STATISTICAL AND MACHINE LEARNING APPROACHES FOR MARKETING

Prof. Minh Phan

IÉSEG
SCHOOL OF MANAGEMENT

Sai Sumanth Sripada

TABLE OF CONTENTS

DATA PRE-PROCESSING :

Outline and Objective :

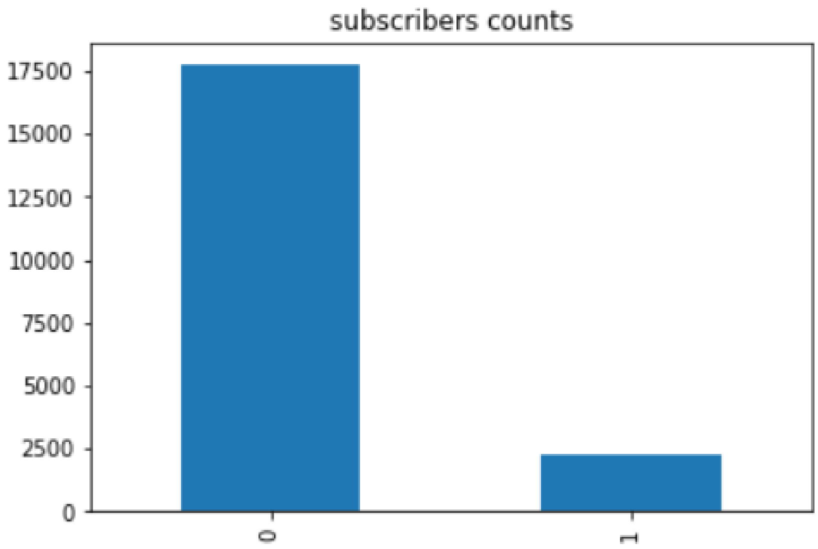
Taiwanese Bank wants to know which customers will subscribe for their next marketing Campaign. Objective is to use the customer data and try different Machine Learning algorithms to find out which best suits the current use case.

Data: The Raw data has 20000 observations and 21 Features initially.

Feature	*Dtype*	*Nunique*	*Action Performed*
client_id	int64	20000 –Unique ID	*Unique IDs*
age	float64	77	*Created Age Groups 20s,30s,40s..*
job	object	12	Replaced NA with Mode and one hot encoded
marital	object	4	Replaced NA with Mode and one hot encoded
education	object	8	Replaced NA with Mode and one hot encoded
default	object	3	Replaced NA with Mode and one hot encoded
housing	object	3	Replaced NA with Mode and one hot encoded
loan	object	3	Replaced NA with Mode and one hot encoded
contact	object	2	Replaced NA with Mode and one hot encoded
month	object	10	Replaced NA with Mode and one hot encoded
day_of_week	object	5	Replaced NA with Mode and one hot encoded
campaign	float64	40	Replaced NA with Median
pdays	float64	25	Replaced NA with Median

previous	float64	7	Replaced NA with Median
poutcome	object	3	Replaced NA with Mode and one hot encoded
emp.var.rate	float64	10	Replaced NA with Median
cons.price.idx	float64	26	Replaced NA with Median
cons.conf.idx	float64	26	Replaced NA with Median
euribor3m	float64	300	Replaced NA with Median
nr.employed	float64	11	Replaced NA with Median
subscribe	int64	2 (Target 1/0)	Target Variable

- Subscribe is the target variable and the distribution is as below:



- Based on the distribution , the threshold is 0.11355 which would help to predict whether the client will subscribe or not (1/0)

LOGISTIC REGRESSION:

Logistic Regression is a Regression technique that is used when the outcome is categorical with possible two or multiple outcomes. it is a predictive analysis algorithm and based on the concept of probability. Logistic Regression is one of the most easily interpretable classification techniques and easy to implement. Unlike Linear Regression, Logistic Regression does not make any assumptions of Normality, Linearity of Variance. This is one of the reasons that Logistic Regression could be more powerful as these assumptions are rarely satisfied.

- The Hypothesis of the Logistic regression tends to limit the outcome between 0 and 1.
- Logistic Regression solves the limitation of Linear Regression in which the outcome variable (y) must be continuous.
- Logistic Regression is a Linear Regression model, but the uses a more complex cost function defined as the **Sigmoid function** or also known as the logistic function.

