1 Experiment No. 1

1.1 1. Preliminary analysis:

a. Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc.

b. Based on these findings, remove duplicates (if any) and treat missing values using an appropriate strategy

Variables explanations:

age (Age in years)

sex : (1 = male, 0 = female)

cp (Chest Pain Type): [0: asymptomatic, 1: atypical angina, 2: non-anginal pain, 3: typical angina]

trestbps (Resting Blood Pressure in mm/hg) chol

(Serum Cholesterol in mg/dl)

fps (Fasting Blood Sugar > 120 mg/dl): [0 = no, 1 = yes]

restecg (Resting ECG): [0: showing probable or definite left ventricular hypertrophy by Estes' criteria, 1: normal, 2: having ST-T wave abnormality] thalach (maximum heart rate achieved)

exang (Exercise Induced Angina): [1 = yes, 0 = no] oldpeak (ST

depression induced by exercise relative to rest)

slope (the slope of the peak exercise ST segment): [0: downsloping; 1: flat; 2: upsloping] ca

[number of major vessels (0–3)]

thal : [1 = normal, 2 = fixed defect, 3 = reversible defect] target:

[0 = disease, 1 = no disease]

[1]:
```
import numpy as np
import pandas as pd
df=pd.read_excel("1645792390_cep1_dataset.xlsx")#importing dataset
```

[2]:
```
df.head()
```

[2] :

	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

[3] :
```
df.describe() # shows statistical summary of data
```

[3]:

	age	sex	cp	trestbps	chol	fbs	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	
std	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	
75%	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	

	thal	target
count	303.000000	303.000000
mean	2.313531	0.544554
std	0.612277	0.498835
min	0.000000	0.000000
25%	2.000000	0.000000
50%	2.000000	1.000000
75%	3.000000	1.000000
max	3.000000	1.000000

[4] :
```
df.isnull().sum() # there are no missing values in the dataset.
```

[4] :
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[5] :
```
df.isna()
```

[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	\
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
..	...	...	...	...	...	..	..	...	...	
298	False	False	False	False	False	False	False	False	False	
299	False	False	False	False	False	False	False	False	False	
300	False	False	False	False	False	False	False	False	False	
301	False	False	False	False	False	False	False	False	False	
302	False	False	False	False	False	False	False	False	False	

	oldpeak	slope	ca	thal	target
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
..	...	...	...	...	...
298	False	False	False	False	False
299	False	False	False	False	False
300	False	False	False	False	False
301	False	False	False	False	False
302	False	False	False	False	False

[303 rows x 14 columns]

[6] :
```
df.info()
```

```
<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
 7    thalach     303 non-null    int64
 8    exang       303 non-null    int64
 9    oldpeak     303 non-null    float64
 10   slope       303 non-null    int64
 11   ca          303 non-null    int64
 12   thal        303 non-null    int64
 13   target      303 non-null    int64
dtypes: float64(1), int64(13) memory usage:
33.3 KB
```

[7] :

```
import pandas as pd
df=pd.read_excel("1645792390_cep1_dataset.xlsx") # Find
duplicate rows based on all columns duplicates =
df.duplicated()
```

2 Experiment No. 2

2. Prepare a report about the data explaining the distribution of the disease and the related factors using the steps listed below:

a. Get a preliminary statistical summary of the data and explore the measures of central ten- dencies and spread of the data

b. Identify the data variables which are categorical and describe and explore these variables using the appropriate tools, such as count plot

c. Study the occurrence of CVD across the Age category

d. Study the composition of all patients with respect to the Sex category

e. Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient

f. Describe the relationship between cholesterol levels and a target variable

g. State what relationship exists between peak exercising and the occurrence of a heart attack

h. Check if thalassemia is a major cause of CVD

i. List how the other factors determine the occurrence of CVD

j. Use a pair plot to understand the relationship between all the given variables

[8] :
```
df.describe() ##preliminary statistical summary
```

[8]:

