Project 1: Heart disease diagnosis

Machine Learning: Logistic Regression

Project build under the guidance of Professor **Nitesh Karmakar**

Machine Learning Mentor(KSI)

Facts

Every day, the average human heart beats around 100,000 times, pumping 2,000 gallons of blood through the body. Inside your body there are 60,000 miles of blood vessels.

The signs of a woman having a heart attack are much less noticeable than the signs of a male.

World Health Organization has estimated that four out of five cardiovascular diseases(CVD) deaths are due to heart attacks.

The prediction of heart disease is considered one of the most important topics in health domain.

Also,

The amount of data in the healthcare industry is huge.

With the machine learning algorithms and having large amounts of data, it is possible to extrapolate information that can help doctors make more accurate predictions.

Dataset

The dataset is a CSV(comma separated value) having 303 rows × 14 columns

Amongst that 14 column, there are 13 features and 1 labelled output column.

Each feature and it’s significance is discussed in the next slide.

Heart disease describes a range of conditions that affect your heart.

- ➔ **Age:** The person's age in years
- ➔ **Sex:** The person's sex (1 = male, 0 = female)
- ➔ **Chest Pain/Angina:** The chest pain experienced
  - ◆ Value 1: typical
  - ◆ Value 2: nontypical
  - ◆ Value 3: asymptomatic
  - ◆ Value 4: nonanginal
- ➔ **RestBP:** The person's resting blood pressure (mm Hg on admission to the hospital)
- ➔ **Chol:** The person's cholesterol measurement in mg/dl
- ➔ **Fbs:** The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)
- ➔ **RestECG:** Resting electrocardiographic measurement
  - ◆ 0 = normal
  - ◆ 1 = having ST-T wave abnormality
  - ◆ 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria

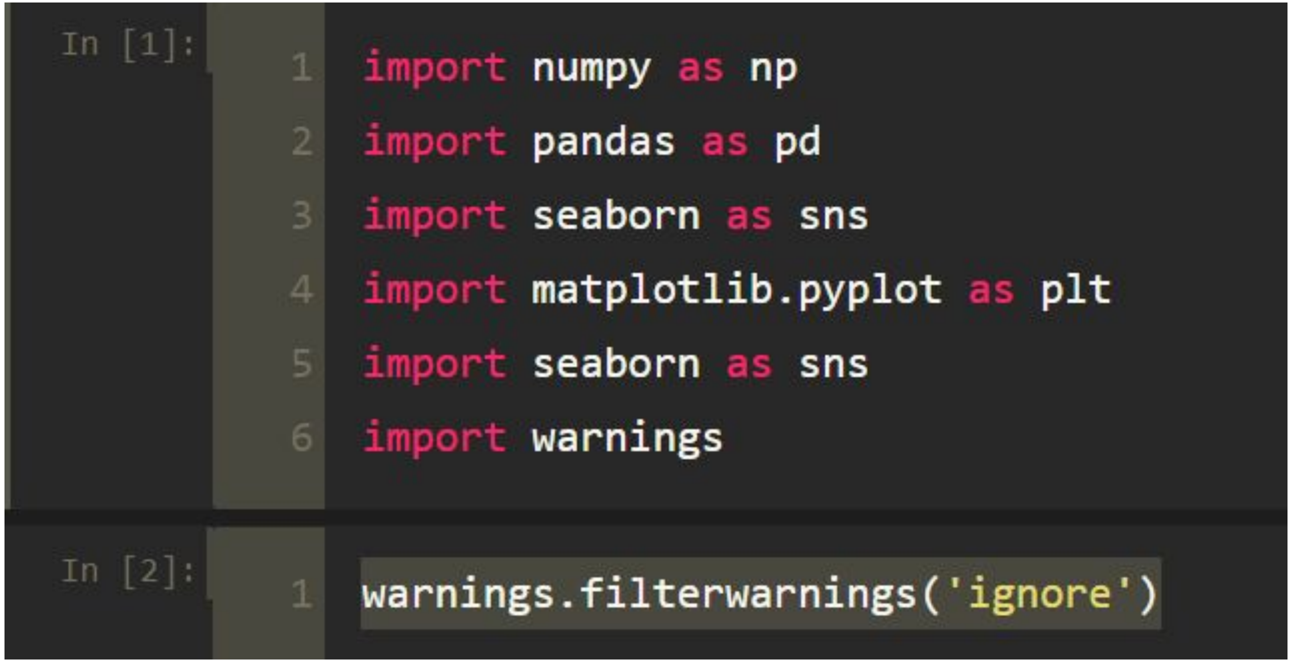
- **MaxHR:** The person's maximum heart rate achieved
- **ExAng:** Exercise induced angina (1 = yes; 0 = no)
- **Oldpeak:** ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot.
- **Slope**: the slope of the peak exercise ST segment
  - Value 1: upsloping
  - Value 2: flat
  - Value 3: downsloping
- **Ca**: The number of major vessels (0-3)
- **Thal**: A blood disorder called thalassemia
  - Value 1: normal
  - Value 2: fixed defect
  - Value 3: reversible defect

Understanding the code

Understanding how to approach this Machine Learning Model

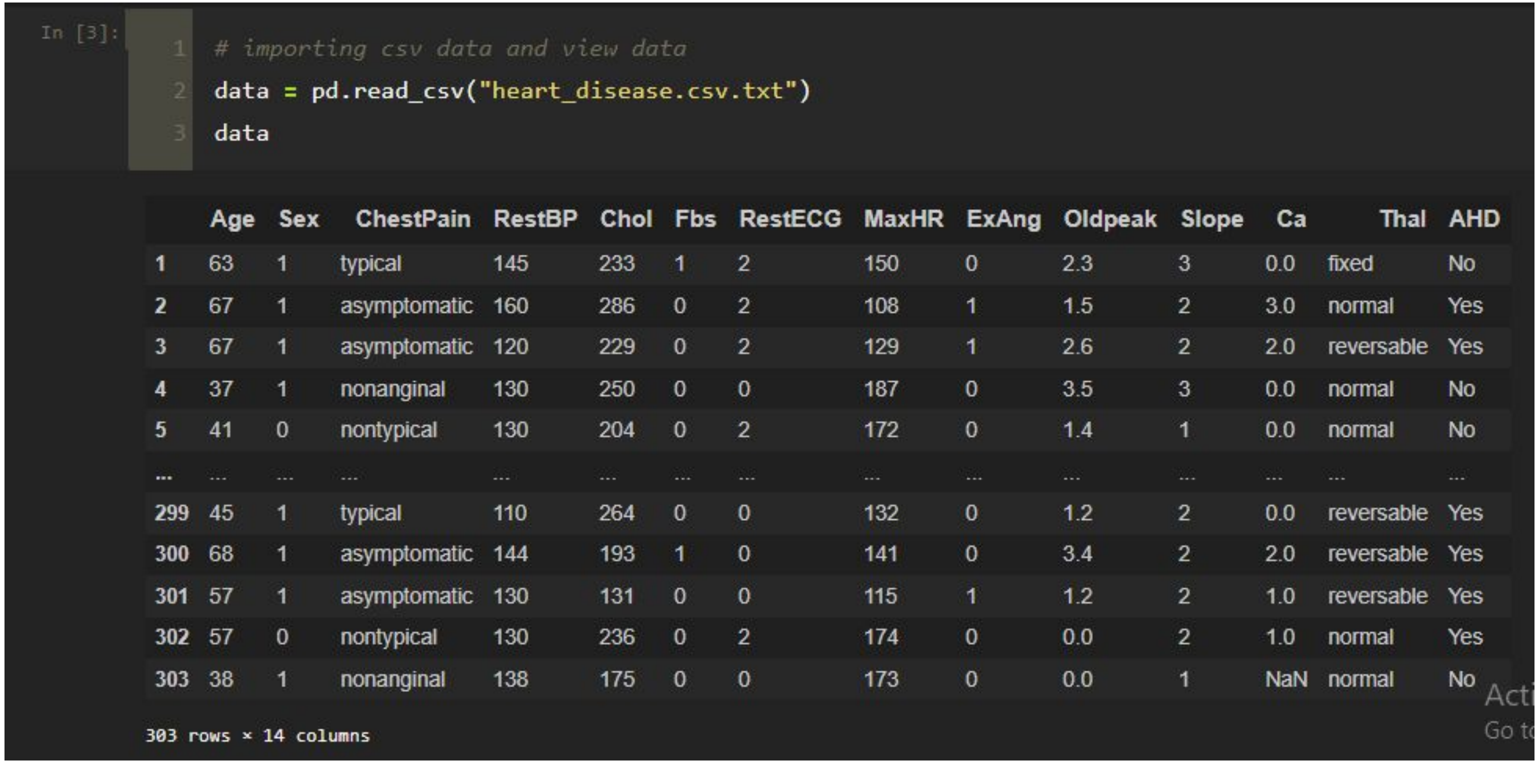
1. Importing libraries
2. Importing Dataset and Read the data (from csv)
3. Identify the dependent and independent variables.
4. Check if the data has missing values or the data is categorical or not.
5. Visualize the data.
6. Now split the data into the groups of training and testing for the respective purpose.
7. After splitting data, fit it to a most suitable model.
8. Prediction
9. Model **Evaluation**

1. Importing Libraries



```
In [1]:  import numpy as np
         import pandas as pd
         import seaborn as sns
         import matplotlib.pyplot as plt
         import seaborn as sns
         import warnings

In [2]:  warnings.filterwarnings('ignore')
```

2. Importing Dataset and Reading the data



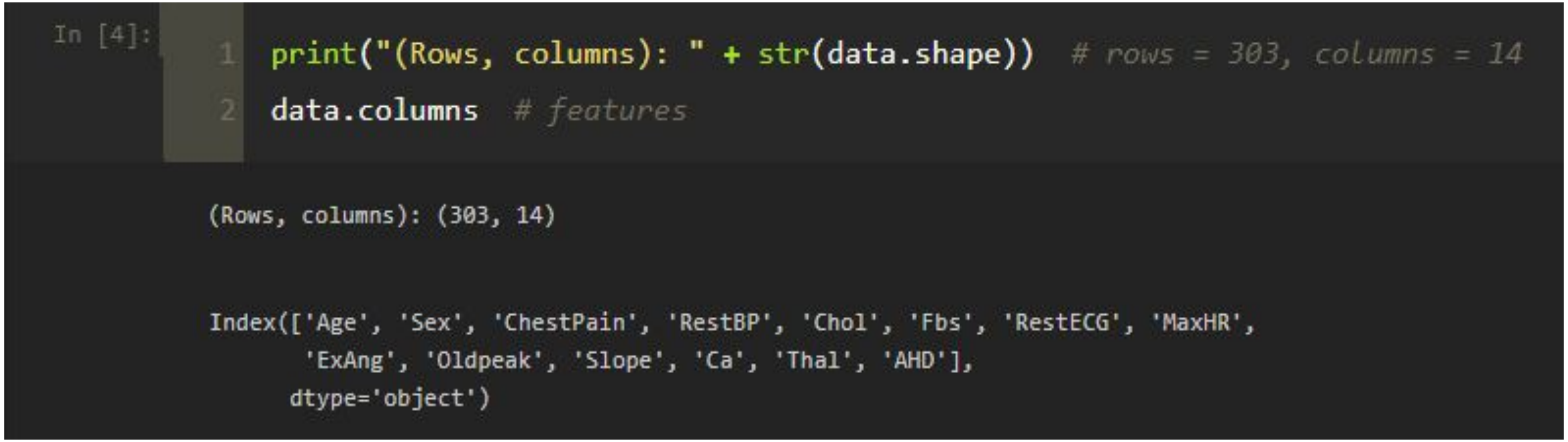
In [3]:

```python
# importing csv data and view data
data = pd.read_csv("heart_disease.csv.txt")
data
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes
301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	Yes
302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.0	normal	Yes
303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	NaN	normal	No

303 rows × 14 columns

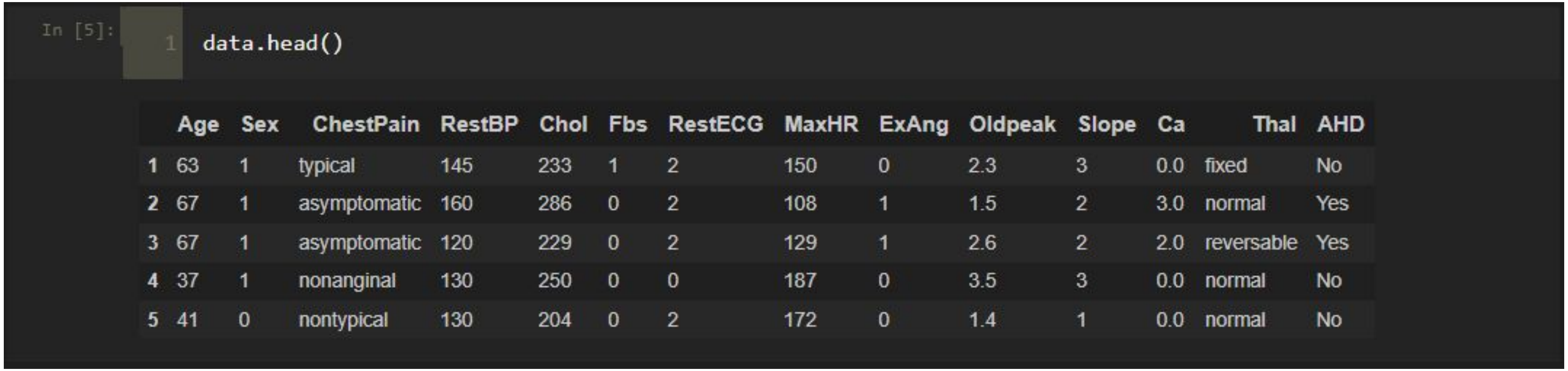
Checking rows and columns