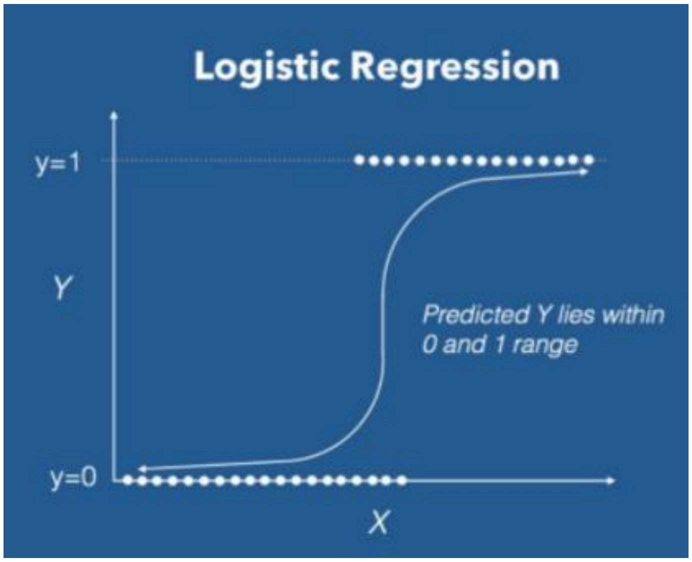


Image Source : Datacamp

Sigmoid Function:

$$\text{Sigmoid Function: } \boldsymbol{f(x)} = \frac{\boldsymbol{e^x}}{\boldsymbol{1+ e^x}}$$

Simple Logistic Regression:

$$\boldsymbol{p(x)} = \frac{\boldsymbol{e^{\beta_\circ + \beta_1 X}}}{\boldsymbol{1 + e^{\beta_\circ + \beta_1 X}}}$$

- e≈2.71828: Constant.
- $p(X)$=Pr(Y=1|X): probability that observation X is in class 1, return value in [0, 1].
- β_0, β_1: the coefficient of Logistic Regression model.

Maximum Likelihood Estimation :

A cost function tells us how close the values are from actual. So here we need a cost function which maximizes the likelihood of getting desired output values. Such a cost function is called as Maximum Likelihood Estimation (MLE) function.

$$L(\beta_0, \beta_1) = \prod_{i:y_i = 1} p(x_i) \prod_{i:y_i = 1} (1 - p(x_i))$$

- β_0, β_1 should be tuned to get the maximum likelihood.

Pros - Logistic Regression:

- Logistic regression is easier to implement, interpret, and very efficient to train.
- Makes no assumptions about distributions of classes.
- With right representation of features it can do a wonderful job and sometimes Logistic Regression is Just enough to make sure the pipeline works.
- Fast and Efficient at classifying unknown records.
- Helps to understand the Neural Networks.

Cons – Logistic Regression:

- Might lead to overfitting if number of features are more than number of Observations.
- Requires no multicollinearity between independent variables.
- Sensitive to outliers.

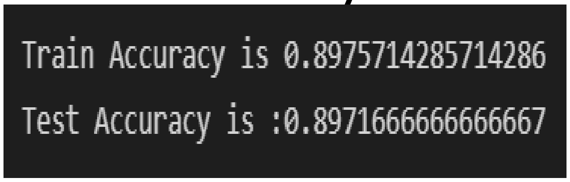
Model Benchmark – Logistic Regression :

Steps Followed :

1. Fit the Logistic model with all the Features and check the performance
2. Calculate the optimum threshold value which is 0.11355 and estimated the predictions based on the threshold.
3. Selected features using Sequential Forward Selection and fit the model again to compare the performance.
4. Cross Validation to verify the performance of the Model.

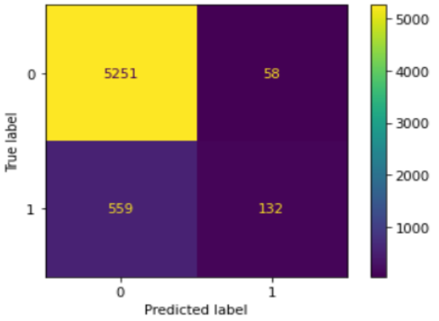
Initial Results with all the Features:

Accuracy

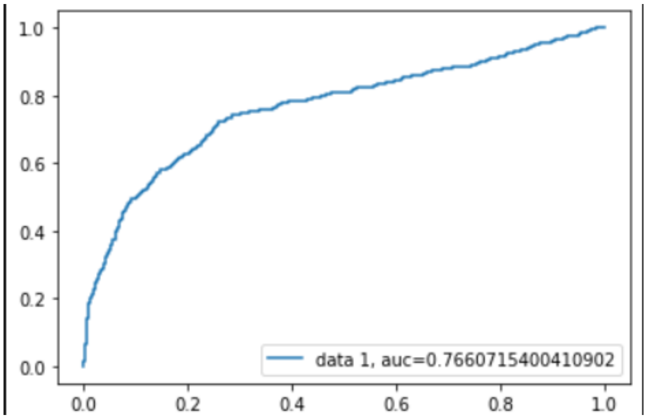


```
Train Accuracy is 0.8975714285714286
Test Accuracy is :0.8971666666666667
```

Confusion Matrix

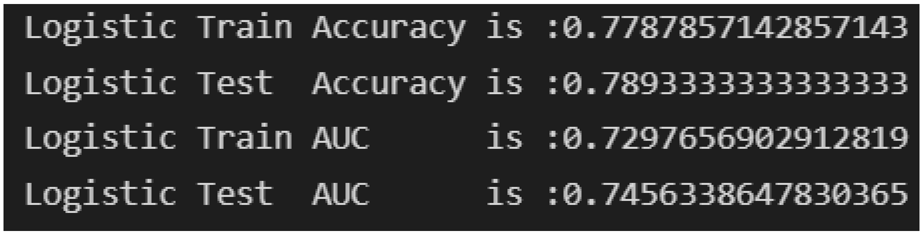


ROC CURVE:



- Upon checking the target count , threshold value of 0.11355 was considered for the probabilities predicted.