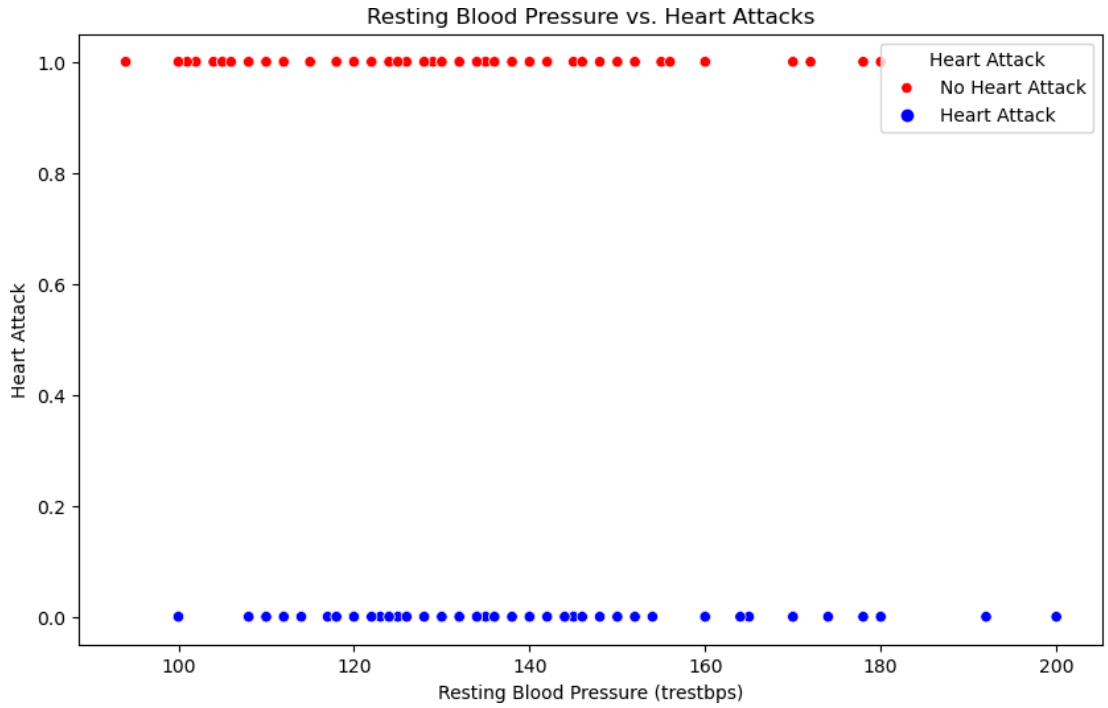
	age	sex	cp	trestbps	chol	fbs	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	
std	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	
75%	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	

	thal	target
count	303.000000	303.000000
mean	2.313531	0.544554
std	0.612277	0.498835
min	0.000000	0.000000
25%	2.000000	0.000000
50%	2.000000	1.000000
75%	3.000000	1.000000
max	3.000000	1.000000

[9]:
```
import pandas as pd import
seaborn as sns
import matplotlib.pyplot as plt

# Read the data from the dataset (replace "data.csv" with your file's name)
data = pd.read_excel("1645792390_cep1_dataset.xlsx")

# Create a scatter plot to visualize the relationship between resting blood␣
 ↪pressure and heart attacks
plt.figure(figsize=(10, 6))
sns.scatterplot(x="trestbps", y="target", data=data, hue="target", palette={0:␣
 ↪"blue", 1: "red"})
```

```
plt.title("Resting Blood Pressure vs. Heart Attacks") plt.xlabel("Resting
Blood Pressure (trestbps)") plt.ylabel("Heart Attack")
plt.legend(title="Heart Attack", loc="upper right", labels=["No Heart Attack",␣
  ↪"Heart Attack"])
plt.show()

# Perform a statistical analysis (t-test) to compare resting blood pressure␣
  ↪between heart attack and no heart attack groups
from scipy.stats import ttest_ind

heart_attack_bp = data[data["target"] == 1]["trestbps"]
no_heart_attack_bp = data[data["target"] == 0]["trestbps"]

t_stat, p_value = ttest_ind(heart_attack_bp, no_heart_attack_bp) print("Statistical Analysis:")
print("t-statistic:", t_stat) print("p-
value:", p_value)
```



Statistical Analysis:
t-statistic: -2.5412927171039
p-value: 0.011546059200233312

A t-test is a statistical test that is used to compare the means of two groups. It is often used

in hypothesis testing to determine whether a process or treatment actually has an effect on the population of interest, or whether two groups are different from one another. The null hypothesis (H0) is that the true difference between these group means is zero. The alternate hypothesis (Ha) is that the true difference is different from zero.

