Predicting Heart disease using Machine Learning

This report tells discusses various Python-based ML and Data Science libraries in an attempt to build a Machine Learning model capable of predicting whether or not someone has heart disease based on their medical attributes.

We're going to take the following approach:

1. Problem Definition
2. Retrieving Data
3. Understanding Features
4. Data Preparation and its tools
5. Exploratory Data Analysis
6. Modelling
7. Model Evaluation

1. **Problem Definition**

Given clinical parameters about a patient, can we predict whether or not they have heart disease?
Our aim is to reach the model accuracy of more than 85% . If the model scores better than 85%, we will select the model

2. **Retrieving Data**

The original data came from the Cleveland data from the UCI Machine Learning Repository and also a version of it available on Kaggle.

https://www.kaggle.com/ronitf/heart-disease-uci?select=heart.csv

3. **Understanding Features**

1. **age:** displays the age of the individual.
2. **sex:** displays the gender of the individual using the following format :
    - 1 = male

- 0 = female

3. **cp (Chest-Pain Type):** displays the type of chest-pain experienced by the individual using the following format :
    - 0 = typical angina
    - 1 = atypical angina
    - 2= non — anginal pain
    - 3 = asymptotic
4. **trestbps(Resting Blood Pressure):** displays the resting blood pressure value of an individual in mmHg (unit)
5. **chol(Serum Cholestrol):** displays the serum cholesterol in mg/dl (unit)
6. **fbs (Fasting Blood Sugar):** compares the fasting blood sugar value of an individual with 120mg/dl.
    - If fasting blood sugar > 120mg/dl then : 1 (true) else : 0 (false)
7. **restecg (Resting ECG):** displays resting electrocardiographic results
    - 0 = normal
    - 1 = having ST-T wave abnormality
    - 2 = left ventricular hyperthrophy
8. **thalach(Max Heart Rate Achieved):** displays the max heart rate achieved by an individual.
9. **exang (Exercise induced angina):**
    - 1 = yes
    - 0 = no
10. **oldpeak (ST depression induced by exercise relative to rest):** displays the value which is an integer or float.
11. **slope (Peak exercise ST segment) :**
    - 0 = upsloping
    - 1 = flat
    - 2 = downsloping
12. **ca (Number of major vessels (0–3) colored by flourosopy):** displays the value as integer or float.
13. **thal :** displays the thalassemia (is an inherited blood disorder that causes your body to have less hemoglobin than normal) :
    - 0 = normal
    - 1 = fixed defect
    - 2 = reversible defect
14. **target (Diagnosis of heart disease):** Displays whether the individual is suffering from heart disease or not :
    - 0 = absence
    - 1 = present.

4. **Data Preparation and its tools**

Pandas & Numpy for Data Analysis and Manipulation

Matplotlib and Seaborn for Data Visualisation

Scikit-Learn for the Modelling and Evaluation

Importing the dataset

```
In [6]: df=pd.read_excel('/content/heart-disease.xlsx')
```

Shape of the dataset (Rows, Columns)

```
In [7]: df.shape
Out[7]: (303, 14)
```

Head of the dataset

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

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
# Import all the tools we need

# Regular EDA(Exploratory data analysis) and plotting libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#We want our plots to appear inside the notebook
%matplotlib inline

#importing package to prepare a report on pandas

# Models from Scikit-Learn
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

# Model Evaluations
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve,plot_roc_curve

#Ignoring the warnings
import warnings
warnings.filterwarnings("ignore")
```

Renaming columns

```
df.rename(columns ={'age':'Age','sex':'Sex','cp':'Chest_pain','trestbps':'Resting_blood_pressure','chol':'Cholesterol','fbs':'Fas
                    'restecg':'ECG_results','thalach':'Maximum_heart_rate','exang':'Exercise_induced_angina','oldpeak':'ST_depres
                   'thal':'Thalassemia_types','target':'Heart_disease'}, inplace = True)
```

