UDACITY MACHINE LEARNING NANODEGREE – 2020

CAPSTONE PROJECT REPORT

PREDICTING THE PRESENCE OF HEART DISEASE IN PATIENTS

HAVISHA KALWAD
MAY, 2020

PROJECT OVERVIEW

Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease; heart rhythm problems (arrhythmias); and heart defects you're born with (congenital heart defects), among others. The term "heart disease" is often used interchangeably with the term "cardiovascular disease." Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart's muscle, valves or rhythm, also are considered forms of heart disease.

Heart disease statistics in United States of America:

- Heart disease is the leading cause of death for men, women, and people of most racial and ethnic groups in the United States.
- One person dies every 37 seconds in the United States from cardiovascular disease.
- About 647,000 Americans die from heart disease each year—that’s 1 in every 4 deaths.
- Heart disease costs the United States about $219 billion each year from 2014 to 2015. This includes the cost of health care services, medicines, and lost productivity due to death.

PROBLEM STATEMENT

- Of all the applications of machine-learning, diagnosing any serious disease using a black box is always going to be a hard sell. If the output from a model is the particular course of treatment (potentially with side-effects), or surgery, or the absence of treatment, people are going to want to know why. This dataset gives a number of variables along with a target condition of having or not having heart disease. Using this dataset I wish to build a model

that helps predict the presence of heart disease in any new patient information which is input into this model.

- The goal of this project is: Given clinical parameters about a patient, can we predict whether or not they have heart disease? Which model would be the best for prediction?

METRICS

The metrics that you choose to evaluate your machine learning model is very important. Choice of metrics influences how the performance of machine learning algorithms is measured and compared. The metrics I have used in my model for performance evaluation are: Accuracy, Recall, Precision and F1-Score from the confusion matrix.

CONFUSION MATRIX

The Confusion matrix is one of the most intuitive and easiest (unless of course, you are not confused)metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes.

Before diving into what the confusion matrix is all about and what it conveys, Let's say we are solving a classification problem where we are predicting whether a person is having cancer or not. Let's give a label of to our target variable: *1: When a person is having cancer 0: When a person is NOT having cancer.*

Alright! Now that we have identified the problem, the confusion matrix, is a table with two dimensions ("Actual" and "Predicted"), and sets of "classes" in both dimensions. Our Actual classifications are columns and Predicted ones are Rows.

Metric	Formula	Evaluation Focus
Accuracy	$ACC = \frac{TP+TN}{TP+TN+FP+FN}$	Overall effectiveness of a classifier
Error rate	$ERR = \frac{FP+FN}{TP+TN+FP+FN}$	Classification error
Precision	$PRC = \frac{TP}{TP+FP}$	Class agreement of the data labels with the positive labels given by the classifier
Sensitivity Specificity	$SNS = \frac{TP}{TP+FN}$ $SPC = \frac{TN}{TN+FP}$	Effectiveness of a classifier to identify positive labels How effectively a classifier identifies negative labels
ROC	$ROC = \frac{\sqrt{SNS^2+SPC^2}}{\sqrt{2}}$	Combined metric based on the Receiver Operating Characteristic (ROC) space [53]
F_1 score	$F_1 = 2\frac{PRC \cdot SNS}{PRC+SNS}$	Combination of precision (PRC) and sensitivity (SNS) in a single metric
Geometric Mean	$GM = \sqrt{SNS \cdot SPC}$	Combination of sensitivity (SNS) and specificity (SPC) in a single metric

		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence = Σ Condition positive / Σ Total population	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV, Precision) = Σ True positive / Σ Test outcome positive	False discovery rate (FDR) = Σ False positive / Σ Test outcome positive
	Test outcome negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = Σ False negative / Σ Test outcome negative	Negative predictive value (NPV) = Σ True negative / Σ Test outcome negative
	Positive likelihood ratio (**LR+**) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = Σ True positive / Σ Condition positive	False positive rate (FPR, Fall-out) = Σ False positive / Σ Condition negative	Accuracy (ACC) = Σ True positive + Σ True negative / Σ Total population	
	Negative likelihood ratio (**LR−**) = FNR/TNR	False negative rate (FNR) = Σ False negative / Σ Condition positive	True negative rate (TNR, Specificity, SPC) = Σ True negative / Σ Condition negative		
	Diagnostic odds ratio (**DOR**) = LR+/LR−				

DATA EXPLORATION

The dataset for this project shall be obtained from the below mentioned links:

1. https://archive.ics.uci.edu/ml/datasets/Heart+Disease
2. https://www.kaggle.com/ronitf/heart-disease-uci

Both the above mentioned link denote the same dataset. The Kaggle version of the dataset was obtained from the UCI dataset.
This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. The names and social security numbers of the patients were removed from the database.
The features/columns used in this dataset are as follows:

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type
    - 0: Typical angina: chest pain related decrease blood supply to the heart
    - 1: Atypical angina: chest pain not related to heart
    - 2: Non-anginal pain: typically, esophageal spasms (non heart related)
    - 3: Asymptomatic: chest pain not showing signs of disease
4. trestbps - resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. chol - serum cholesterol in mg/dl
    - serum = LDL + HDL + .2 * triglycerides
    - above 200 is cause for concern
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
    - '>126' mg/dL signals diabetes
7. restecg - resting electrocardiographic results

- 0: Nothing to note
- 1: ST-T Wave abnormality
    - can range from mild symptoms to severe problems
    - signals non-normal heart beat
- 2: Possible or definite left ventricular hypertrophy
    - Enlarged heart's main pumping chamber

8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during exercise unhealthy heart will stress more
11. slope - the slope of the peak exercise ST segment
    - 0: Upsloping: better heart rate with exercise (uncommon)
    - 1: Flatsloping: minimal change (typical healthy heart)
    - 2: Downslopins: signs of unhealthy heart
12. ca - number of major vessels (0-3) colored by fluoroscopy
    - colored vessel means the doctor can see the blood passing through
    - the more blood movement the better (no clots)
13. thal - thallium stress result
    - 1,3: normal
    - 6: fixed defect: used to be defect but ok now
    - 7: reversable defect: no proper blood movement when exercising
14. target - have disease or not (1=yes, 0=no) (= the predicted attribute)

IMPORTING THE DATASET AND READING THE VALUES

1. Importing the dataset:

```
# Loading the dataset
df = pd.read_csv("heart.csv")

# Preview of the dataset
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.30	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.50	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.40	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.80	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.60	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.20	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.20	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.40	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.20	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.00	1	1	2	0

303 rows × 14 columns

Reading the .csv file into a pandas data frame.

We can see that the dataset has 14 columns and 303 rows.

2. Data Types of the features:

```
# Summary of the dataset
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
```

Most of the features are integers. Our target column used for classification is also an integer.

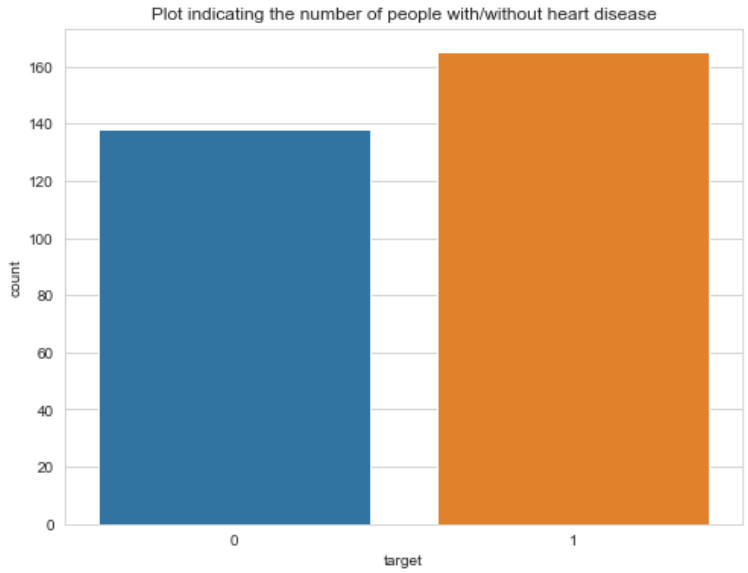
3. Understanding the statistics of the values in the dataset: (Applicable only for the numerical columns)