A negative t-value indicates a reversal in the directionality of the effect, which has no bearing on the significance of the difference between groups.

The p value, or probability value, tells you how likely it is that your data could have occurred under the null hypothesis. It does this by calculating the likelihood of your test statistic, which is the number calculated by a statistical test using your data.

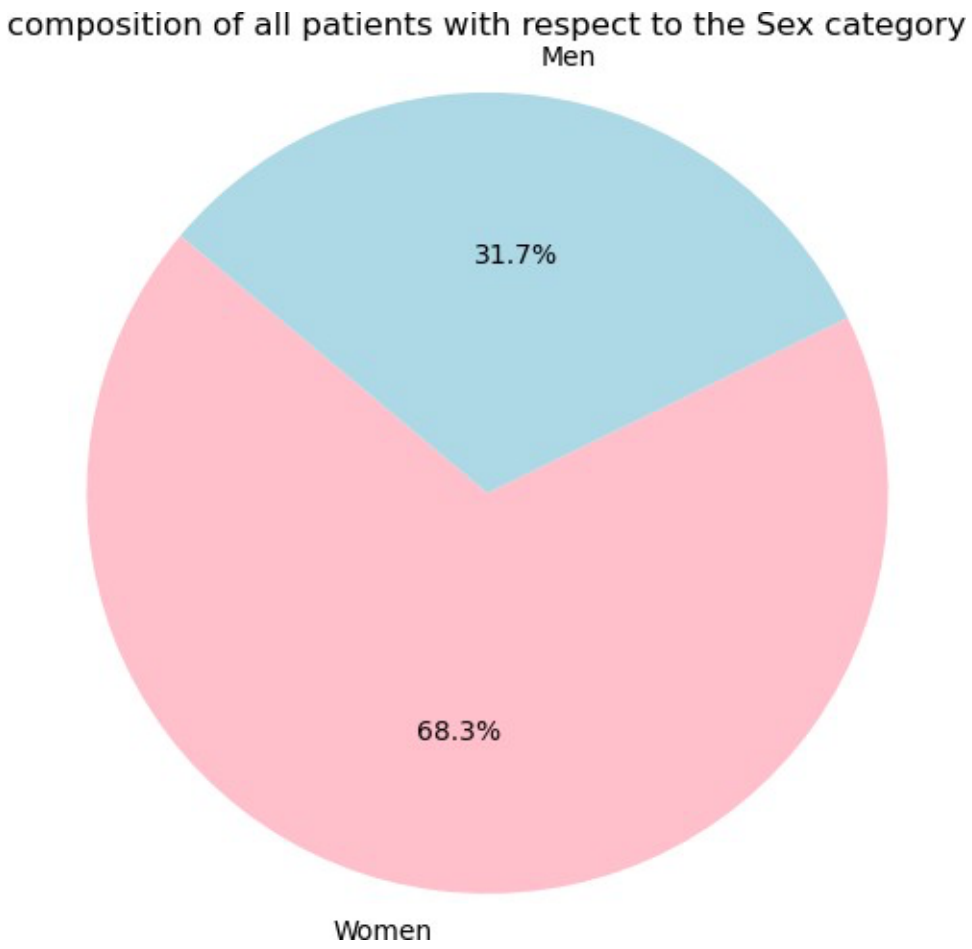
P-value

if P-value > 0.05 then The result is not statistically significant and hence don't reject the null hypothesis.

if P-value < 0.05 then The result is statistically significant. Generally, reject the null hypothesis in favour of the alternative hypothesis.

