y1_test,color="blue")
plt.scatter(X2_test,y2_test,color="red")
plt.plot(X1_test,y_pred1,color="blue",linewidth=2)
plt.plot(X2_test,y_pred2,color="red",linewidth=2)
plt.xlabel("X")
plt.ylabel("y")
plt.title("Multiple Linear Regression")
plt.show()
```

```
Coefficients:  [[1.88199746]]
Coefficients:  [[2.00317266]]
Intercept:  [1.26395959]
Intercept:  [0.68918563]
Mean_Squared_Error:  2.9573753913252188
Mean_Squared_Error:  5.15919190916186
R-squared Score:  0.8663947329351374
R-squared Score:  -1.5452856283756056
```



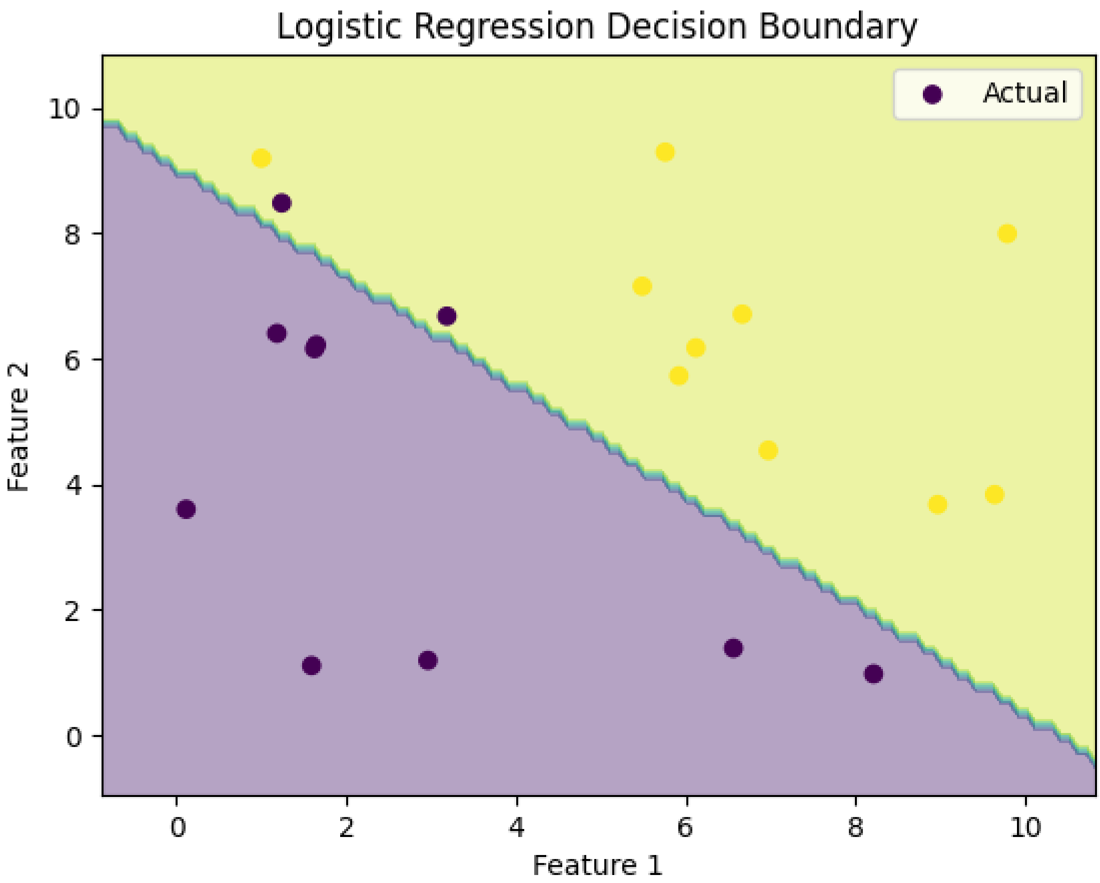
Assignment. 15

Ass. 15: Using Python develop Logistic Regression Model for a given dataset.

In [ ]:

```
# Question 15. Logistic Regression
import numpy as np
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

np.random.seed(0)
X = np.random.rand(100, 2) * 10

y  = (X[:, 0] + X[:, 1] > 10).astype(int)

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

model = LogisticRegression()

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)

print("Accuracy: ", accuracy)
print("Cofusion Matrix: ")
print(conf_matrix)

if X_train.shape[1] == 2:
  X_min, X_max = X[:,0].min() - 1,X[:, 0].max() + 1
  y_min, y_max = X[:,1].min() - 1,X[:, 1].max() + 1
  XX, yy = np.meshgrid(np.arange(X_min, X_max, 0.1), np.arange(y_mi
n, y_max, 0.2))

  Z = model.predict(np.c_[XX.ravel(), yy.ravel()])
  Z = Z.reshape(XX.shape)
  plt.contourf(XX, yy, Z, alpha=0.4)
  plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, marker='o', lab
el='Actual')
  plt.xlabel("Feature 1")
  plt.ylabel("Feature 2")
  plt.title("Logistic Regression Decision Boundary")
  plt.legend()
  plt.show()
```

```
Accuracy:  0.9
Cofusion Matrix:
[[ 8  2]
 [ 0 10]]
```



Assignment. 16

Ass. 16: Using Python develop Decision Tree Classification model for a given dataset and use it to classify a new sample.

In [ ]:

```
# Question 16. Write a python program to implement Decision tree us
ing sklearn and its parameter tuning
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
=0.2, random_state=42)

# Create a Decision Tree Classifier
dt_classifier = DecisionTreeClassifier()

# Define the hyperparameters to tune
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create a GridSearchCV object
grid_search = GridSearchCV(dt_classifier, param_grid, cv=5)

# Fit the model to the training data
grid_search.fit(X_train, y_train)

# Print the best parameters found by GridSearchCV
print("Best Parameters:", grid_search.best_params_)

# Make predictions on the test set
y_pred = grid_search.predict(X_test)

# Calculate and print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Best Parameters: {'criterion': 'entropy', 'max_depth': 5, 'min_samp
les_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'}
Accuracy: 0.9666666666666667
```