MACHINE LEARNING PROJECT

HEART DISEASE CLASSIFICATION

Abhishek Banerjee

INDEX

TABLE OF GRAPHS:

1. First 5 rows of the dataset

```
#Printing a few columns
dataset.head()
```

	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

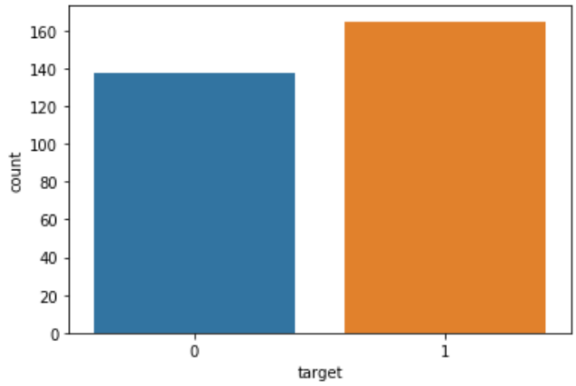
2. Description of statistics of the dataset

```
#Description
dataset.describe()
```

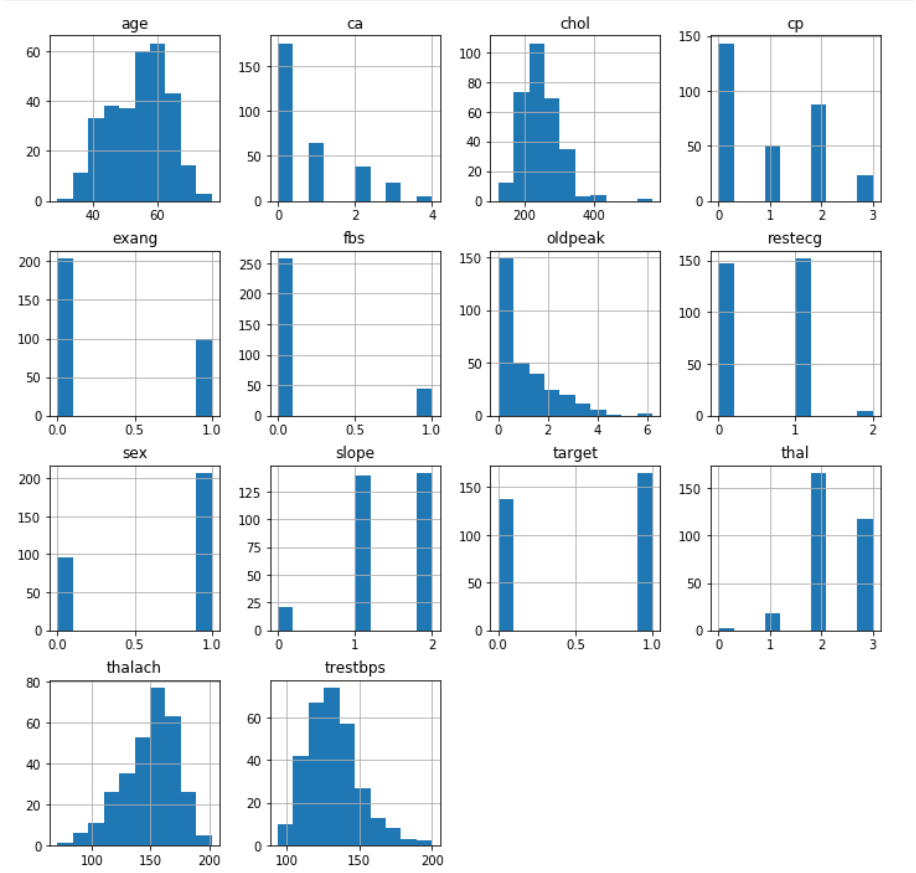
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000

3. Heatmap of the correlation matrix

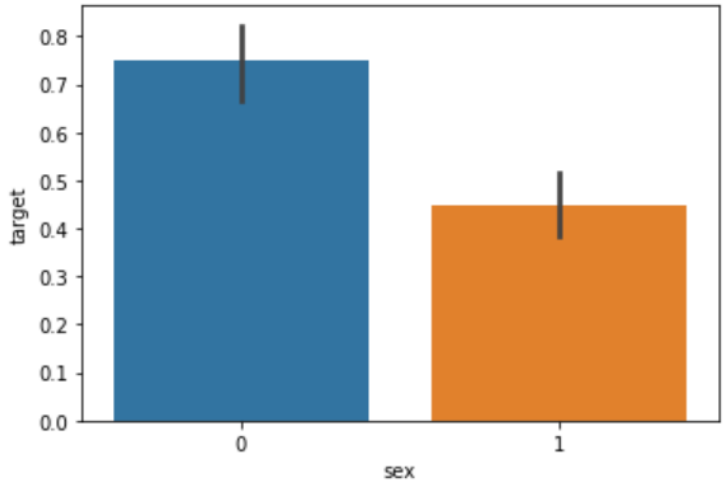
4. Number of patients NOT having heart disease (0) vs having heart disease (1)



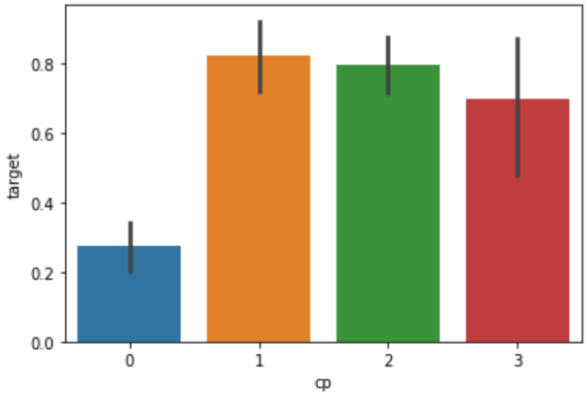
5. Histograms of all the features



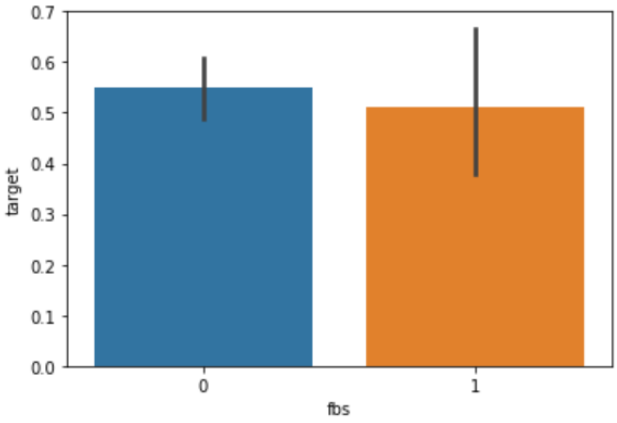
6. No of females (0) vs no of males (1) more likely to have heart disease