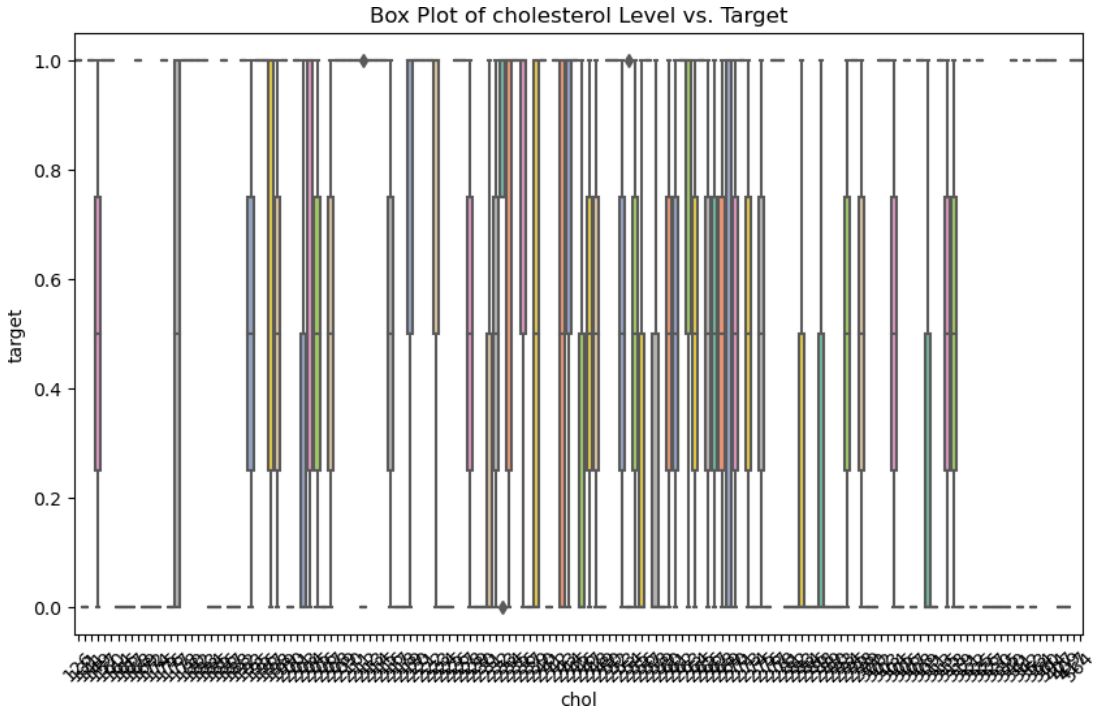
if P-value < 0.01 then The result is highly statistically significant, and thus rejects the null hy- pothesis in favour of the alternative hypothesis.

[10]:
```
##d. Study the composition of all patients with respect to the Sex category
import pandas as pd import
seaborn as sns
import matplotlib.pyplot as plt

# Create a pie chart to show composition by Sex sex_counts =
df["sex"].value_counts() plt.figure(figsize=(6, 6))
lll=['Women','Men']
plt.pie(sex_counts, labels=lll, autopct="%1.1f%%", startangle=140,␣
  ↪colors=["pink",  "lightblue"])
plt.title("composition of all patients with respect to the Sex category")
plt.axis("equal")
plt.show()
```