```
In [4]:
print("(Rows, columns): " + str(data.shape))  # rows = 303, columns = 14
data.columns  # features
```

```
(Rows, columns): (303, 14)

Index(['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs', 'RestECG', 'MaxHR',
       'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal', 'AHD'],
      dtype='object')
```

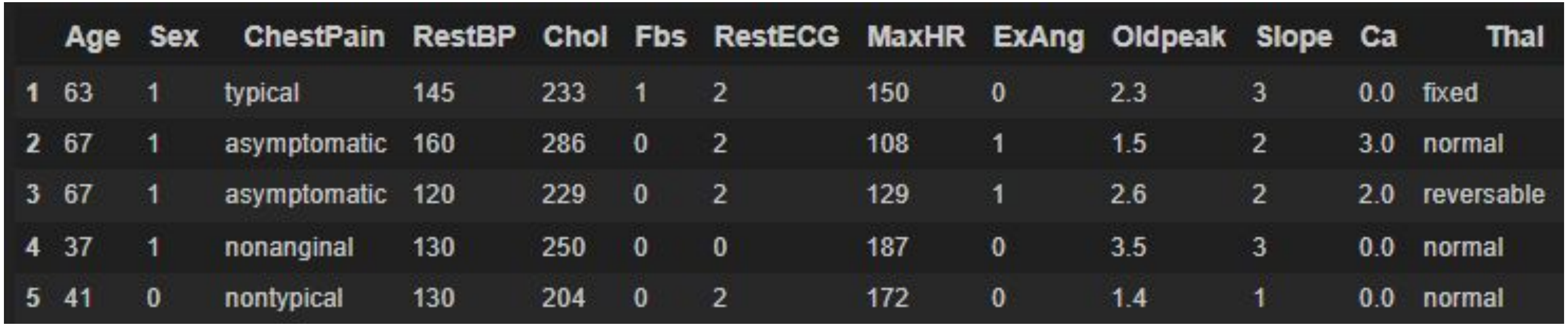
Checking first 5 values from the imported dataset



```
In [5]:  data.head()
```

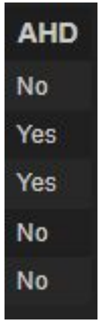
	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

3. Identify the dependent and independent variables.



	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal

Dependent Variable:



AHD
No
Yes
Yes
No
No

Dependent variable exists

- Since the AHD column exists, which depends on the independent 13 features. Hence, it is dependent variable.
- As dependent variable exists, it falls into the category of supervised learning.
- Also, the output of the model is either **Yes** or **No.** It is a binary classification.
- Hence, we will use Logistic Regression for this problem.

Why choose Logistic Regression ?

Logistic Regression is a statistical and machine-learning techniques classifying records of a dataset based on the values of the input fields.

It predicts a dependent variable based on one or more set of independent variables to predict outcomes .

It can be used both for binary classification and multi-class classification

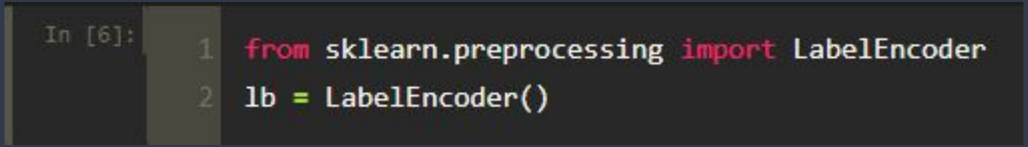
In this project, we have to predict whether the person has heart disease or not, which means, it falls into the category of binary classification(Since, we have to predict- Yes/No).

4. Check if the data has missing values or the data is categorical or not.

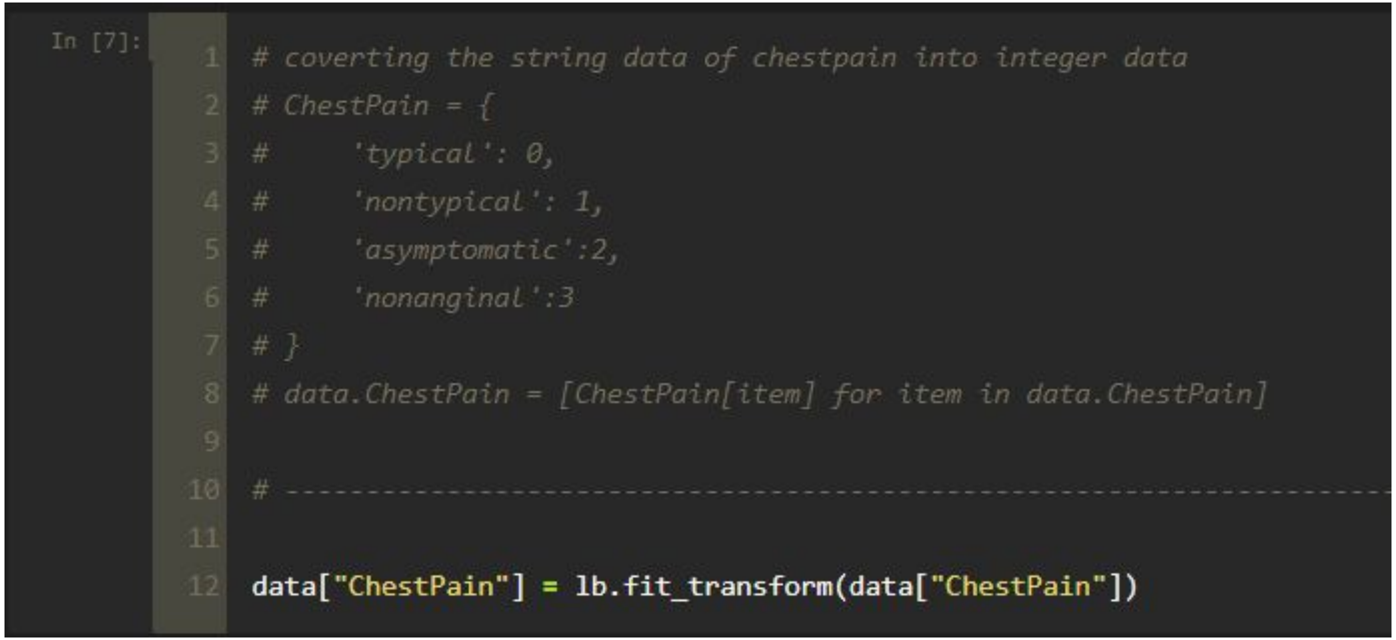
Changing string data type to numeric data type:

- Data generally needs to be put into numeric form for machine learning algorithms to use the data to make predictions.

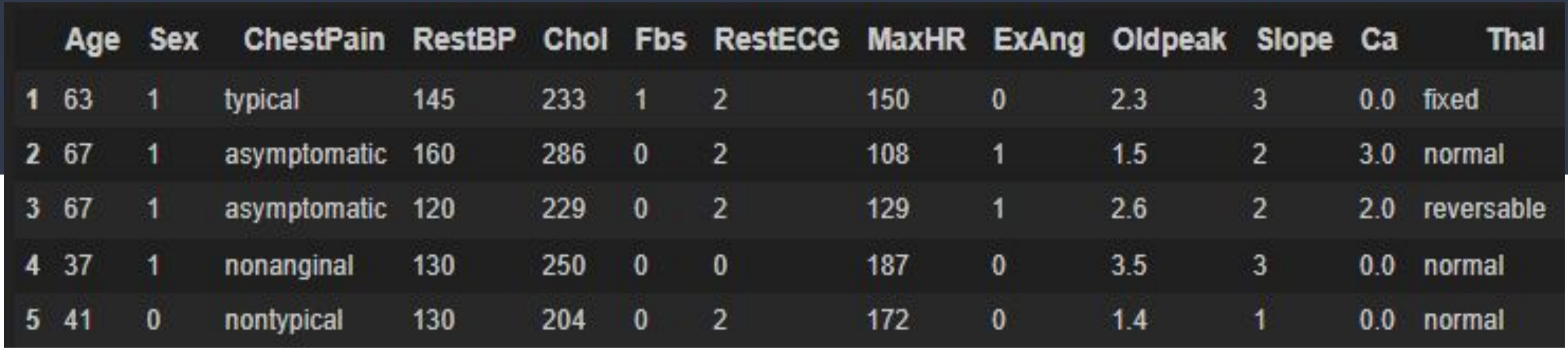
- Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.