```
# View of the Renamed Dataframe
df.head()
```

	Age	Sex	Chest_pain	Resting_blood_pressure	Cholesterol	Fasting_blood_sugar	ECG_results	Maximum_heart_rate	Exercise_induced_angina	ST_depressic
0	63	1	3	145	233	1	0	150	0	2
1	37	1	2	130	250	0	1	187	0	3
2	41	0	1	130	204	0	0	172	0	1
3	56	1	1	120	236	0	1	178	0	0
4	57	0	0	120	354	0	1	163	1	0

5. **Exploratory Data Analysis**

Information about the data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
Age                        303 non-null int64
Sex                        303 non-null int64
Chest_pain                 303 non-null int64
Resting_blood_pressure     303 non-null int64
Cholesterol                303 non-null int64
Fasting_blood_sugar        303 non-null int64
ECG_results                303 non-null int64
Maximum_heart_rate         303 non-null int64
Exercise_induced_angina    303 non-null int64
ST_depression              303 non-null float64
ST_slope                   303 non-null int64
Major_vessels              303 non-null int64
Thalassemia_types          303 non-null int64
Heart_disease              303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Are there any missing values?

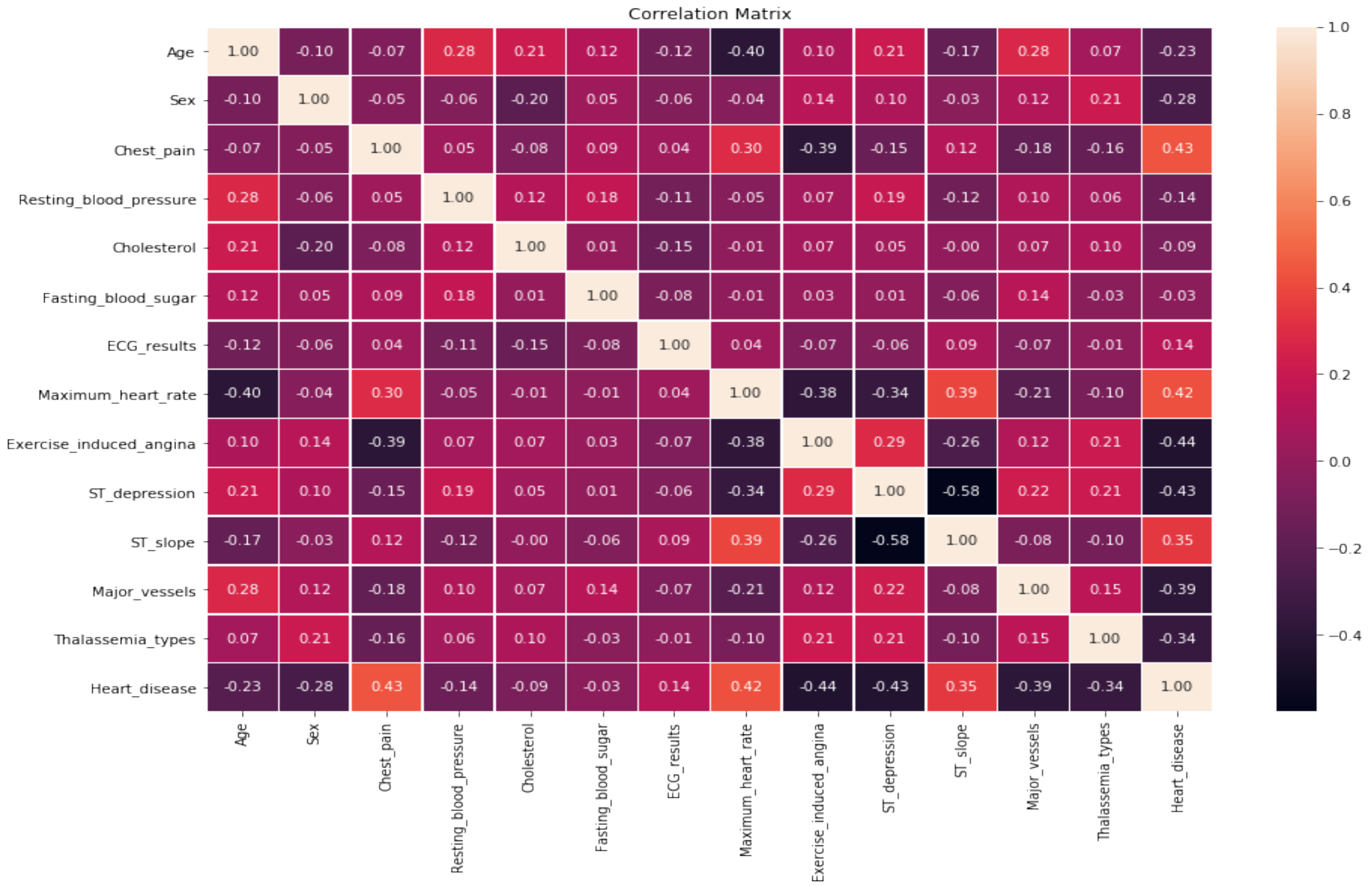
```
df.isna().sum()
```

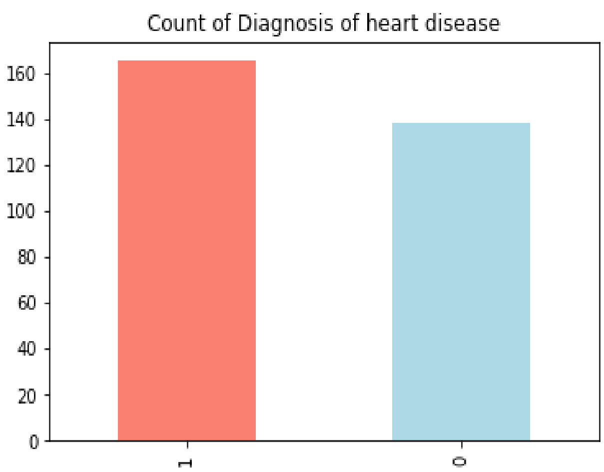
```
Age                        0
Sex                        0
Chest_pain                 0
Resting_blood_pressure     0
Cholesterol                0
Fasting_blood_sugar        0
ECG_results                0
Maximum_heart_rate         0
Exercise_induced_angina    0
ST_depression              0
ST_slope                   0
Major_vessels              0
Thalassemia_types          0
