```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import plotly.graph_objects as go
import plotly.express as px
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay
```

```
!pip install griddb_python
```

```
Collecting griddb_python
  Downloading griddb_python-0.8.5-cp310-cp310-manylinux1_x86_64.whl (784 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 784.9/784.9 kB 6.6 MB/s eta 0:00:00
Installing collected packages: griddb_python
Successfully installed griddb_python-0.8.5
```

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

```
heart_dataset.shape
```

```
(918, 12)
```

```
heart_dataset.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exe
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

```
heart_dataset.dtypes
```

```
Age                 int64
Sex                object
ChestPainType      object
RestingBP           int64
Cholesterol         int64
FastingBS           int64
RestingECG         object
MaxHR               int64
ExerciseAngina     object
Oldpeak           float64
ST_Slope           object
HeartDisease        int64
dtype: object
```

```
heart_dataset.isna().sum()
```

```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

```
heart_dataset.describe(include='all')
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingE(
count	918.000000	918	918	918.000000	918.000000	918.000000	91
unique	NaN	2	4	NaN	NaN	NaN	
top	NaN	M	ASY	NaN	NaN	NaN	Norm
freq	NaN	725	496	NaN	NaN	NaN	5(
mean	53.510893	NaN	NaN	132.396514	198.799564	0.233115	Na
std	9.432617	NaN	NaN	18.514154	109.384145	0.423046	Na
min	28.000000	NaN	NaN	0.000000	0.000000	0.000000	Na

```
categorical_cols= heart_dataset.select_dtypes(include=['object'])
categorical_cols.columns
```

```
    Index(['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope'], dtype='object')
```

```
for cols in categorical_cols.columns:
    print(cols,'-', len(categorical_cols[cols].unique()),'Labels')
```

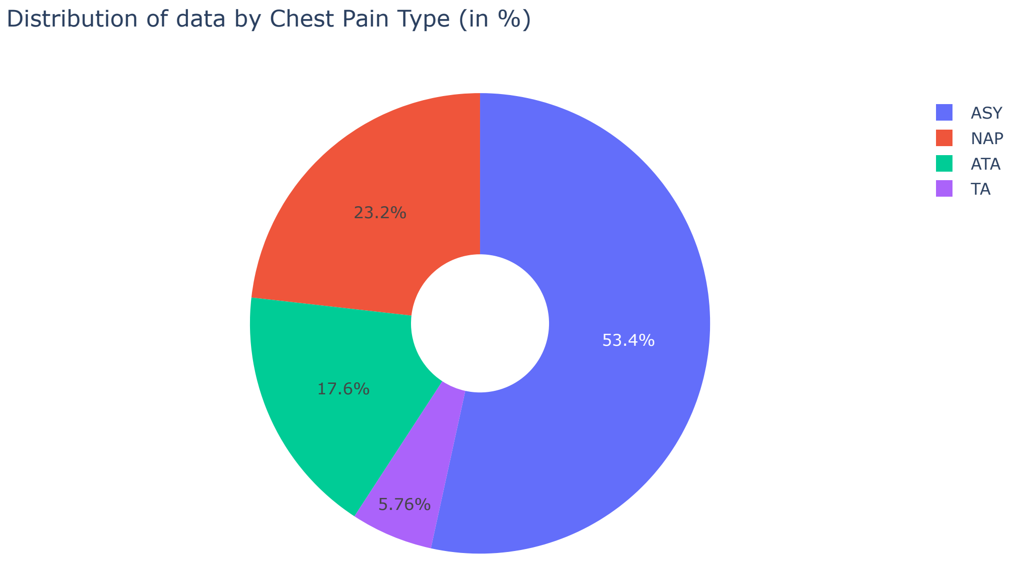
```
    Sex - 2 Labels
    ChestPainType - 4 Labels
    RestingECG - 3 Labels
    ExerciseAngina - 2 Labels
    ST_Slope - 3 Labels
```