```
df.describe().transpose()
```

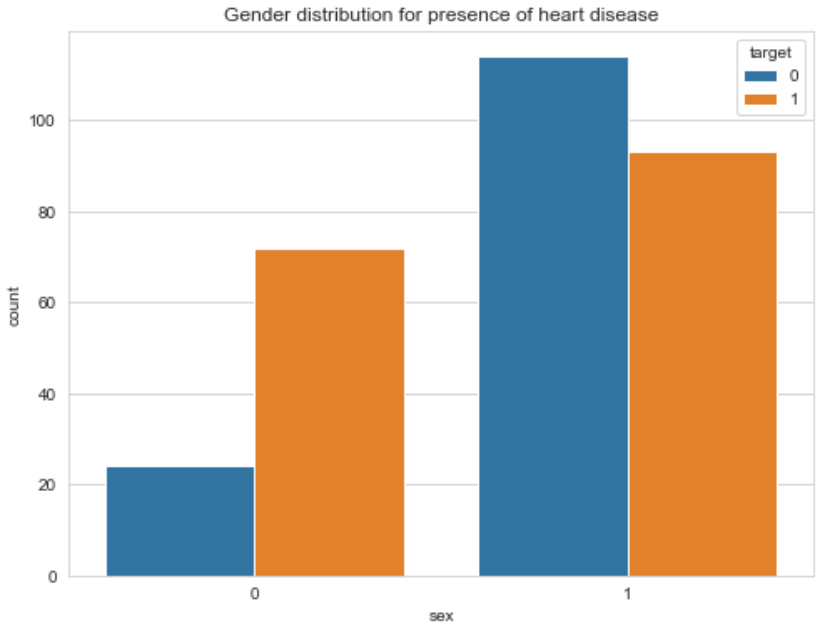
	count	mean	std	min	25%	50%	75%	max
age	303.00	54.37	9.08	29.00	47.50	55.00	61.00	77.00
sex	303.00	0.68	0.47	0.00	0.00	1.00	1.00	1.00
cp	303.00	0.97	1.03	0.00	0.00	1.00	2.00	3.00
trestbps	303.00	131.62	17.54	94.00	120.00	130.00	140.00	200.00
chol	303.00	246.26	51.83	126.00	211.00	240.00	274.50	564.00
fbs	303.00	0.15	0.36	0.00	0.00	0.00	0.00	1.00
restecg	303.00	0.53	0.53	0.00	0.00	1.00	1.00	2.00
thalach	303.00	149.65	22.91	71.00	133.50	153.00	166.00	202.00
exang	303.00	0.33	0.47	0.00	0.00	0.00	1.00	1.00
oldpeak	303.00	1.04	1.16	0.00	0.00	0.80	1.60	6.20
slope	303.00	1.40	0.62	0.00	1.00	1.00	2.00	2.00
ca	303.00	0.73	1.02	0.00	0.00	0.00	1.00	4.00
thal	303.00	2.31	0.61	0.00	2.00	2.00	3.00	3.00
target	303.00	0.54	0.50	0.00	0.00	1.00	1.00	1.00

EXPLORATORY VISUALIZATION

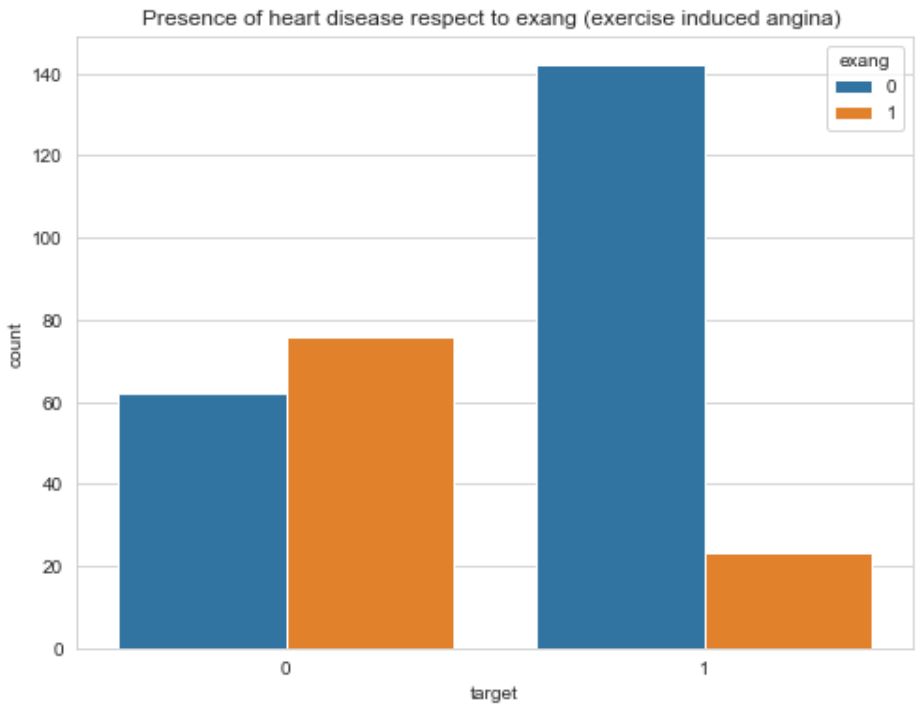
1. Understanding the target column:
   'Target' here refers to the presence of heart disease in the patient. 1: Heart disease present 0: Heart disease absent.



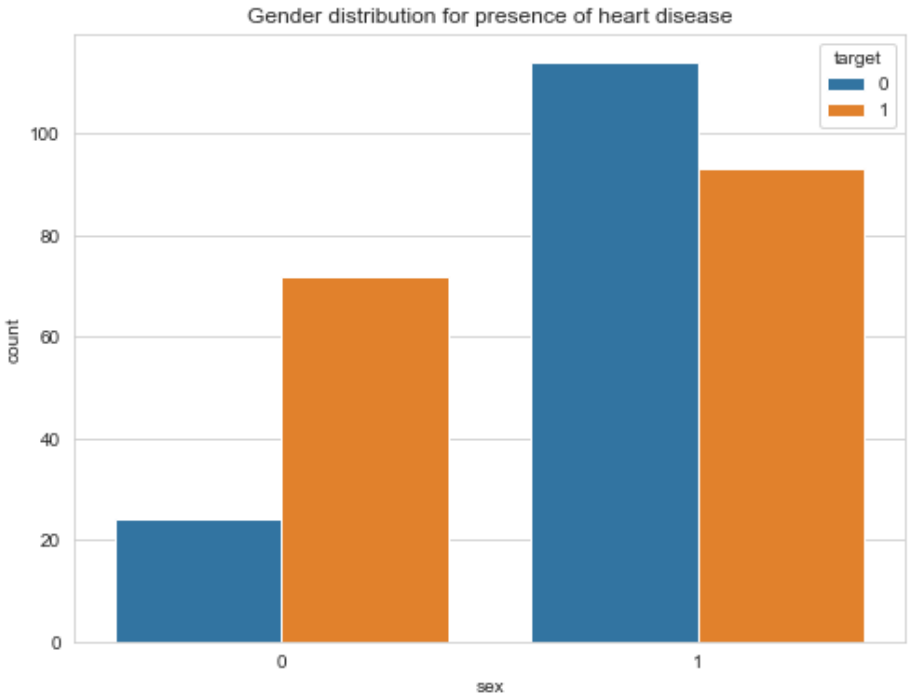
2. Distribution of heart disease based on gender
   Out of 96 females - 72 have heart disease and 24 do not have heart disease.
   Out of 207 males - 93 have heart disease and 114 do not have heart disease.



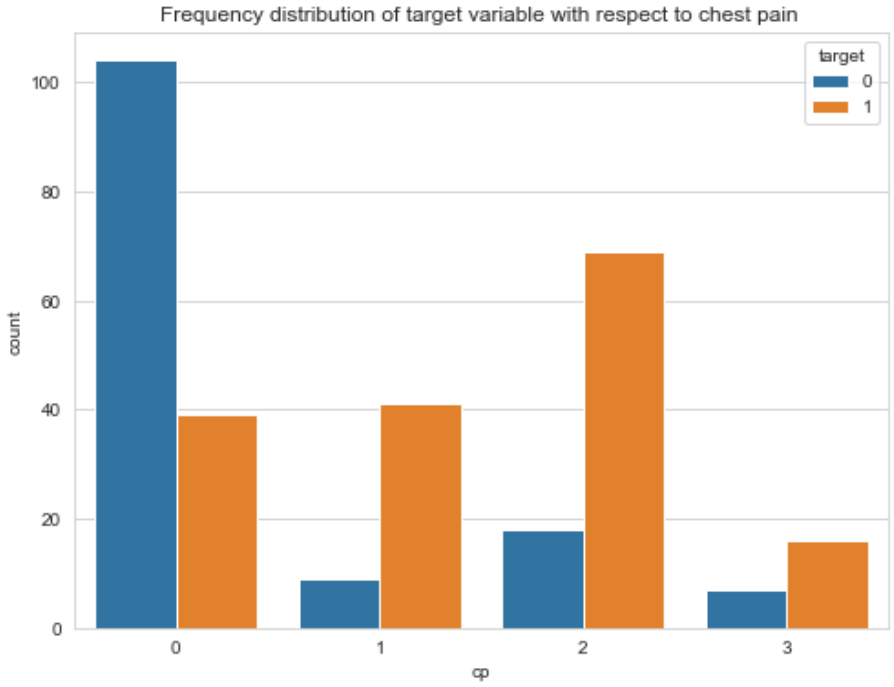
3. Visualize the target values distribution with respect to exang (exercise induced angina)



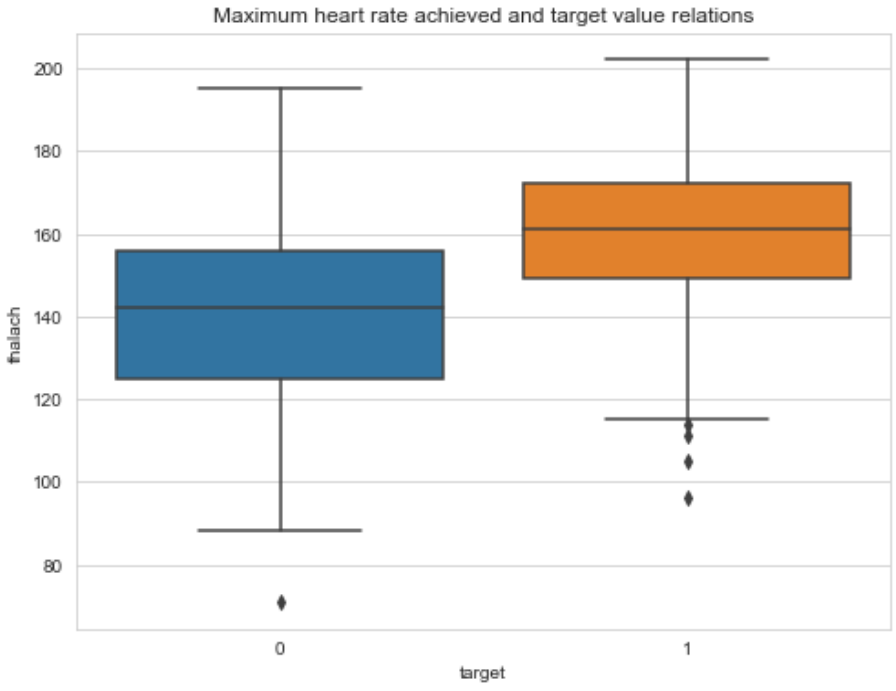
4. Visualize the target values distribution with respect to fbs (fasting blood sugar)



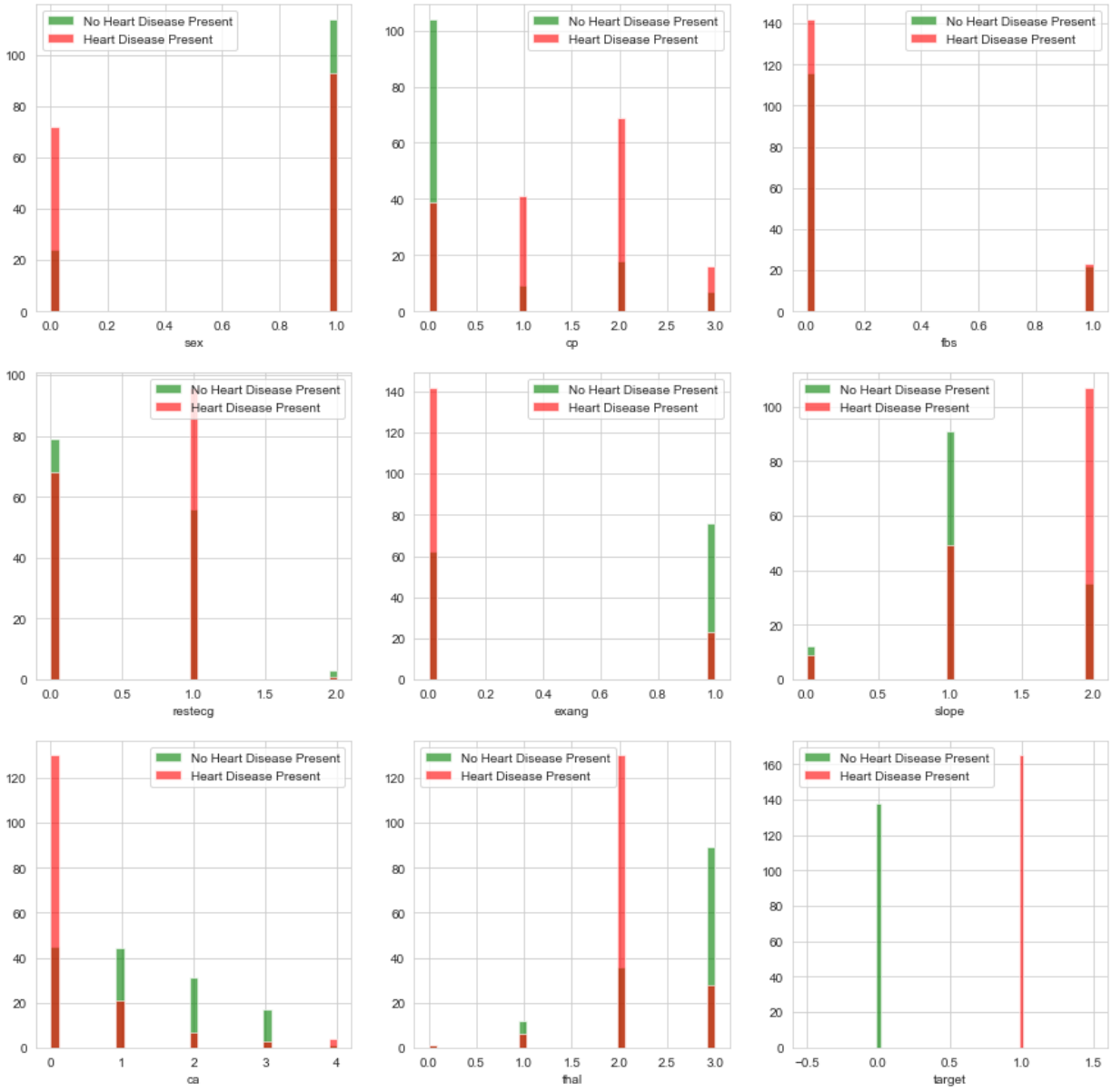
5. Relation between chest pain and presence of heart disease:



6. Relation between maximum heart rate and presence of heart disease: The boxplot confirms our finding that people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).