```
In [6]:
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
```

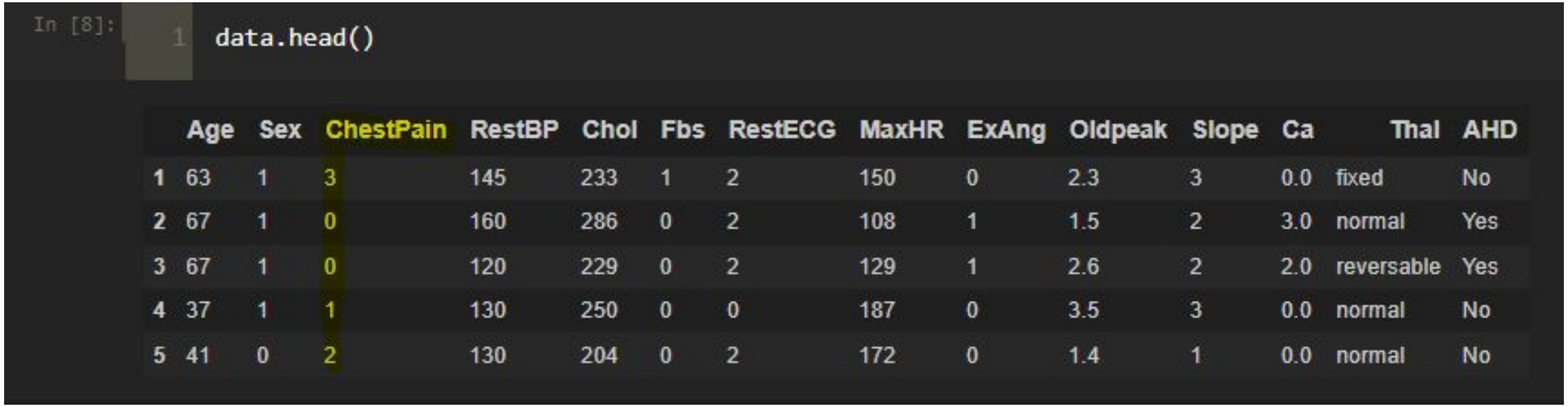


```
In [7]:
# coverting the string data of chestpain into integer data
# ChestPain = {
#     'typical': 0,
#     'nontypical': 1,
#     'asymptomatic':2,
#     'nonanginal':3
# }
# data.ChestPain = [ChestPain[item] for item in data.ChestPain]

# --------------------------------------------------------------------------------------------

data["ChestPain"] = lb.fit_transform(data["ChestPain"])
```



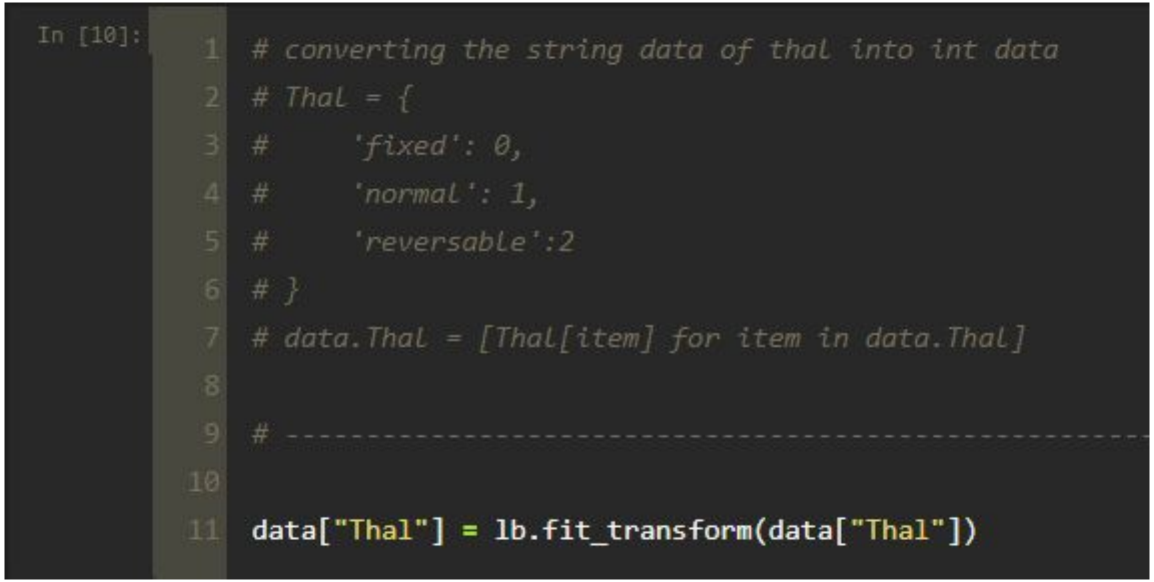
	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal

After Label Encoding:

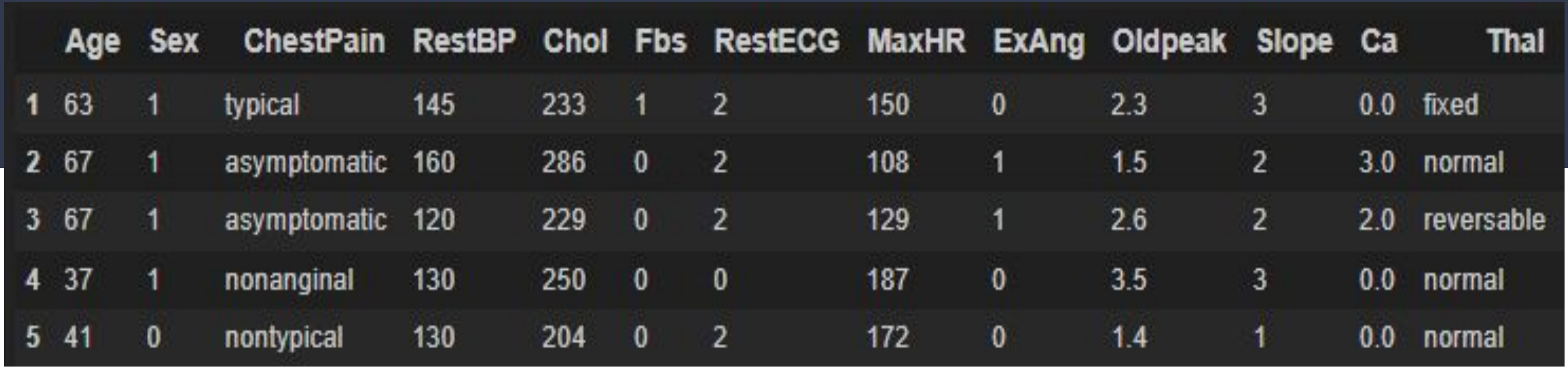


```
In [8]:  data.head()
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	3	145	233	1	2	150	0	2.3	3	0.0	fixed	No
2	67	1	0	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
3	67	1	0	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
4	37	1	1	130	250	0	0	187	0	3.5	3	0.0	normal	No
5	41	0	2	130	204	0	2	172	0	1.4	1	0.0	normal	No

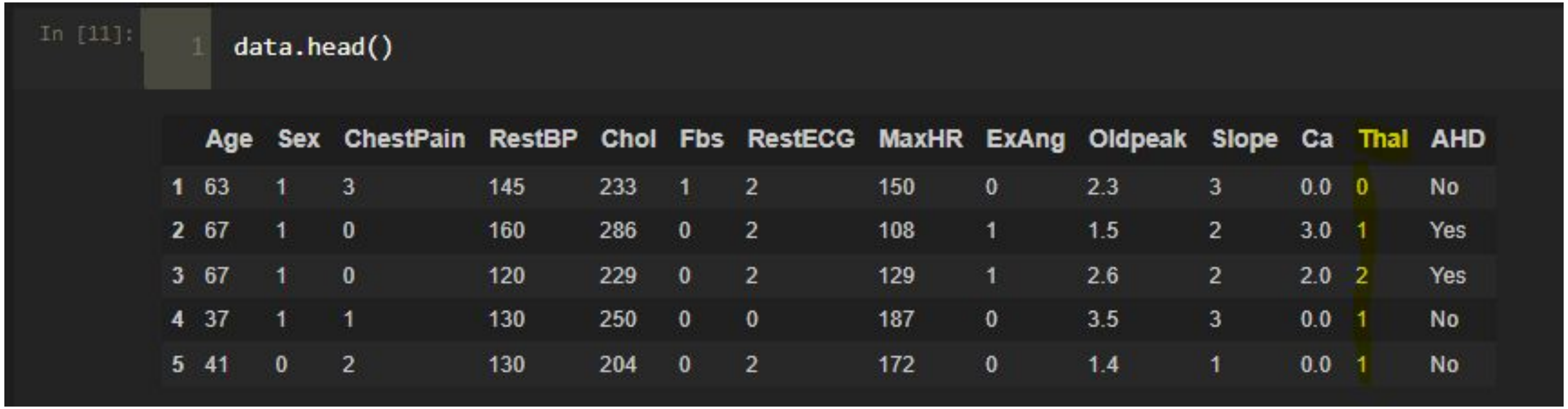


```
In [10]:
# converting the string data of thal into int data
# Thal = {
#     'fixed': 0,
#     'normal': 1,
#     'reversable':2
# }
# data.Thal = [Thal[item] for item in data.Thal]

# -------------------------------------------------------

data["Thal"] = lb.fit_transform(data["Thal"])
```



	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal

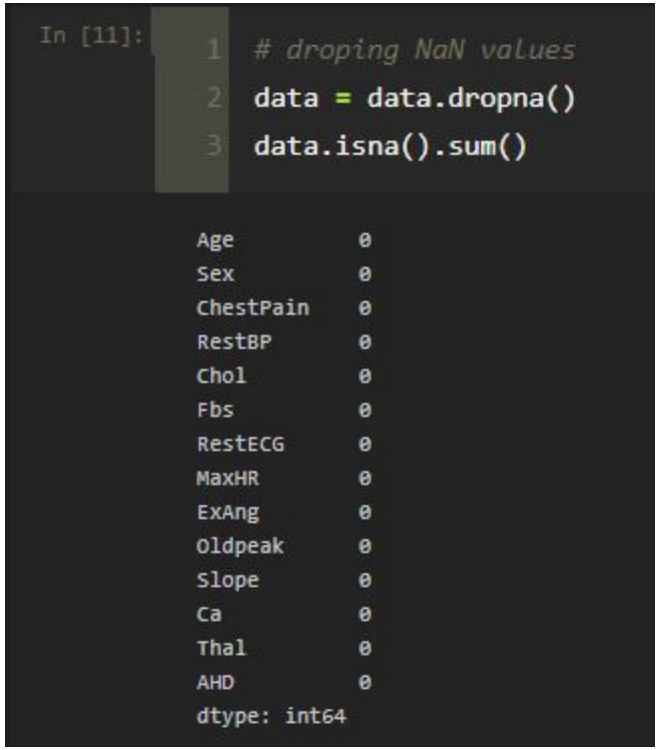
After Label Encoding:



```
In [11]: data.head()
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	3	145	233	1	2	150	0	2.3	3	0.0	0	No
2	67	1	0	160	286	0	2	108	1	1.5	2	3.0	1	Yes
3	67	1	0	120	229	0	2	129	1	2.6	2	2.0	2	Yes
4	37	1	1	130	250	0	0	187	0	3.5	3	0.0	1	No
5	41	0	2	130	204	0	2	172	0	1.4	1	0.0	1	No

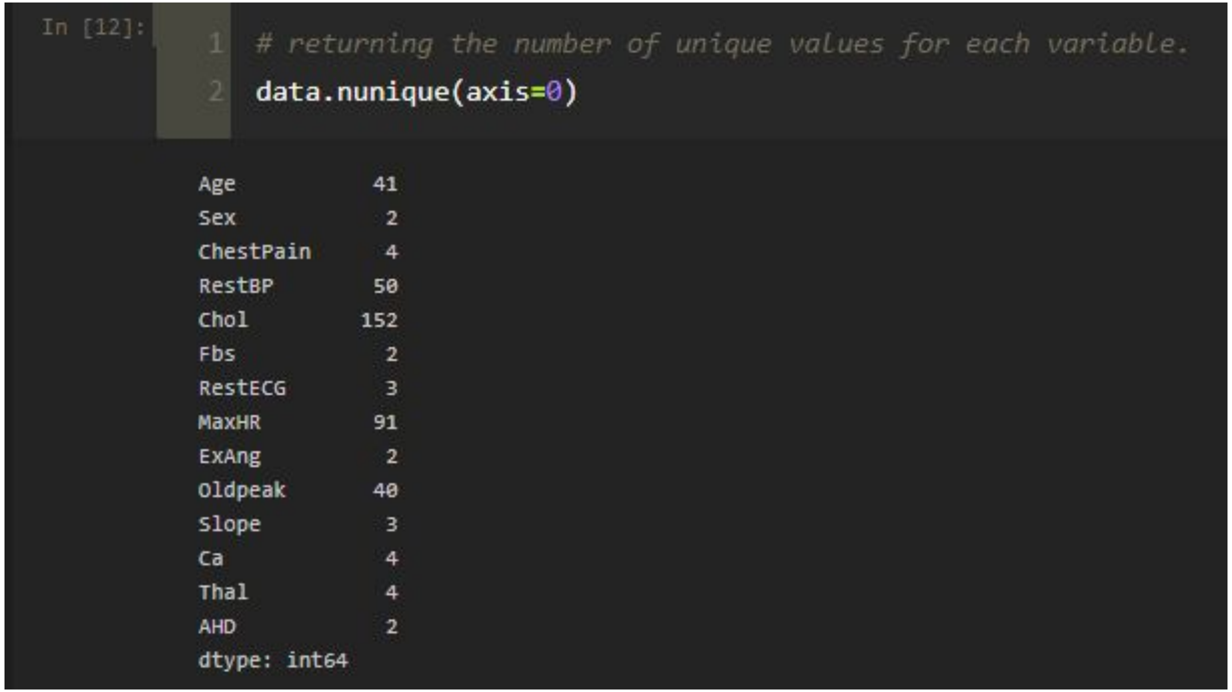
Dropping NaN/Null Values:



```
In [11]:
# droping NaN values
data = data.dropna()
data.isna().sum()
```

```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           0
Thal         0
AHD          0
dtype: int64
```

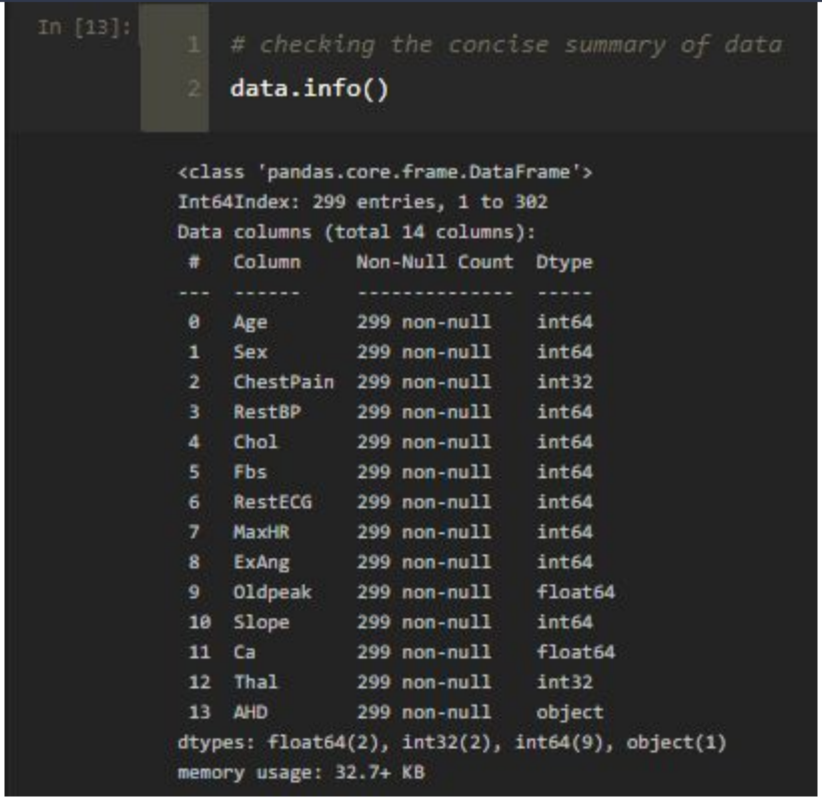
Unique values for each variable:



```
In [12]:
# returning the number of unique values for each variable.
data.nunique(axis=0)
```

```
Age          41
Sex           2
ChestPain     4
RestBP       50
Chol        152
Fbs           2
RestECG       3
MaxHR        91
ExAng         2
Oldpeak      40
Slope         3
Ca            4
Thal          4
AHD           2
dtype: int64
```

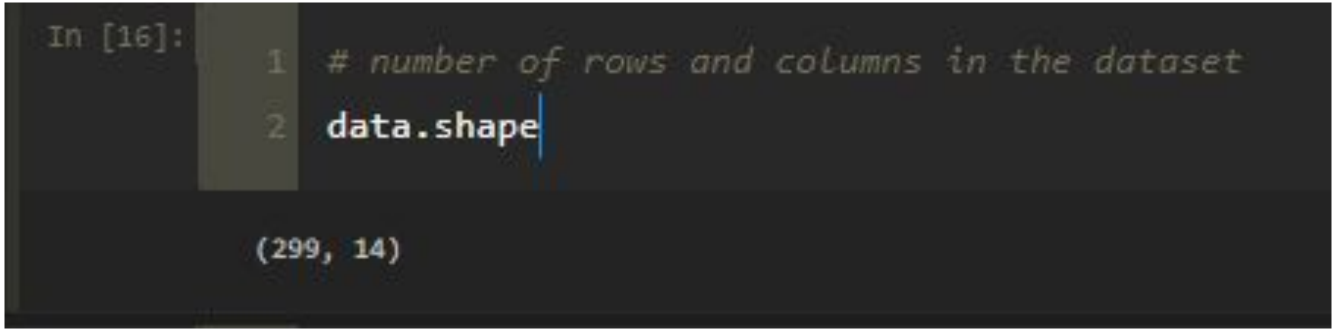
Concise summary of data:



```
In [13]:
# checking the concise summary of data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 299 entries, 1 to 302
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Age        299 non-null    int64  
 1   Sex        299 non-null    int64  
 2   ChestPain  299 non-null    int32  
 3   RestBP     299 non-null    int64  
 4   Chol       299 non-null    int64  
 5   Fbs        299 non-null    int64  
 6   RestECG    299 non-null    int64  
 7   MaxHR      299 non-null    int64  
 8   ExAng      299 non-null    int64  
 9   Oldpeak    299 non-null    float64
 10  Slope      299 non-null    int64  
 11  Ca         299 non-null    float64
 12  Thal       299 non-null    int32  
 13  AHD        299 non-null    object 
dtypes: float64(2), int32(2), int64(9), object(1)
memory usage: 32.7+ KB
```

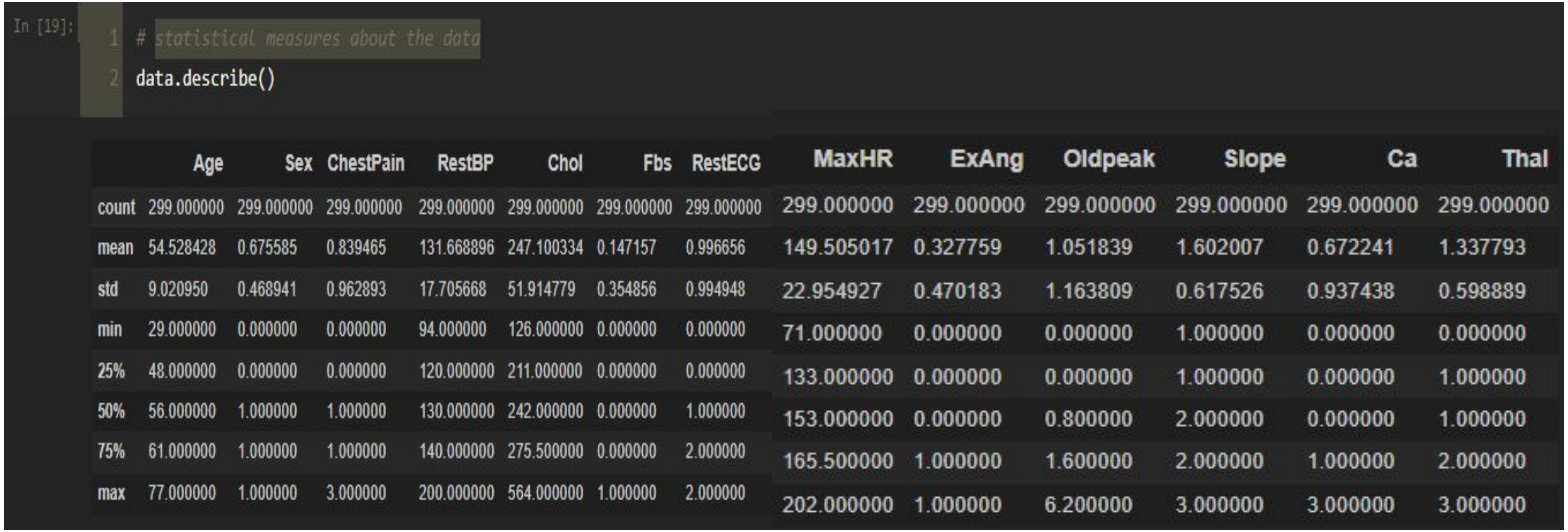
Rows and columns in the dataset after dropping NaN values:



```
In [16]:
# number of rows and columns in the dataset
data.shape
```

```
(299, 14)
```

Statistical measures about the data:

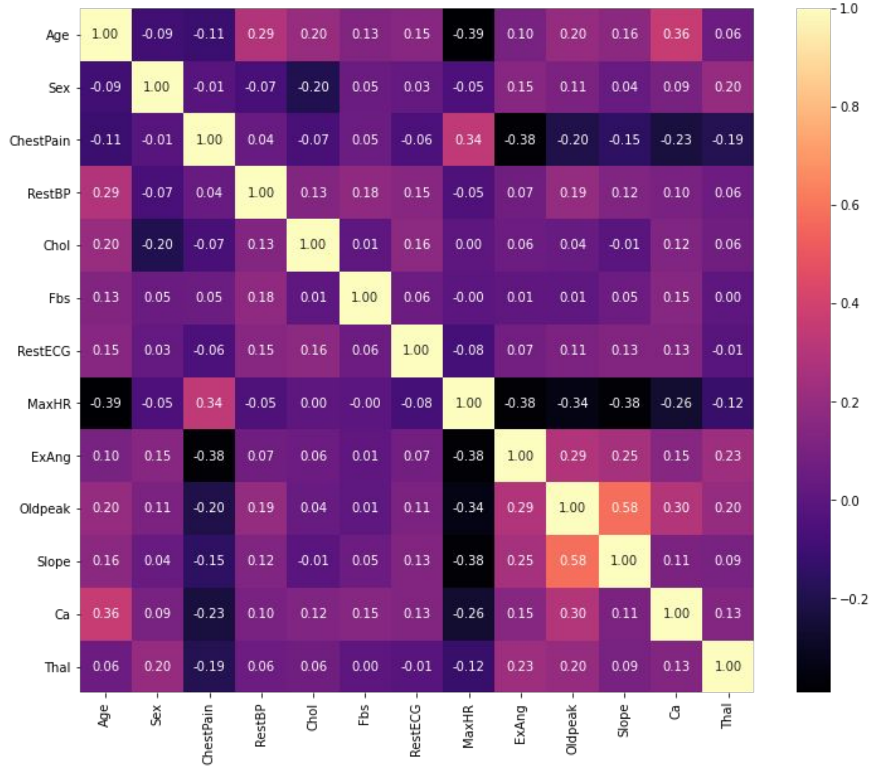


In [19]:

```python
# statistical measures about the data
data.describe()
```

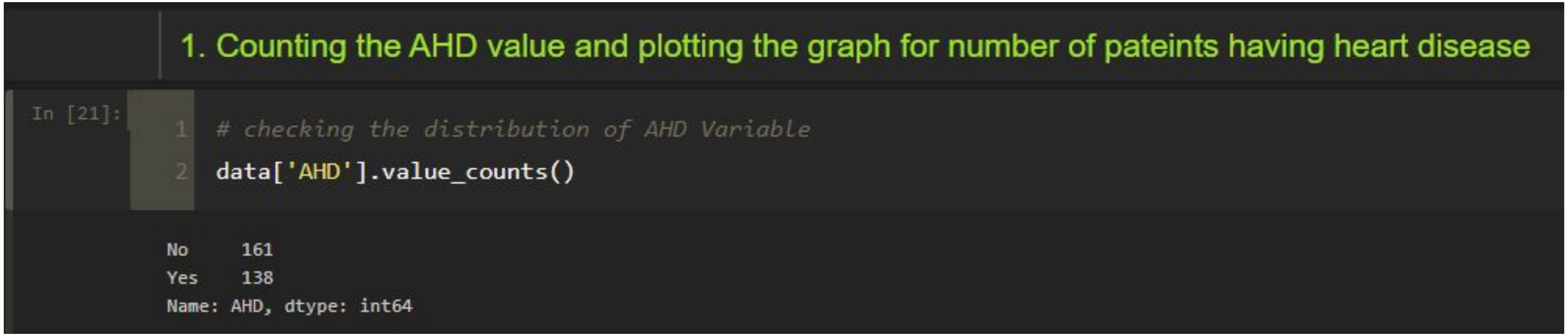
	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000
mean	54.528428	0.675585	0.839465	131.668896	247.100334	0.147157	0.996656	149.505017	0.327759	1.051839	1.602007	0.672241	1.337793
std	9.020950	0.468941	0.962893	17.705668	51.914779	0.354856	0.994948	22.954927	0.470183	1.163809	0.617526	0.937438	0.598889
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000	1.000000
50%	56.000000	1.000000	1.000000	130.000000	242.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000	1.000000
75%	61.000000	1.000000	1.000000	140.000000	275.500000	0.000000	2.000000	165.500000	1.000000	1.600000	2.000000	1.000000	2.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	3.000000

Correlation matrix: Correlation indicates how the features are related to each other or to the target variable.



	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
Age	1.00	-0.09	-0.11	0.29	0.20	0.13	0.15	-0.39	0.10	0.20	0.16	0.36	0.06
Sex	-0.09	1.00	-0.01	-0.07	-0.20	0.05	0.03	-0.05	0.15	0.11	0.04	0.09	0.20
ChestPain	-0.11	-0.01	1.00	0.04	-0.07	0.05	-0.06	0.34	-0.38	-0.20	-0.15	-0.23	-0.19
RestBP	0.29	-0.07	0.04	1.00	0.13	0.18	0.15	-0.05	0.07	0.19	0.12	0.10	0.06
Chol	0.20	-0.20	-0.07	0.13	1.00	0.01	0.16	0.00	0.06	0.04	-0.01	0.12	0.06
Fbs	0.13	0.05	0.05	0.18	0.01	1.00	0.06	-0.00	0.01	0.01	0.05	0.15	0.00
RestECG	0.15	0.03	-0.06	0.15	0.16	0.06	1.00	-0.08	0.07	0.11	0.13	0.13	-0.01
MaxHR	-0.39	-0.05	0.34	-0.05	0.00	-0.00	-0.08	1.00	-0.38	-0.34	-0.38	-0.26	-0.12
ExAng	0.10	0.15	-0.38	0.07	0.06	0.01	0.07	-0.38	1.00	0.29	0.25	0.15	0.23
Oldpeak	0.20	0.11	-0.20	0.19	0.04	0.01	0.11	-0.34	0.29	1.00	0.58	0.30	0.20
Slope	0.16	0.04	-0.15	0.12	-0.01	0.05	0.13	-0.38	0.25	0.58	1.00	0.11	0.09
Ca	0.36	0.09	-0.23	0.10	0.12	0.15	0.13	-0.26	0.15	0.30	0.11	1.00	0.13
Thal	0.06	0.20	-0.19	0.06	0.06	0.00	-0.01	-0.12	0.23	0.20	0.09	0.13	1.00

5. Visualizing the data



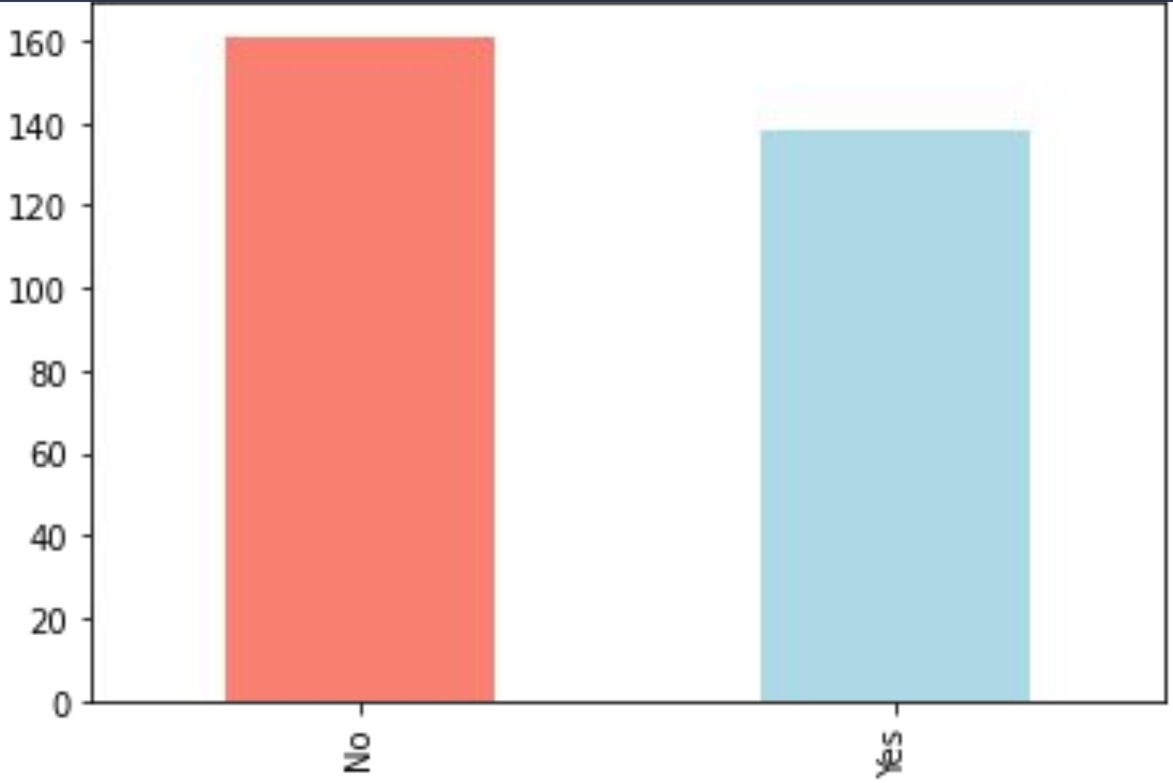
1. Counting the AHD value and plotting the graph for number of pateints having heart disease

In [21]:

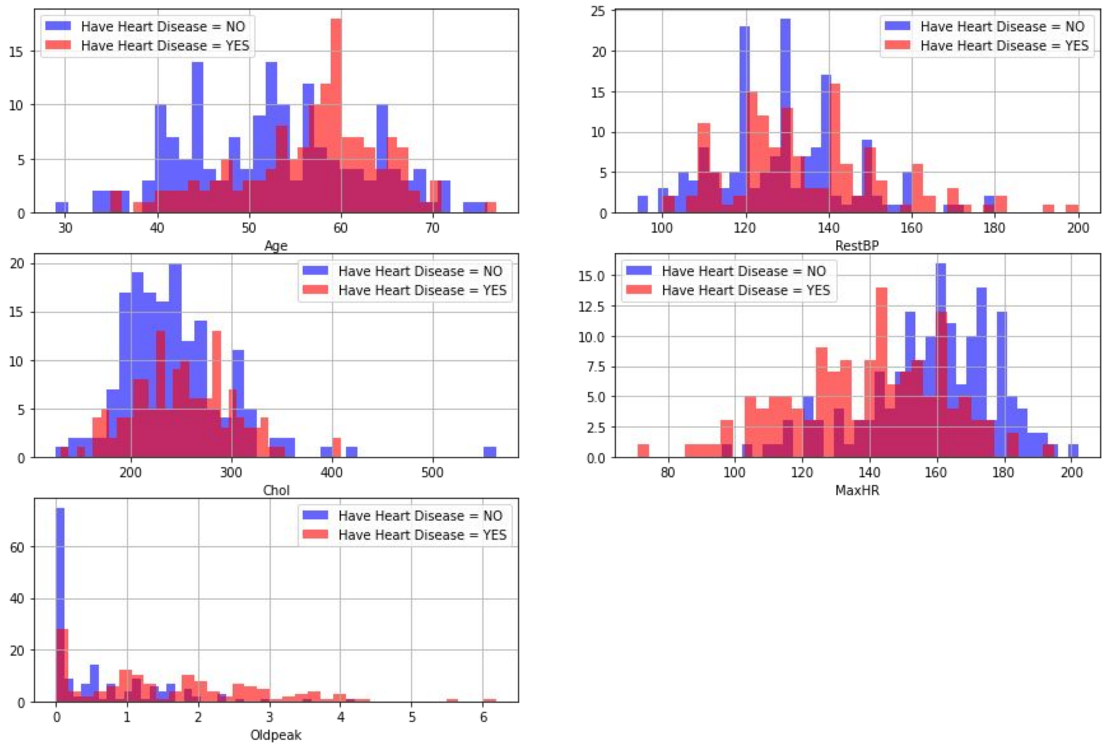
```
# checking the distribution of AHD Variable
data['AHD'].value_counts()
```

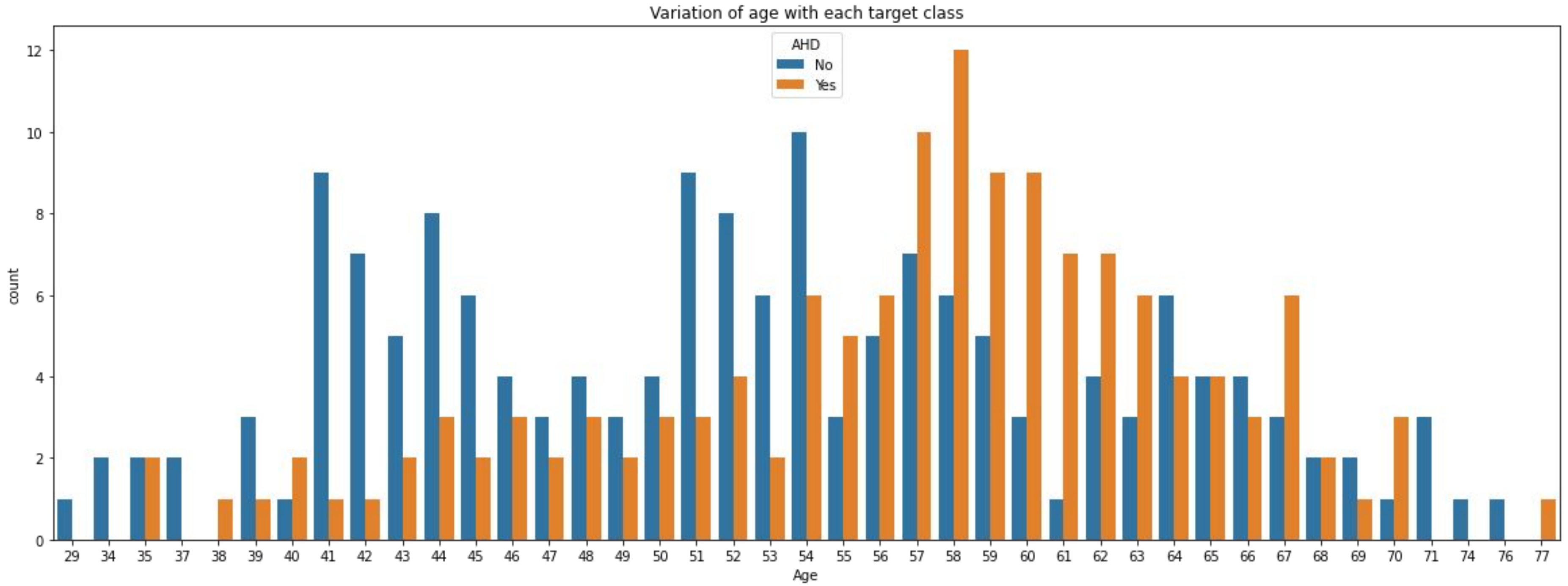
```
No     161
Yes    138
Name: AHD, dtype: int64
```

Graph for number of patients having heart disease:

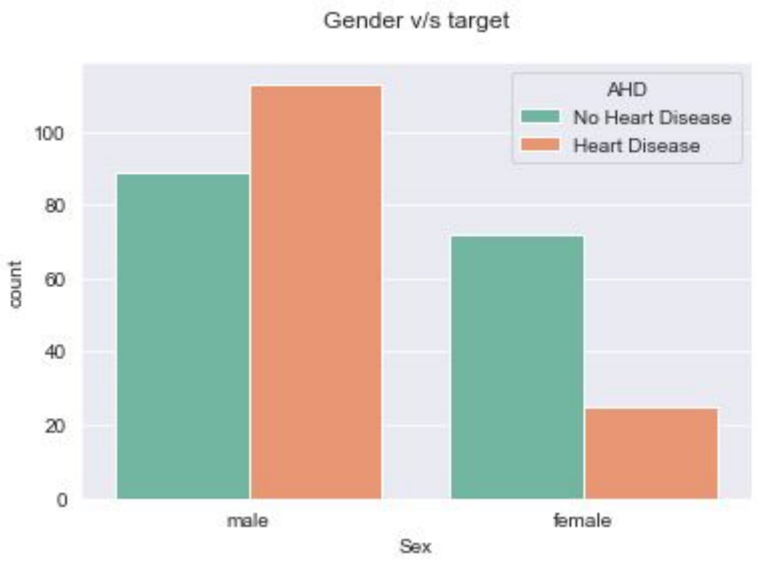


Relation of various attributes with the target:



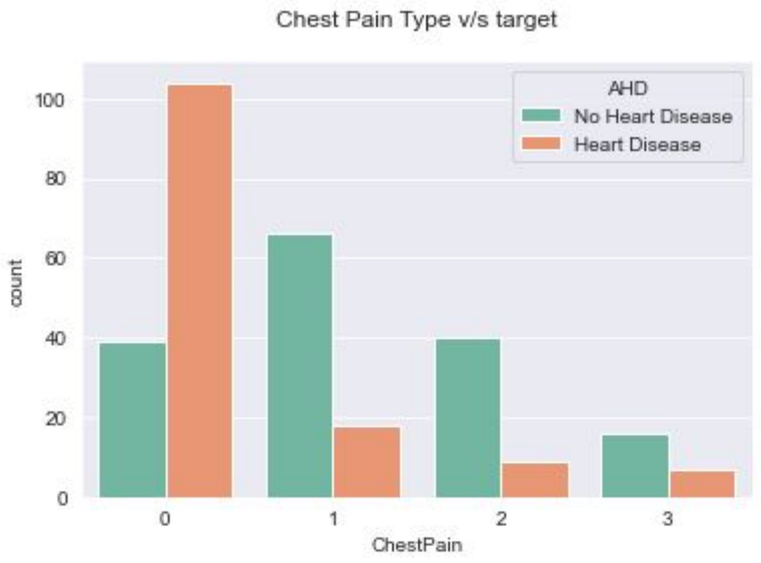

Variation of age with each target class
AHD
No
Yes
count
Age
0
2
4
6
8
10
12
29 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 74 76 77

According to this dataset, **Males** are more susceptible to get Heart Disease than **Females**.

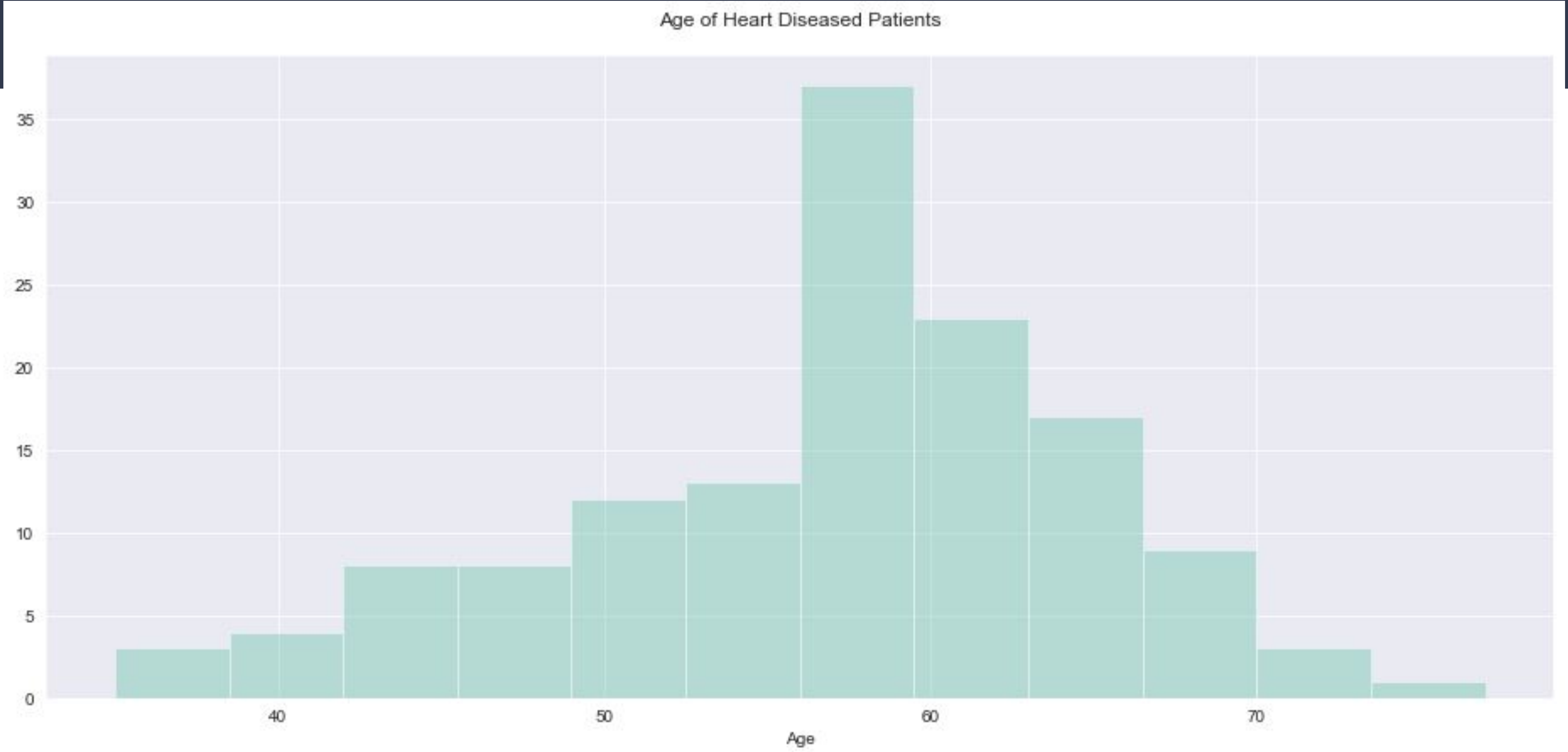


- There are four types of chest pain, typical, non typical, asymptomatic and nonanginal.
- Most of the Heart Disease patients are found to have asymptomatic chest pain

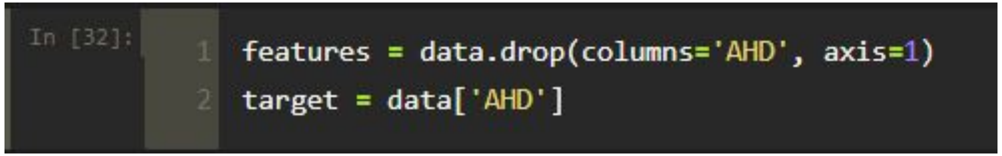
- Value 1: typical
- Value 2: non typical
- Value 3: asymptomatic
- Value 4: nonanginal



Heart Disease is very common in the seniors which is composed of **age group 60 and above** and common among adults which belong to the age group of **41 to 60**. But it’s rare among the age group of **19 to 40** and very rare among the age group of 0 to 18.

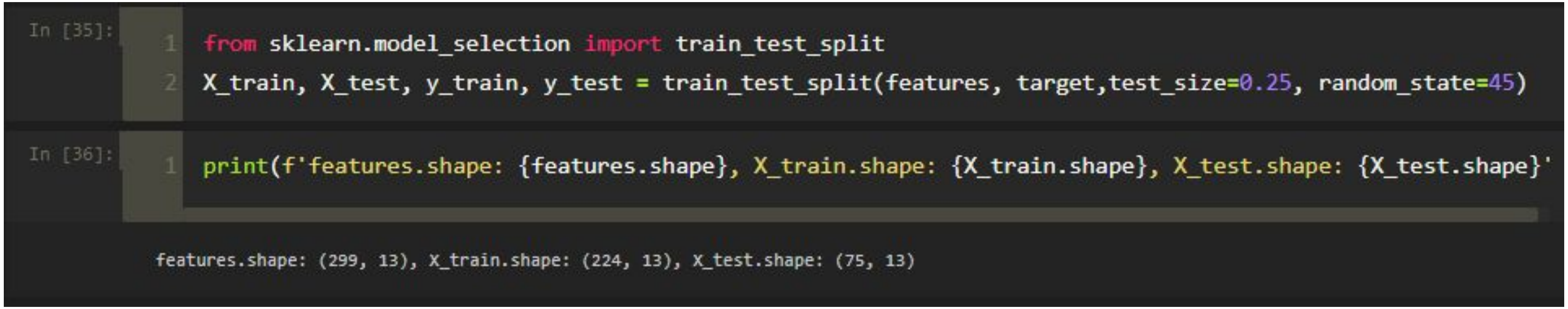


5. Prepare Data for Modeling



