#unzip dataset

import zipfile zip\_ref =zipfile.ZipFile("/content/archive (8).zip",'r') zip\_ref.extractall('/content') zip\_ref.close()

import pandas as pd

import train\_test\_split from sklearn.linear\_model

import LogisticRegression from sklearn.metrics import accuracy\_score, classification\_report

df=pd.read\_csv("heart.csv")

df

**age**

**sex**

**cp**

**trestbps**

**chol**

**fbs**

**restecg**

**thalach**

**exang**

**oldpeak**

**slope**

**ca**

**thal**

**target**

1. 52 1 0 125 212 0 1 168 0 1.0 2 2 3 0
2. 53 1 0 140 203 1 0 155 1 3.1 0 0 3 0
3. 70 1 0 145 174 0 1 125 1 2.6 0 0 3 0
4. 61 1 0 148 203 0 1 161 0 0.0 2 1 3 0
5. 62 0 0 138 294 1 1 106 0 1.9 1 3 2 0

**...** ... ... ... ... ... ... ... ... ... ... ... ... ... ...

1. 59 1 1 140 221 0 1 164 1 0.0 2 0 2 1
2. 60 1 0 125 258 0 0 141 1 2.8 1 1 3 0
3. 47 1 0 110 275 0 0 118 1 1.0 1 1 2 0
4. 50 0 0 110 254 0 0 159 0 0.0 2 0 2 1
5. 54 1 0 120 188 0 1 113 0 1.4 1 1 3 0
6. rows × 14 columns

no\_samples = len(df) no\_samples

1025

num\_classes = df['target'].nunique() print("Number of classes:", num\_classes)

Number of classes: 2

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1025 entries, 0 to 1024 Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

1. age 1025 non-null int64
2. sex 1025 non-null int64
3. cp 1025 non-null int64
4. trestbps 1025 non-null int64
5. chol 1025 non-null int64
6. fbs 1025 non-null int64
7. restecg 1025 non-null int64
8. thalach 1025 non-null int64
9. exang 1025 non-null int64
10. oldpeak 1025 non-null float64
11. slope 1025 non-null int64
12. ca 1025 non-null int64
13. thal 1025 non-null int64 13 target 1025 non-null int64 dtypes: float64(1), int64(13) memory usage: 112.2 KB

df.shape

(1025, 14)

df.isnull().sum()

**0**

**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

X= df.drop('target',axis=1) #feature(x) y=df['target'] #labely

X

**age**

**sex**

**cp**

**trestbps**

**chol**

**fbs**

**restecg**

**thalach**

**exang**

**oldpeak**

**slope**

**ca**

**thal**

1. 52 1 0 125 212 0 1 168 0 1.0 2 2 3
2. 53 1 0 140 203 1 0 155 1 3.1 0 0 3
3. 70 1 0 145 174 0 1 125 1 2.6 0 0 3
4. 61 1 0 148 203 0 1 161 0 0.0 2 1 3
5. 62 0 0 138 294 1 1 106 0 1.9 1 3 2

**...** ... ... ... ... ... ... ... ... ... ... ... ... ...

1. 59 1 1 140 221 0 1 164 1 0.0 2 0 2
2. 60 1 0 125 258 0 0 141 1 2.8 1 1 3
3. 47 1 0 110 275 0 0 118 1 1.0 1 1 2
4. 50 0 0 110 254 0 0 159 0 0.0 2 0 2
5. 54 1 0 120 188 0 1 113 0 1.4 1 1 3
6. rows × 13 columns

y

**target**

* 1. 0
  2. 0
  3. 0
  4. 0
  5. 0

**...** ...

1. 1
2. 0
3. 0
4. 1
5. 0
6. rows × 1 columns

|  |  |  |
| --- | --- | --- |
|  |  |  |

**dtype:** int64

# Split the data into 50% training and 50% testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=40)

#  Train the model using Randomforest

# Initialize and train the classifier

from sklearn.ensemble import RandomForestClassifier model = RandomForestClassifier(random\_state=42) model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred) report = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}') print('Classification Report:') print(report)

Accuracy: 0.97

Classification Report:

precision recall f1-score support

0 0.95 0.99 0.97 246

1 0.99 0.96 0.97 267

accuracy 0.97 513

macro avg 0.97 0.97 0.97 513

weighted avg 0.97 0.97 0.97 513

Q2.Apply min-max,z-score and normalization technique on dataset and then apply Classification

from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

# Min-Max Scaling

min\_max\_scaler = MinMaxScaler()

X\_train\_minmax = min\_max\_scaler.fit\_transform(X\_train)

X\_test\_minmax = min\_max\_scaler.transform(X\_test)

# Train Random Forest with Min-Max scaled data rf\_model\_minmax = RandomForestClassifier(random\_state=42) rf\_model\_minmax.fit(X\_train\_minmax, y\_train) y\_pred\_minmax = rf\_model\_minmax.predict(X\_test\_minmax)

# Evaluate Min-Max Random Forest

accuracy\_minmax = accuracy\_score(y\_test, y\_pred\_minmax) report\_minmax = classification\_report(y\_test, y\_pred\_minmax)

print("Min-Max Scaling:") print(f'Accuracy: {accuracy\_minmax:.2f}') print('Classification Report:') print(report\_minmax)

Min-Max Scaling:

Accuracy: 0.97

Classification Report:

precision recall f1-score support

0 0.95 0.99 0.97 246

1 0.99 0.96 0.97 267

accuracy 0.97 513

macro avg 0.97 0.97 0.97 513

weighted avg 0.97 0.97 0.97 513

standard\_scaler = StandardScaler()

X\_train\_standard = standard\_scaler.fit\_transform(X\_train)

X\_test\_standard = standard\_scaler.transform(X\_test)

# Train Random Forest with Z-score standardized data rf\_model\_standard = RandomForestClassifier(random\_state=42) rf\_model\_standard.fit(X\_train\_standard, y\_train) y\_pred\_standard = rf\_model\_standard.predict(X\_test\_standard)

# Evaluate Z-score Random Forest

accuracy\_standard=accuracy\_score(y\_test,y\_pred\_standard)

report\_standard=classification\_report(y\_test,y\_pred\_standard)

print("\nZ-score Standardization:") print(f'Accuracy: {accuracy\_standard:.2f}') print('Classification Report:') print(report\_standard)

Z-score Standardization:

Accuracy: 0.97

Classification Report:

precision recall f1-score support

0 0.95 0.99 0.97 246

1 0.99 0.96 0.97 267

accuracy 0.97 513

macro avg 0.97 0.97 0.97 513

weighted avg 0.97 0.97 0.97 513