```
train, test = train_test_split(heart_dataset,test_size=0.3,random_state= 1234)
```

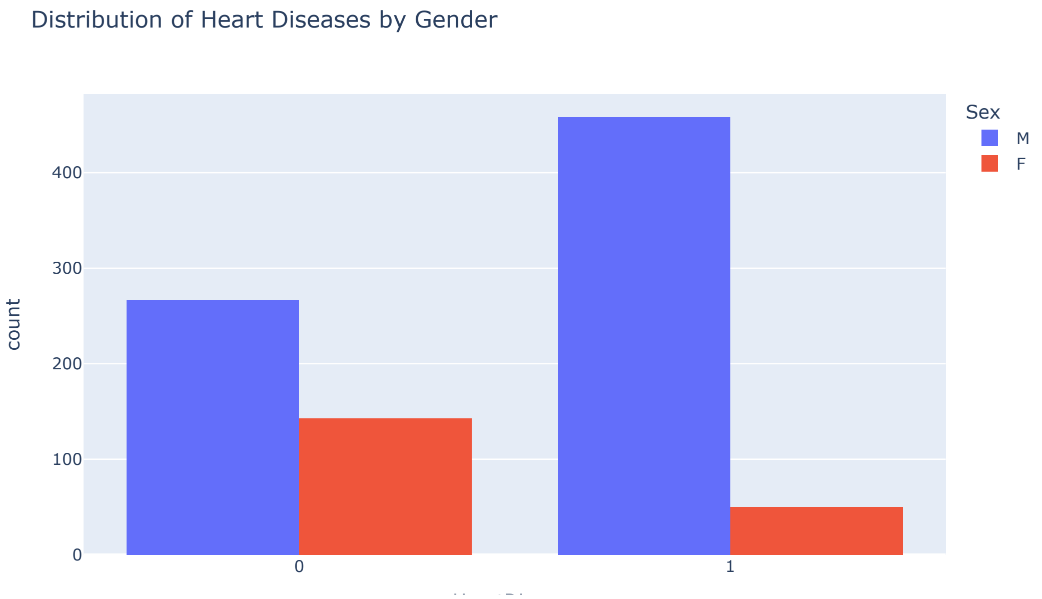
```
labels = [x for x in train.ChestPainType.value_counts().index]
values = train.ChestPainType.value_counts()
```

```
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])

fig.update_layout(
    title_text="Distribution of data by Chest Pain Type (in %)")
fig.update_traces()
fig.show()
```


Distribution of data by Chest Pain Type (in %)

```
fig=px.histogram(heart_dataset,
                 x="HeartDisease",
                 color="Sex",
                 hover_data=heart_dataset.columns,
                 title="Distribution of Heart Diseases by Gender",
                 barmode="group")
fig.show()
```


Distribution of Heart Diseases by Gender

```
train['Sex'] = np.where(train['Sex'] == "M", 0, 1)
train['ExerciseAngina'] = np.where(train['ExerciseAngina'] == "N", 0, 1)
test['Sex'] = np.where(test['Sex'] == "M", 0, 1)
test['ExerciseAngina'] = np.where(test['ExerciseAngina'] == "N", 0, 1)
```

```
train.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Ex
578	57	0	ASY	156	173	0	LVH	119	
480	58	0	ATA	126	0	1	Normal	110	
512	35	0	NAP	123	161	0	ST	153	
634	40	0	TA	140	199	0	Normal	178	
412	56	0	ASY	125	0	1	Normal	103	

```
train=pd.get_dummies(train)
test=pd.get_dummies(test)
```

```
train.head()
```

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	Hear
578	57	0	156	173	0	119	1	3.0	
480	58	0	126	0	1	110	1	2.0	
512	35	0	123	161	0	153	0	-0.1	
634	40	0	140	199	0	178	1	1.4	
412	56	0	125	0	1	103	1	1.0	

```
test.head()
```