Heart_disease              0
dtype: int64
```

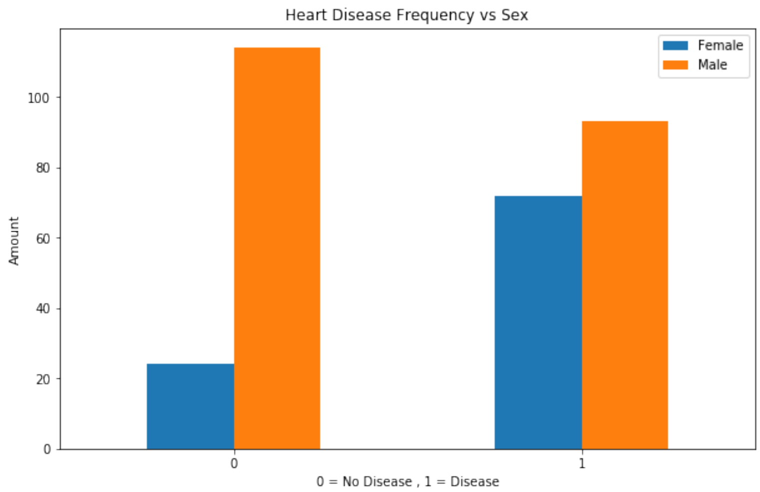
Description about the dataset

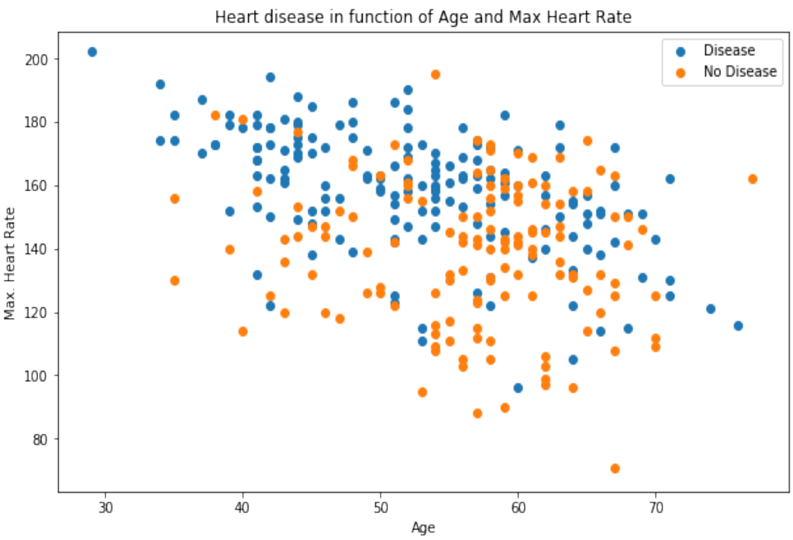
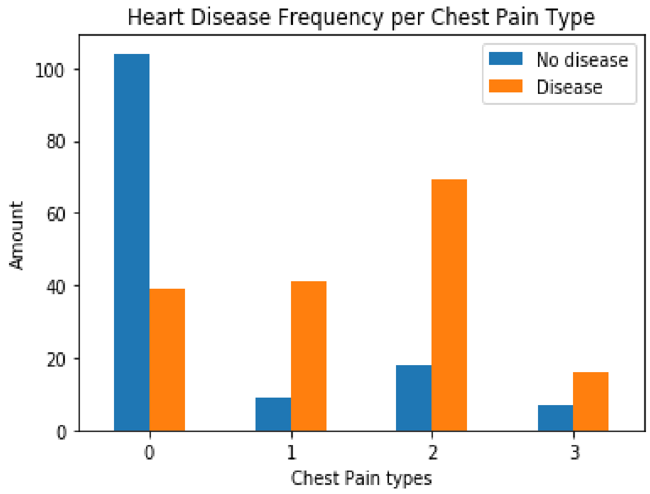
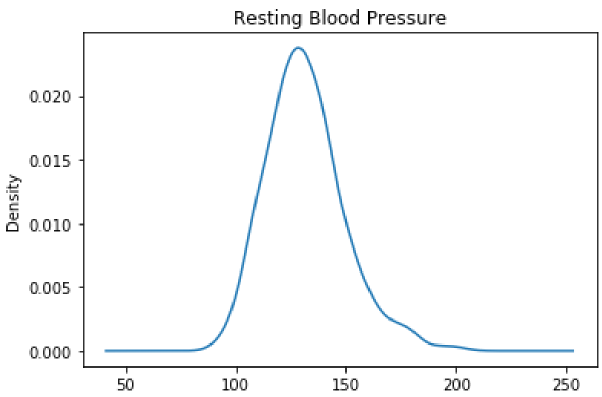
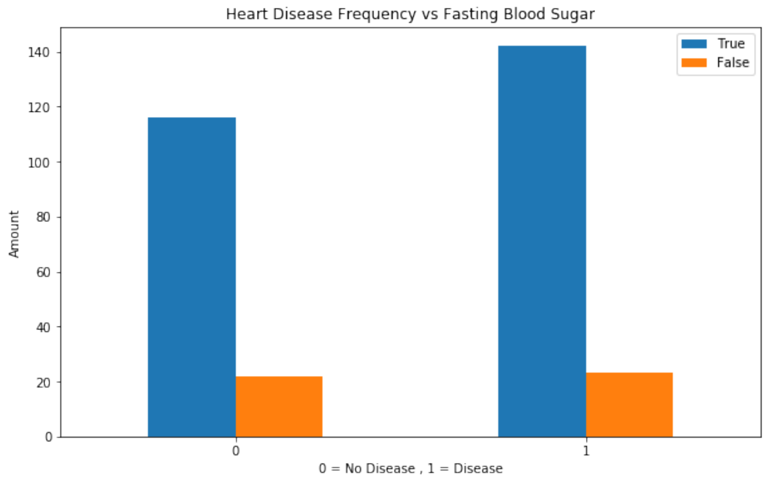
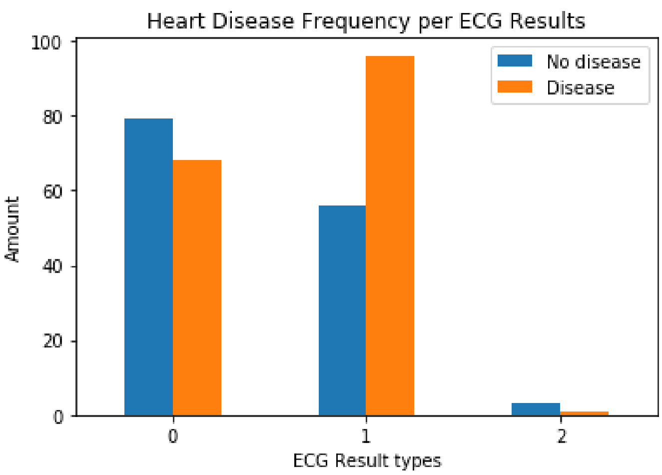
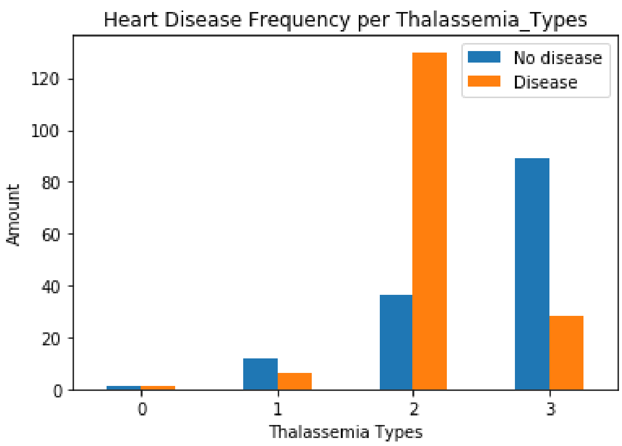
```
df.describe()
```

	Age	Sex	Chest_pain	Resting_blood_pressure	Cholesterol	Fasting_blood_sugar	ECG_results	Maximum_heart_rate
count	303.000000	303.000000	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	0.528053	149.646865
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000



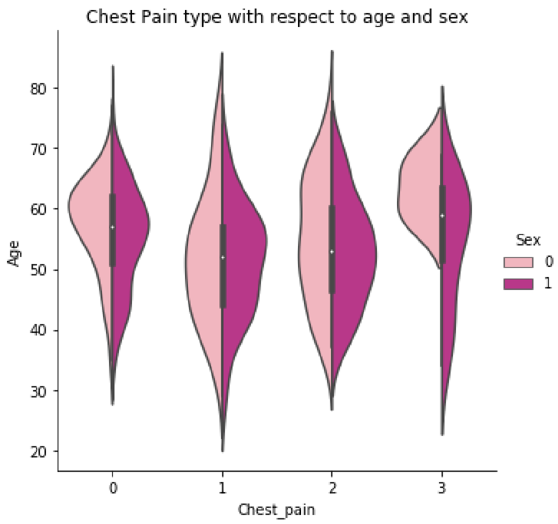


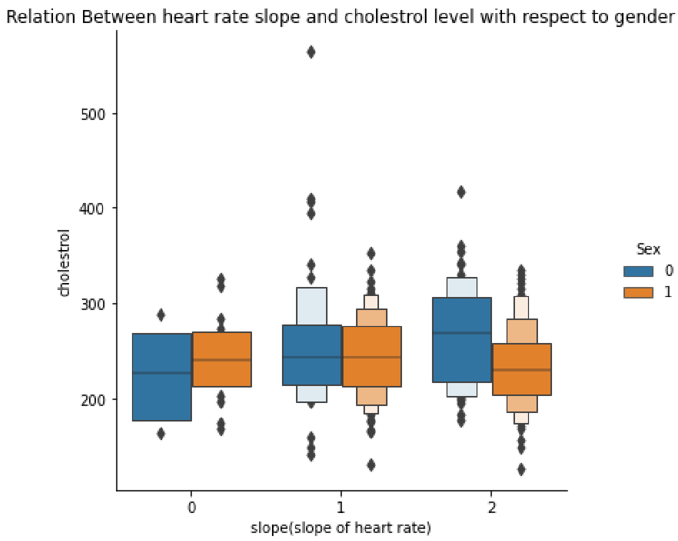



Heart disease in function of Age and Max Heart Rate
Disease
No Disease
Max. Heart Rate
Age

Heart Disease Frequency per Chest Pain Type
No disease
Disease
Amount
Chest Pain types

Resting Blood Pressure
Density

Heart Disease Frequency vs Fasting Blood Sugar
True
False
Amount
0 = No Disease , 1 = Disease

Heart Disease Frequency per ECG Results
No disease
Disease
Amount
ECG Result types

Heart Disease Frequency per Thalassemia_Types
No disease
Disease
Amount
Thalassemia Types

Relation Between Maximum heart rate and chest pain types with respect to gender
maximum heart rate
Chest_pain
gender

Relation Between resting chest pain types and Blood Pressure Rate with respect to gender
Blood Pressure Rate
Sex
chest pain types





6. Modelling

We will experiment with the models, trying 3 different models and getting the results from them and comparing them later

Split data using Train-Test Split