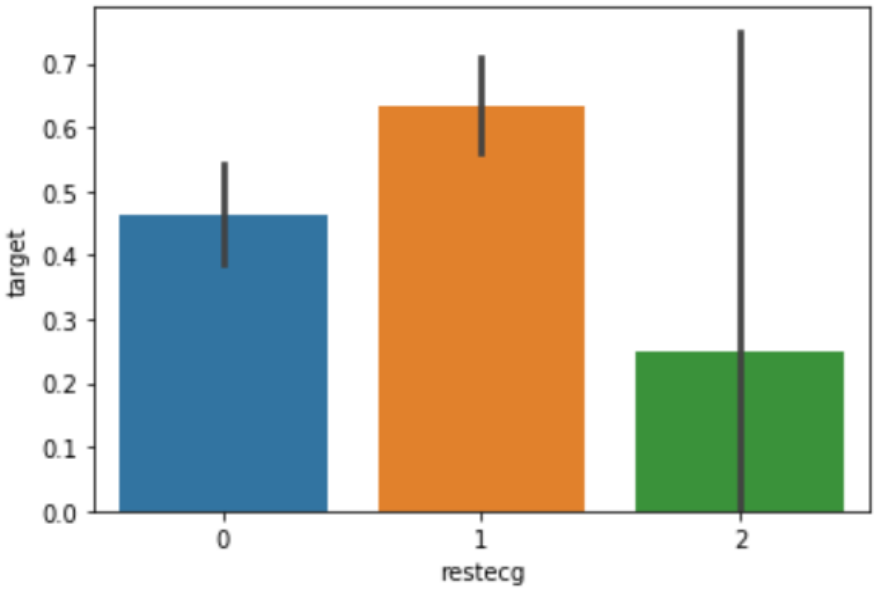
7. Types of chest pains associated with more likely to have heart disease



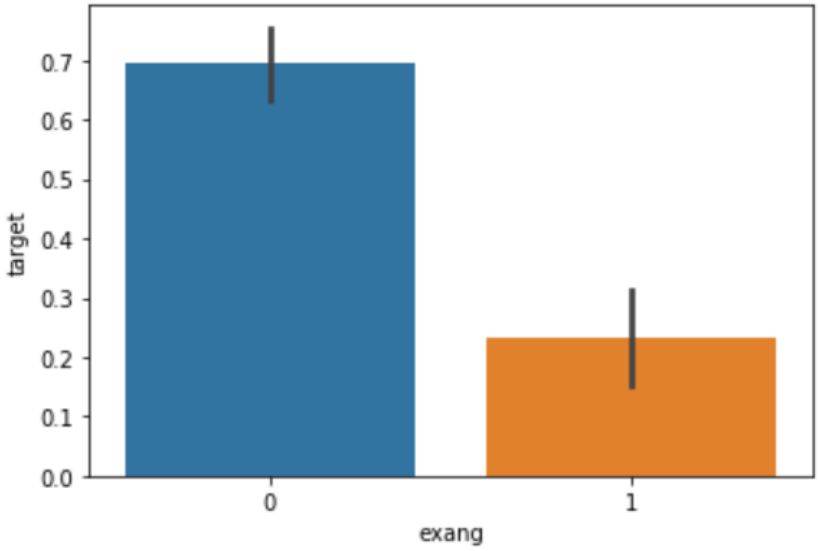
8. Fasting blood sugar > 120 mg/dl (1) VS fasting blood sugar < 120 mg/dl (0) associated with more likely to having a heart disease



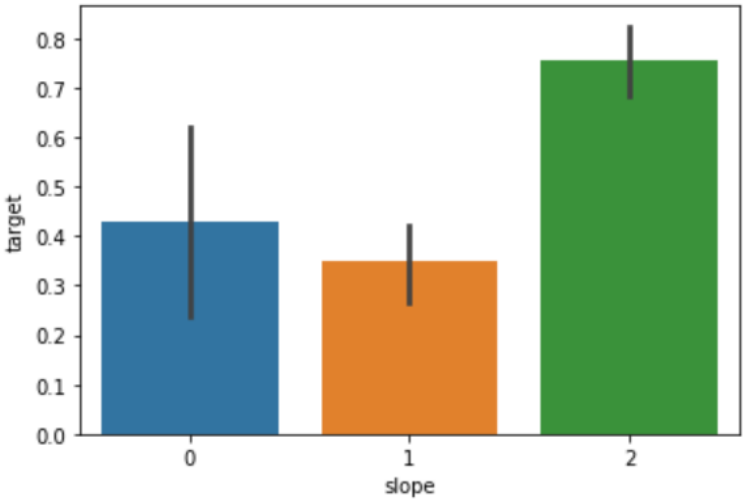
9. Different values of resting electrocardiographic results (restecg) vs more likely to having a heart disease



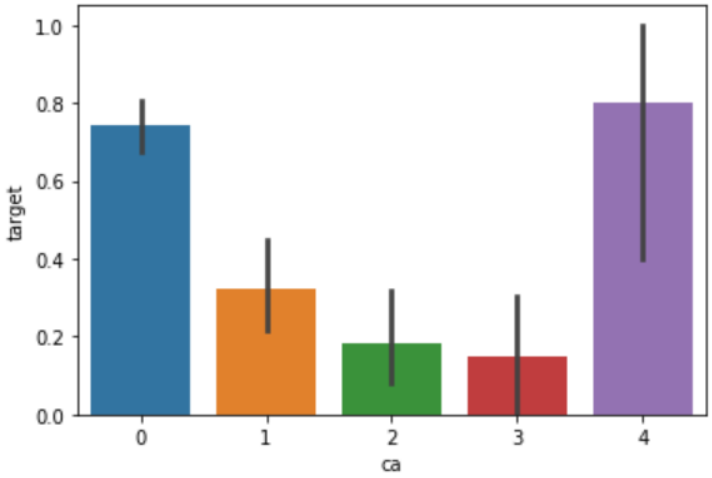
10. Exercise induced angina(1) vs NO Exercise induced angina (0) with more likely to having a heart disease



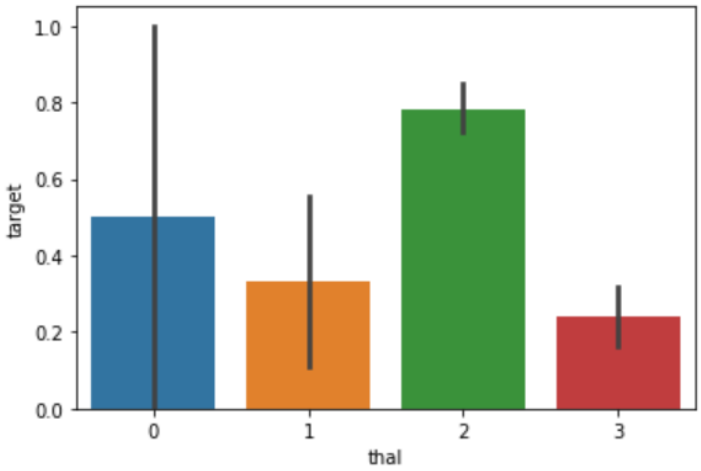
11. slope: the slope of the peak exercise ST segment
    -- Value 1: upsloping
    -- Value 2: flat
    -- Value 3: downsloping VS more likely to have heart disease



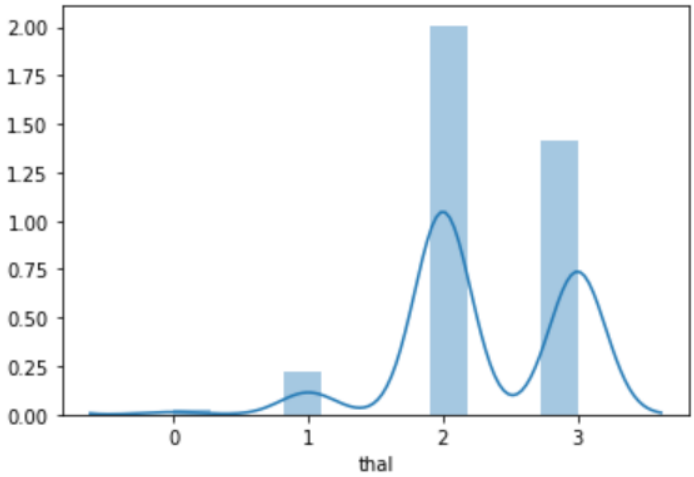
12. number of major vessels (0-3) colored by flourosopy vs more likely to have heart disease



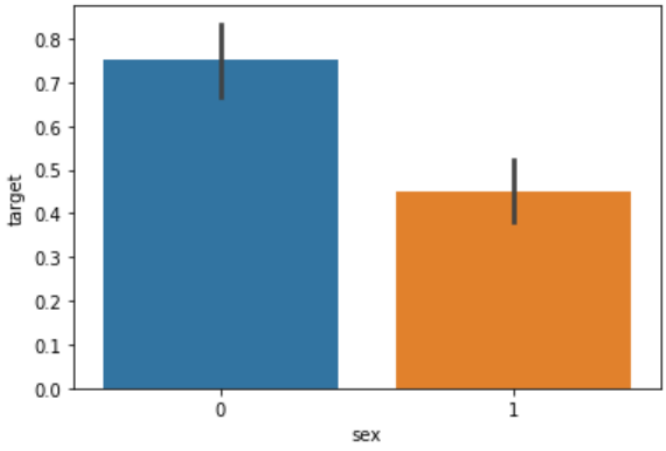
13.thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
vs chances of having a heart disease



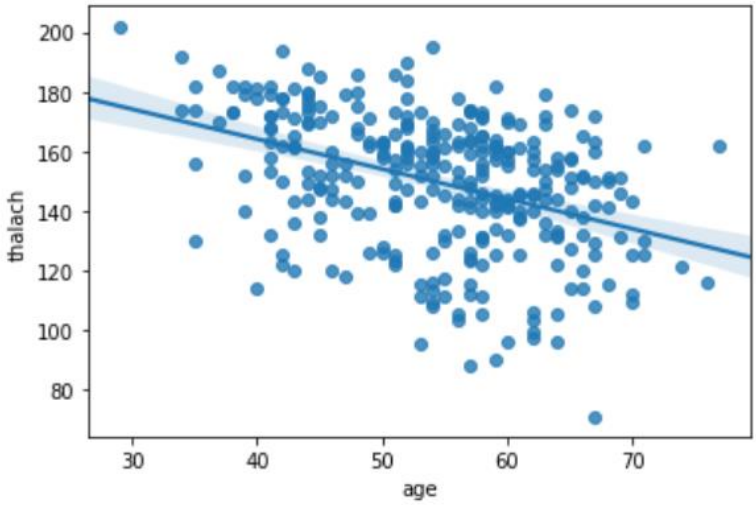
14. Distplot of thalassemia



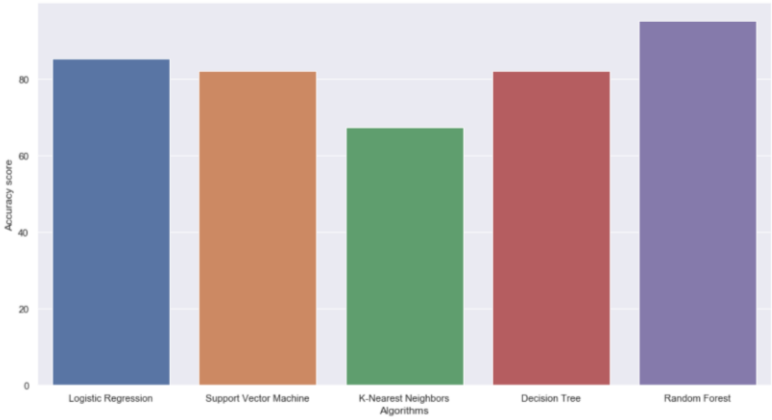
15.Bar Plot of Male & Female having Heart Disease



16.Scatter Plot Age & Maximum Heart rate



17.Accuracy score of different models



Introduction

Heart disease has become a regular problem with people approaching old age. It is also affecting middle aged adults making it fatal for them. These people might not be diagnosed with coronary artery disease until have a heart attack, angina, stroke or heart failure. It's important to watch for cardiovascular symptoms and discuss concerns with your doctor. Cardiovascular disease can sometimes be found early with regular evaluations.
Thus, preventing heart diseases has become more than necessary. Good data-driven systems for predicting heart diseases can improve the entire research and prevention process, making sure that more people can live healthy lives. This is where Machine Learning comes into play. Machine Learning helps in predicting the heart diseases, and the predictions made are quite accurate.

The project involved analysis of the heart disease patient dataset with proper data processing. Then, different models were trained and predictions are made with different algorithms KNN, Decision Tree, Random Forest, SVM, Logistic Regression etc. This is the Jupyter notebook code and dataset I've used is given.

I've used a variety of Machine Learning algorithms, implemented in Python, to predict the presence of heart disease in a patient. This is a classification problem, with input features as a variety of parameters, and the target variable as a binary variable, predicting whether heart disease is present or not.
Machine Learning algorithms used:

1. Logistic Regression (Scikit-learn)
2. Support Vector Machine (Linear) (Scikit-learn)
3. K-Nearest Neighbors (Scikit-learn)
4. Decision Tree (Scikit-learn)
5. Random Forest (Scikit-learn)

Accuracy achieved: 95% (Random Forest)

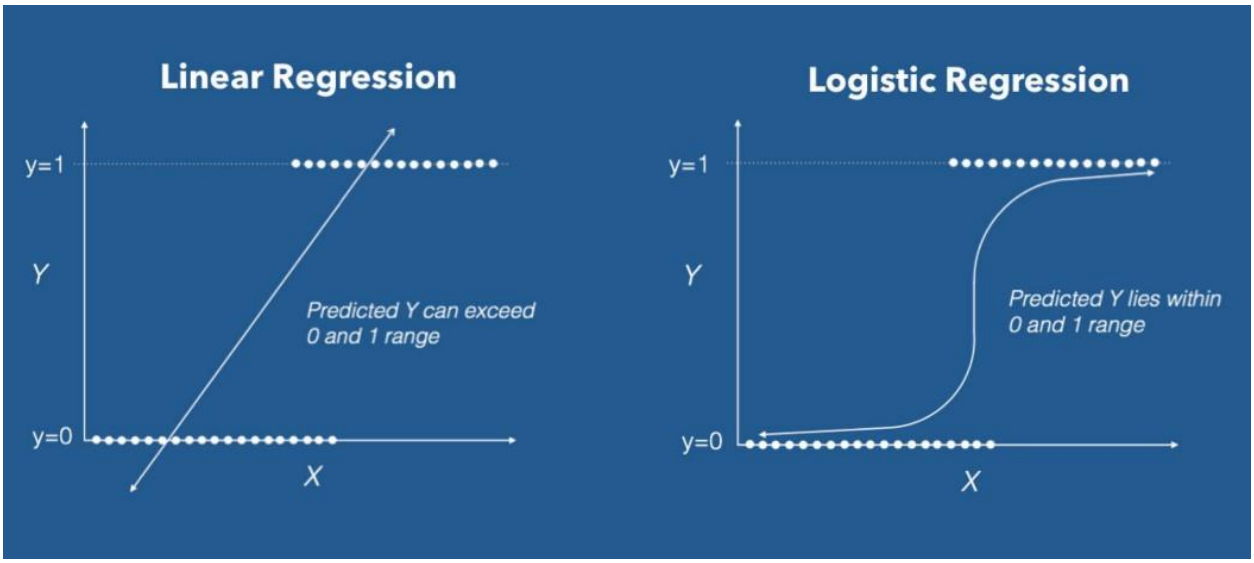
Software Libraries Used

- numpy
- seaborn
- pandas
- sklearn.datasets
- sklearn.model_selection
- sklearn.linear_model
- sklearn.metrics
- matplotlib.pyplot
- sklearn.neighbours
- sklearn.tree
- sklearn.ensemble

Mathematics

- **Logistic Regression**

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.



We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the '**Sigmoid function**' or also known as the 'logistic function' instead of a linear function.

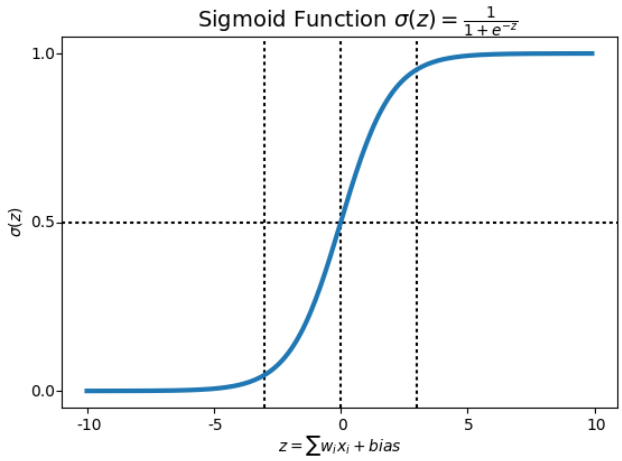
The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$$0 \leq h_\theta(x) \leq 1$$

Logistic regression hypothesis expectation

-What is the Sigmoid Function?

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.



Sigmoid Function Graph

$$f(x) = \frac{1}{1+e^{-(x)}}$$

Formula of a sigmoid function

-Hypothesis Representation

When using *linear regression* we used a formula of the hypothesis i.e.

$h\Theta(x) = \beta_0 + \beta_1 X$

For logistic regression we are going to modify it a little bit i.e.

$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$

We have expected that our hypothesis will give values between 0 and 1.

$Z = \beta_0 + \beta_1 X$

$h\Theta(x) = \text{sigmoid}(Z)$

i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$

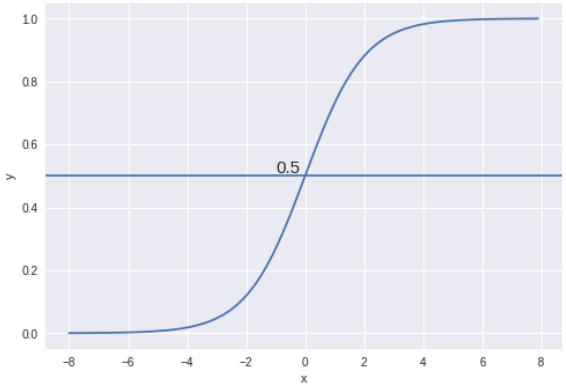
$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

The Hypothesis of logistic regression

-Decision Boundary

We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.

For Example, We have 2 classes, let's take them like cats and dogs(1 — dog , 0 — cats). We basically decide with a threshold value above which we classify values into Class 1 and of the value goes below the threshold then we classify it in Class 2.

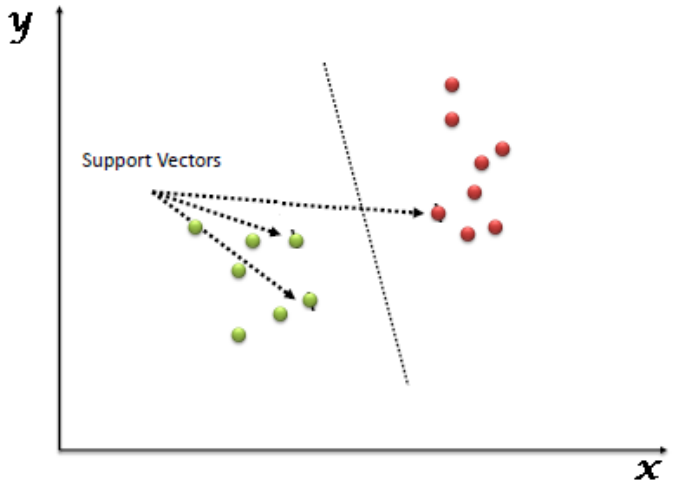


Example

As shown in the above graph we have chosen the threshold as 0.5, if the prediction function returned a value of 0.7 then we would classify this observation as Class 1(DOG). If our prediction returned a value of 0.2 then we would classify the observation as Class 2(CAT).

- **Support Vector Machine**

"Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



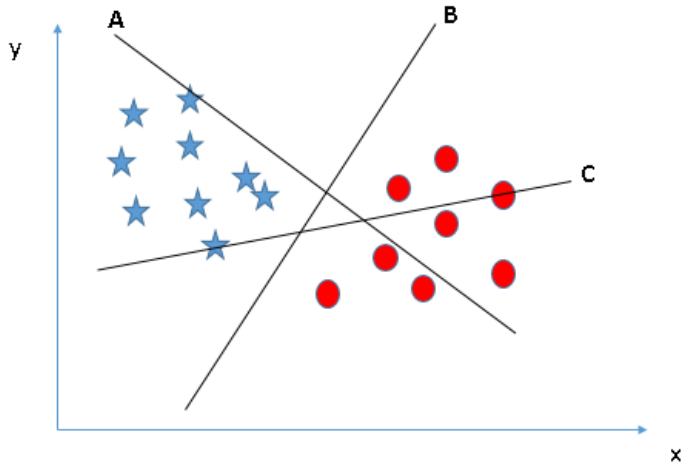
Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

How does it work?

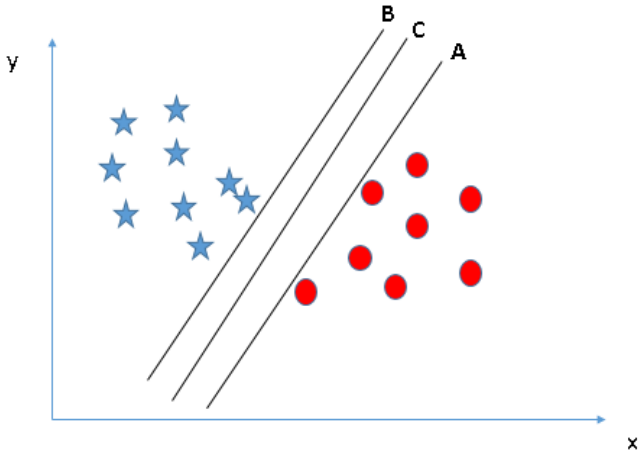
Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is "How can we identify the right hyper-plane?".

Let's understand:

- Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.

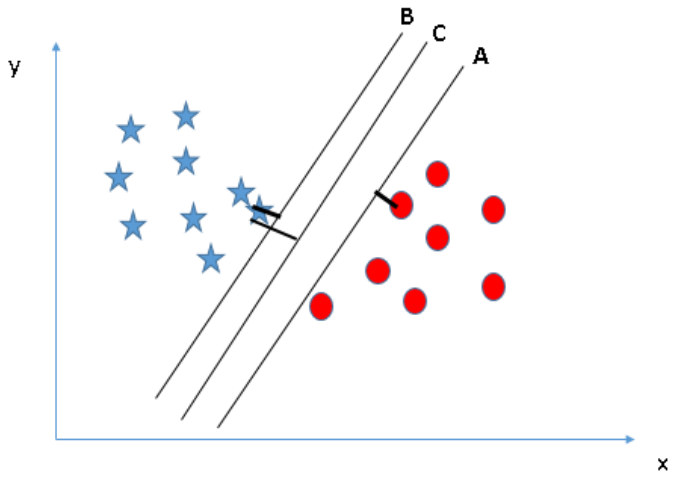


- You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.
- Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



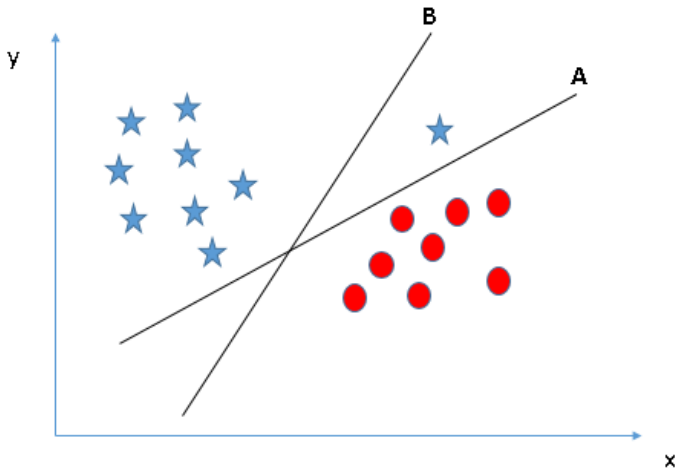
Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.

Let’s look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

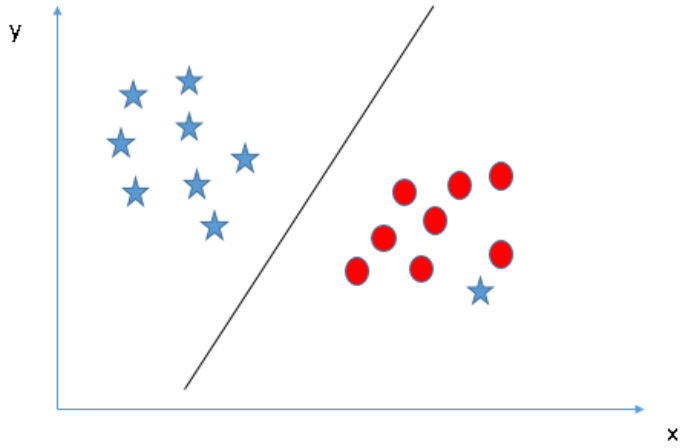
- Identify the right hyper-plane (Scenario-3):Hint: Use the rules as discussed in previous section to identify the right hyper-plane



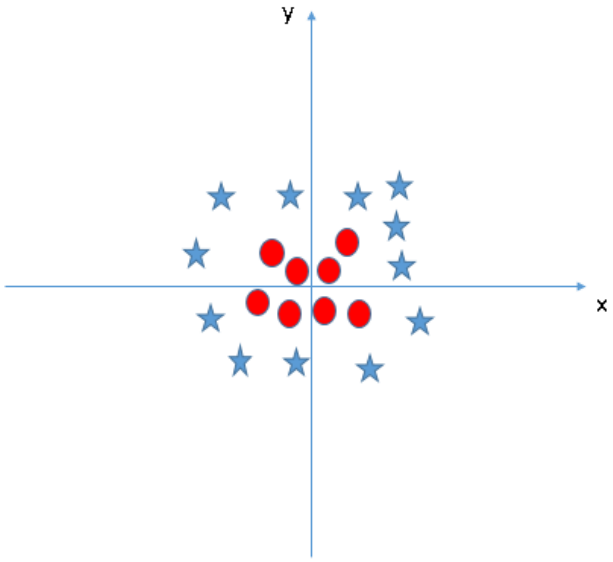
Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

- Can we classify two classes (Scenario-4)?: Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.

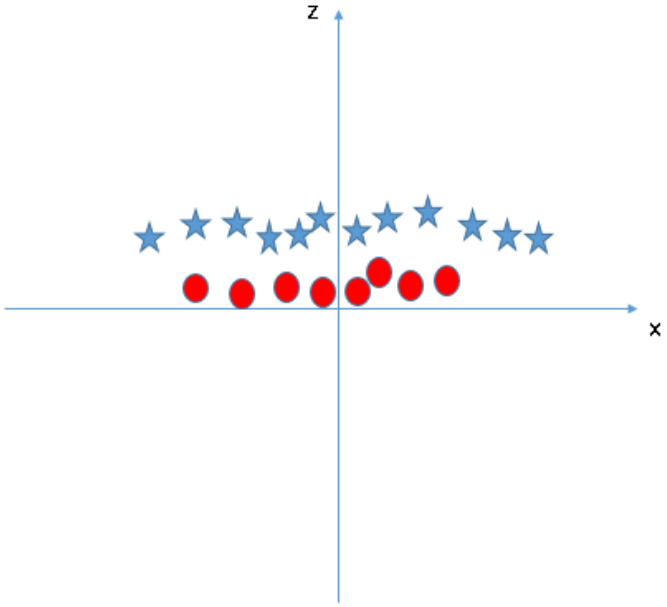
- As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



- Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



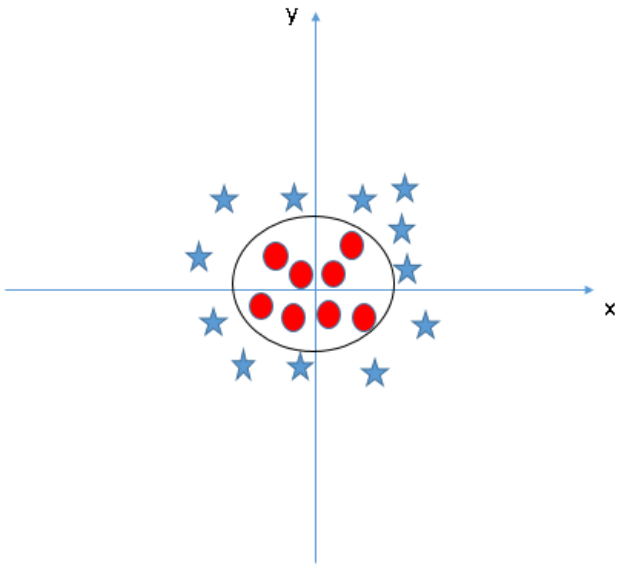
In above plot, points to consider are:

  - All values for z would be positive always because z is the squared sum of both x and y

- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the kernel trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



- **K- Nearest Neighbour**

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.

The algorithm's learning is:

1. Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.

2. Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.

3. Non -Parametric: In KNN, there is no predefined form of the mapping function.

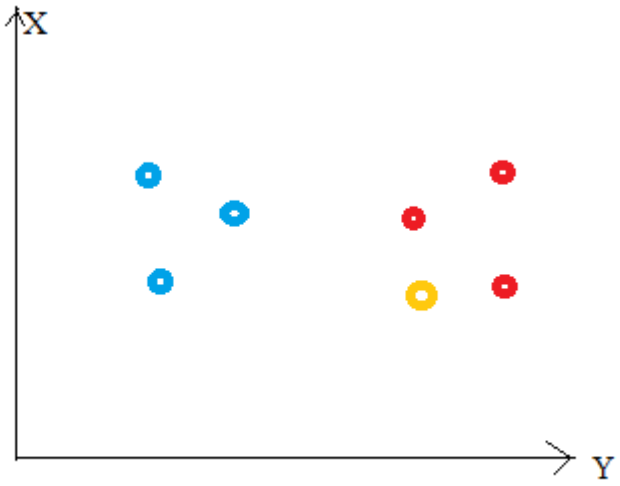
How does KNN Work?

1. *Principle:*

   Consider the following figure. Let us say we have plotted data points from our training set on a two-dimensional feature space. As shown, we have a total of 6 data points (3 red and 3 blue). Red data points belong to 'class1' and blue data points belong to 'class2'. And yellow data point in a feature space represents the new point for which a class is to be predicted. Obviously, we say it belongs to 'class1' (red points)

Why?

Because its nearest neighbors belong to that class!



Yes, this is the principle behind K Nearest Neighbors. Here, nearest neighbors are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm. Euclidean distance is the most popular distance metric. You can also use Hamming distance, Manhattan distance, Minkowski distance as per your need. For predicting class/ continuous value for a new data point, it considers all the data points in the training dataset. Finds new data point's 'K' Nearest Neighbors (Data points) from feature space and their class labels or continuous values.

Then:

For classification: A class label assigned to the majority of K Nearest Neighbors from the training dataset is considered as a predicted class for the new data point.

For regression: Mean or median of continuous values assigned to K Nearest Neighbors from training dataset is a predicted continuous value for our new data point

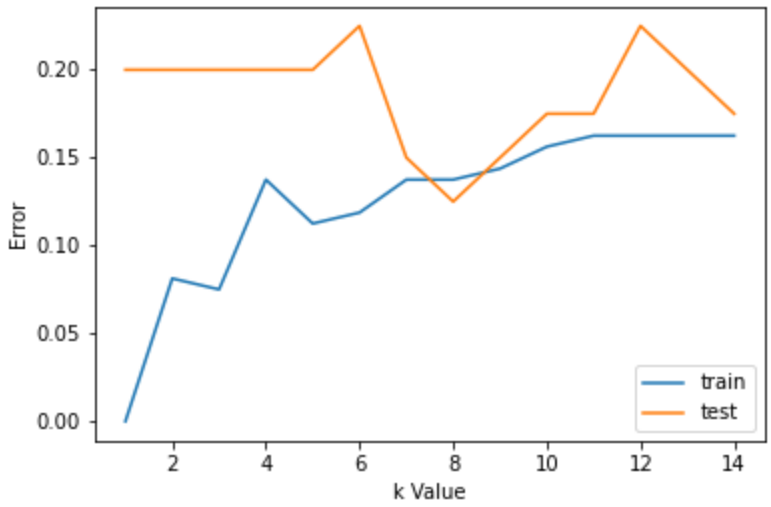
2. *Model Representation*

   Here, we do not learn weights and store them, instead, the entire training dataset is stored in the memory. Therefore, model representation for KNN is the entire training dataset.

How to choose the value for K?

K is a crucial parameter in the KNN algorithm. Some suggestions for choosing K Value are:

1. Using error curves: The figure below shows error curves for different values of K for training and test data.



Choosing a value for K

At low K values, there is overfitting of data/high variance. Therefore test error is high and train error is low. At K=1 in train data, the error is always zero, because the nearest neighbor to that point is that point itself. Therefore though training error is low test error is high at lower K values. This is called overfitting. As we increase the value for K, the test error is reduced.

But after a certain K value, bias/ underfitting is introduced and test error goes high. So we can say initially test data error is high(due to variance) then it goes low and stabilizes and with further increase in K value, it again increases(due to bias). The K value when test error stabilizes and is low is considered as optimal value for K. From the above error curve we can choose K=8 for our KNN algorithm implementation.

2. Also, domain knowledge is very useful in choosing the K value.

3. K value should be odd while considering binary(two-class) classification.

Required Data Preparation:

1. Data Scaling: To locate the data point in multidimensional feature space, it would be helpful if all features are on the same scale. Hence normalization or standardization of data will help.

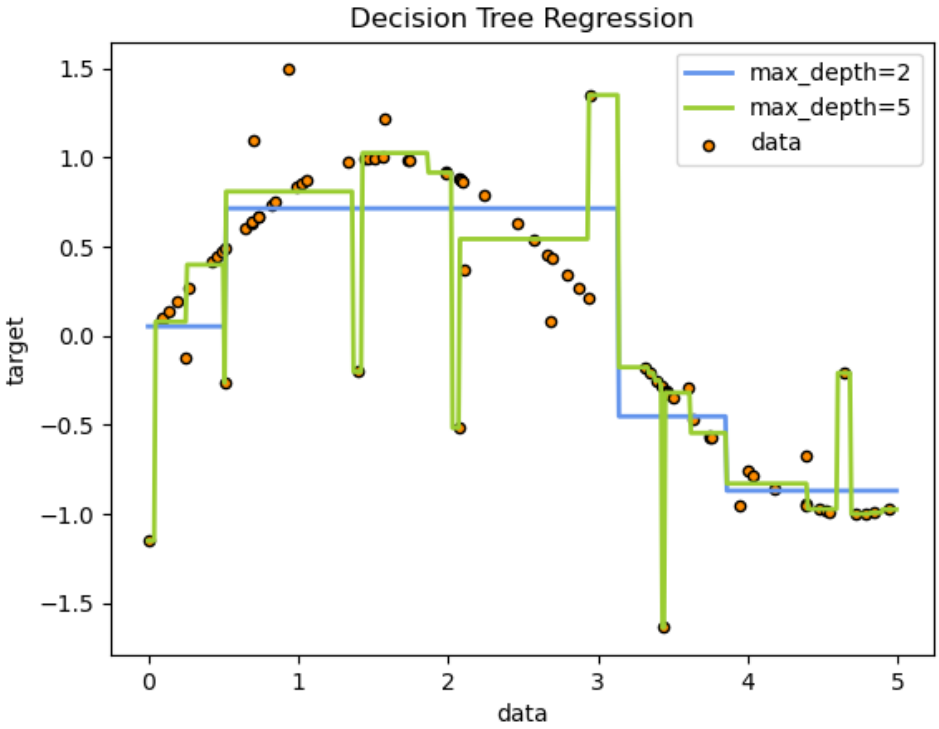
2. Dimensionality Reduction: KNN may not work well if there are too many features. Hence dimensionality reduction techniques like feature selection, principal component analysis can be implemented.

3. Missing value treatment: If out of M features one feature data is missing for a particular example in the training set then we cannot locate or calculate distance from that point. Therefore deleting that row or imputation is required.

- ## Decision Tree

  Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

  For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.



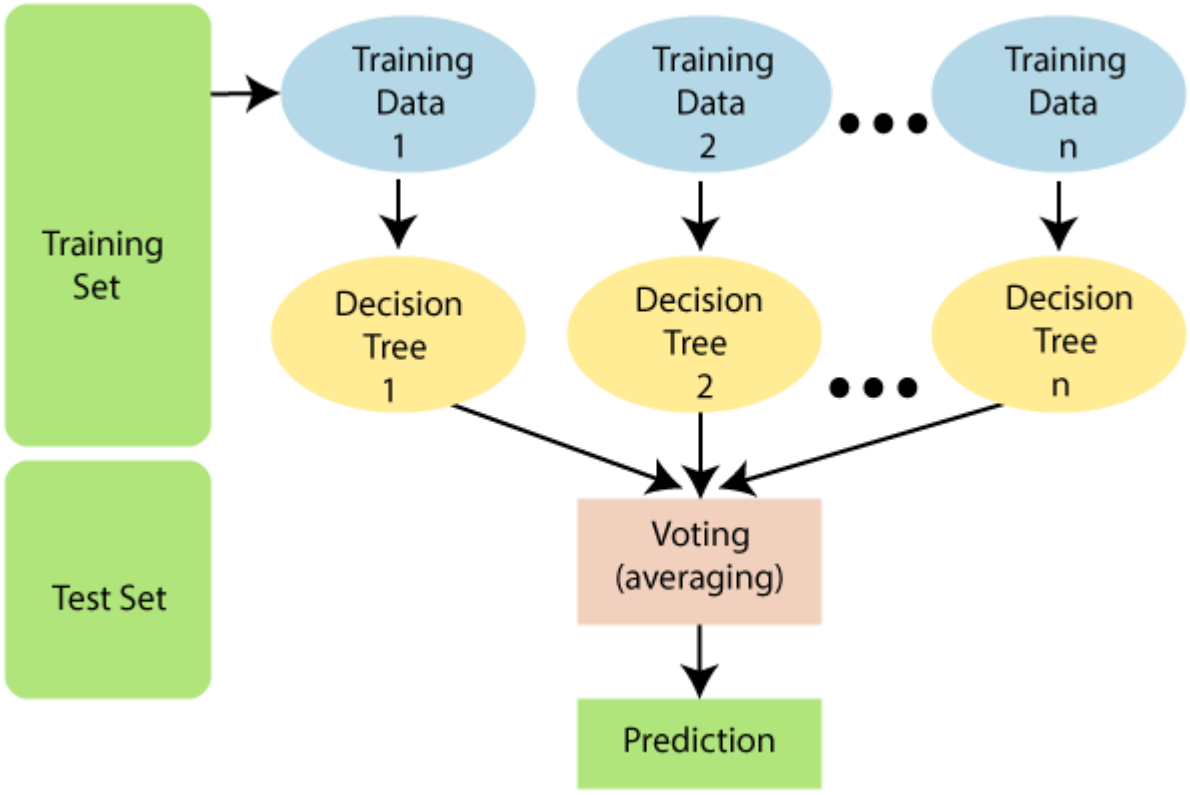
- ## Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, *"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."* Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Conclusion:

We investigated the data, checking for data unbalancing, visualizing the features and understanding the relationship between different features

We used train-validation split to evaluate the model effectiveness to classify the target value, Le detecting if heart disease is present.

We then investigated four classification models

We started with Logistic Regression, SVM, KNN, Decision Tree and Random Forest.

The highest accuracy was achieved was Random Forest (95%).