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	Hear
581	48	0	140	208	0	159	1	1.5	
623	60	0	140	293	0	170	0	1.2	
60	49	0	100	253	0	174	0	0.0	
613	58	0	140	385	1	135	0	0.3	
40	54	1	150	230	0	130	0	0.0	

```
train.shape
```

```
(642, 19)
```

```
x_train=train.drop(['HeartDisease'],1)
x_test=test.drop(['HeartDisease'],1)

y_train=train['HeartDisease']
y_test=test['HeartDisease']
```

```
<ipython-input-21-7f118799b7b0>:1: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.

<ipython-input-21-7f118799b7b0>:2: FutureWarning:

In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
```

```
print(x_train.shape)
print(x_test.shape)
```

```
(642, 18)
(276, 18)
```

```
lr = LogisticRegression(max_iter=10000)
model1=lr.fit(x_train, y_train)
```

```
print("Train accuracy:",model1.score(x_train, y_train))
```

```
Train accuracy: 0.8582554517133957
```

```
print("Test accuracy:",model1.score(x_test,y_test))
```

```
Test accuracy: 0.894927536231884
```

```
lrpred = lr.predict(x_test)
```

```
print(classification_report(lrpred,y_test))
```

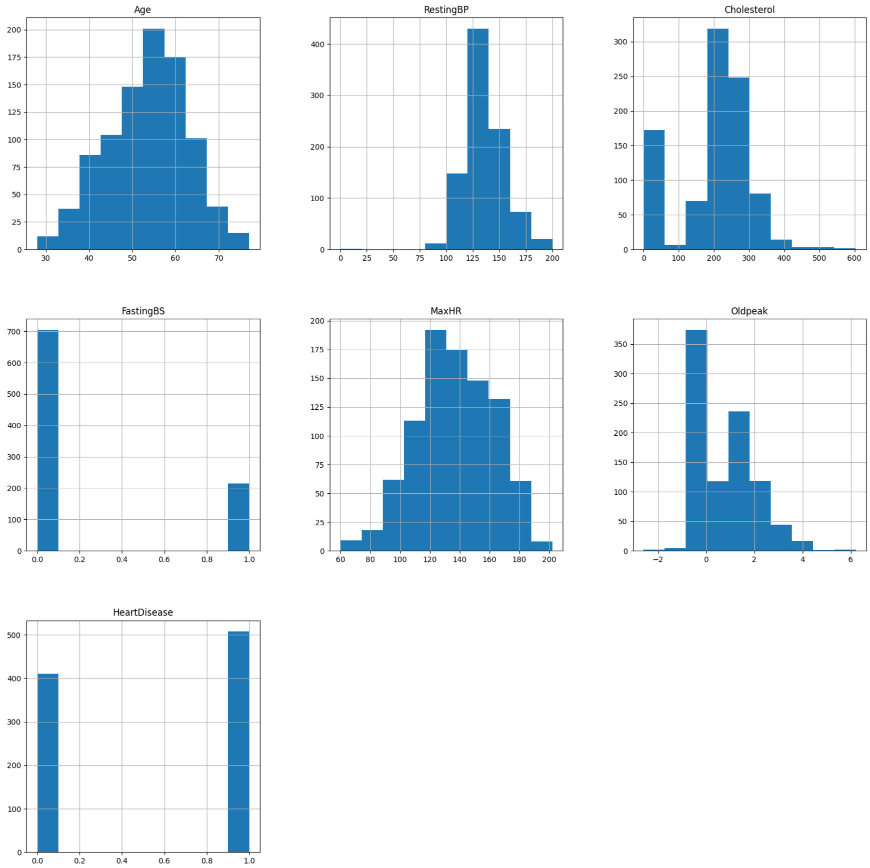
```
              precision    recall  f1-score   support

           0       0.85      0.90      0.88       114
           1       0.93      0.89      0.91       162

    accuracy                           0.89       276
   macro avg       0.89      0.90      0.89       276
weighted avg       0.90      0.89      0.90       276
```

```
import matplotlib.pyplot as plt
heart_dataset.hist(figsize = (20,20))

plt.show()
```


Age
0
25
50
75
100
125
150
175
200
30
40
50
60
70
RestingBP
0
100
200
300
400
0
25
50
75
100
125
150
175
200
Cholesterol
0
50
100
150
200
250
300
0
100
200
300
400
500
600
FastingBS
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
MaxHR
0
25
50
75
100
125
150
175
200
60
80
100
120
140
160
180
200
Oldpeak
0
50
100
150
200
250
300
350
−2
0
2
4
6
HeartDisease
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0