7. Relation between all categorical variables and presence of heart disease:
   Categorical Variables: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']



<u>Interpretation of the chart:</u>

- cp (Chest Pain) : People with cp = 1, 2, 3 are more likely to have heart disease than people with cp = 0.

- restecg (resting electrocardiographic results) : People with value 1 (signals non-normal heart beat, can range from mild symptoms to severe problems) are more likely to have heart disease.

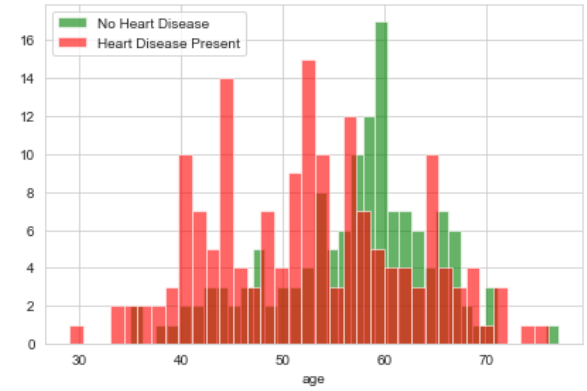
- exang (exercise induced angina) : People with value 0 (No ==> exercice induced angina) have heart disease more than people with value 1 (Yes ==> exercice induced angina)

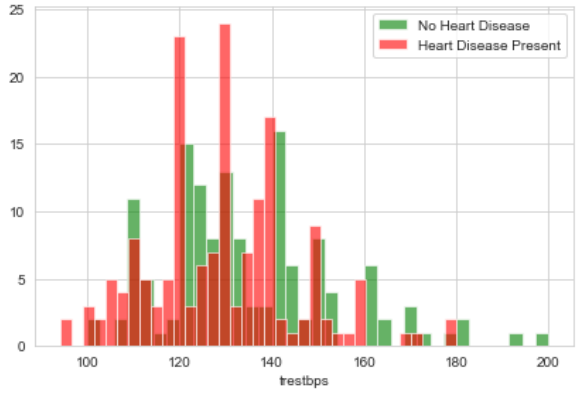
- slope (the slope of the peak exercise ST segment) : People with slope value equal to 2 (Downslopins: signs of unhealthy heart) are more likely to have heart disease than people with slope value equal to 0 (Upsloping: better heart rate with excercise) or 1 (Flatsloping: minimal change (typical healthy heart)).

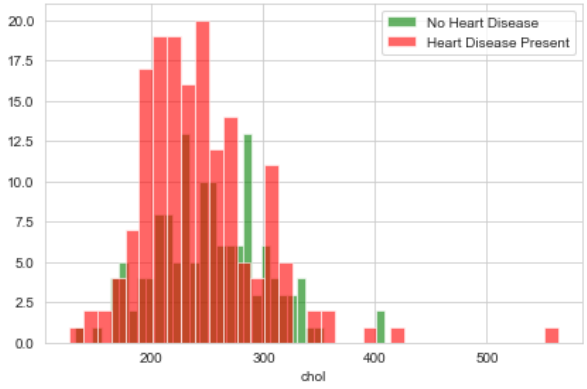
- ca (number of major vessels (0-3) colored by flourosopy) : the more blood movement the better so people with ca equal to 0 are more likely to have heart disease.

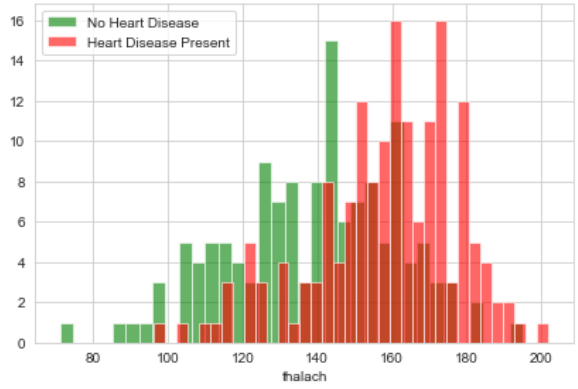
- thal (thalium stress result) : People with thal value equal to 2 (fixed defect: used to be defect but ok now) are more likely to have heart disease.

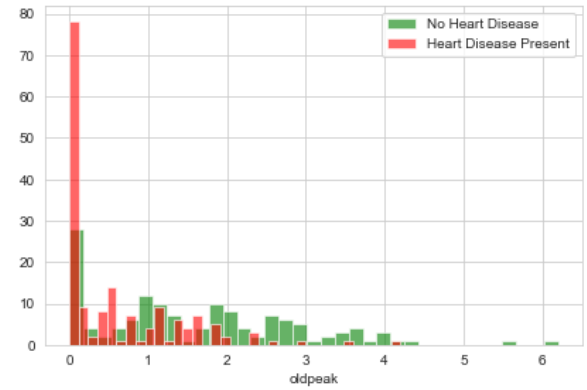
8. Relation between all numerical variables and presence of heart disease: Numerical Variables: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']











Interpretation of the chart:

- age: younger people seem to have heart disease more frequently than older people