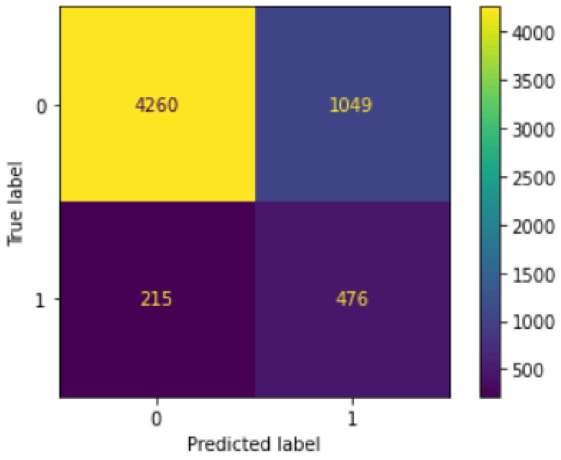
Results with the selected featured using Forward Stepwise :



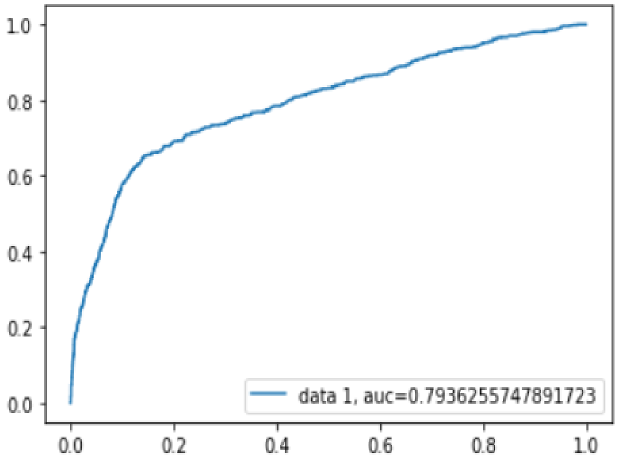
```
Logistic Train Accuracy is :0.7787857142857143
Logistic Test  Accuracy is :0.7893333333333333
Logistic Train AUC      is :0.7297656902912819
Logistic Test  AUC      is :0.7456338647830365
```

- Initially we can notice the overfitting from the results and after taking the threshold and feature selection through forward stepwise, now the overfitting is taken care of.
- Below Confusion Matrix shows that the true positive values are now more accurate than the first approach.

Confusion Matrix:



ROC Curve



DECISION TREE :

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly used for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node**. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem or decision based on given conditions.
- It is called a decision tree because similar to a tree it starts with the root node which expands on further branches and constructs a tree-like structure.
- A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

Basic Decision Tree :

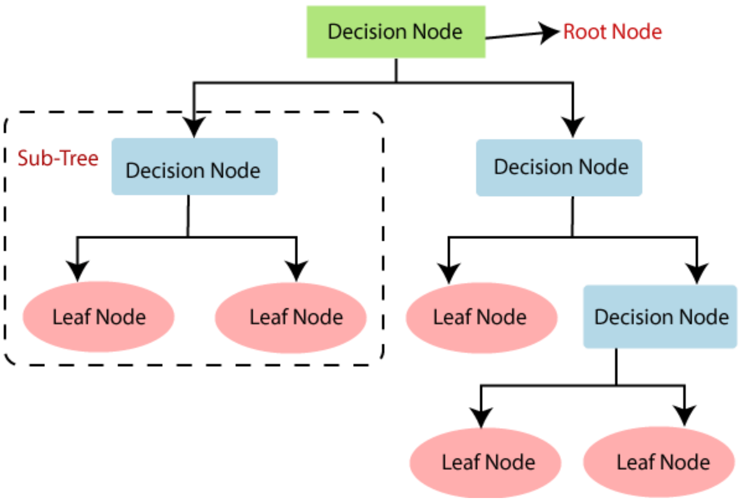


Image Source : Google

- **Decision Node:** Is the Main Node and the root node of the data from where tree starts.
- **Leaf Node:** Leaf nodes are the final nodes and cannot be divided further.
- **Sub Tree:** A tree formed by splitting the tree.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

Example : Below we can see an example of simple decision tree where ,based on patients symptoms, decision is taken and suggested by doctor.