```
In [32]:
features = data.drop(columns='AHD', axis=1)
target = data['AHD']
```

6. Splitting the data

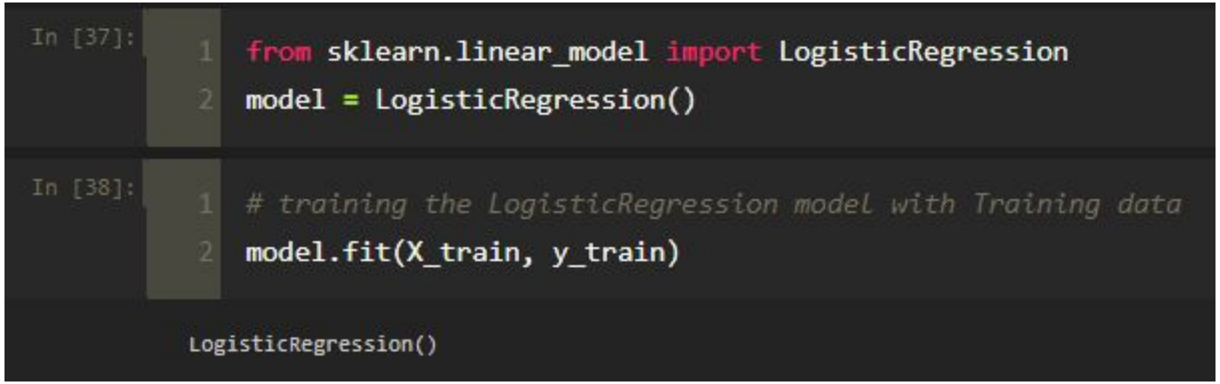