- trestbps : resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern

- chol (serum cholestoral in mg/dl) : above 200 is cause for concern. People with high cholesterol are more likely to have a heart disease.

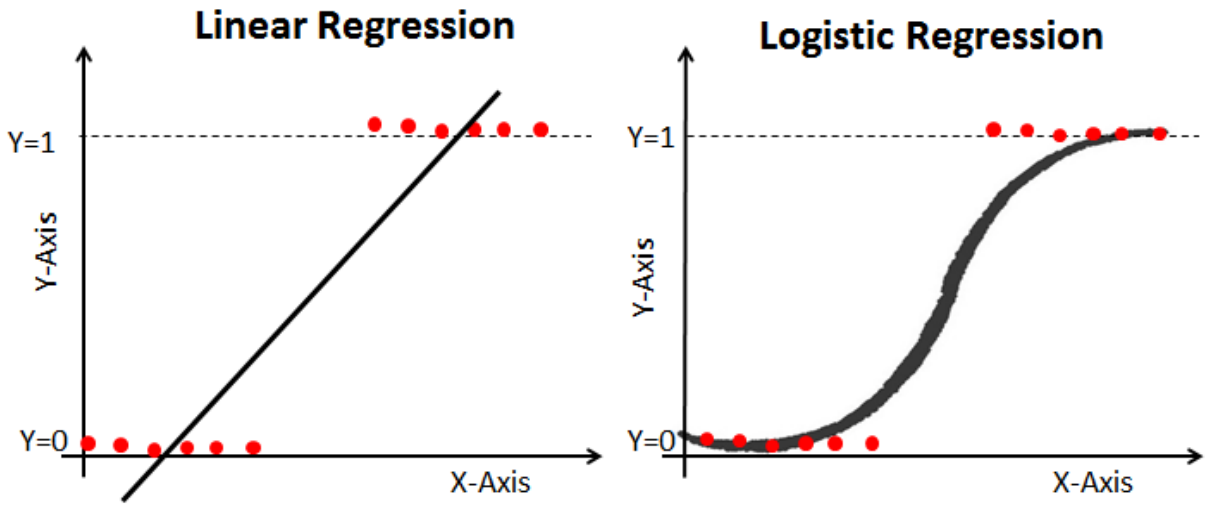
- thalach (maximum heart rate achieved) : People who acheived a maximum heart rate more than 140 are more likely to have heart disease.

- oldpeak: ST depression induced by exercise relative to rest looks at stress of heart during excercise unhealthy heart will stress more.

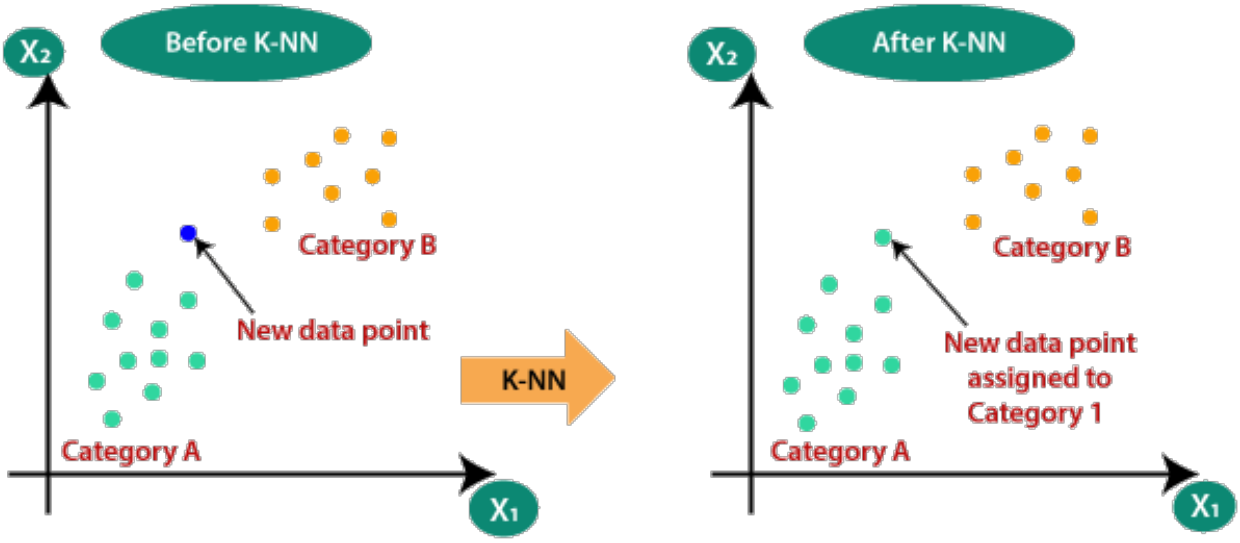
ALGORITHMS AND TECHNIQUES

I have used six machine learning algorithms for this model. They are as follows:

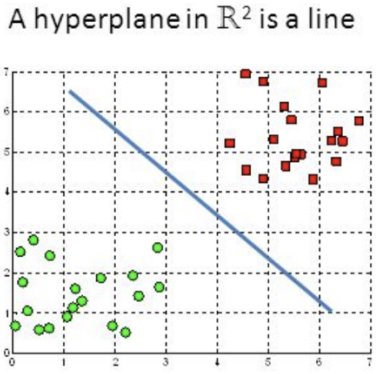
1. Logistic Regression: We identify problem as classification problem when independent variables are continuous in nature and dependent variable is in categorical form i.e. in classes like positive class and negative class. The real life example of classification example would be, to categorize the mail as spam or not spam, to categorize the tumor as malignant or benign and to categorize the transaction as fraudulent or genuine. All these problem's answers are in categorical form i.e. Yes or No. Such problems use Logistic Regression.

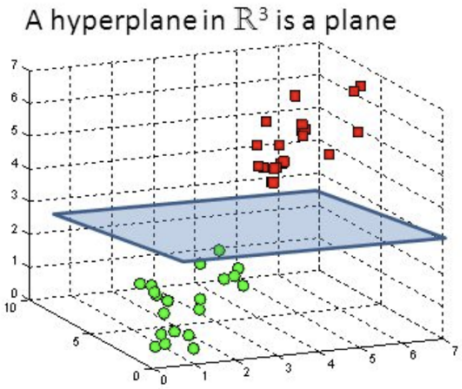


2. <u>K-Nearest Neighbors:</u> KNN captures the idea of similarity (sometimes called distance, proximity, or closeness).



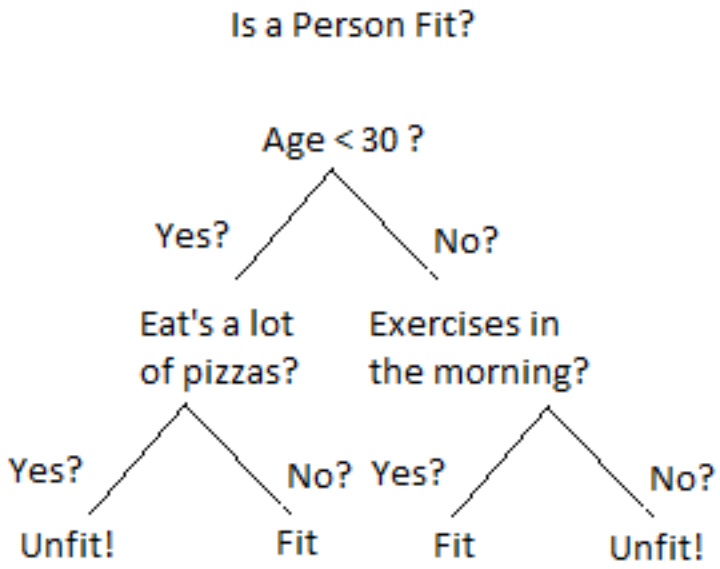
3. Support Vector Machines: The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



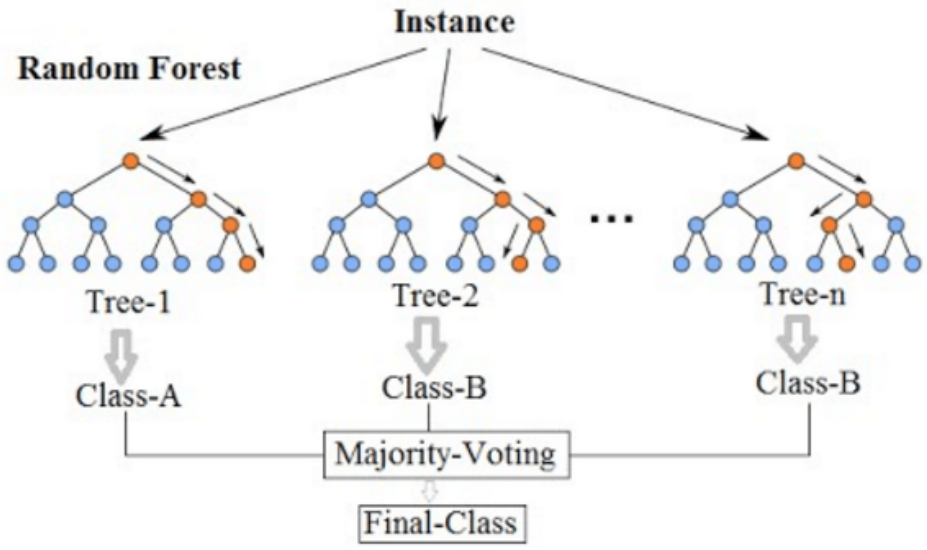


4. Decision Tree Classifier: A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter. A Decision Tree consists of :

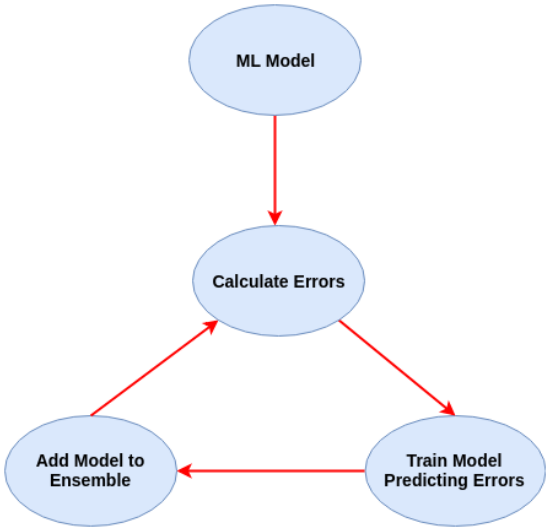
- Nodes : Test for the value of a certain attribute.
- Edges/ Branch : Correspond to the outcome of a test and connect to the next node or leaf.
- Leaf nodes : Terminal nodes that predict the outcome (represent class labels or class distribution).



5. Random Forest Classifier: It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. Ensemble algorithms are those which combines more than one algorithms of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object.



6. XGBoost Classifier: Gradient Boosting specifically is an approach where new models are trained to predict the residuals (i.e errors) of prior models. I've outlined the approach in the diagram below. The advantage of this iterative approach is that the new models being added are focused on correcting the mistakes which were caused by other models. In a standard ensemble method where models are trained in isolation, all of the models might simply end up making the same mistakes!



BENCHMARK

I wish to compare the results obtained from these machine learning models with the results obtained by using hyperparameter tuning for these six machine learning models.
When creating a machine learning model, there would be many design choices as to how to define your model architecture. Often times, we don't immediately know what the optimal model architecture should be for a given model, and thus we'd like to be able to explore a range of possibilities. In true machine learning fashion, we'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically. Parameters which define the model architecture are referred to as "hyperparameters" and thus this process of searching for the ideal model architecture is referred to as "hyperparameter tuning".

DATA PREPROCESSING

1. No null values were obtained in the dataset.