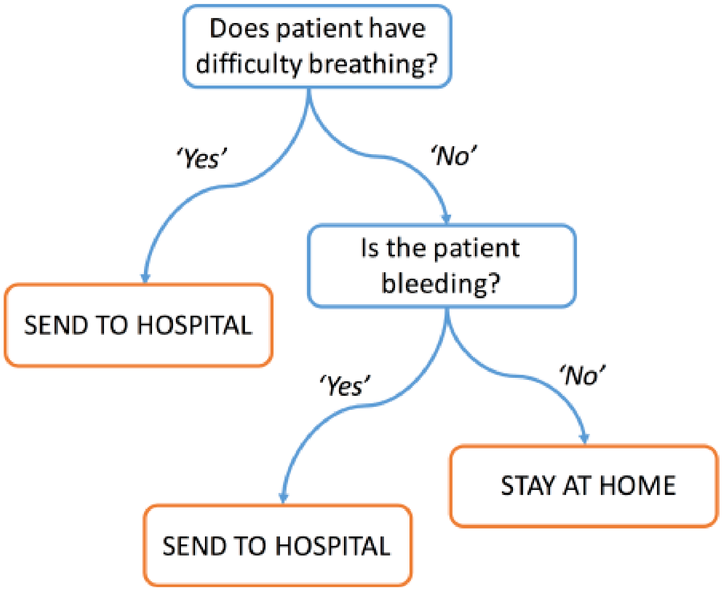


Image Source : class slide (https://www.samtalksml.net/helping-doctors-validate-decision-trees/)

Decision Tree working:

1. Tree Begins with Root Node
2. Find the Best attribute (GINI , Information Gain)
3. Make that attribute decision node and breaks dataset into smaller subsets.
4. Start tree building by iterating the process.

- GINI is the measure of impurity or purity used while creating a tree
- Lower the GINI better the performance
- Information gain is the measurement of the changes in entropy after the segmenttion

Pros – Decision Tree:

- Simple to understand as it works like a human.
- Useful for decision related problems.
- Less preprocessing compared to other models.

Cons – Decision Tree:

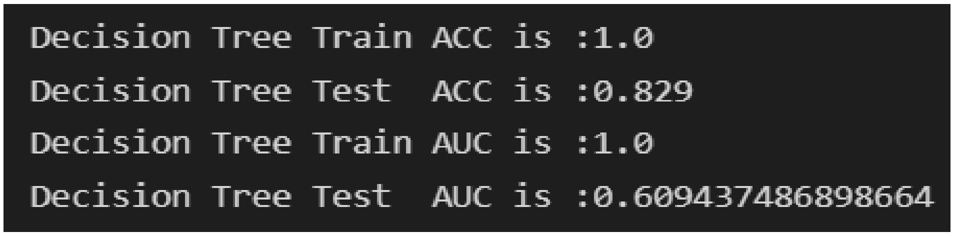
- Complex
- Hard to interpret.
- OverFitting

Model Benchmark – Decision Tree:

Steps Followed:

1. First Decision Model is applied with all the features.
2. Then Features are selected using Sequential Forward Stepwise.
3. Parameters are selected and the criterion is Gini with depth 7 and sample leafs 5

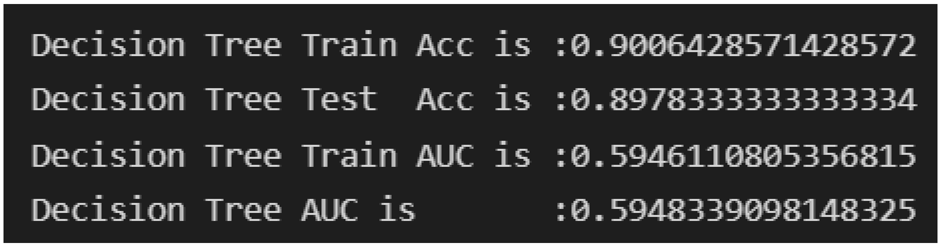
Initial Results :



```
Decision Tree Train ACC is :1.0
Decision Tree Test  ACC is :0.829
Decision Tree Train AUC is :1.0
Decision Tree Test  AUC is :0.609437486898664
```

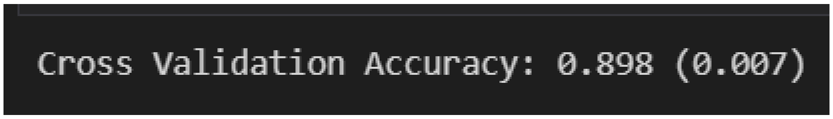
- It is clearly evident that decision tree is overfitting the model and this needs to be addressed.