```
X=df.drop('Heart_disease',axis=1)
y=df['Heart_disease']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Now we have got our data split into training and test sets, it is time to build a Machine Learning model.

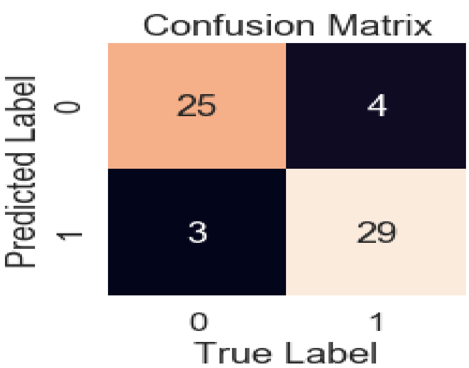
We will train it (find the patterns) on the training set.

And we will test it (use the patterns) on the test set.

We're going to try 3 different Machine Learning models:

1. Logistic Regression
2. K-Nearest Neighbours Classifier
3. Random Forest Classifier

1. Logistic Regression (Accuracy of 88.5%)



Classification Report

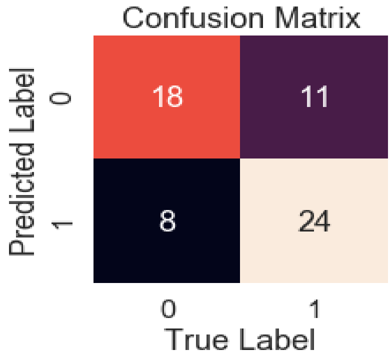
```
print(classification_report(y_test,lr_y_preds))
```

```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61
```

2. K-Nearest Neighbour Classifier (Accuracy of 68.8%)



Classification Report

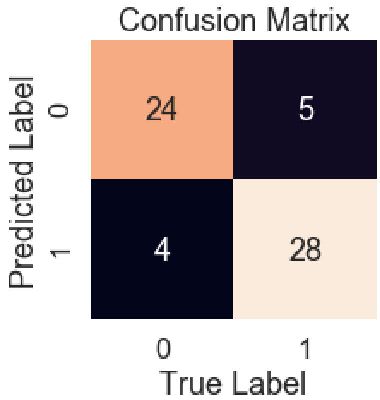
```
print(classification_report(y_test,knn_y_preds))
```

```
              precision    recall  f1-score   support

           0       0.69      0.62      0.65        29
           1       0.69      0.75      0.72        32

    accuracy                           0.69        61
   macro avg       0.69      0.69      0.69        61
weighted avg       0.69      0.69      0.69        61
```

3. Random Forest Classifier (Accuracy of 85.2%)



Classification Report

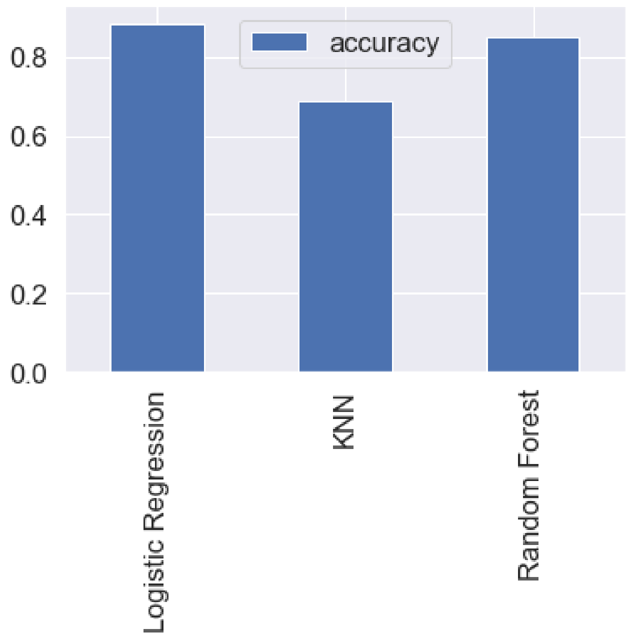
```
print(classification_report(y_test,rf_y_preds))
```

```
              precision    recall  f1-score   support

           0       0.80      0.83      0.81        29
           1       0.84      0.81      0.83        32

    accuracy                           0.82        61
   macro avg       0.82      0.82      0.82        61
weighted avg       0.82      0.82      0.82        61
```

Comparison Among Models




```
#Based on accuracy
model_compare=pd.DataFrame(model_scores,index=['accuracy'])
model_compare
```