```
# Checking for missing values in the dataset

df.isnull().sum()
```

```
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
```

2. Performed the action of scaling on the dataset

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = scaler.fit_transform(dataset[columns_to_scale])
```

```
# After scaling the variables

dataset.head()
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	0.95	0.76	-0.26	0.02	1.09	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	-1.92	-0.09	0.07	1.63	2.12	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	-1.47	-0.09	-0.82	0.98	0.31	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	0.18	-0.66	-0.20	1.24	-0.21	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	0.29	-0.66	2.08	0.58	-0.38	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0

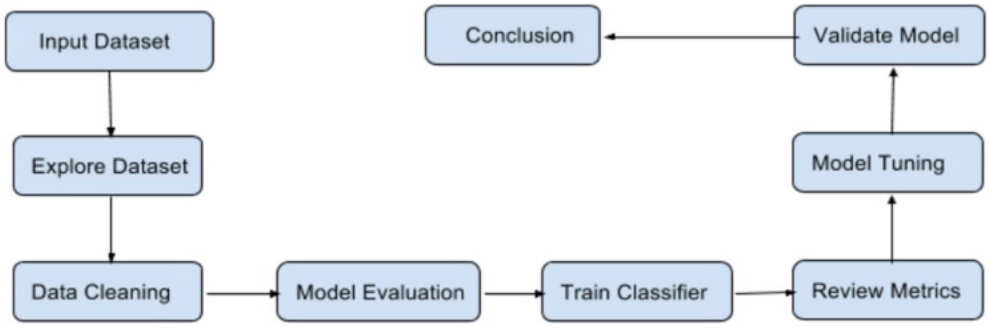
5 rows × 31 columns

3. Splitting the dataset into training and test sets with 70% of the data on the training set and 30% of the data on test set.

```
# Splitting the dataset into training and test set

from sklearn.model_selection import train_test_split

X = dataset.drop('target', axis=1)
y = dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

IMPLEMENTATION AND REFINEMENT

The process of implementation is as follows:



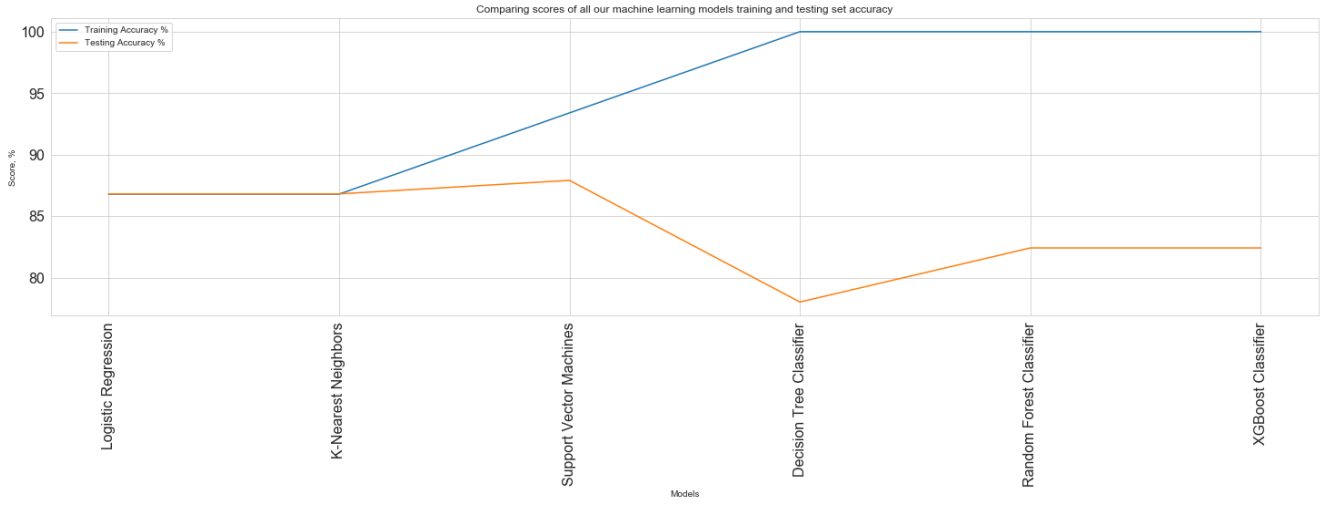
I shall first train the six machine learning models. I shall calculate the performance metrics for all the six models which would be to measure the Accuracy, Precision, Recall and F1-Score. I shall then evaluate the training and test set accuracies for all the six machine learning models to see which model performs better. I shall then tune the model using hyperparameter tuning and then do the same evaluation methods as done for the above machine learning models.

MODEL EVALUATION AND JUSTIFICATION

1. Performance of the Machine Learning models:

The six machine learning gave the following performance metrics for accuracy on the training and test set.