- After feature selection and parameter tuning, we can still see there is a bit of overfitting but still better than the initial results



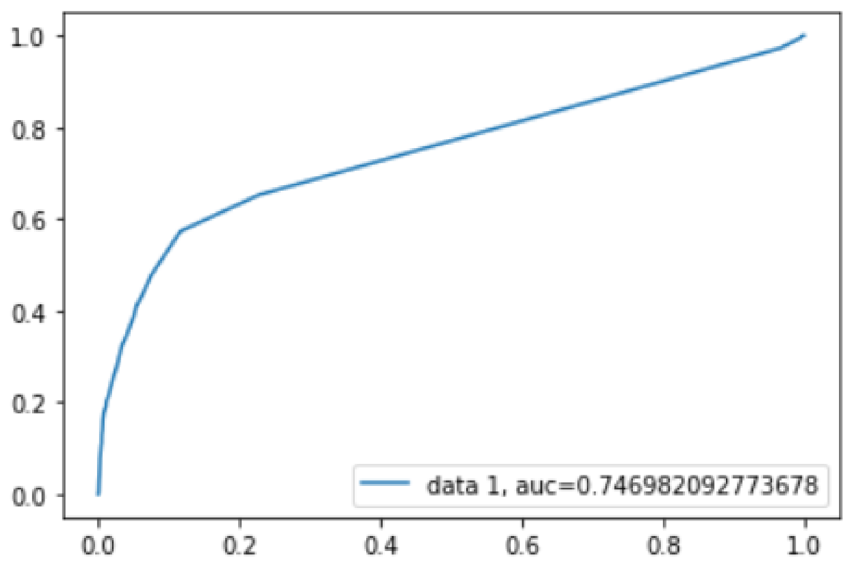
```
Decision Tree Train Acc is :0.9006428571428572
Decision Tree Test  Acc is :0.8978333333333334
Decision Tree Train AUC is :0.5946110805356815
Decision Tree AUC is       :0.5948339098148325
```

- Cross Validation accuracy is close to the test accuracy but still suggests overfitting still exists.



```
Cross Validation Accuracy: 0.898 (0.007)
```

ROC Curve :



RANDOM FOREST :

Random Forest is Supervised Machine Learning Approach which can be used for both Classification and Regression problems. It is based on ensemble learning where multiple classifiers are combined to solve the problem. From random data samples Random Forest creates decision Trees and gets predictions for each tree and selects the best Solution by means of Voting.

- Better to have some actual values in the features of the data to predict accurate results
- The predictions must be less correlated from each tree
- Greater the number of trees in the forest, higher is the Accuracy and prevents overfitting.
- Also helps for the feature selection.

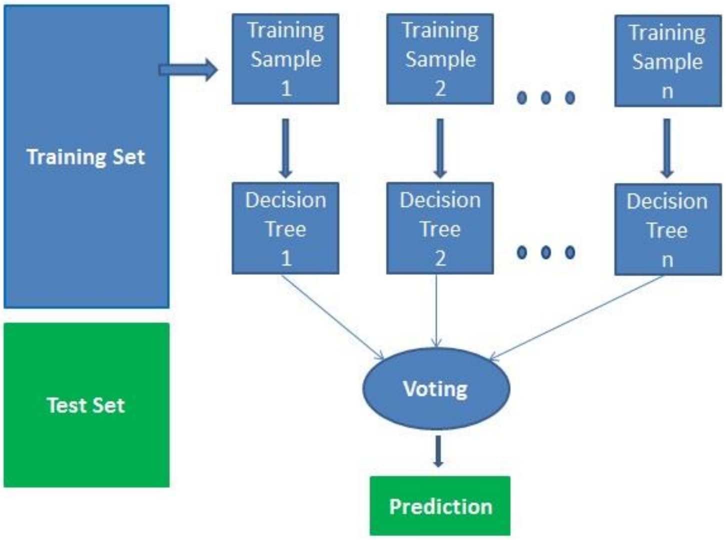


Image Source : Datacamp

WORKING – RANDOM FOREST

- Random Samples were selected from the data

- Tree is built for each model and the prediction result will be collection from each tree.
- Voting for each prediction
- Prediction of the tree with most voting will be considered as final Prediction.

Pros – Random Forest :

- Highly accurate and efficient.
- Can overcome overfitting
- Can be used for both classification and Regression Problems

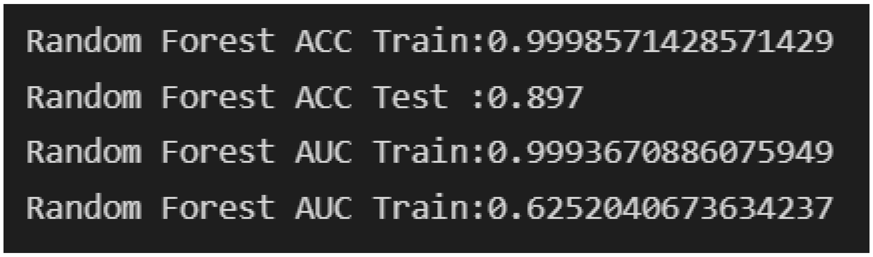
Cons – Random Forest:

- Complex and Time consuming.
- Difficult to interpret.