	Logistic Regression	KNN	Random Forest
accuracy	0.885246	0.688525	0.852459

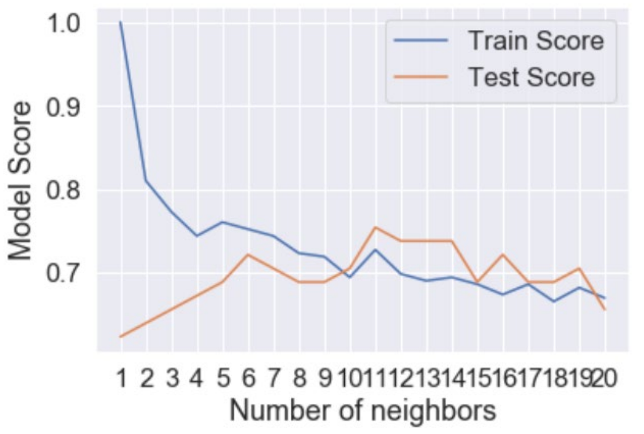
Now we have got baseline model...and we know a model's first prediction aren't always based our next steps off. What should we do?

Let's look at the following:

- HyperParameter tuning
- Feature Importance
- Confusion Matrix
- Cross-Validation
- Precision
- Recall
- F1-Score
- Classification Report
- ROC Curve
- Area under the curve(AUC)

HyperParameter Tuning (Manually) - K-Nearest Neighbour

```
Maximum KNN score on the test data: 75.41
```



After KNN tuning also, KNN model got improved but still it is not predicting better than Random Forest and Logistic Regression. So we will discard it

HyperParameter tuning with Randomized Search CV

We are going to tune:

- Logistic Regression
- Random Forest Classifier

```
#Create a hyperparameter grid for Logistic Regression

log_reg_grid={"C":np.logspace(-4,4,20),
              "solver":['liblinear']}

#Create a hyperparameter grid for RandomForestClassifier(it is recommended to use continuous distributions for hyperparameter
# tuning for RandomForestClassifier i.e. why using "arange")

rf_grid={"n_estimators":np.arange(10,1000,50),
         "max_depth":[None,3,5,10],
         "min_samples_split":np.arange(2,20,2),
         "min_samples_leaf":np.arange(1,20,2)}
```

Logistic Regression

```
#Tune Logistic Regression
np.random.seed(42)

#Setup random hyperparameter search for Logistic Regression
rs_log_reg=RandomizedSearchCV(LogisticRegression(),
                              param_distributions=log_reg_grid,
                              cv=5,
                              n_iter=20,
                              verbose=2)

#Fit random hyperparameter search for Logistic Regression
rs_log_reg.fit(X_train,y_train)
```

```
#checking the best parameters we got from RandomizedSearchCV

rs_log_reg.best_params_
```

```
{'solver': 'liblinear', 'C': 0.23357214690901212}
```

```
#Finding the score

rs_log_reg.score(X_test,y_test)
```

```
0.8852459016393442
```

Random Forest

```
#Setup Random Seed
np.random.seed(42)

#Setup random hyperparameter search for Logistic Regression, the combinations are many so randomly try 20.
rs_rf=RandomizedSearchCV(RandomForestClassifier(),
                         param_distributions=rf_grid,
                         cv=5,
                         n_iter=20,
                         verbose=2)

#Fit random hyperparameter search for Logistic Regression
rs_rf.fit(X_train,y_train)
```

```
rs_rf.best_params_
```

```
{'n_estimators': 210,
 'min_samples_split': 4,
 'min_samples_leaf': 19,
 'max_depth': 3}
```

```
rs_rf.score(X_test,y_test)
```

```
0.8688524590163934
```

So now we have done RandomizedSearchCV, we will **eliminate RandomForest** as it's score is not much as compared to logistic Regression

HyperParameter tuning with GridSearchCV

Since our Logistic Regression model provides the best scores so far, we will try and improve it again using GridSearchCV.

```
#Tune Logistic Regression
np.random.seed(42)

#Setup random hyperparameter search for Logistic Regression
gs_log_reg=GridSearchCV(LogisticRegression(),
                        param_grid=log_reg_grid,
                        cv=5,
                        verbose=2)

#Fit random hyperparameter search for Logistic Regression
gs_log_reg.fit(X_train,y_train)
```

```
gs_log_reg.best_params_
```

```
{'C': 0.20433597178569418, 'solver': 'liblinear'}
```