	Model	Training Accuracy %	Testing Accuracy %	(TrainScore - TestScore)
0	Logistic Regression	86.79	86.81	-0.02
1	K-Nearest Neighbors	86.79	86.81	-0.02
2	Support Vector Machine	93.40	87.91	5.48
3	Decision Tree Classifier	100.00	78.02	21.98
4	Random Forest Classifier	100.00	82.42	17.58
5	XGBoost Classifier	100.00	82.42	17.58



```
ml_models.sort_values(by=['Training_Scores'], ascending=False)
```

	Models	Training_Scores	Testing_Scores
3	Decision Tree Classifier	100.00	78.02
4	Random Forest Classifier	100.00	82.42
5	XGBoost Classifier	100.00	82.42
2	Support Vector Machines	93.40	87.91
0	Logistic Regression	86.79	86.81
1	K-Nearest Neighbors	86.79	86.81

```
ml_models.sort_values(by=['Testing_Scores'], ascending=False)
```

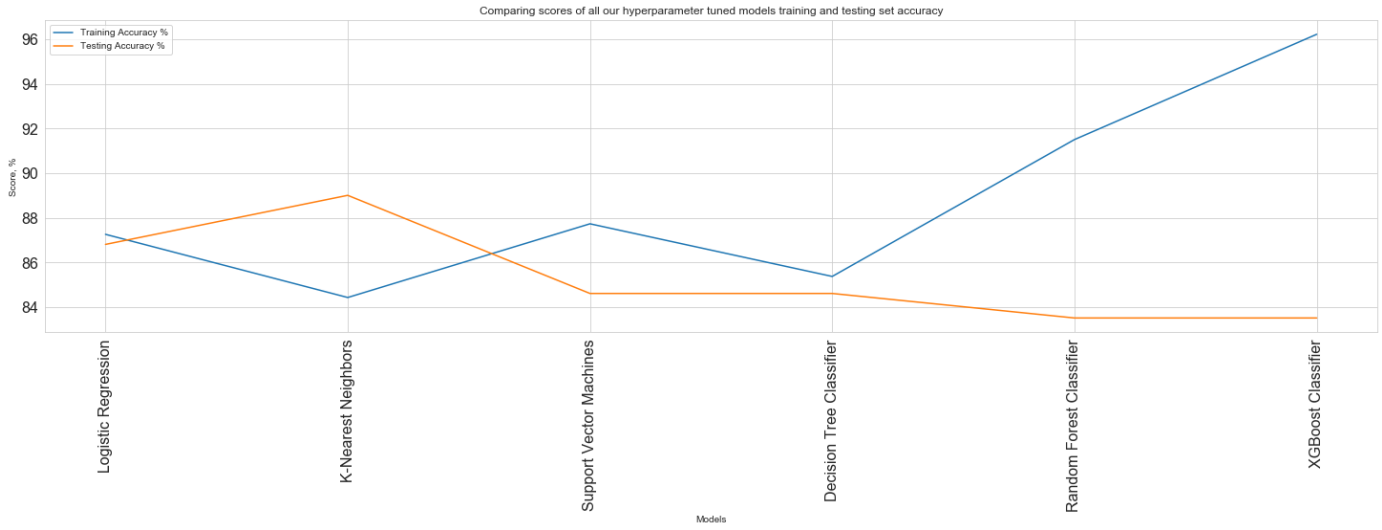
	Models	Training_Scores	Testing_Scores
2	Support Vector Machines	93.40	87.91
0	Logistic Regression	86.79	86.81
1	K-Nearest Neighbors	86.79	86.81
4	Random Forest Classifier	100.00	82.42
5	XGBoost Classifier	100.00	82.42
3	Decision Tree Classifier	100.00	78.02

Interpretation of results on the machine learning models:

- Random Forest, Decicion Tree and XGBoost classifier seems to perform the best on training set with an accuracy of 100%.
- SVM classifier seems to perform best on the test set with an accuracy of 87.91%.
- Logistic Regression and K-Nearest Neighbors showed the least difference between the training and testing set accuracy of just 0.02% and their test scores beat the training scores.

2. Performance of the Machine Learning models using Hyperparameter Tuning:

	Model	Training Accuracy %	Testing Accuracy %	(TrainScore - TestScore)
0	Tuned Logistic Regression	87.26	86.81	0.45
1	Tuned K-nearest neighbors	84.43	89.01	-4.58
2	Tuned Support Vector Machine	87.74	84.62	3.12
3	Tuned Decision Tree Classifier	85.38	84.62	0.76
4	Tuned Random Forest Classifier	91.51	83.52	7.99
5	Tuned XGBoost Classifier	96.23	83.52	12.71



```
models.sort_values(by=['Training_Accuracy_Scores'], ascending=False)
```

	Model	Training_Accuracy_Scores	Testing_Accuracy_Scores
5	XGBoost Classifier	96.23	83.52
4	Random Forest Classifier	91.51	83.52
2	Support Vector Machines	87.74	84.62
0	Logistic Regression	87.26	86.81
3	Decision Tree Classifier	85.38	84.62
1	K-Nearest Neighbors	84.43	89.01

```
models.sort_values(by=['Testing_Accuracy_Scores'], ascending=False)
```

	Model	Training_Accuracy_Scores	Testing_Accuracy_Scores
1	K-Nearest Neighbors	84.43	89.01
0	Logistic Regression	87.26	86.81
2	Support Vector Machines	87.74	84.62
3	Decision Tree Classifier	85.38	84.62
4	Random Forest Classifier	91.51	83.52
5	XGBoost Classifier	96.23	83.52

Interpretation of the results:

- XGBoost classifier seems to perform the best on training set with an accuracy of 96.23%.
- K-Nearest Neighbors classifier seems to perform best on the test set with an accuracy of 89.01% beating its test set accuracy of 84.43%
- Logistic Regression showed the least difference between the training and testing set accuracy of just 0.45%

Overall, I believe K-Nearest Neighbor classification would be ideal for this model as it has shown pretty good performance on the training set for both machine learning model and with hyperparameter tuning. KNN model has also

performed better on the test set than on the training set. Hence, I beliebe KNN machine learning model with hyperparameter tuning would be the best fit.

KNN model performance on machine learning model:

```
TRAIN RESULT:
-------------
Accuracy Score: 84.43%
Classification Report:  Precision Score: 82.54%
                        Recall Score: 90.43%
                        F1 score: 86.31%
Confusion Matrix:
 [[ 75  22]
 [ 11 104]]

************************************************

TEST RESULT:
-------------
Accuracy Score: 89.01%
Classification Report:  Precision Score: 87.04%
                        Recall Score: 94.00%
                        F1 score: 90.38%
Confusion Matrix:
 [[34  7]
 [ 3 47]]
```

KNN model performance with hyperparameter tuning:

```
TRAIN RESULT:
-------------
Accuracy Score: 84.43%
Classification Report:  Precision Score: 82.54%
                        Recall Score: 90.43%
                        F1 score: 86.31%
Confusion Matrix:
 [[ 75  22]
 [ 11 104]]

************************************************

TEST RESULT:
-------------
Accuracy Score: 89.01%
Classification Report:  Precision Score: 87.04%
                        Recall Score: 94.00%
                        F1 score: 90.38%
Confusion Matrix:
 [[34  7]
 [ 3 47]]
```