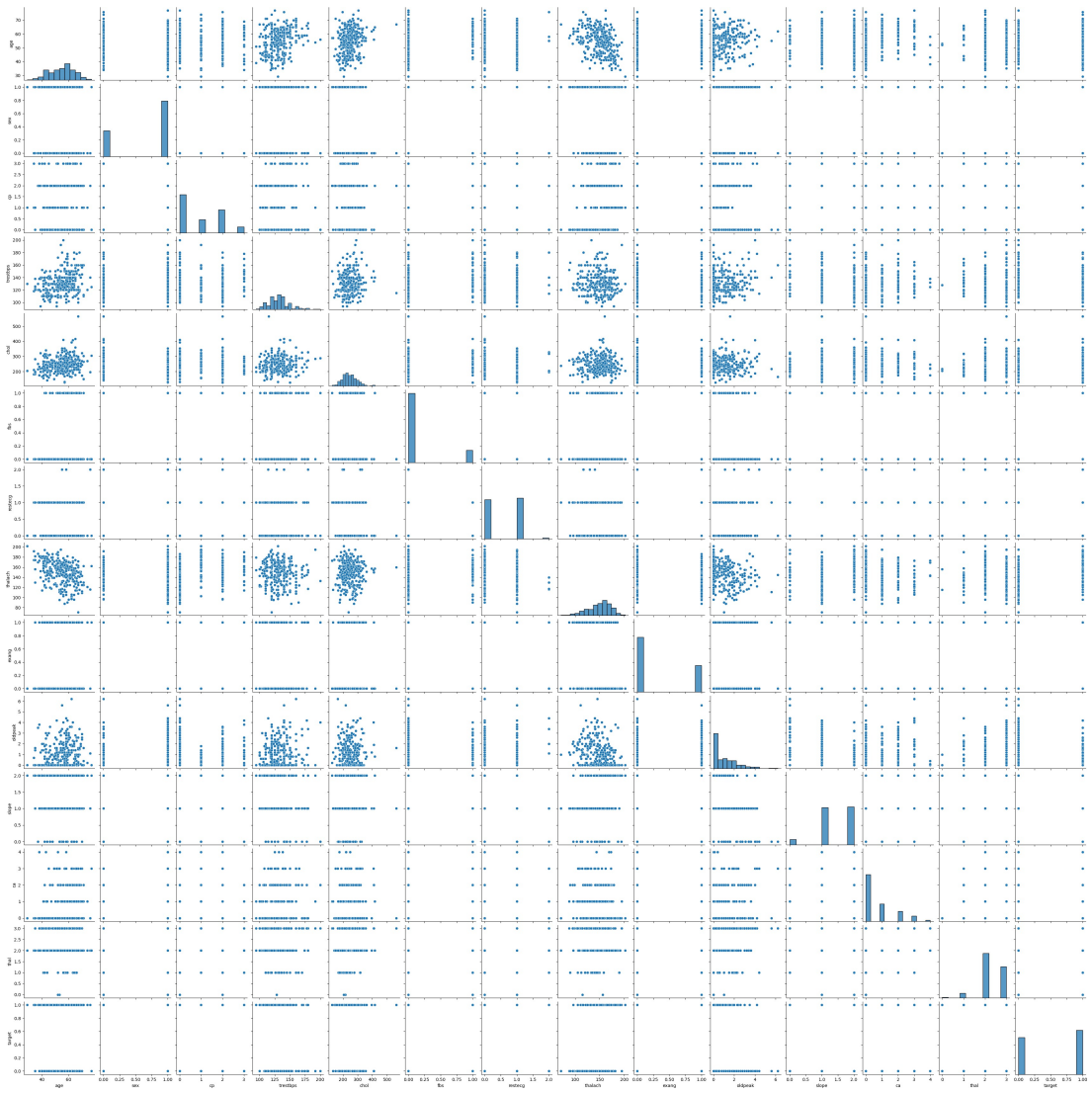
[11]:

```
##f. Describe the relationship between cholesterol levels and a target variable
import pandas as pd import
seaborn as sns
import matplotlib.pyplot as plt #
Creating a box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='chol', y='target', data=df, palette='Set2') plt.title('Box Plot of
cholesterol Level vs. Target') plt.xticks(rotation=45)
plt.show()
```



[12]:
```
##J. Use a pair plot to understand the relationship between all the given␣
 ↪variables
import seaborn as sns
sns.pairplot(df)
```

[12] : <seaborn.axisgrid.PairGrid at 0x1e167168670>



3 Logistic Regression

[13] :
```
df["target"].value_counts()
```

[13]: 1 165
0 138
Name: target, dtype: int64

[14] :
```
X=df.drop("target",axis=1) y =
df["target"]
```

[15] :
```
import matplotlib.pyplot as plt from
matplotlib import rcParams from
matplotlib.cm import rainbow
%matplotlib inline
import seaborn as sns

from sklearn.model_selection import train_test_split X_train,X_test,y_train,y_test
= train_test_split(X,y,test_size=0.

  s20,stratify=y,random_state=7)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression() lr.fit(X_train,
y_train)
```

```
C:\Users\Student\anaconda3\lib\site- packages\sklearn\linear_model\_logistic.py:814:
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 shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options: https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[15] : LogisticRegression()

[16] :
```
pred = lr.predict(X_test)
from sklearn.metrics import accuracy_score, confusion_matrix,␣

  sclassification_report
```

[17] :
```
from sklearn.metrics import accuracy_score, confusion_matrix,␣