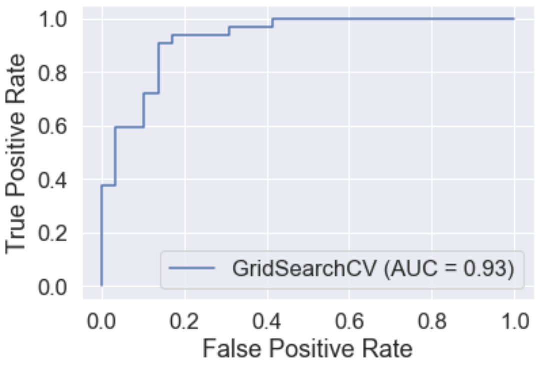
```
gs_log_reg.score(X_test,y_test)
```

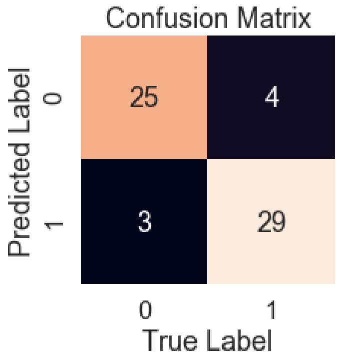
```
0.8852459016393442
```

7. Evaluating our tuned Logistic Regression model, beyond accuracy

- ROC curve and AUC score
- Confusion matrix
- Classification report
- precision
- recall
- f1-score
- Cross Validation

ROC and AUC





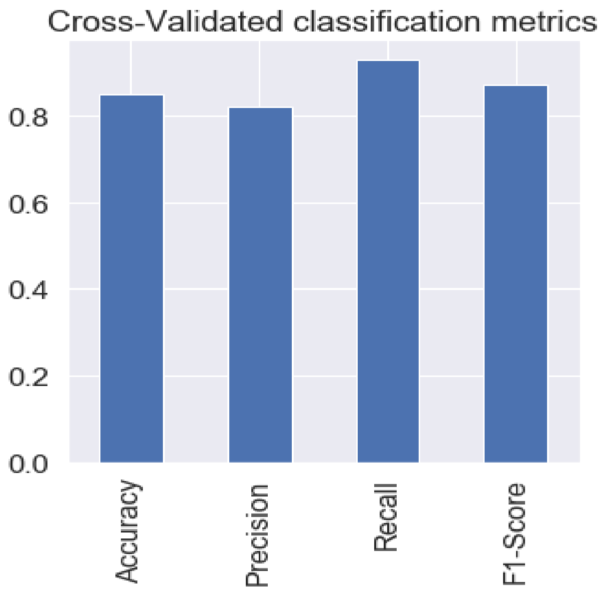
Classification Report

```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61
```

Cross Validation

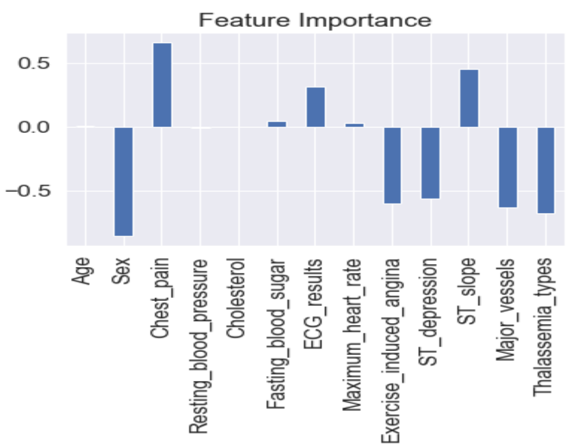


Accuracy	Precision	Recall	F1-Score
0.847978	0.821587	0.927273	0.87054

Feature Importance

Feature Importance is another way of asking, "which features contributed most to the outcomes of the model and how did they contribute?" Finding feature importance is different for each machine learning model.

Let's find the feature importance for our Logistic Regression model




```
{'Age': 0.003167280199343156З,
 'Sex': -0.8604465072345515,
 'Chest_pain': 0.6606704082033799,
 'Resting_blood_pressure': -0.01156993168080875,
 'Cholesterol': -0.001663744504776871,
 'Fasting_blood_sugar': 0.043861071652469864,
 'ECG_results': 0.31275846822418324,
 'Maximum_heart_rate': 0.024593613737779126,
 'Exercise_induced_angina': -0.6041308000615746,
 'ST_depression': -0.5686280368396555,
 'ST_slope': 0.4505162797258308,
 'Major_vessels': -0.636098967608622З,
 'Thalassemia_types': -0.6766337263029825}
```