Model Benchmark – Random Forest :

Steps Followed:

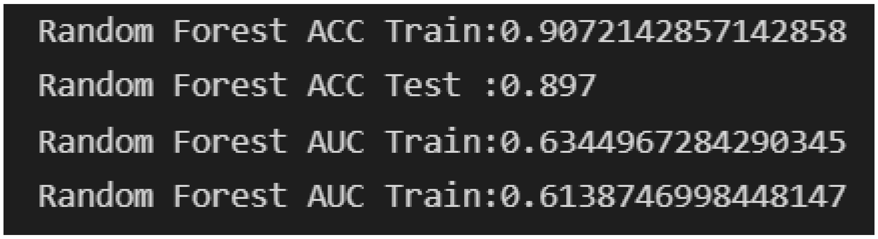
1. First, Model is applied on the whole dataset
2. Selected the features using forward Stepwise
3. RandomSearch for the best parameters
4. Fit the model again with the selected features and parameters.
5. Cross validate to check the accuracy.
   Initial Metrics:



```
Random Forest ACC Train:0.9998571428571429
Random Forest ACC Test :0.897
Random Forest AUC Train:0.9993670886075949
Random Forest AUC Train:0.6252040673634237
```

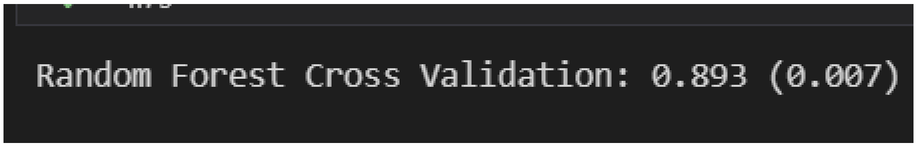
- Initial results definitely suggest the Random Forest model is overfitting and needs to be addressed.
- Below is the result after the feature selection and the parameter tuning using randomSearchCV.

- Overfitting has been handled to some extend but still we can the train acc and auc are higher than test data.



```
Random Forest ACC Train:0.9072142857142858
Random Forest ACC Test :0.897
Random Forest AUC Train:0.6344967284290345
Random Forest AUC Train:0.6138746998448147
```

- Cross Validation Accuracy :



```
Random Forest Cross Validation: 0.893 (0.007)
```

K-NEAREST NEIGHBOURS :

It is one of the simple supervised Machine Learning Model. No assumption is made by KNN for the data distribution. Model assumes the similarity between new data and existing data and categorize the new data based on existing categories. Like Random Forest and most of the tree-based Models, KNN can be used for both Regression and Classification problems. Model performs better with the smaller number of features.

KNN-Working: For KNN, number of neighbors is the main factor

- Select K
- Calculate distance (Euclidean, Manhattan etc.)
- *Find out Nearest Neighbors based on the calculated Distance.*
- Assign categories to the new data.

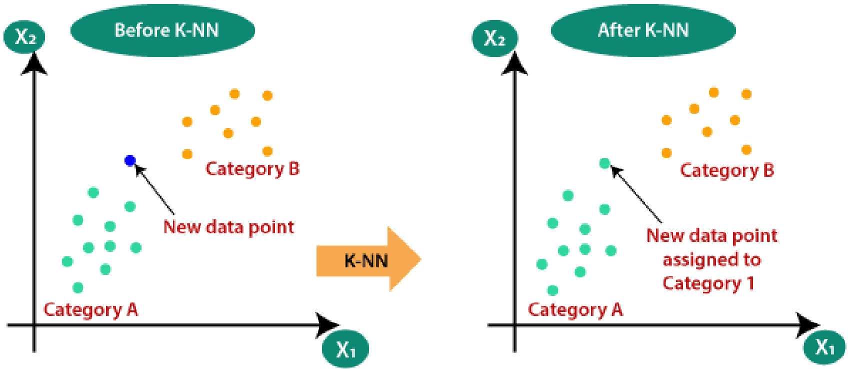


Image Source : TowardsDatascience

From the above image, we can see that the new data point which is blue in color is classified to category A, after applying KNN. KNN finds it as the nearest neighbor to the Category A based on the distance.

Pros – KNN :

- Simple to Implement
- High Accuracy
- Can handle noise well
- Can be used for regression and classification Problems.

Cons – KNN:

- Need to determine the value of K every time and sometimes it might become complex
- Works well with a smaller number of features but struggles with more features.
- Sensitive to outliers
- Cannot handle Missing values

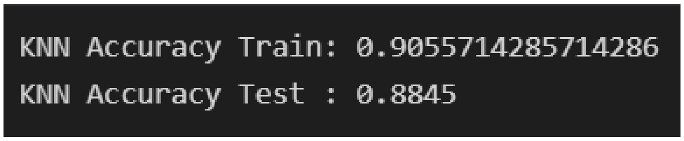
Model Benchmark – KNN :

Steps Followed :

1. Initially applied the model with all the Features.

2. Using Forward Step feature selection, selected 20 Features.
3. Selected the Parameters 7 nearest neighbors and algorithm is brute.
4. Fit the model with selected parameters
5. Cross Validate to verify the accuracy.