  sclassification_report
# Accuracy on Test data
accuracy_score(y_test, pred)
```

[17]: 0.8032786885245902

[18] :
```
# Accuracy on Train data
accuracy_score(y_train, lr.predict(X_train))
```

[18]: 0.8512396694214877

4 Linear Regression

[19] :
```
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 mean_squared_error, r2_score
```

[20] :
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[21] :
```
X=df.drop("target",axis=1) y =
df["target"]
```

[22] :
```
model = LinearRegression()
```

[23] :
```
model.fit(X_train, y_train)
```

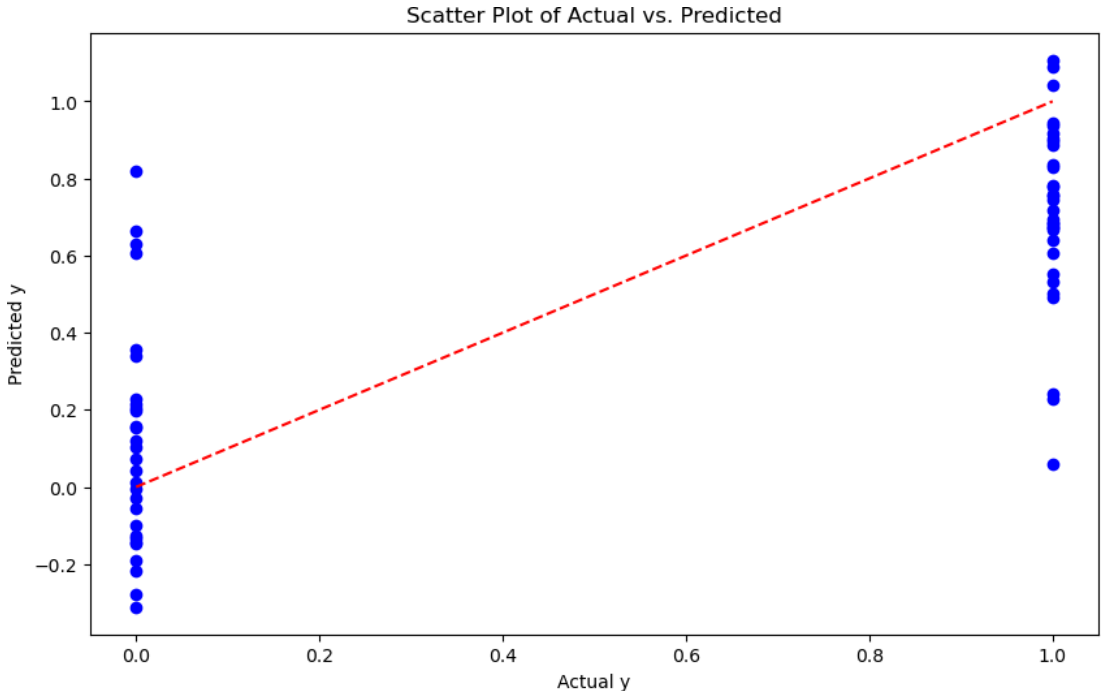
[23] : LinearRegression()

[24] :
```
y_pred = model.predict(X_test)
```

[25] :
```
mse=mean_squared_error(y_test, y_pred) r2 =
r2_score(y_test, y_pred)

#model performance metrics print("Mean
Squared Error:", mse) print("R-squared:",
r2)
```

Mean Squared Error: 0.11627071992880018 R-squared: 0.5337894947682484

[26] :
```
plt.figure(figsize=(10, 6)) plt.scatter(y_test, y_pred,
color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',␣
 ↪linestyle='--')
plt.xlabel('Actual y')
plt.ylabel('Predicted y')
plt.title('Scatter Plot of Actual vs. Predicted') plt.show()
```



[27] :
```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
␣classification_report
# Accuracy on Test data
accuracy_score(y_test, pred)
```

[27]: 0.4918032786885246

[28] :
```
# Accuracy on Train data
accuracy_score(y_train, lr.predict(X_train))
```

[28]: 0.8305785123966942

in above dataset logistic regression has high accuracy than linear regression

when we compare logistic regression with random forest, random forest having more accuracy.

Linear Regression and Logistic Regression are simpler and offer interpretability, but they are limited in handling complex data.

Random Forest is more flexible and can capture complex relationships but may be less interpretable due to its ensemble nature.

The choice depends on the nature of data, the task at hand, and your goals for accuracy, inter- pretability, and handling complexity.