```
In [35]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, target,test_size=0.25, random_state=45)
```

```
In [36]:
print(f'features.shape: {features.shape}, X_train.shape: {X_train.shape}, X_test.shape: {X_test.shape}'
```

```
features.shape: (299, 13), X_train.shape: (224, 13), X_test.shape: (75, 13)
```

7. Fitting into Model

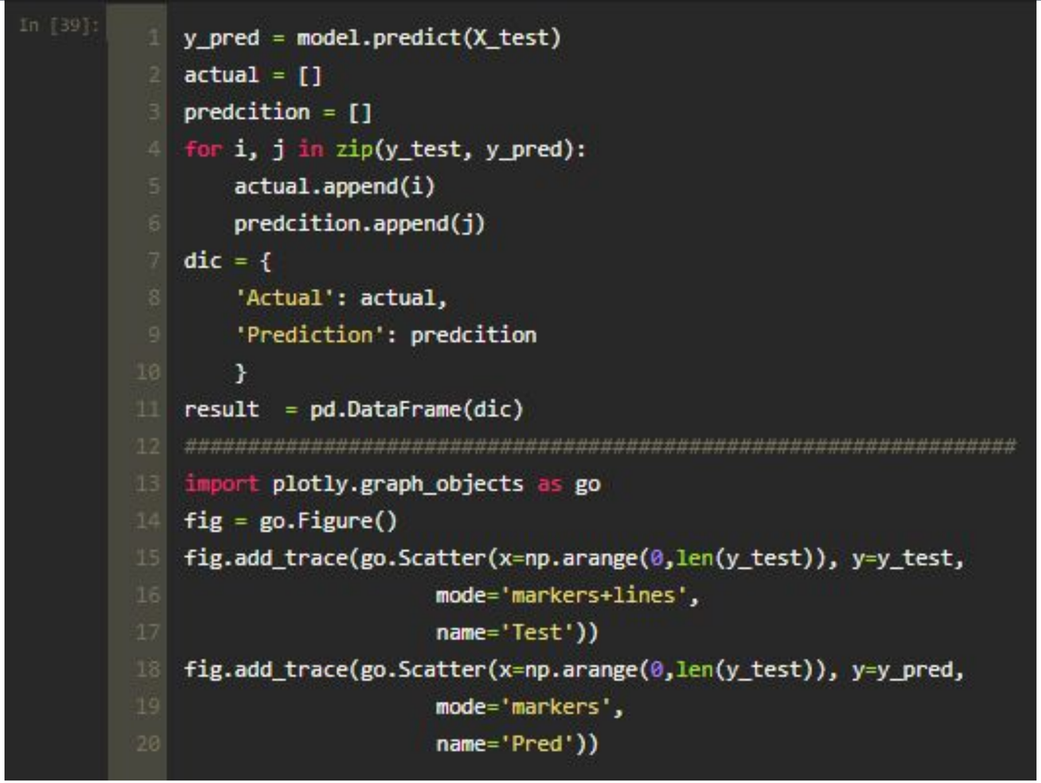


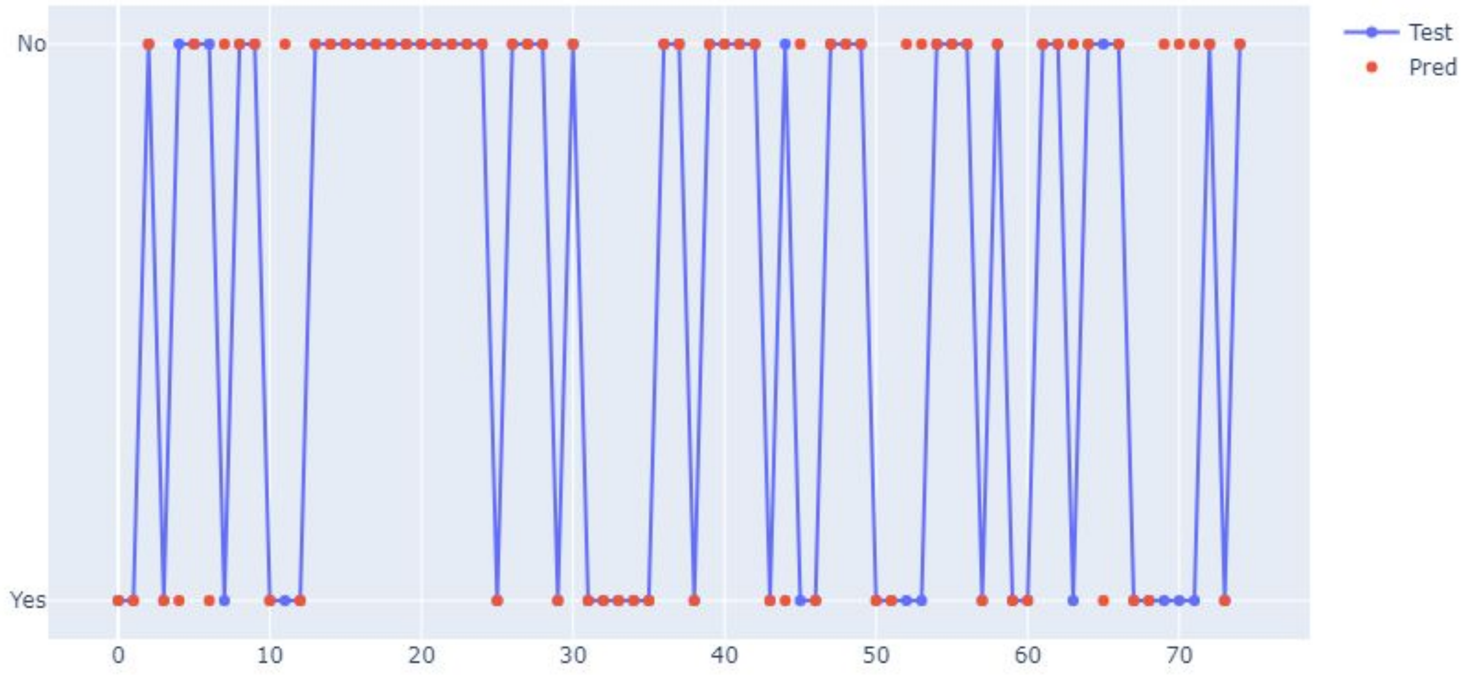
```
In [37]:
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
In [38]:
# training the LogisticRegression model with Training data
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

8. Prediction and plotting ‘predicted and actual’ graph

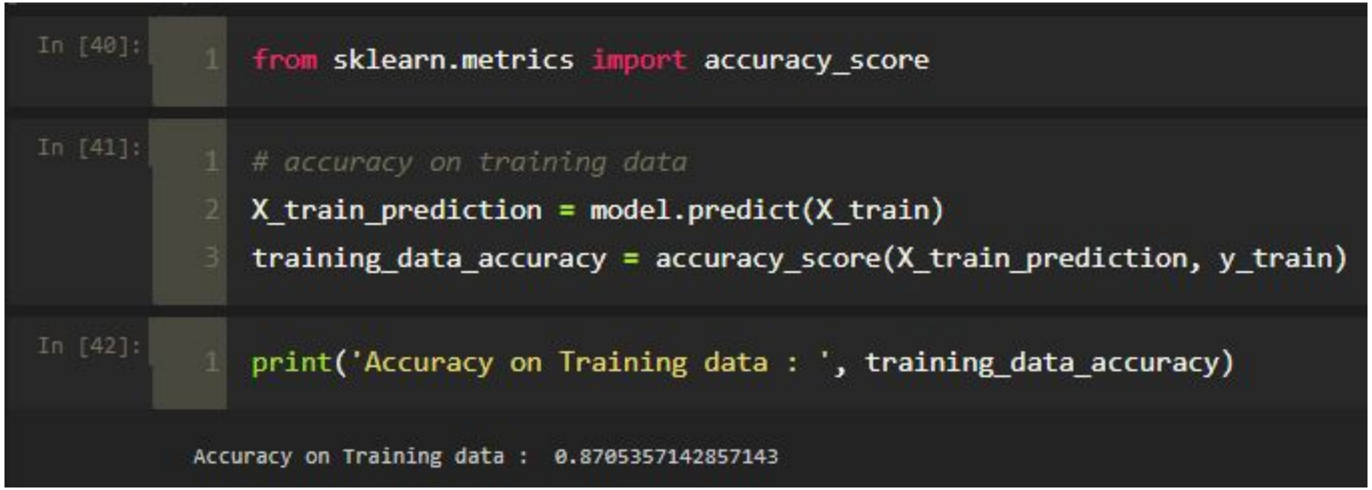

```
In [39]:
y_pred = model.predict(X_test)
actual = []
predcition = []
for i, j in zip(y_test, y_pred):
    actual.append(i)
    predcition.append(j)
dic = {
    'Actual': actual,
    'Prediction': predcition
    }
result  = pd.DataFrame(dic)
###################################################################
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter(x=np.arange(0,len(y_test)), y=y_test,
                    mode='markers+lines',
                    name='Test'))
fig.add_trace(go.Scatter(x=np.arange(0,len(y_test)), y=y_pred,
                    mode='markers',
                    name='Pred'))
```


No
Yes
0
10
20
30
40
50
60
70
Test
Pred

- The red dots represent the predicted values that are either 0 or 1 and the blue line & and dot represents the actual value of that particular patient.

- In the places where the red dot and blue dot do not overlap are the wrong predictions and where the both dots overlap those are the right predicted values.

9. Model **Evaluation**

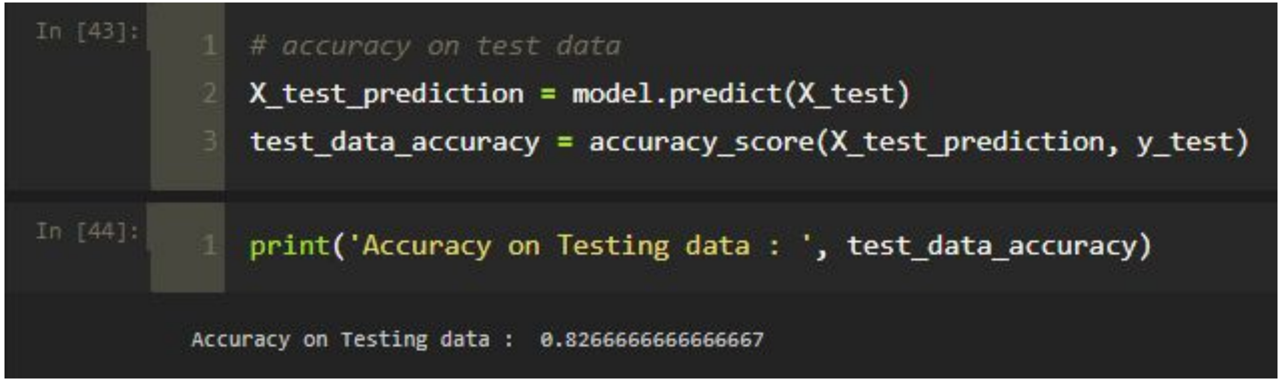


```
In [40]:
from sklearn.metrics import accuracy_score
```

```
In [41]:
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, y_train)
```

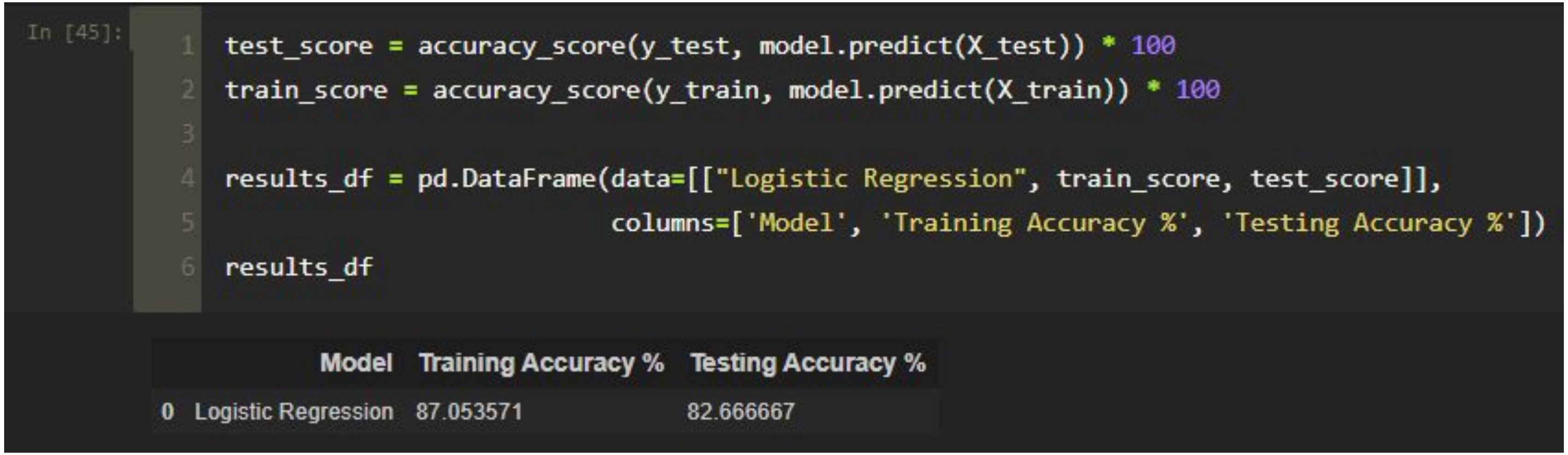
```
In [42]:
print('Accuracy on Training data : ', training_data_accuracy)

Accuracy on Training data :  0.8705357142857143
```

Accuracy on Training data : 0.8705357142857143



```
In [43]:  # accuracy on test data
          X_test_prediction = model.predict(X_test)
          test_data_accuracy = accuracy_score(X_test_prediction, y_test)

In [44]:  print('Accuracy on Testing data : ', test_data_accuracy)

          Accuracy on Testing data :  0.8266666666666667
```

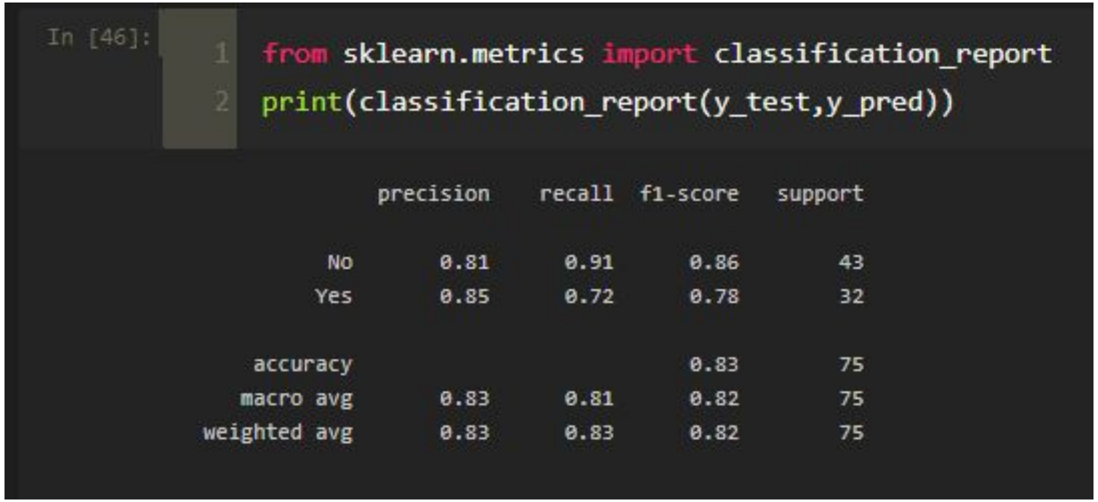
Accuracy on Testing data : 0.8266666666666667



```
In [45]:
test_score = accuracy_score(y_test, model.predict(X_test)) * 100
train_score = accuracy_score(y_train, model.predict(X_train)) * 100

results_df = pd.DataFrame(data=[["Logistic Regression", train_score, test_score]],
                          columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df
```

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	87.053571	82.666667

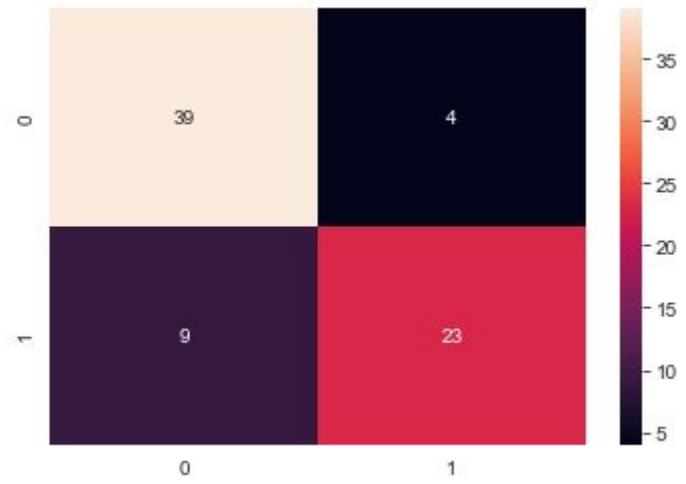
Classification Report:



```
In [46]:
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.81      0.91      0.86        43
         Yes       0.85      0.72      0.78        32

    accuracy                           0.83        75
   macro avg       0.83      0.81      0.82        75
weighted avg       0.83      0.83      0.82        75
```

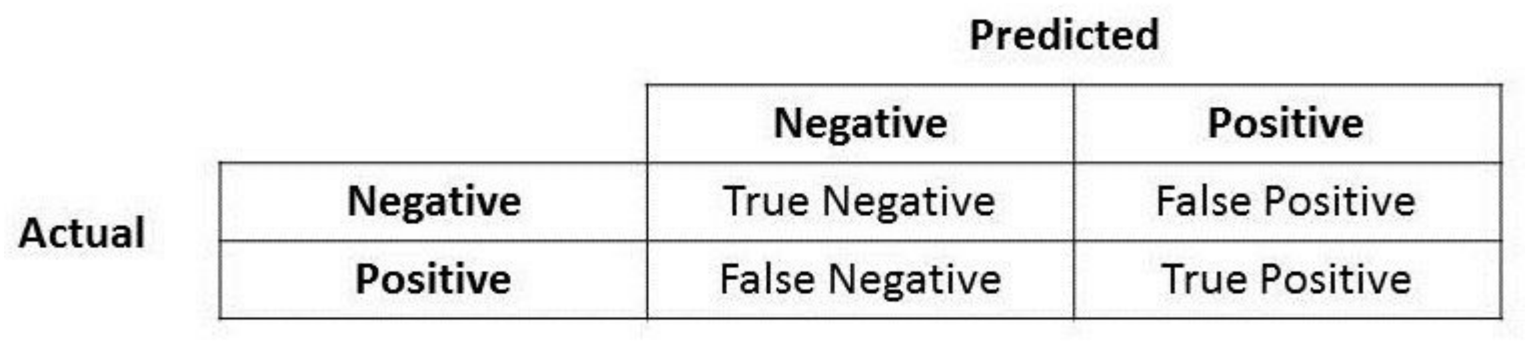
Confusion Matrix



Thank You!

- By Mohammed Faraz Hussain
  - Terna Engineering College, Nerul
  - ID: TU3F1920077
  - Roll No.: TE-B-24

Confusion Matrix



		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Confusion Matrix:

True Positive (TP) — model correctly predicts the positive class (prediction and actual both are positive). In the above example, 10 people who have tumors are predicted positively by the model.

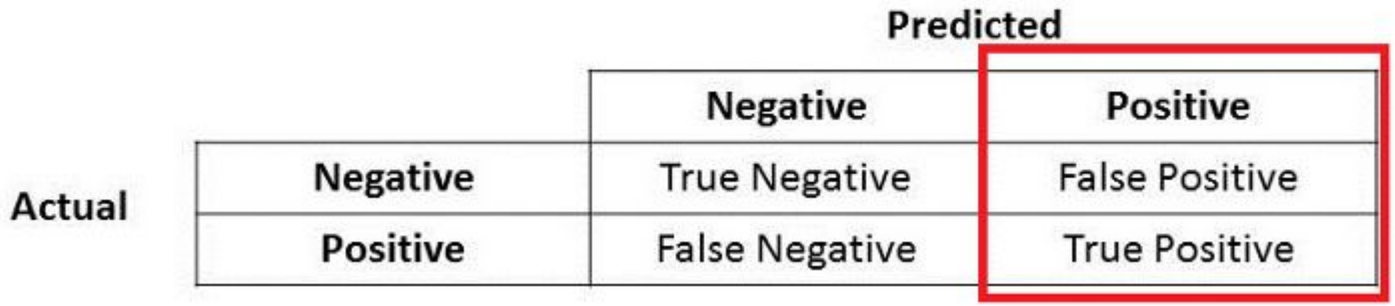
True Negative (TN) — model correctly predicts the negative class (prediction and actual both are negative). In the above example, 60 people who don't have tumors are predicted negatively by the model.

False Positive (FP) — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). In the above example, 22 people are predicted as positive of having a tumor, although they don't have a tumor. FP is also called a TYPE I error.

False Negative (FN) — model wrongly predicts the positive class (predicted-negative, actual-positive). In the above example, 8 people who have tumors are predicted as negative. FN is also called a TYPE II error.

Precision:

Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.

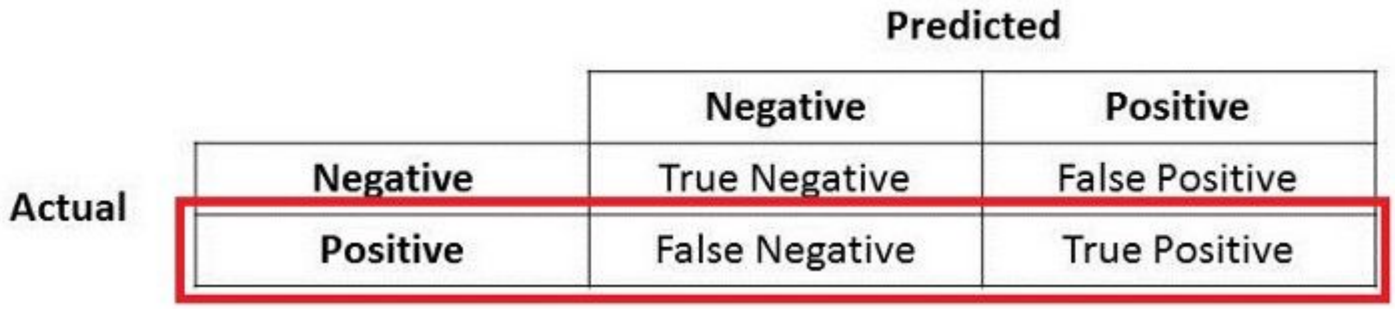


		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

$$= \frac{\textit{True Positive}}{\textit{Total Predicted Positive}}$$

Recall:
Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive)



		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

F1 Score:
F1 Score is needed when you want to seek a balance between Precision and Recall.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$