Initial Results with all the Features :

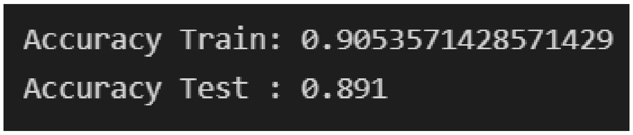


```
KNN Accuracy Train: 0.9055714285714286
KNN Accuracy Test : 0.8845
```

- KNN is overfitting the model as can be seen from the results.

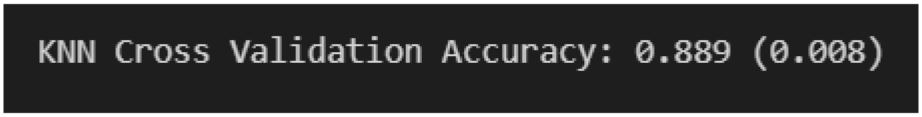
Results with the selected features and Parameters:

- Like RandomForest and Decision Tree, KNN tends to overfit the model a bit.

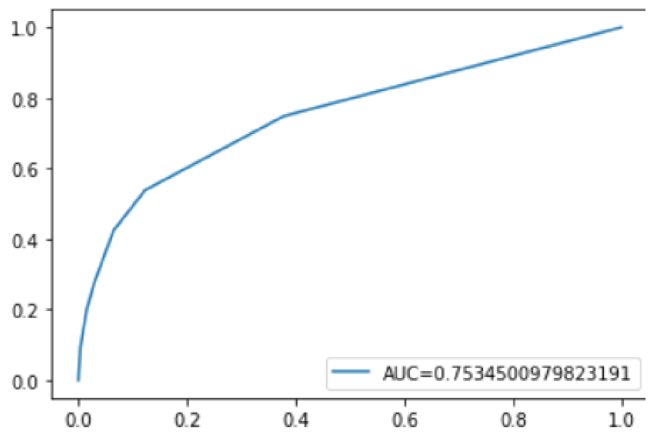


```
Accuracy Train: 0.9053571428571429
Accuracy Test : 0.891
```

Cross Validation – KNN:



```
KNN Cross Validation Accuracy: 0.889 (0.008)
```



ADAPTIVE BOOST(ADABOOST) :

When it comes to achieving higher accuracies and accurate predictions boosting algorithms are the better choice because of the way they handle the data. Boosting Models combine multiple low accuracy models to create a high accuracy model. AdaBoost is one of the ensemble boosting algorithm.

Boosting : are the set of low accurate classifiers to create a high accurate classifier and have a better accuracy than a random guess. They can deal with the overfitting problem.

- Adaboost sets weights of classifiers and trains the data sample in an iteration to ensure accurate predictions

Working of Adaboost :

1. Selects training subset randomly.
2. Iteratively trains the Model based on the predictions from the previous training.
3. Wrongly classified observations will be given more weight so that in next iteration they will get high probability for classification.
4. This continues till the training data fits without any error.
5. In the end , perform voting on all the built learning algorithms.

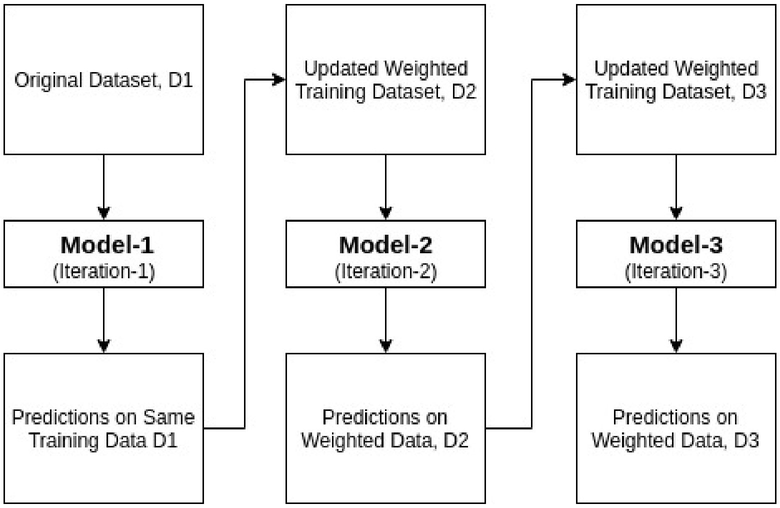

Original Dataset, D1
Model-1
(Iteration-1)
Predictions on Same Training Data D1
Updated Weighted Training Dataset, D2
Model-2
(Iteration-2)
Predictions on Weighted Data, D2
Updated Weighted Training Dataset, D3
Model-3
(Iteration-3)
Predictions on Weighted Data, D3

Image Source : Datacamp

Pros – AdaBoost :

- Easy to implement.
- Highly efficient and accuracte.
- Not prône to Overfitting.

Cons -AdaBoost :

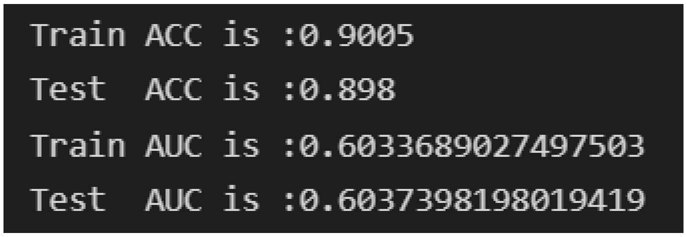
- Sensitive to noise.
- Sensitive to Outliers.
- Latency, compared to other Boosting Algorithms.

Model Benchmark – AdaBoost :

Steps Followed:

1. Just like all the models, Adaboost is applied to the whole data set initially.
2. Selected Features using Forward Stepwise.
3. Applied GridSearchCv to get the best set of parameters for tuning the model.
4. With the parameters and the selected Features applied the model again.
5. Cross validated with the accuracy using K-Fold CV.

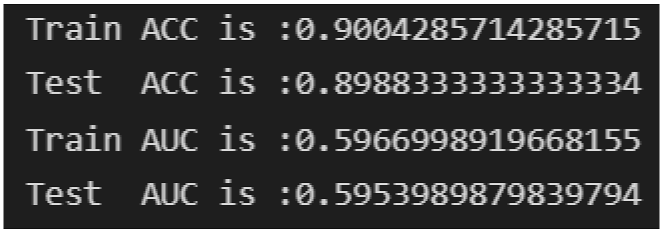
Initial Results Metrics:



```
Train ACC is :0.9005
Test  ACC is :0.898
Train AUC is :0.6033689027497503
Test  AUC is :0.6037398198019419
```

Results with the Features selected and Parameters:

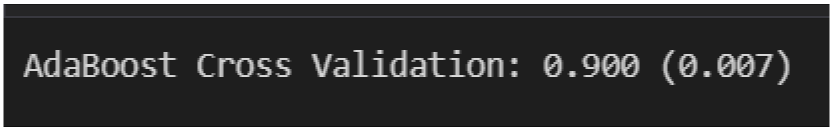
- Below is the results after selecting the features using Forward Stepwise and tuning the parameters using GridsearchCV
- We can see there is no much difference in the results and the model has bit of overfitting.



```
Train ACC is :0.9004285714285715
Test  ACC is :0.8988333333333334
Train AUC is :0.5966998919668155
Test  AUC is :0.5953989879839794
```

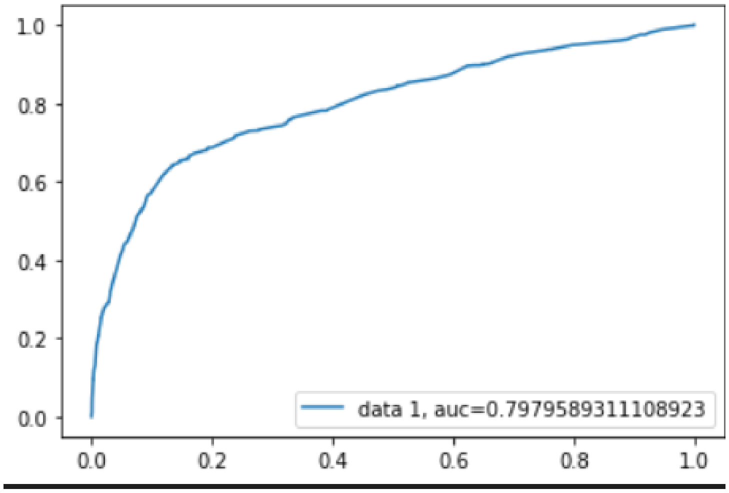
Cross Validation for AdaBoost:

- Cross Validation score is close to the Final Adaboost scores.



```
AdaBoost Cross Validation: 0.900 (0.007)
```

ROC Curve:



REFERENCES:

- https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148
- *https://machinelearningmastery.com/adaboost-ensemble-in-python/*
- https://www.javatpoint.com/machine-learning-random-forest-algorithm
- https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
- https://medium.com/@anuuz.soni/advantages-and-disadvantages-of-knn-ee06599b9336
- https://medium.com/@chaudhurysrijani/tuning-of-adaboost-with-computational-complexity-8727d01a9d20
- https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/
- https://www.kdnuggets.com/2020/02/decision-tree-intuition.html
- https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm