INTERVIEW PREDICTION

BY MACHINE LEARNING USING PYTHON

Project Done by:

PRAVASH RANJAN NAYAK, KALYANI GOVT. ENGG. COLLEGE, 161020110024 of 2016-17

SYEDA ZARIF AFRIN, KALYANI GOVT. ENGG. COLLEGE, 161020110047 of 2016-17

AMAN KUMAR GUPTA, KALYANI GOVT. ENGG. COLLEGE, 161020110004 of 2016-17

AKASH RAY, KALYANI GOVT. ENGG. COLLEGE, 161020110051 of 2016-17

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Table of Contents

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their co-operation during the period of my assignment.

- **Syeda Zarif Afrin**
- **Akash Ray**
- **Aman Kumar Gupta**
- **Pravash Ranjan Nayak**

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

PROJECT SCOPE

Interview Attendance prediction is an important assessment conducted by the HR companies to minimize the overall expenditure with respect to money as well as manpower.

HR companies conducting interviews are required to make prior arrangements based on the number of candidates that are expected to turn up for the interview. A rough idea has to be taken so that there are neither any shortcomings nor excessive disbursement of funds for the preparation.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

PROJECT OBJECTIVE

One part of the task is that we need to predict whether a candidate who has applied for the interview will turn up or not. Another part of the task includes observing any hidden patterns in the dataset ,i.e, if a candidate appears for the interview then what factors influence his decision to be present on the date of the interview. For example, it might happen that candidate from a particular location have a high chance of not turning up for the interview. So, the second aspect of the project was to observe these patterns.

[illegible]

Globsyn Knowledge Foundation Kolkata

Performance Metrics

Ours is a classification problem, therefore, we need to take into account the number of misclassifications into account. This can be achieved through Confusion Matrix. Delving deep into the problem statement , we need to consider the Precision score for choosing our model, since if we predict a candidate to not come (0) but if he arrives (1) for the interview then it might not harm us. However, if we predict him to come (1) but he doesnt turn up (0), then it will be problematic situation. Hence we need to minimise False Positives in order to keep the Precision high.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

LOGISTIC REGRESSION MODEL

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Linear Regression Equation:

$$\boxed{y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn}$$

Where, y is dependent variable and x1, x2 ... and Xn are explanatory variables.

Sigmoid Function:

$$\boxed{p = 1/1 + e^{-y}}$$

Apply Sigmoid function on linear regression:

$$\boxed{p = 1/1 + e^{-(\beta 0 + \beta 1X1 + \beta 2X2 \ldots\ldots \beta nXn)}}$$

Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.
- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

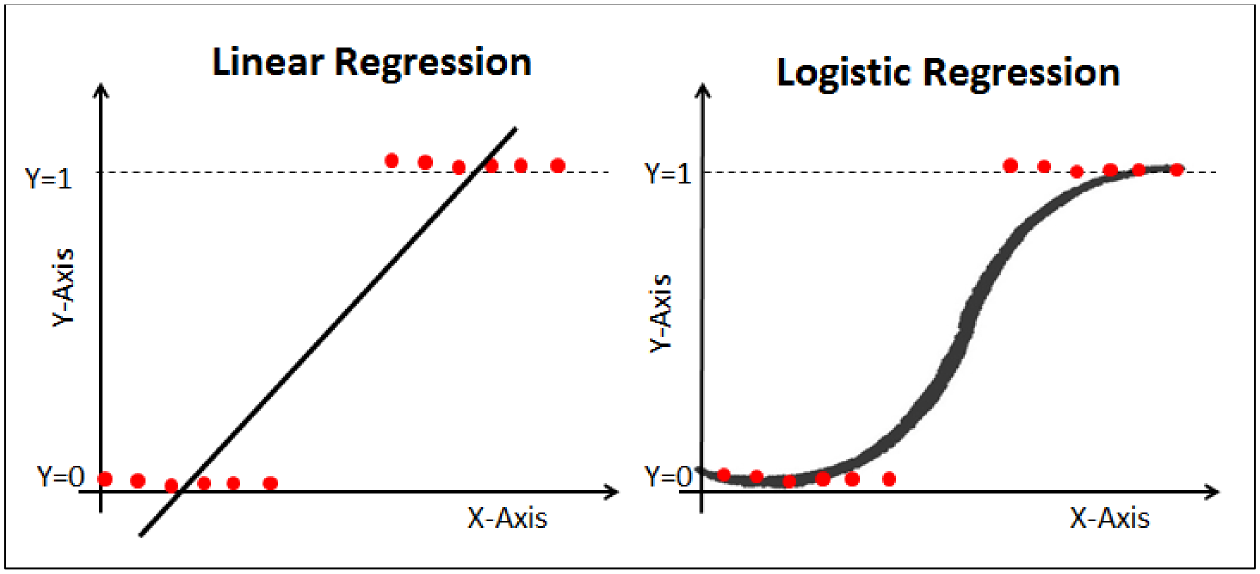
[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Linear Regression Vs. Logistic Regression

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.



Maximum Likelihood Estimation Vs. Least Square Method

The MLE is a "likelihood" maximization method, while OLS is a distance-minimizing approximation method. Maximizing the likelihood function determines the parameters that are most likely to produce the observed data. From a statistical point of view, MLE sets the mean and variance as parameters in determining the specific parametric values for a given model. This set of parameters can be used for predicting the data needed in a normal distribution.

Ordinary Least squares estimates are computed by fitting a regression line on given data points that has the minimum sum of the squared deviations (least square error). Both are used to estimate the parameters of a linear regression model. MLE assumes a joint probability mass function, while OLS doesn't require any stochastic assumptions for minimizing distance.

[illegible]

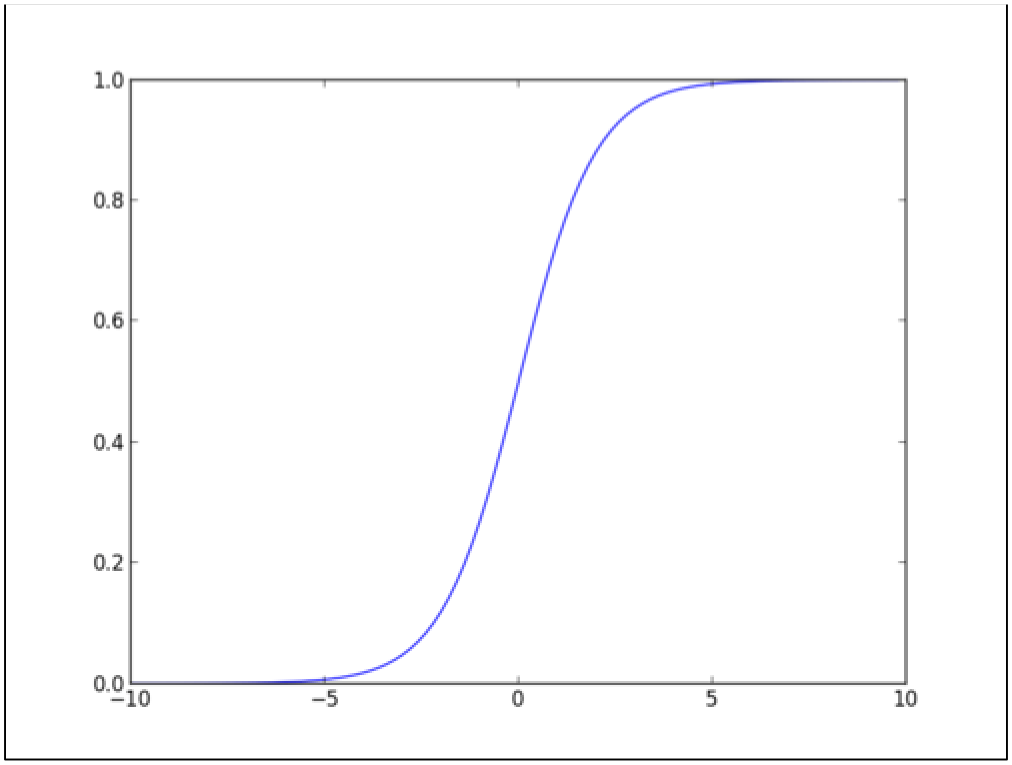
Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Sigmoid Function

The sigmoid function, also called logistic function gives an ‘S’ shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The outputcannotFor example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that patient will suffer from cancer.

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



Types of Logistic Regression

[illegible]

Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

NAIVE-BAYES MODEL

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

$$\boxed{P(h|D) = \frac{P(D|h)P(h)}{P(D)}}$$

- P(h): the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h.
- P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.
- P(D|h): the probability of data d given that the hypothesis h was true. This is known as posterior probability.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

How Naive Bayes classifier works?

Let’s understand the working of Naive Bayes through an example. Given an example of weather conditions and playing sports. You need to calculate the probability of playing sports. Now, you need to classify whether players will play or not, based on the weather condition.

First Approach (In case of a single feature)

Naive Bayes classifier calculates the probability of an event in the following steps:

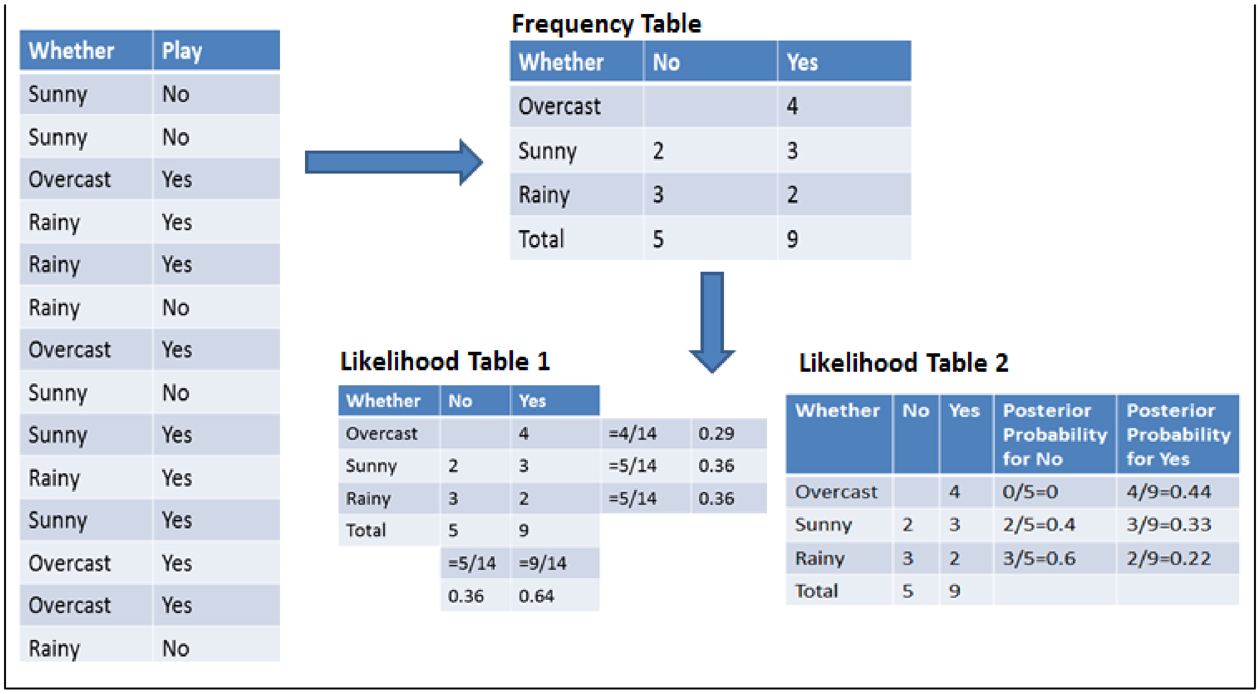
- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for each class
- Step 3: Put these value in Bayes Formula and calculate posterior probability.
- Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

For simplifying prior and posterior probability calculation you can use the two tables frequency and likelihood tables. Both of these tables will help you to calculate the prior and posterior probability. The Frequency table contains the occurrence of labels for all features. There are two likelihood tables. Likelihood Table 1 is showing prior probabilities of labels and Likelihood Table 2 is showing the posterior probability.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019



Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table

Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1

Whether	No	Yes		
Overcast		4	=4/14	0.29
Sunny	2	3	=5/14	0.36
Rainy	3	2	=5/14	0.36
Total	5	9		
	=5/14	=9/14		
	0.36	0.64		

Likelihood Table 2

Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

Now suppose you want to calculate the probability of playing when the weather is overcast.

Probability of playing:

P(Yes | Overcast) = P(Overcast | Yes) P(Yes) / P (Overcast) ....(1)

1. Calculate Prior Probabilities:
2. P(Overcast) = 4/14 = 0.29
3. P(Yes)= 9/14 = 0.64

1. Calculate Posterior Probabilities:
2. P(Overcast |Yes) = 4/9 = 0.44

1. Put Prior and Posterior probabilities in equation (1)
2. P (Yes | Overcast) = 0.44 * 0.64 / 0.29 = 0.98(Higher)

Similarly, you can calculate the probability of not playing:

Probability of not playing:

P(No | Overcast) = P(Overcast | No) P(No) / P (Overcast) ....(2)

[illegible]

Globsyn Knowledge Foundation Kolkata

1. Calculate Prior Probabilities:
2. P(Overcast) = 4/14 = 0.29
3. P(No)= 5/14 = 0.36

1. Calculate Posterior Probabilities:
2. P(Overcast |No) = 0/9 = 0

1. Put Prior and Posterior probabilities in equation (2)
2. P (No | Overcast) = 0 * 0.36 / 0.29 = 0

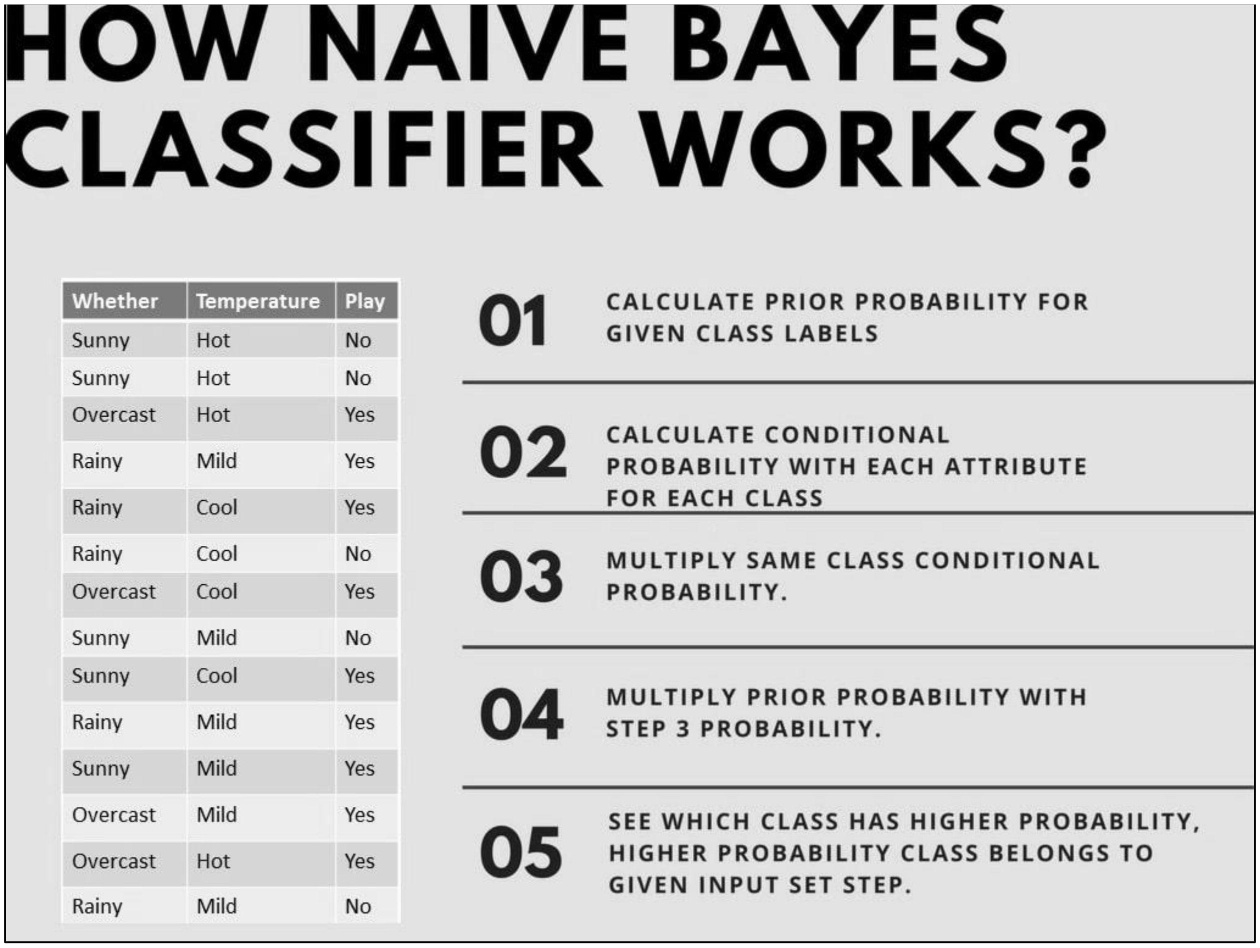
The probability of a 'Yes' class is higher. So you can determine here if the weather is overcast than players will play the sport.

Second Approach (In case of multiple features)

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019



HOW NAIVE BAYES CLASSIFIER WORKS?

Whether	Temperature	Play
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	No
Sunny	Cool	Yes
Rainy	Mild	Yes
Sunny	Mild	Yes
Overcast	Mild	Yes
Overcast	Hot	Yes
Rainy	Mild	No

01 CALCULATE PRIOR PROBABILITY FOR GIVEN CLASS LABELS

02 CALCULATE CONDITIONAL PROBABILITY WITH EACH ATTRIBUTE FOR EACH CLASS

03 MULTIPLY SAME CLASS CONDITIONAL PROBABILITY.

04 MULTIPLY PRIOR PROBABILITY WITH STEP 3 PROBABILITY.

05 SEE WHICH CLASS HAS HIGHER PROBABILITY, HIGHER PROBABILITY CLASS BELONGS TO GIVEN INPUT SET STEP.

Now suppose you want to calculate the probability of playing when the weather is overcast, and the temperature is mild.

Probability of playing:

*P(Play= Yes | Weather=Overcast, Temp=Mild) = P(Weather=Overcast, Temp=Mild | Play= Yes)*P(Play=Yes) ....(1)

P(Weather=Overcast, Temp=Mild | Play= Yes)= P(Overcast |Yes) P(Mild |Yes) ………..(2)

1. Calculate Prior Probabilities: P(Yes)= 9/14 = 0.64
2. Calculate Posterior Probabilities: P(Overcast |Yes) = 4/9 = 0.44 P(Mild |Yes) = 4/9 = 0.44
3. Put Posterior probabilities in equation (2) P(Weather=Overcast, Temp=Mild | Play= Yes) = 0.44 * 0.44 = 0.1936(Higher)

[illegible]

Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

4. Put Prior and Posterior probabilities in equation (1) P(Play= Yes | Weather=Overcast, Temp=Mild) = 0.1936*0.64 = 0.124

Similarly, you can calculate the probability of not playing:

Probability of not playing:

*P(Play= No | Weather=Overcast, Temp=Mild) = P(Weather=Overcast, Temp=Mild | Play= No)*P(Play=No) ....(3)

P(Weather=Overcast, Temp=Mild | Play= No)= P(Weather=Overcast |Play=No) P(Temp=Mild | Play=No) ………..(4)

1. Calculate Prior Probabilities: P(No)= 5/14 = 0.36
2. Calculate Posterior Probabilities: P(Weather=Overcast |Play=No) = 0/9 = 0 P(Temp=Mild | Play=No)=2/5=0.4
3. Put posterior probabilities in equation (4) P(Weather=Overcast, Temp=Mild | Play= No) = 0 * 0.4= 0
4. Put prior and posterior probabilities in equation (3) P(Play= No | Weather=Overcast, Temp=Mild) = 0*0.36=0

The probability of a 'Yes' class is higher. So you can say here that if the weather is overcast than players will play the sport.

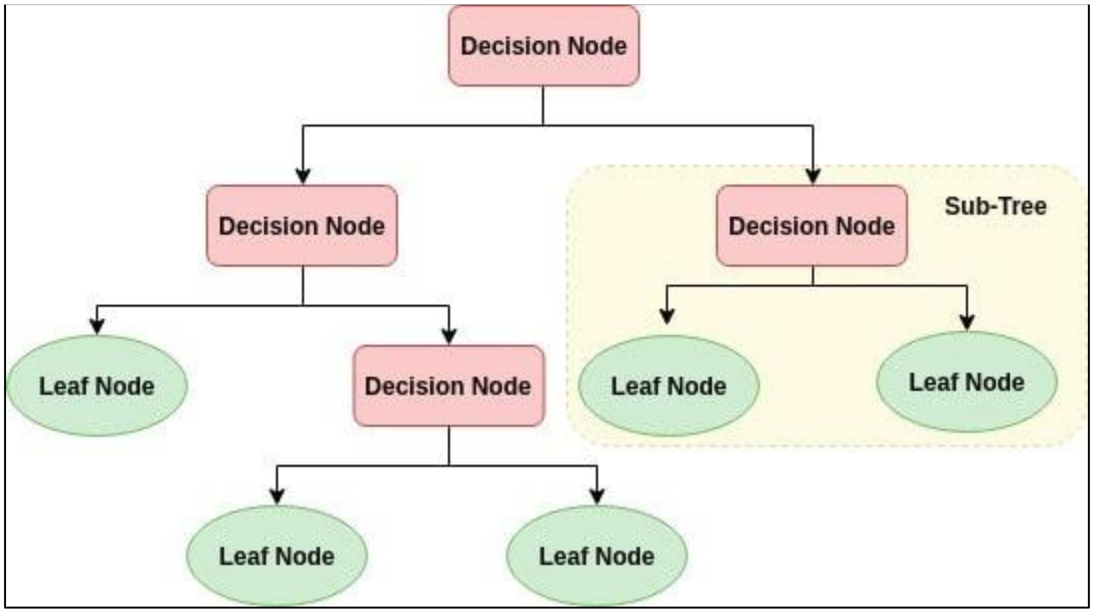
DECISION TREE

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

How does the Decision Tree algorithm work?

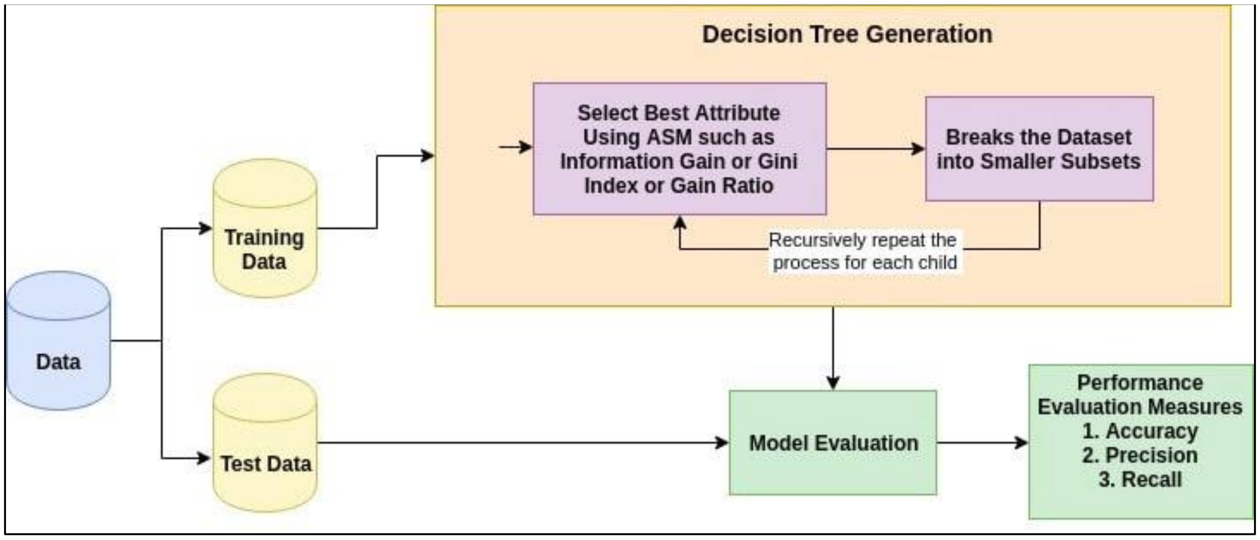
The basic idea behind any decision tree algorithm is as follows:

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
    - All the tuples belong to the same attribute value.
    - There are no more remaining attributes.
    - There are no more instances.


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Attribute Selection Measures

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute. In the case of a continuous-valued attribute, split points for branches also need to define. Most popular selection measures are Information Gain, Gain Ratio, and Gini Index.

Information Gain

[illegible]

Globsyn Knowledge Foundation Kolkata

Shannon invented the concept of entropy, which measures the impurity of the input set. In physics and mathematics, entropy referred as the randomness or the impurity in the system. In information theory, it refers to the impurity in a group of examples. Information gain is the decrease in entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

$$\boxed{\text{Info(D)} = - \sum_{i=1}^{m} pi \log_2 pi}$$

Where, Pi is the probability that an arbitrary tuple in D belongs to class Ci.

$$\boxed{\text{Info}_A(\text{D}) = \sum_{j=1}^{V} \frac{|Dj|}{|D|} \quad \text{X Info}(\text{D}_j)}$$

$$\boxed{\text{Gain(A)} = \text{Info(D)} - \text{Info}_A(\text{D})}$$

Where,

- Info(D) is the average amount of information needed to identify the class label of a tuple in D.
- |Dj|/|D| acts as the weight of the jth partition.
- InfoA(D) is the expected information required to classify a tuple from D based on the partitioning by A.

The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N().

Gain Ratio

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Information gain is biased for the attribute with many outcomes. It means it prefers the attribute with a large number of distinct values. For instance, consider an attribute with a unique identifier such as customer_ID has zero info(D) because of pure partition. This maximizes the information gain and creates useless partitioning.

C4.5, an improvement of ID3, uses an extension to information gain known as the gain ratio. Gain ratio handles the issue of bias by normalizing the information gain using Split Info. Java implementation of the C4.5 algorithm is known as J48, which is available in WEKA data mining tool.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Where,

- |Dj|/|D| acts as the weight of the jth partition.
- v is the number of discrete values in attribute A.

The gain ratio can be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the highest gain ratio is chosen as the splitting attribute.

Gini index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$\text{Gini(D)}=1 - \sum_{i=1}^{m} \text{Pi}^2$$

Where, pi is the probability that a tuple in D belongs to class Ci.

[illegible]

Globsyn Knowledge Foundation Kolkata

The Gini Index considers a binary split for each attribute. You can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into D1 and D2, the Gini index of D is:

$$\boxed{\mathrm{Gini}_A(D)=\frac{|D1|}{|D|}\,\mathrm{Gini}(D_1)+\frac{|D2|}{|D|}\,\mathrm{Gini}(D_2)}$$

In case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split-point and point with smaller gini index chosen as the splitting point.

$$\boxed{\Delta Gini(A) = Gini(D) - Gini_A(D).}$$

The attribute with minimum Gini index is chosen as the splitting attribute.

[illegible]

Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

RANDOM FOREST

Let’s understand the algorithm in layman’s terms. Suppose you want to go on a trip and you would like to travel to a place which you will enjoy.

So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends.

Let’s suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip.

In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far.

The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forests algorithm.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

[illegible]

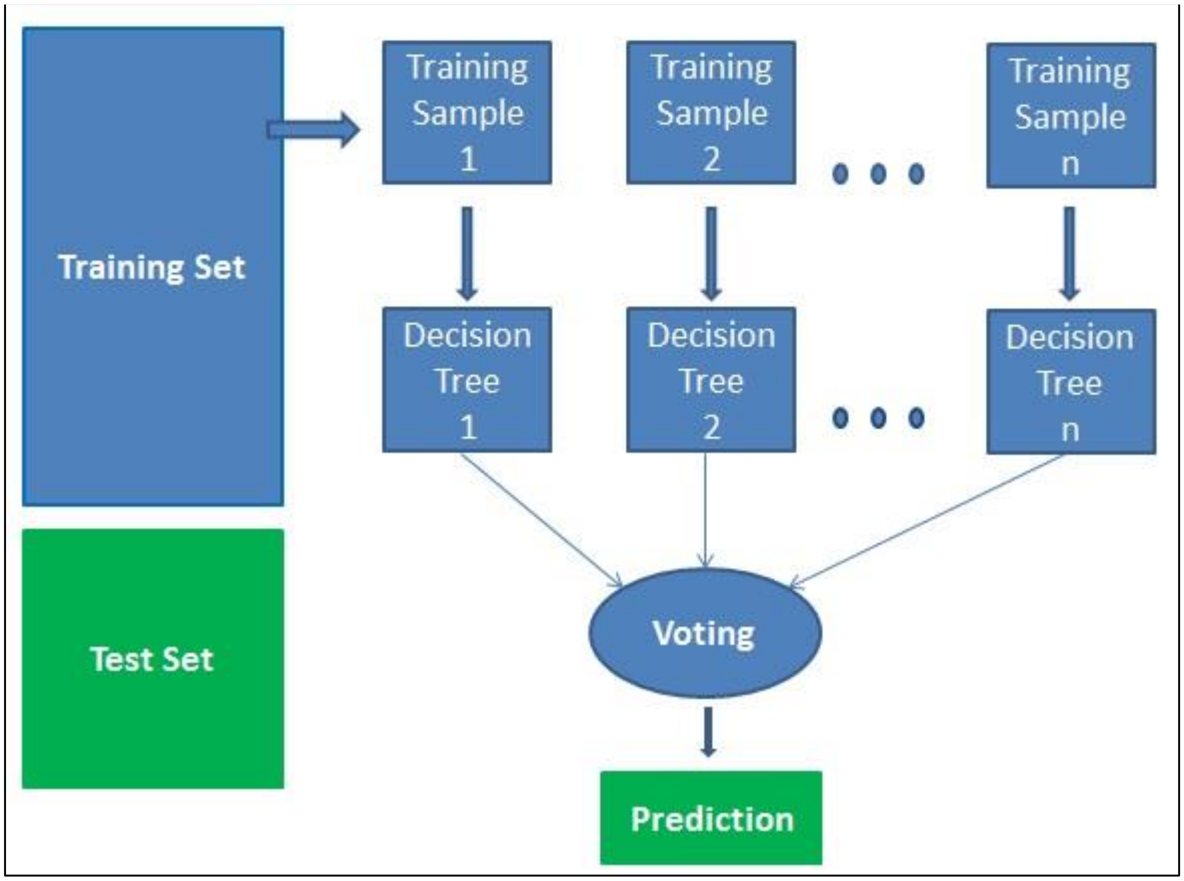
Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

How does the algorithm work?

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.



Advantages:

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages:

- Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Finding important features

Random forests also offers a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.
This score will help you choose the most important features and drop the least important ones for model building.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

Random Forests vs Decision Trees

- Random forests is a set of multiple decision trees.
- Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- Decision trees are computationally faster.
- Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

K-NEAREST NEIGHBOR

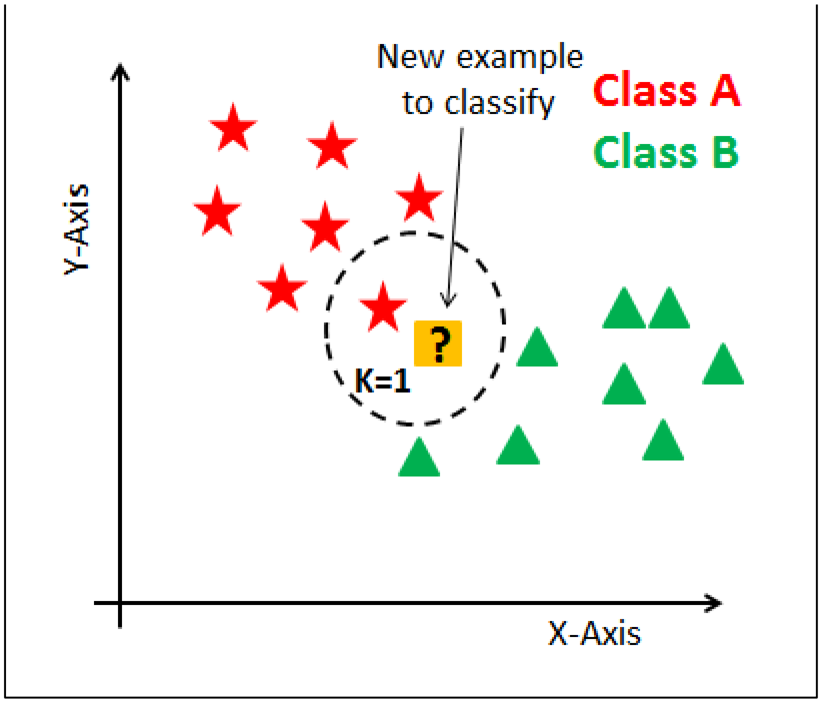
KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

How does the KNN algorithm work?

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.

[illegible]

Globsyn Knowledge Foundation Kolkata



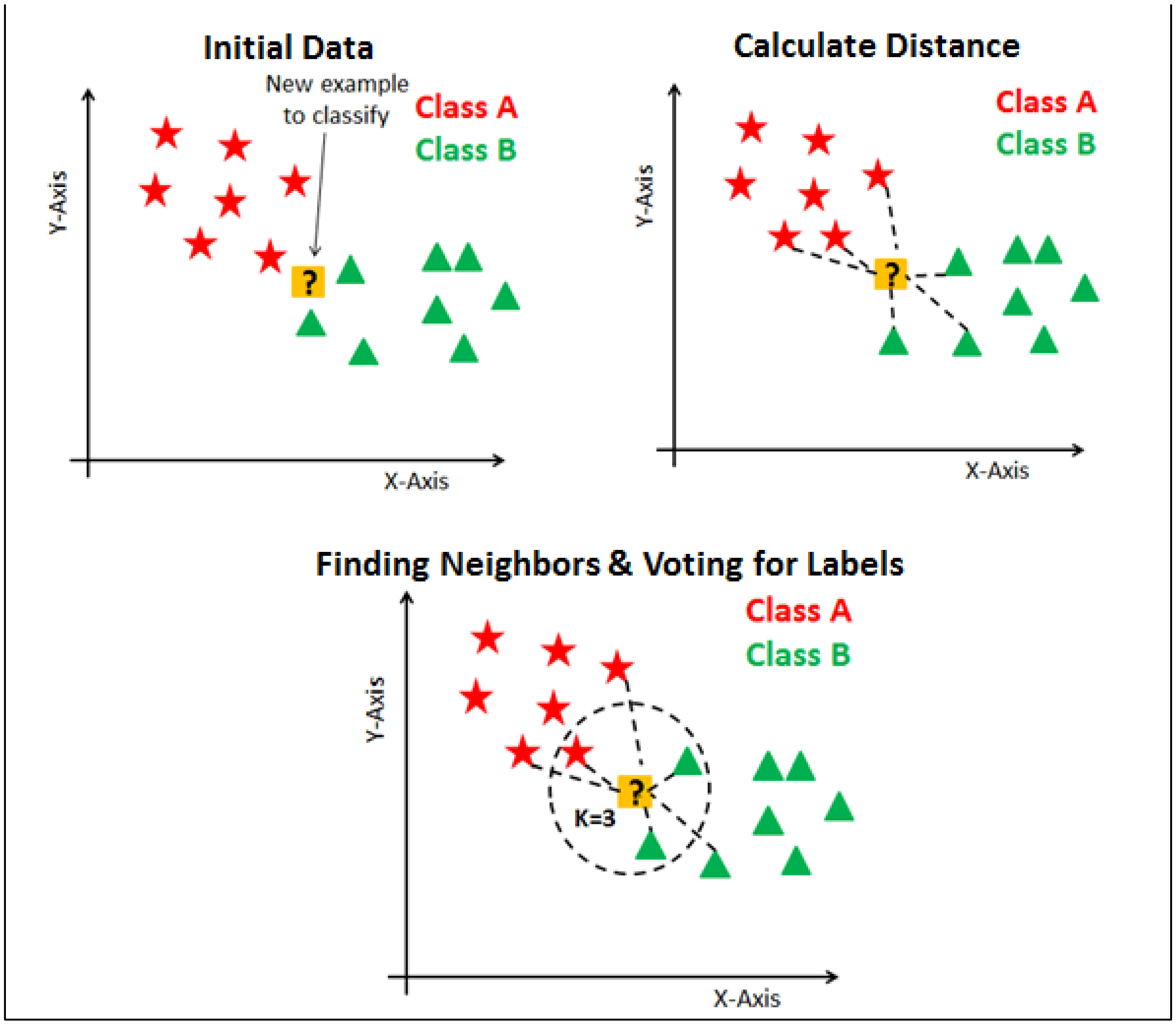
Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels

[illegible]

Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019



Eager Vs. Lazy Learners

Eager learners mean when given training points will construct a generalized model before performing prediction on given new points to classify. You can think of such learners as being ready, active and eager to classify unobserved data points.

Lazy Learning means there is no need for learning or training of the model and all of the data points used at the time of prediction. Lazy learners wait until the last minute before classifying any data point. Lazy learner stores merely the training dataset and waits until classification needs to perform. Only when it sees the test tuple does it perform generalization to classify the tuple based on its similarity to the stored training tuples. Unlike eager learning methods, lazy

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

learners do less work in the training phase and more work in the testing phase to make a classification. Lazy learners are also known as instance-based learners because lazy learners store the training points or instances, and all learning is based on instances.

Curse of Dimensionality

KNN performs better with a lower number of features than a large number of features. You can say that when the number of features increases than it requires more data. Increase in dimension also leads to the problem of overfitting. To avoid overfitting, the needed data will need to grow exponentially as you increase the number of dimensions. This problem of higher dimension is known as the Curse of Dimensionality.

To deal with the problem of the curse of dimensionality, you need to perform principal component analysis before applying any machine learning algorithm, or you can also use feature selection approach. Research has shown that in large dimension Euclidean distance is not useful anymore. Therefore, you can prefer other measures such as cosine similarity, which get decidedly less affected by high dimension.

How do you decide the number of neighbors in KNN?

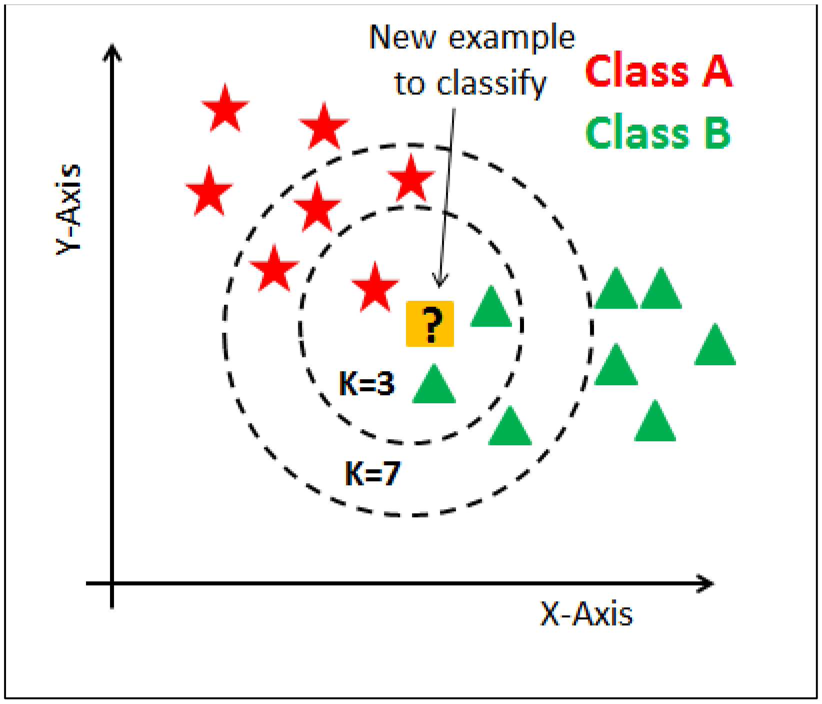
Now, you understand the KNN algorithm working mechanism. At this point, the question arises that How to choose the optimal number of neighbors? And what are its effects on the classifier? The number of neighbors(K) in KNN is a hyperparameter that you need choose at the time of model building. You can think of K as a controlling variable for the prediction model.

Research has shown that no optimal number of neighbors suits all kind of data sets. Each dataset has it's own requirements. In the case of a small number of neighbors, the noise will have a higher influence on the result, and a large number of neighbors make it computationally expensive. Research has also shown that a small amount of neighbors are most flexible fit which will have low bias but high variance and a large number of neighbors will have a smoother decision boundary which means lower variance but higher bias.

[illegible]

Globsyn Knowledge Foundation Kolkata

Generally, Data scientists choose as an odd number if the number of classes is even. You can also check by generating the model on different values of k and check their performance. You can also try Elbow method here.



[illegible]

Globsyn Knowledge Foundation
Kolkata

Document sign date :May 3, 2019

DATA DESCRIPTION

1. Date of Interview : date

Date of Interview of the candidate.

Data Type : String

Value Type : Categorical

Null Value : 1

2. Client Name : client

Name of the client for which the interview is being conducted.

Data Type : String

Value Type : Categorical

Null Value : 0

3. Industry : industry

The type of industry for which the interview is being conducted.

Data Type : String

Value Type : Categorical

Null Value : 1

4. Location : location

Interview Location.

Data Type : String

Value Type : Categorical

Null Value : 1

5. Position to be closed : position

[illegible]

Globsyn Knowledge Foundation Kolkata

The position for which the interview is being conducted.

Data Type : String

Value Type : Categorical

Null Value : 1

6. Nature of Skillset : skills

The skills that the candidate possesses.

Data Type : String

Value Type : Categorical

Null Value : 1

7. Interview Type : inter_type

The type of interview that will be conducted.

Data Type : String

Value Type : Categorical

Null Value : 1

8. Name(Cand ID) : ID

ID of the candidate.

Data Type : String

Value Type : Categorical

Null Value : 1

9. Gender : gender

Gender of the candidate.

Data Type : String

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Value Type : Categorical

Null Value : 1

10. Candidate Current Location : curr_location

Current Location of the candidate.

Data Type : String

Value Type : Categorical

Null Value : 1

11. Candidate Job Location : job_location

The location where the candidate will be offered the job.

Data Type : String

Value Type : Categorical

Null Value : 1

12. Interview Venue : venue

The venue where the interview will be conducted.

Data Type : String

Value Type : Categorical

Null Value : 1

13. Candidate Native Location : native_location

The native location of the candidate.

Data Type : String

Value Type : Categorical

Null Value : 1

14. Have you obtained the necessary permission to start at the required time : permission

[illegible]

Globsyn Knowledge Foundation Kolkata

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 205

15. Hope there will be no unscheduled meetings : hope

Is there any possibility of an unscheduled meeting.

Data Type : String

Value Type : Categorical

Null Value : 248

16. Can I Call you three hours before the interview and follow up on your attendance for the interview : three_hours

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 248

17. Can I have an alternative number/ desk number. I assure you that I will not trouble you too much : alt_number

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 248

18. Have you taken a printout of your updated resume. Have you read the JD and understood the same : resume

Self-explanatory.

Data Type : String

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Value Type : Categorical

Null Value : 249

19. Are you clear with the venue details and the landmark. : clear_about_venue

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 249

20. Has the call letter been shared : letter

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 246

21. Expected Attendance : expected

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 6

22. Observed Attendance : observed

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 1

[illegible]

Globsyn Knowledge Foundation Kolkata

23. Marital Status : married

Self-explanatory.

Data Type : String

Value Type : Categorical

Null Value : 1

24. Unnamed : 23

Garbage Values.

25. Unnamed : 24

Garbage Values.

26. Unnamed : 25

Garbage Values.

27. Unnamed : 26

Garbage Values.

28. Unnamed : 27

Garbage Values.

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

DATA ANALYSIS:

Observations:

- Looking into the data set , we can say that we have 1234 records with 28 columns/features.
- Firstly, we can clearly elIminate the last 5 columns , which have only null values.
- Secondly, we need to shorten and modify the column in order to make the data more interpretable.
- We need to clean the data before modelling.

For now we are removing all the 253 null values we have in our data set. Later on, we shall impute the null values and see if we get any better results.

After cleaning we have :

- 23 columns all of which are non-ordinal categorical in nature
- We have 12 columns which are of object-data type

[illegible]

Globsyn Knowledge Foundation Kolkata

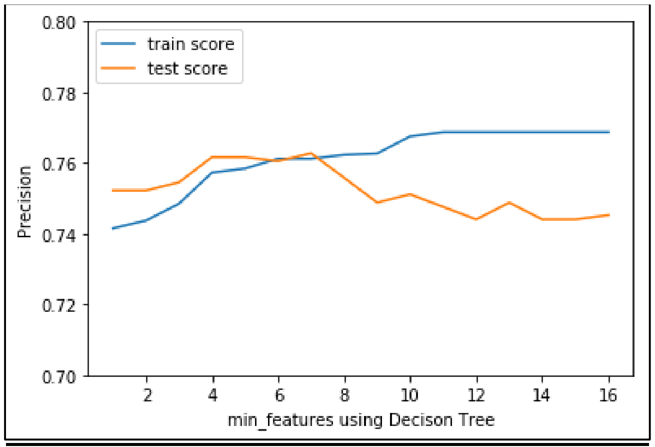
Document sign date :May 3, 2019

MODEL BUILDING

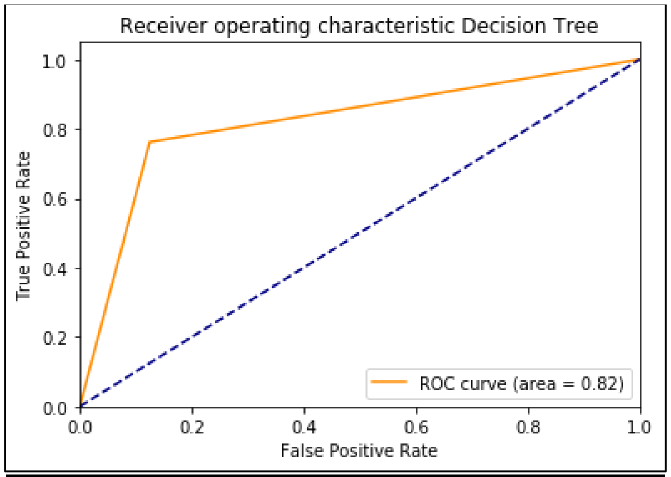
- **Label Encoding Approach**

It is a feature under the sklearn.preprocessing module which when applied to a data transforms it into numerical values. Each unique value is assigned a number and every occurence of that value is replaced by its respective number.

Decision Tree with RFECV



ROC-AUC CURVE

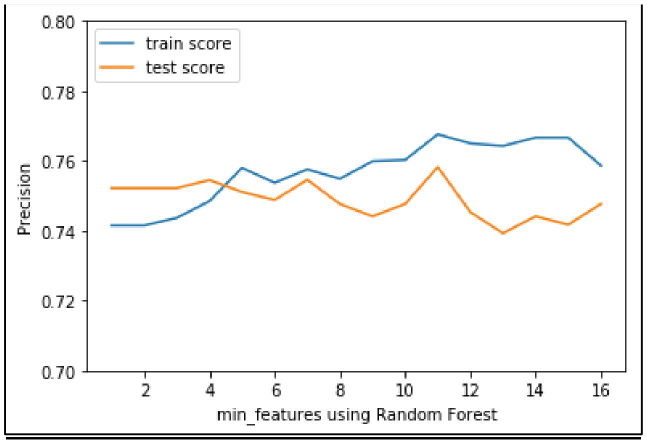


[illegible]

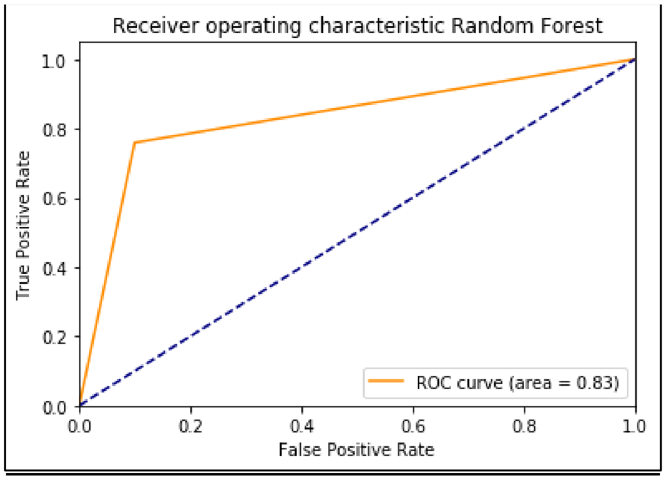
Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Random Forest with RFECV



ROC-AUC CURVE

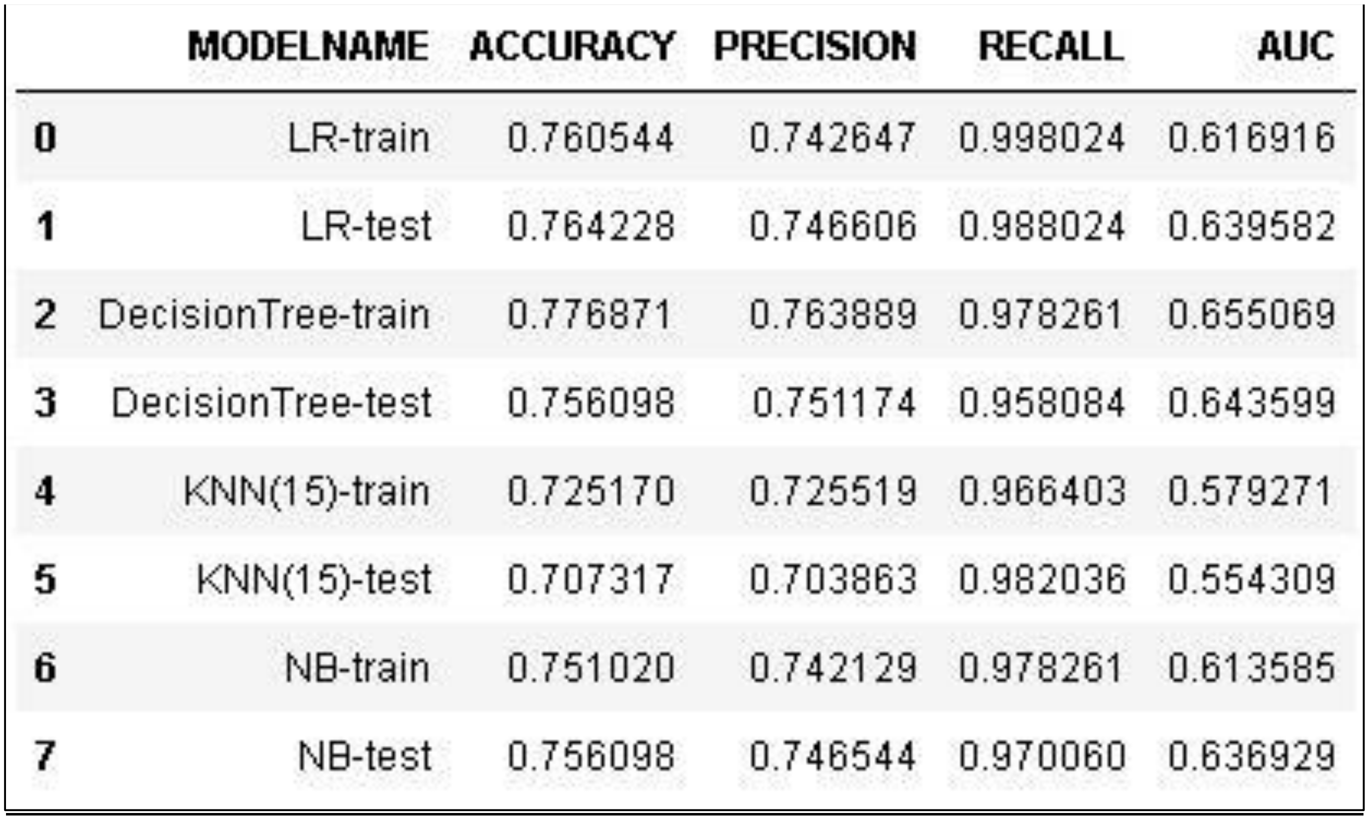


[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Other Models (On top 10 features)



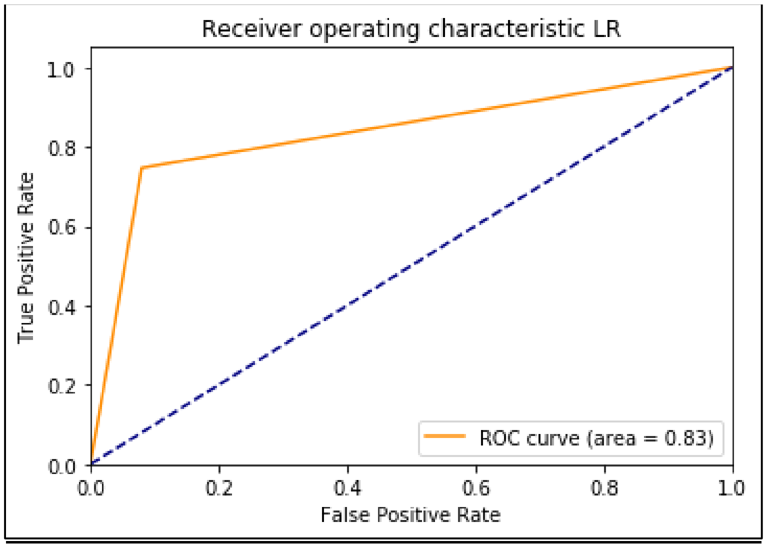
	MODELNAME	ACCURACY	PRECISION	RECALL	AUC
0	LR-train	0.760544	0.742647	0.998024	0.616916
1	LR-test	0.764228	0.746606	0.988024	0.639582
2	DecisionTree-train	0.776871	0.763889	0.978261	0.655069
3	DecisionTree-test	0.756098	0.751174	0.958084	0.643599
4	KNN(15)-train	0.725170	0.725519	0.966403	0.579271
5	KNN(15)-test	0.707317	0.703863	0.982036	0.554309
6	NB-train	0.751020	0.742129	0.978261	0.613585
7	NB-test	0.756098	0.746544	0.970060	0.636929

[illegible]

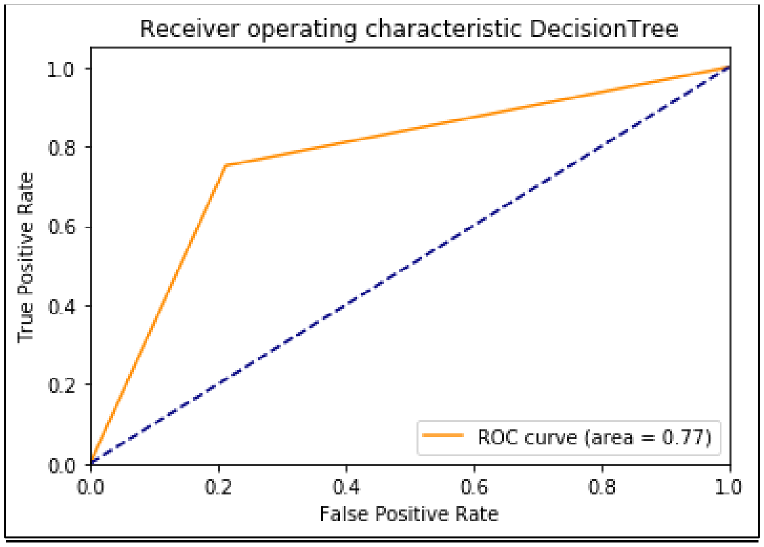
Globsyn Knowledge Foundation Kolkata

ROC-AUC CURVES

- **Logistic Regression**



- **Decision Tree**

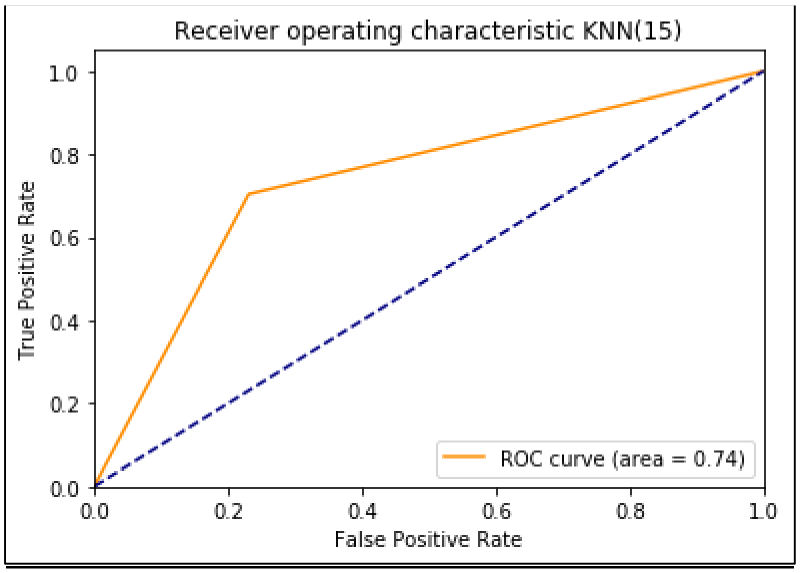


[illegible]

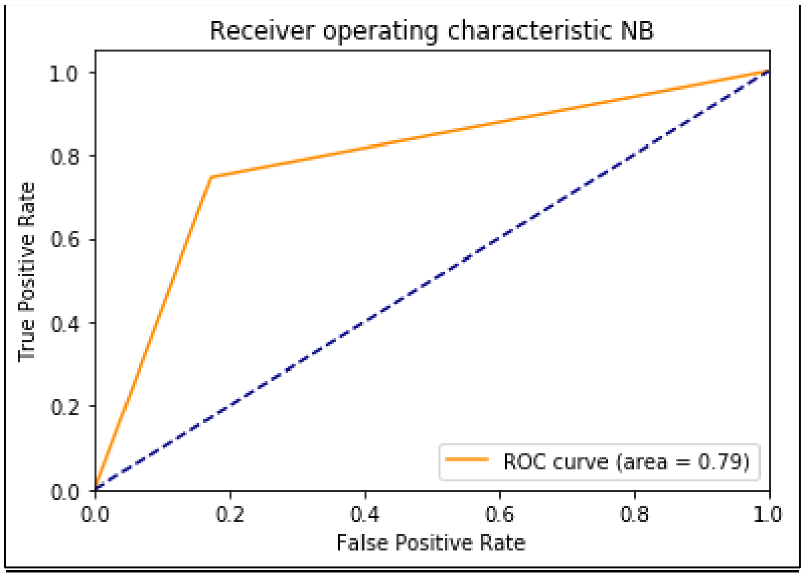
Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

- **KNN (15)**



Naive Bayes



[illegible]

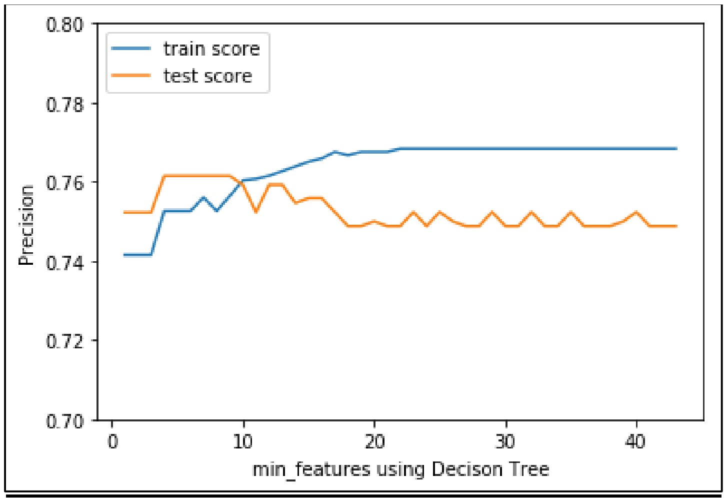
Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

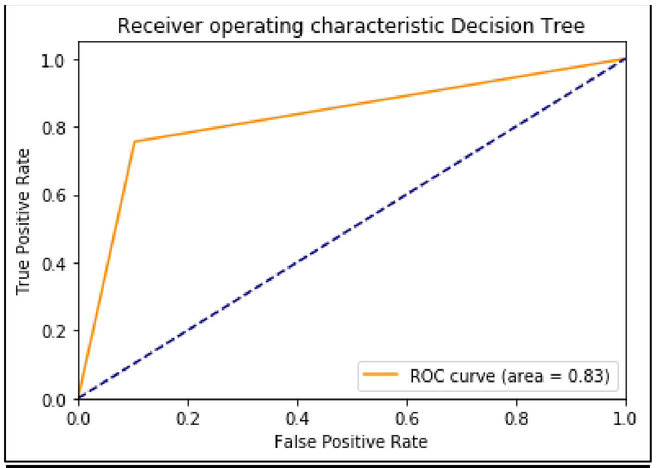
● One Hot Encoding

A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

Decision Tree with RFECV



ROC-AUC CURVE

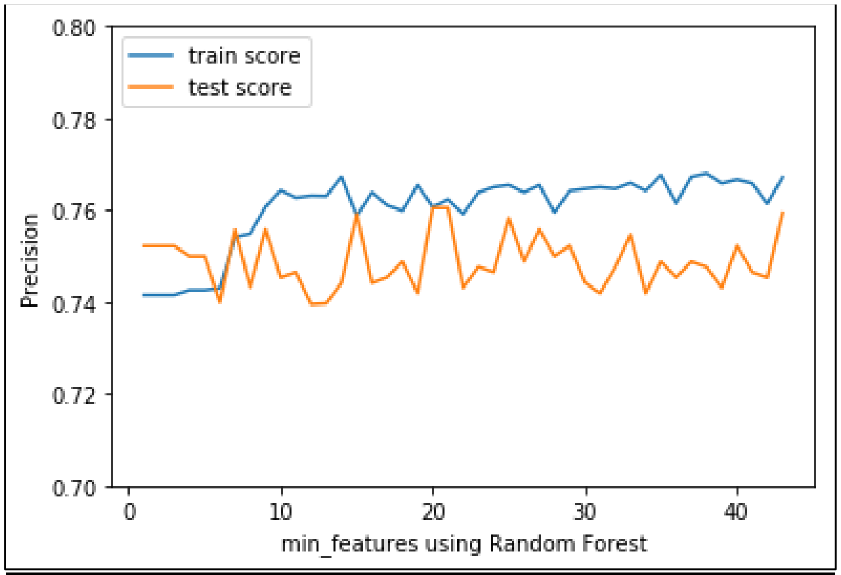


[illegible]

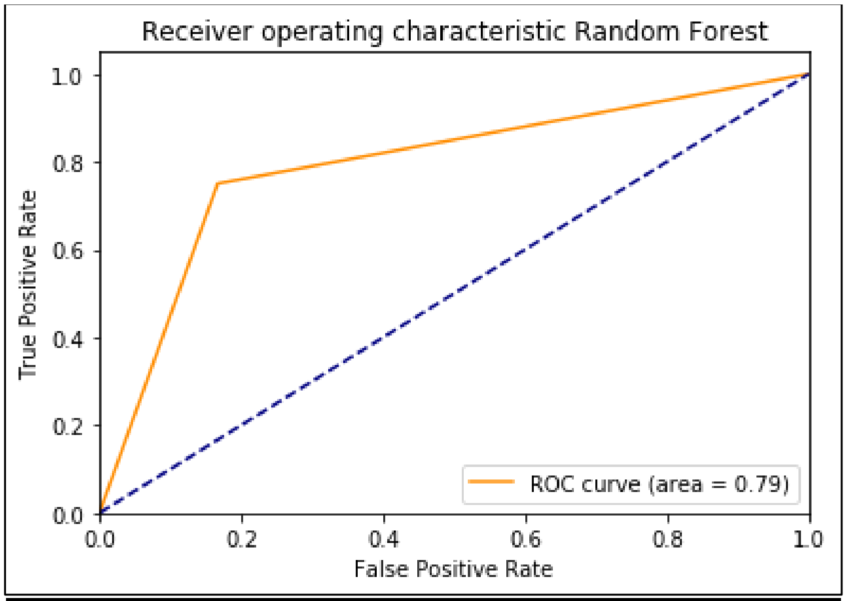
Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Random Forest with RFECV



ROC-AUC CURVE

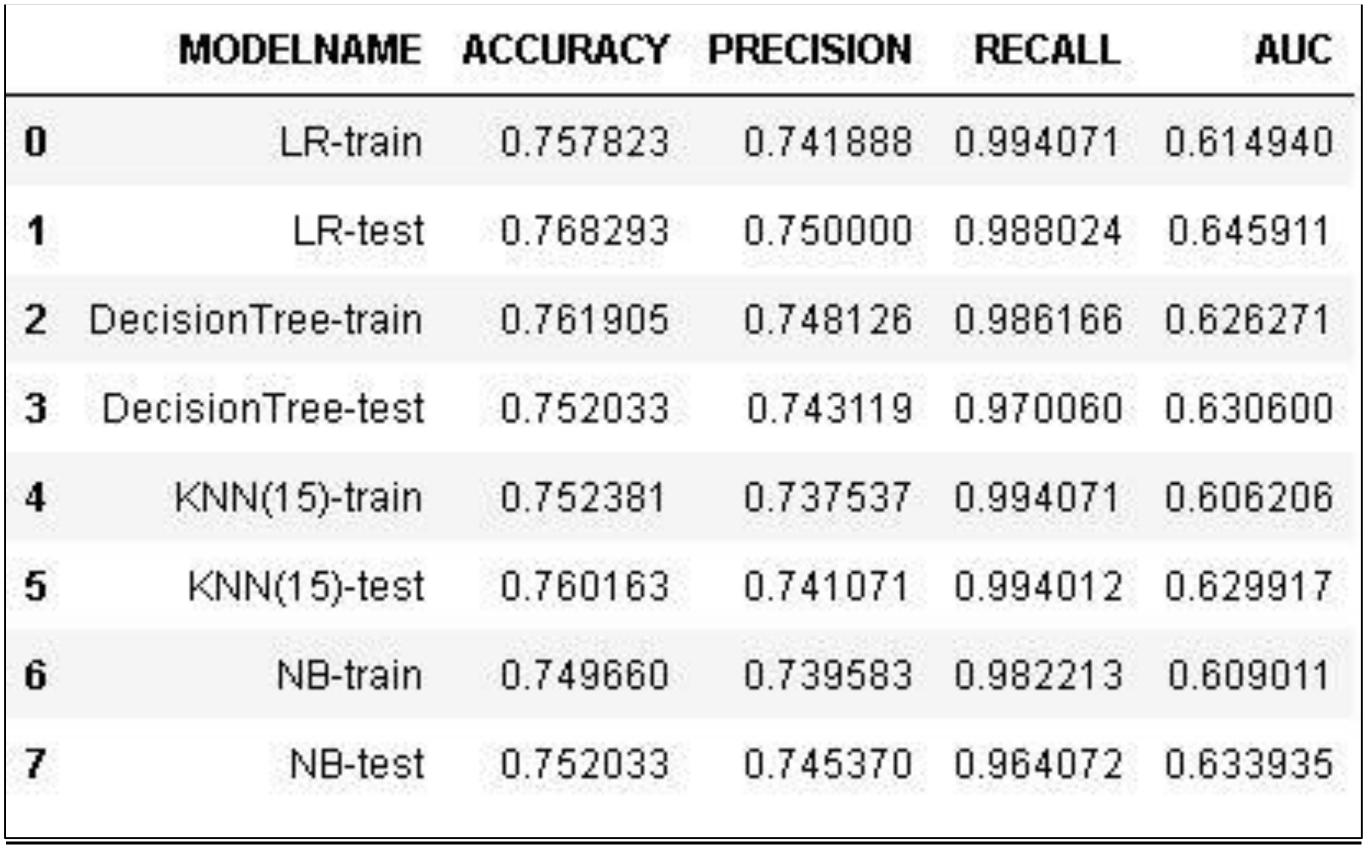


[illegible]

Globsyn Knowledge Foundation
Kolkata

Document sign date :May 3, 2019

Other Models (On top 10 features)



	MODELNAME	ACCURACY	PRECISION	RECALL	AUC
0	LR-train	0.757823	0.741888	0.994071	0.614940
1	LR-test	0.768293	0.750000	0.988024	0.645911
2	DecisionTree-train	0.761905	0.748126	0.986166	0.626271
3	DecisionTree-test	0.752033	0.743119	0.970060	0.630600
4	KNN(15)-train	0.752381	0.737537	0.994071	0.606206
5	KNN(15)-test	0.760163	0.741071	0.994012	0.629917
6	NB-train	0.749660	0.739583	0.982213	0.609011
7	NB-test	0.752033	0.745370	0.964072	0.633935

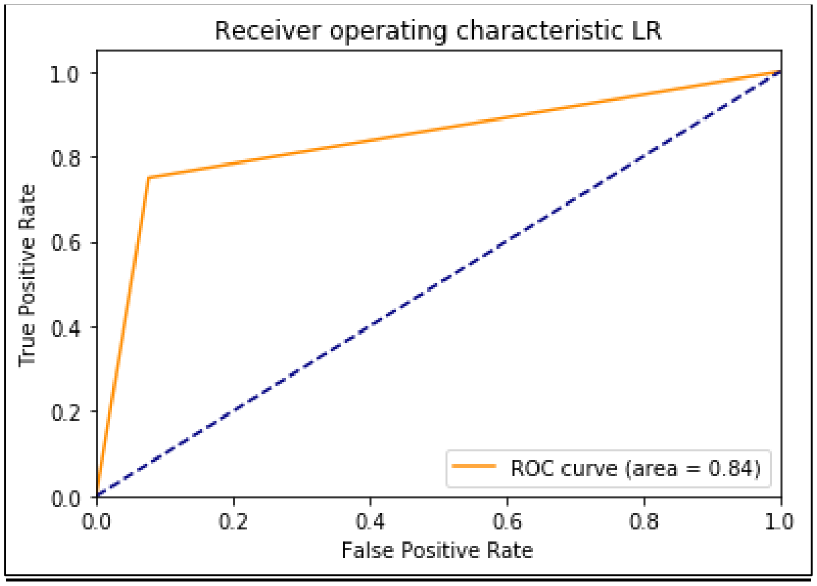
[illegible]

Globsyn Knowledge Foundation Kolkata

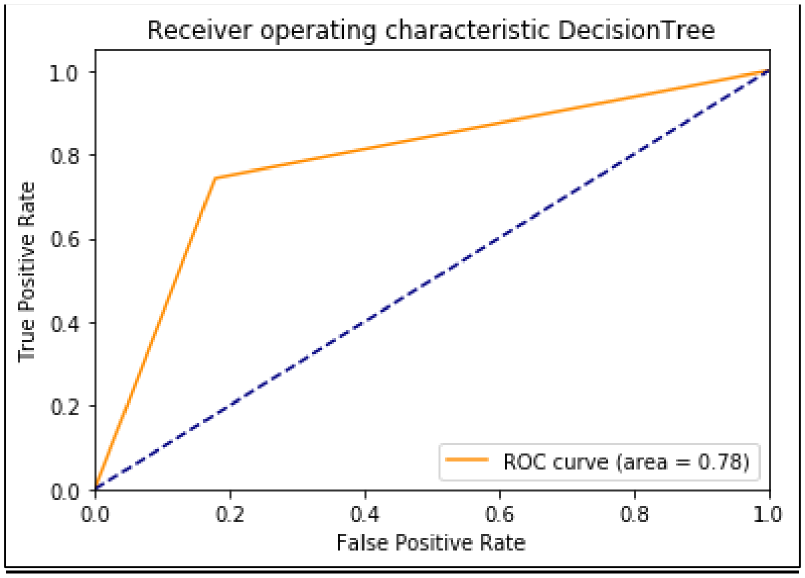
Document sign date :May 3, 2019

ROC-AUC CURVES

- **Logistic Regression**



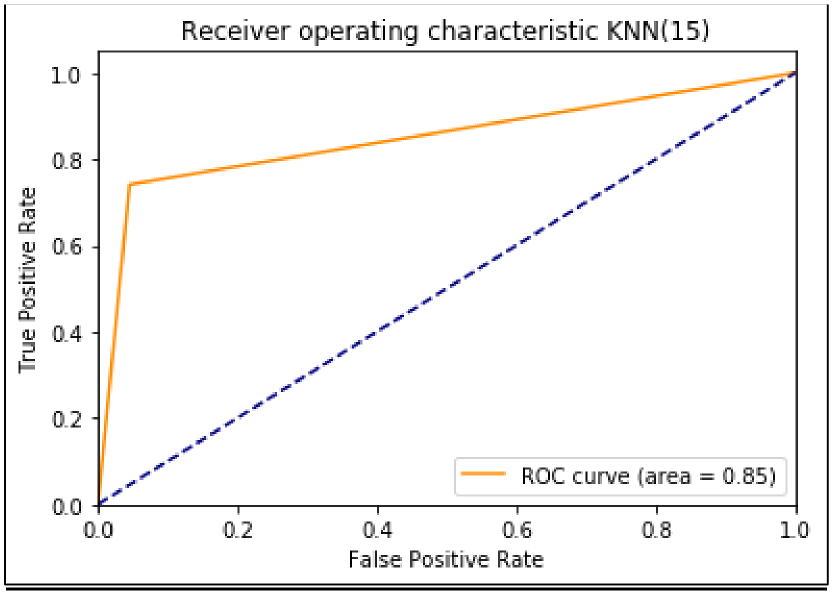
- **Decision Tree**



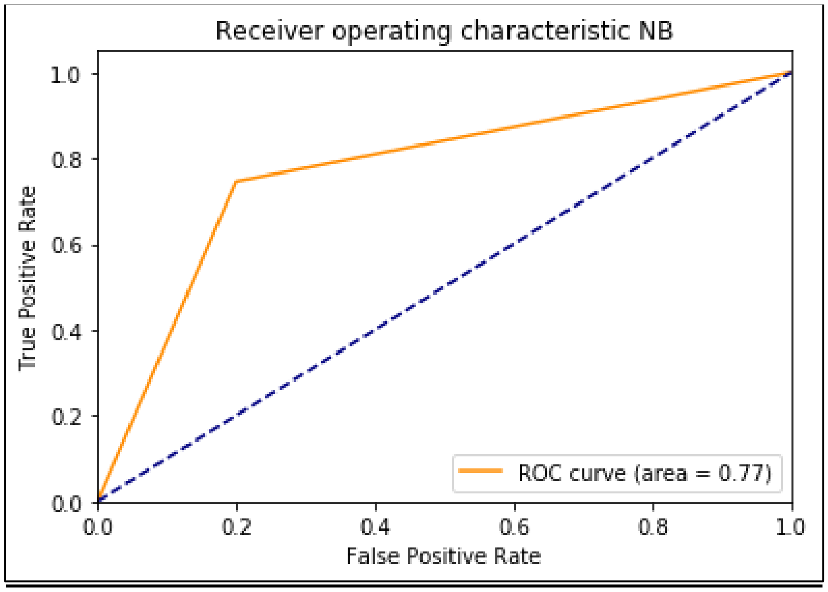
[illegible]

Globsyn Knowledge Foundation Kolkata

- **KNN (15)**



- **Naive Bayes**



[illegible]

Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

This time, we shall impute the null values and run our models over it. We shall not remove any row with null values.

Observations:

- Most of the null values are present in the columns [permission,three_hours,resume,letter,hope,alt_number,clear_about_venue]
- From above results we see that from all the people who turned up for the interview 83% had shared all their details of the above mentioned columns
- Therefore, we may assume that if a person has arrived for the interview , then there is a high chance(83%) that he must have shared all his details
- Interestingly, those who did not turn up for the interview have a 51% chance of not turning up
- From the above outputs it reflects that conditions which made people come for the interviews are the same reasons for them not turning up for the interview
- This means people who had shared their details but still didn't turn up, may be caused by different factors which is not present in our current data set
- Therefore, we just fill No for those who did not turn up for the interview

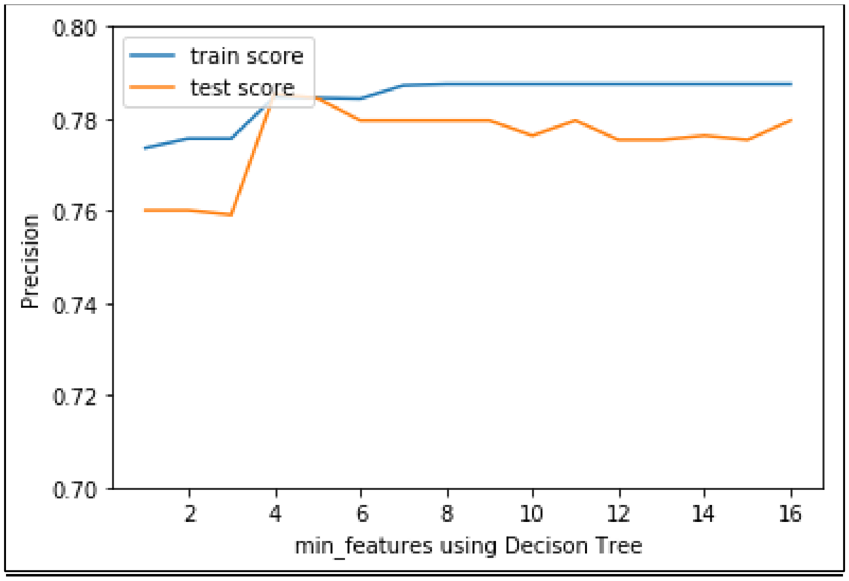
[illegible]

Globsyn Knowledge Foundation ★ Kolkata

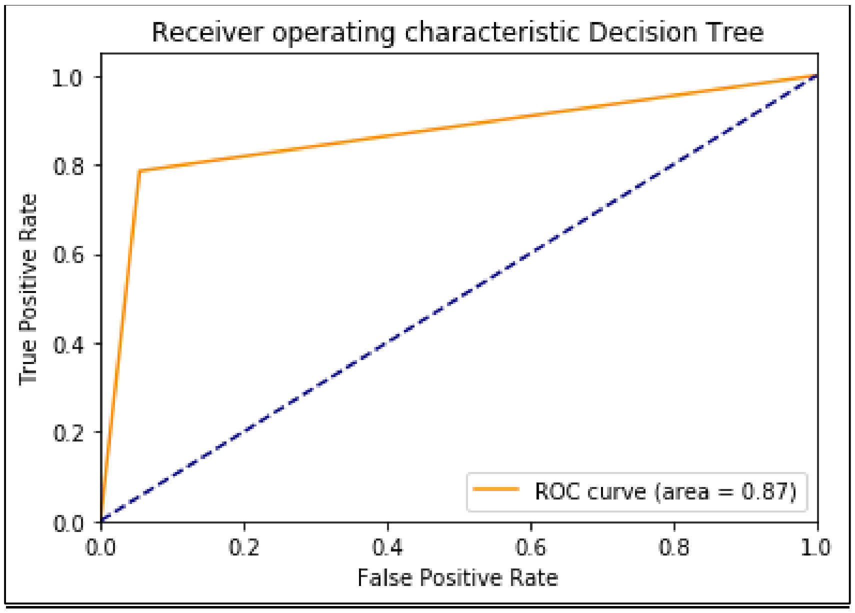
Document sign date :May 3, 2019

- **Label Encoding**

Decision Tree with RFECV



ROC-AUC CURVE

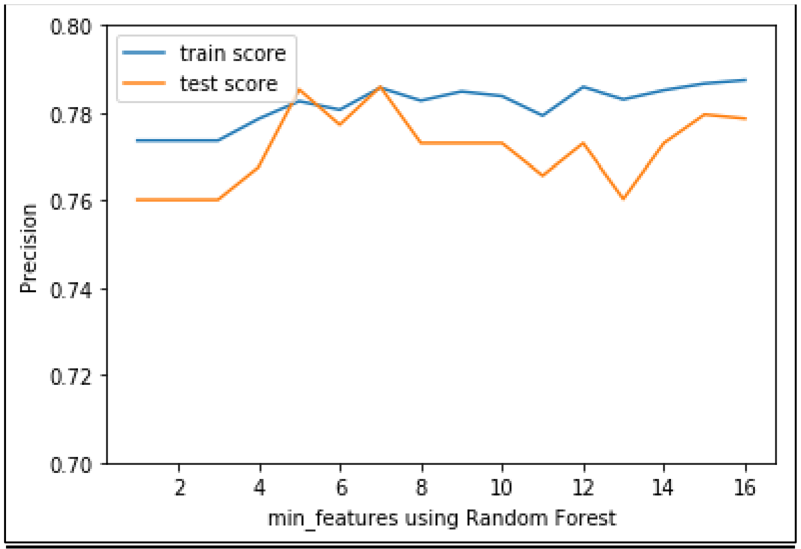


[illegible]

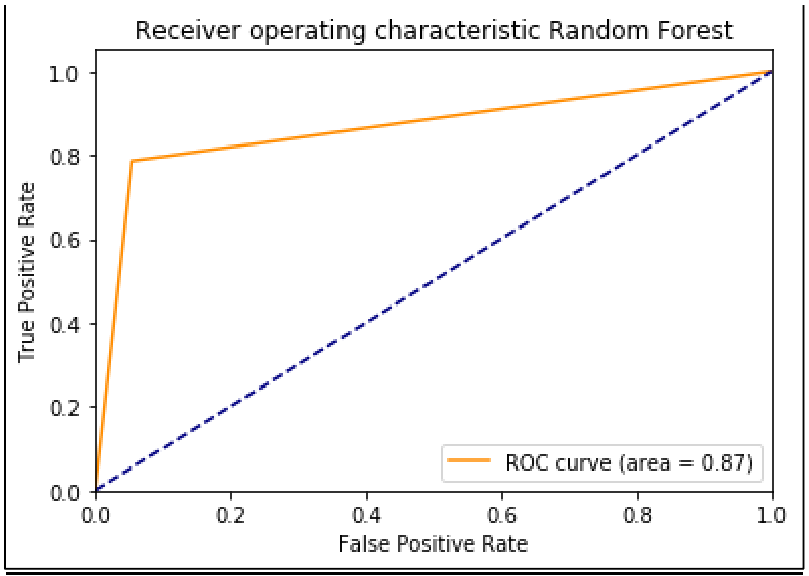
Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

Random Forest with RFECV



ROC-AUC CURVE



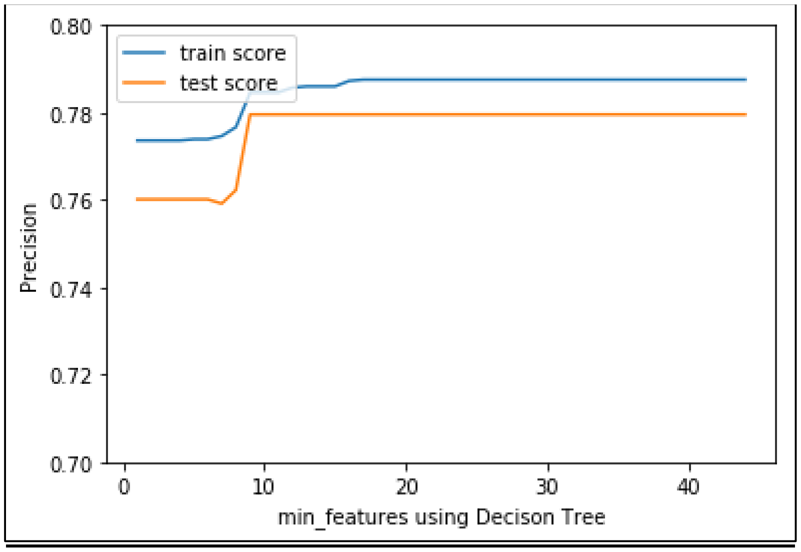
[illegible]

Globsyn Knowledge Foundation Kolkata

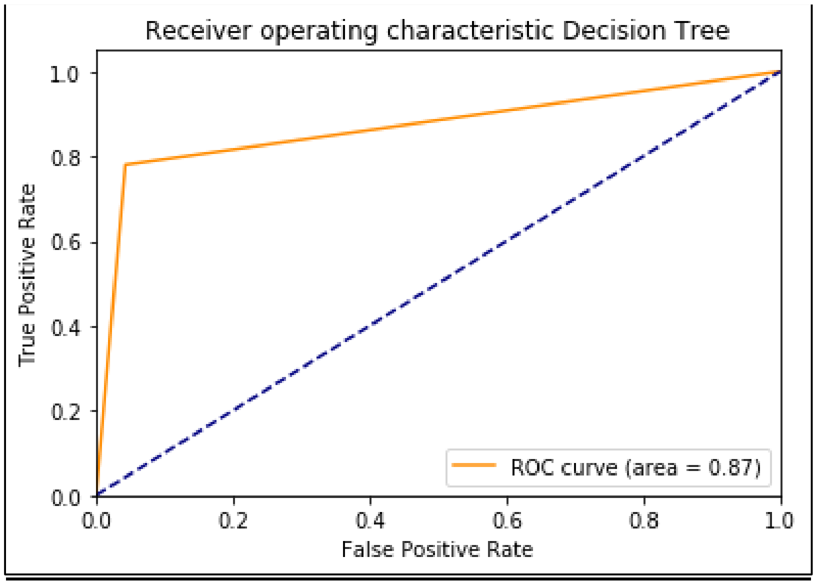
Document sign date :May 3, 2019

- **One Hot Encoding**

Decision Tree with RFECV



ROC-AUC CURVE

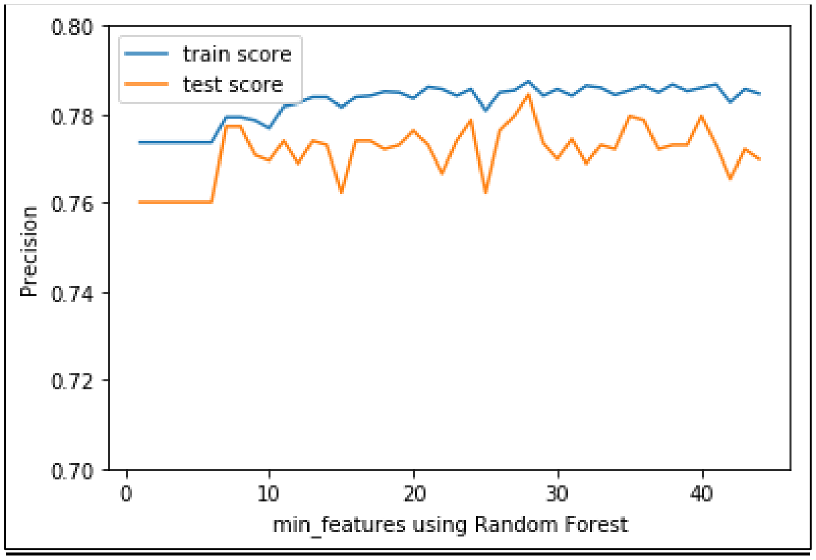


[illegible]

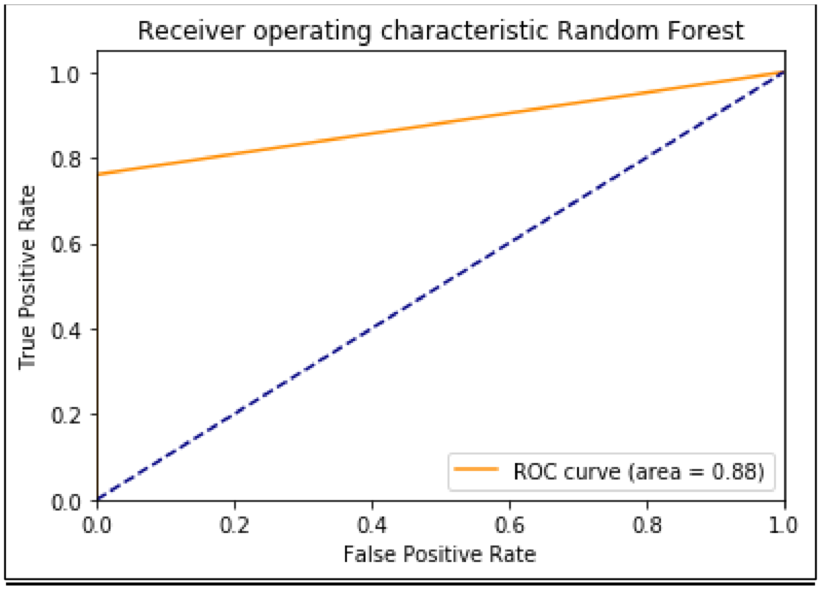
Globsyn Knowledge Foundation ★ Kolkata

Document sign date :May 3, 2019

Random Forest with RFECV



ROC-AUC CURVE



[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Final Model (Code)

- **First Approach**

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
df=pd.read_csv('C:/Users/Dell/Desktop/Interview.csv')
import warnings
warnings.filterwarnings('ignore')
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn import feature_selection
from sklearn import tree
from sklearn import utils
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
from sklearn import naive_bayes
from sklearn import feature_selection
from sklearn import ensemble
from sklearn import linear_model
from sklearn import neighbors
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
import re

df.info()
df.drop(["Unnamed: 23",'Unnamed: 24','Unnamed: 25','Unnamed: 26','Unnamed: 27'],axis=1,inplace=True)
df.columns=['date','client','industry','location','closedpos','skills','type','id','gender','currloc','jobloc','venue','natloc','permit','meeting','call','no.','resume','clear','letter','expat','obsat','marit']
df.drop(df.index[df["date"].isnull()],axis=0,inplace=True)
df.date=df.date.str.replace(" ","")
df.date=df.date.str.replace(r'&\d[\d].\d\d\w\w','')
df.date=df.date.str.replace(r'&\d.\d\d\w\w','')
df.date=df.date.str.replace('/','.')
df.date=df.date.str.replace('-Apr-','.04.')
df.date=df.date.str.replace('–Apr-','.04.')
df.date=df.date.str.replace('Apr','.04.')
df.date=df.date.str.replace(r'\W16','.2016')
df.date=df.date.str.replace(r'\W15','.2015')
df.date=df.date.str.replace('-','.')
from datetime import datetime
df['date']=pd.to_datetime(df['date'],format='%d.%m.%Y')
df["date"]=df["date"].dt.date
```

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

```
df["date"].unique()
df2=pd.DataFrame({"year":[],"month":[],"day":[]})
df=pd.concat([df,df2],sort=False)
df2=pd.DataFrame({"dayofyear":[]})
df=pd.concat([df,df2],sort=False)
df2=pd.DataFrame({"dayofweek":[]})
df=pd.concat([df,df2],sort=False)
for i,s in df.iterrows():
    k=df["date"][i].timetuple()
    df["year"][i]=k.tm_year
    df["month"][i]=k.tm_mon
    df["day"][i]=k.tm_mday
    df["dayofweek"][i]=datetime.isoweekday(df["date"][i])
    df.dayofyear[i]=k.tm_yday
df.info()
df.obsat.replace({'No':'N','Yes':'Y','yes':'Y','no':'N','yes                ':'Y','No                ':'N','NO':'N','no
':'N'},inplace=True)
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
df.loc[:,'obsat']=le.fit_transform(df.loc[:,'obsat'])
for i in df.month.unique():
    print(i,(df[(df['month']==i)          &          (df.obsat==1)].shape[0]),(df[(df['month']==i)          &
(df.obsat==0)].shape[0]))
for i in df.dayofweek.unique():
    print(i,(df[(df['dayofweek']==i)      &      (df.obsat==1)].shape[0]),(df[(df['dayofweek']==i)      &
(df.obsat==0)].shape[0]))
for i in df.year.unique():
    print (i)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
    print(df[(df['year']==i) & (df.obsat==1)].shape[0])
    print(df[(df['year']==i) & (df.obsat==0)].shape[0])



df["natloc"].unique()
df['natloc'].replace({'Delhi /NCR':'Delhi','- Cochin- ':'Cochin'},inplace=True)
df['natloc'].unique()
df['client'].replace({'Hospira':'H','Aon Hewitt':'A','UST':'U','Standard Chartered Bank':'S','ANZ':'A1','Pfizer':'P','Standard Chartered Bank Chennai':'S','Aon hewitt Gurgaon':'A','Astrazeneca':'As','Flextronics':'F','Prodapt':'P1','Williams Lea':'W','Barclays':'B','Hewitt':'A','Woori Bank':'Wb'},inplace=True)
for i in df["meeting"].unique():
        print(i,(df[(df['meeting']==i) & (df.obsat==1)].shape[0]),(df[(df['meeting']==i) & (df.obsat==0)].shape[0]),sep=" ")

df.meeting.isnull().sum()

df.meeting.replace({'Yes':'Y','Na':'N','No':'N','yes':'Y','Not sure':'N','Not Sure':'N','cant Say':'N'},inplace=True)

df.drop(df.index[df.meeting.isnull()],axis=0,inplace=True)
for i in df["meeting"].unique():
        print(i,(df[(df['meeting']==i) & (df.obsat==1)].shape[0]),(df[(df['meeting']==i) & (df.obsat==0)].shape[0]),sep=" ")
for i in df["permit"].unique():
        print(i,(df[(df['permit']==i) & (df.obsat==1)].shape[0]),(df[(df['permit']==i) & (df.obsat==0)].shape[0]),sep=" ")
df.drop(df.index[df.permit.isnull()],axis=0,inplace=True)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
df.permit.replace({'Yes':'Y','No':'N','Not yet':'N','Yet to
confirm':'N','yes':'Y','Na':'N'},inplace=True)
for i in df["permit"].unique():
      print(i,(df[(df['permit']==i) & (df.obsat==1)].shape[0]),(df[(df['permit']==i) &
(df.obsat==0)].shape[0]),sep=" ")
for i in df["resume"].unique():
      print(i,(df[(df['resume']==i) & (df.obsat==1)].shape[0]),(df[(df['resume']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df['resume'].replace({'Yes':'Y', 'No':'N', 'No- will take it soon':'N', 'Not yet':'N', 'yes':'Y', 'Na':'N',
    'Not Yet':'N'},inplace=True)
df.drop(df.index[df.resume.isnull()],axis=0,inplace=True)

for i in df["resume"].unique():
      print(i,(df[(df['resume']==i) & (df.obsat==1)].shape[0]),(df[(df['resume']==i) &
(df.obsat==0)].shape[0]),sep=" ")
for i in df["expat"].unique():
      print(i,(df[(df['expat']==i) & (df.obsat==1)].shape[0]),(df[(df['expat']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df['expat'].replace({'Yes':'Y','No':'N','yes':'Y','11:00 AM':'Y','10.30 Am':'Y'},inplace=True)

for i,series in df.iterrows():
    if((series.expat=='Uncertain') & (series.obsat==1)):
      df.loc[i,'expat']='Y'
    elif ((series.expat=='Uncertain') &( series.obsat==0)):
      df.loc[i,'expat']='N'
for i in df["expat"].unique():
      print(i,(df[(df['expat']==i) & (df.obsat==1)].shape[0]),(df[(df['expat']==i) &
(df.obsat==0)].shape[0]),sep=" ")
for i in df["letter"].unique():
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
    print(i,(df[(df['letter']==i) & (df.obsat==1)].shape[0]),(df[(df['letter']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df['letter'].replace({'Yes':'Y', 'Havent Checked':'Y', 'No':'N', 'Need To Check':'N', 'Not
sure':'Y','Not Sure':'N', 'Not yet':'N', 'no':'N', 'yes':'Y', 'Na':'N'},inplace=True)
for i in df["letter"].unique():
    print(i,(df[(df['letter']==i) & (df.obsat==1)].shape[0]),(df[(df['letter']==i) &
(df.obsat==0)].shape[0]),sep=" ")
for i in df["clear"].unique():
    print(i,(df[(df['clear']==i) & (df.obsat==1)].shape[0]),(df[(df['clear']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df['clear'].replace({'Yes':'Y', 'No':'N', 'No- I need to check':'N', 'yes':'Y', 'Na':'N',
'no':'N'},inplace=True)
for i in df["clear"].unique():
    print(i,(df[(df['clear']==i) & (df.obsat==1)].shape[0]),(df[(df['clear']==i) &
(df.obsat==0)].shape[0]),sep=" ")
for i in df["no."].unique():
    print(i,(df[(df['no.']==i) & (df.obsat==1)].shape[0]),(df[(df['no.']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df['no.'].replace({'Yes':'Y', 'No':'N', 'No I have only thi number':'N', 'yes':'Y',
'Na':'N'},inplace=True)
for i in df["no."].unique():
    print(i,(df[(df['no.']==i) & (df.obsat==1)].shape[0]),(df[(df['no.']==i) &
(df.obsat==0)].shape[0]),sep=" ")
for i in df["call"].unique():
    print(i,(df[(df['call']==i) & (df.obsat==1)].shape[0]),(df[(df['call']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df['call'].replace({'Yes':'Y', 'No':'N', 'No Dont':'Y', 'yes':'Y', 'Na':'N'},inplace=True)
for i in df["call"].unique():
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
        print(i,(df[(df['call']==i) & (df.obsat==1)].shape[0]),(df[(df['call']==i) &
(df.obsat==0)].shape[0]),sep=" ")
df.currloc.replace({'Chennai':'C','Gurgaon':'G','Bangalore':'B','Hyderabad':'H','Delhi':'D','chennai'
:'C','- Cochin- ':'K','Noida':'N','CHENNAI':'C','chennai ':'C'},inplace=True)
df.venue.replace({'Hosur':'H', 'Gurgaon':'G', 'Bangalore':'B', 'Chennai':'C', 'Hyderabad':'H',
    '- Cochin- ':'K', 'Noida':'N'},inplace=True)
df.jobloc.replace({'Hosur':'H', 'Bangalore':'B', 'Chennai':"C", 'Gurgaon':'G', 'Visakapatinam':'V',
    '- Cochin- ':'K', 'Noida':'N'},inplace=True)
df['type'].replace({'Scheduled Walkin':'SW', 'Scheduled ':'S', 'Walkin':'W', 'Scheduled Walk
In':'SW',
    'Walkin ':'W'},inplace=True)
df.location.replace({'Chennai':'C', 'Gurgaon':'G', 'Bangalore':'B', 'Hyderabad':'H', 'Gurgaonr':'G',
    'Delhi':'D', 'chennai':'C', '- Cochin- ':'K', 'Noida':'N', 'CHENNAI':'C', 'chennai
':'C'},inplace=True)
df.closedpos.replace({'Production- Sterile':'P','Selenium testing':'S','Dot Net':'D','AML':'A','Trade
Finance':'T','Routine':'R','Niche':'N'},inplace=True)
df.head()
df.drop("id",axis=1,inplace=True)
for i in df.drop(["date","skills","obsat"],axis=1):
  df.loc[:,i]=le.fit_transform(df.loc[:,i])
X=df.drop(["skills","date","obsat"],axis=1)
y=df["obsat"]
def get_importance(X,y):
  rf=ensemble.RandomForestClassifier()
  rf.fit(X,y)
  return pd.DataFrame(rf.feature_importancX=df.drop(["skills","date","obsat"],axis=1)
y=df["obsat"]es_,index =
X.columns,columns=['importance']).sort_values('importance',ascending=False)
def modelstats1(X,y):
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
np.random.seed(42)
Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=.25,random_state=42)
stats=[]
modelnames=["LR","DecisionTree","KNN","NB"]
models=list()
models.append(linear_model.LogisticRegression(C=10))
models.append(tree.DecisionTreeClassifier())
models.append(neighbors.KNeighborsClassifier())
models.append(naive_bayes.GaussianNB())
for name,model in zip(modelnames,models):
    if name=="KNN":
        k=[l for l in range(5,17,2)]
        grid={"n_neighbors":k}
        grid_obj = GridSearchCV(estimator=model,param_grid=grid,scoring="f1")
        grid_fit =grid_obj.fit(Xtrain,ytrain)
        model = grid_fit.best_estimator_
        model.fit(Xtrain,ytrain)
        name=name+"("+str(grid_fit.best_params_["n_neighbors"])+")"
        print(grid_fit.best_params_)
    else:
        model.fit(Xtrain,ytrain)
    trainprediction=model.predict(Xtrain)
    testprediction=model.predict(Xtest)
    scores=list()
    scores.append(name+"-train")
    scores.append(metrics.accuracy_score(ytrain,trainprediction))
    scores.append(metrics.precision_score(ytrain,trainprediction))
    scores.append(metrics.recall_score(ytrain,trainprediction))
    scores.append(metrics.roc_auc_score(ytrain,trainprediction))
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
        stats.append(scores)
        scores=list()
        scores.append(name+"-test")
        scores.append(metrics.accuracy_score(ytest,testprediction))
        scores.append(metrics.precision_score(ytest,testprediction))
        scores.append(metrics.recall_score(ytest,testprediction))
        scores.append(metrics.roc_auc_score(ytest,testprediction))
        stats.append(scores)

    colnames=["MODELNAME","ACCURACY","PRECISION","RECALL","AUC"]
    return pd.DataFrame(stats,columns=colnames)
imp=get_importance(X,y)
imp
def rfe_selection(X,y):
    np.random.seed(42)
    rf=ensemble.RandomForestClassifier()
    train=[]
    test=[]
    loss=[]
    m=[rf]
    s=["Random Forest"]
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
    for j in range(0,len(m)):
        #print(s[j])
        model=m[j]
        for i in range(1,30):
            rfecv=feature_selection.RFE(estimator=model,n_features_to_select=i)
            rfecv.fit(X_train,y_train)
            cols=X.columns[rfecv.get_support()]
            X_=X_train[cols]
            X_t=X_test[cols]
            rfecv.fit(X_,y_train)

            #model.fit(X_,y)
            #print(cols)
            train.append(metrics.precision_score(y_train,rfecv.predict(X_)))
            test.append(metrics.precision_score(y_test,rfecv.predict(X_t)))
            loss.append(metrics.log_loss(y_test,rfecv.predict_proba(X_t)))
    plt.figure()
    plt.plot(np.arange(1,30,1),train)
    plt.plot(np.arange(1,30,1),test)
    plt.title("Optimal Number of Features (Random Forest) ")
    plt.legend(['train score', 'test score'], loc='upper left')
    plt.ylim(.7,.8)
    plt.show()
from sklearn import feature_selection
def get_feat_dt(X,y):
    np.random.seed(42)
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
    model=tree.DecisionTreeClassifier()
    n=X.shape[1]
    tr_score=[]
    ts_score=[]
    for i in range(1,n):
        rfecv=feature_selection.RFE(model,n_features_to_select=i)
        rfecv.fit(X_train,y_train)
        col=X.columns[rfecv.get_support()]
        model.fit(X_train[col],y_train)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
        tr_p=model.predict(X_train[col])
        ts_p=model.predict(X_test[col])
        tr_score.append(metrics.precision_score(y_train,tr_p))
        ts_score.append(metrics.precision_score(y_test,ts_p))
    plt.figure()
    plt.plot(range(1,n),tr_score)
    plt.plot(range(1,n),ts_score)
    plt.xlabel("min_features using Decison Tree")
    plt.ylabel("Precision")
    plt.legend(['train score', 'test score'], loc='upper left')
    plt.ylim(.7,.8)
    plt.show()

def modelstats1(X,y):
    np.random.seed(42)
    Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=.25,random_state=42)
    stats=[]
    modelnames=["LR","DecisionTree","KNN","NB"]
    models=list()
    models.append(linear_model.LogisticRegression(C=10))
    models.append(tree.DecisionTreeClassifier())
    models.append(neighbors.KNeighborsClassifier())
    models.append(naive_bayes.GaussianNB())
    for name,model in zip(modelnames,models):
        if name=="KNN":
            k=[l for l in range(5,17,2)]
            grid={"n_neighbors":k}
            grid_obj = GridSearchCV(estimator=model,param_grid=grid,scoring="f1")
            grid_fit =grid_obj.fit(Xtrain,ytrain)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
            model = grid_fit.best_estimator_
            model.fit(Xtrain,ytrain)
            name=name+"("+str(grid_fit.best_params_["n_neighbors"])+")"
            print(grid_fit.best_params_)
        else:
            model.fit(Xtrain,ytrain)
        trainprediction=model.predict(Xtrain)
        testprediction=model.predict(Xtest)
        scores=list()
        scores.append(name+"-train")
        scores.append(metrics.accuracy_score(ytrain,trainprediction))
        scores.append(metrics.precision_score(ytrain,trainprediction))
        scores.append(metrics.recall_score(ytrain,trainprediction))
        scores.append(metrics.roc_auc_score(ytrain,trainprediction))
        stats.append(scores)
        scores=list()
        scores.append(name+"-test")
        scores.append(metrics.accuracy_score(ytest,testprediction))
        scores.append(metrics.precision_score(ytest,testprediction))
        scores.append(metrics.recall_score(ytest,testprediction))
        scores.append(metrics.roc_auc_score(ytest,testprediction))
        stats.append(scores)

    colnames=["MODELNAME","ACCURACY","PRECISION","RECALL","AUC"]
    return pd.DataFrame(stats,columns=colnames)
def run_rf(X,y,i):
    np.random.seed(42)
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
    model=ensemble.RandomForestClassifier()
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
    rfecv=feature_selection.RFE(model,n_features_to_select=i)
    rfecv.fit(X_train,y_train)
    col=X.columns[rfecv.get_support()]
    model.fit(X_train[col],y_train)
    tr_p=model.predict(X_train[col])
    ts_p=model.predict(X_test[col])
    print (col)
    print(i)
    print("Train Set")
    print("precison",metrics.precision_score(y_train,tr_p)*100)
    print("recall",metrics.recall_score(y_train,tr_p)*100)
    print("accuracy",metrics.accuracy_score(y_train,tr_p)*100)
    print("log loss",metrics.log_loss(y_train,model.predict_proba(X_train[col])))
    print("Test Set")
    print("precision",metrics.precision_score(y_test,ts_p)*100)
    print("recall",metrics.recall_score(y_test,ts_p)*100)
    print("acuuracy",metrics.accuracy_score(y_test,ts_p)*100)
    print("log loss",metrics.log_loss(y_test,model.predict_proba(X_test[col])))
def run_dt(X,y,i):
    np.random.seed(42)
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
    model=tree.DecisionTreeClassifier()
    rfecv=feature_selection.RFE(model,n_features_to_select=i)
    rfecv.fit(X_train,y_train)
    col=X.columns[rfecv.get_support()]
    model.fit(X_train[col],y_train)
    tr_p=model.predict(X_train[col])
    ts_p=model.predict(X_test[col])
    print(i)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
    print("Train Set")
    print("precison",metrics.precision_score(y_train,tr_p)*100)
    print("recall",metrics.recall_score(y_train,tr_p)*100)
    print("accuracy",metrics.accuracy_score(y_train,tr_p)*100)
    print("log loss",metrics.log_loss(y_train,model.predict_proba(X_train[col])))
    print("Test Set")
    print("precision",metrics.precision_score(y_test,ts_p)*100)
    print("recall",metrics.recall_score(y_test,ts_p)*100)
    print("acuuracy",metrics.accuracy_score(y_test,ts_p)*100)
    print("log loss",metrics.log_loss(y_test,model.predict_proba(X_test[col])))
get_feat_dt(df.drop(["date","obsat","skills"],axis=1),df.obsat)
rfe_selection(df.drop(["date","skills","obsat"],axis=1),df.obsat)
modelstats1(df.drop(["date","obsat","skills"],axis=1),df.obsat)
run_rf(df.drop(["date","obsat","skills"],axis=1),df.obsat,8)
run_rf(df.drop(["date","obsat","skills"],axis=1),df.obsat,9)
```

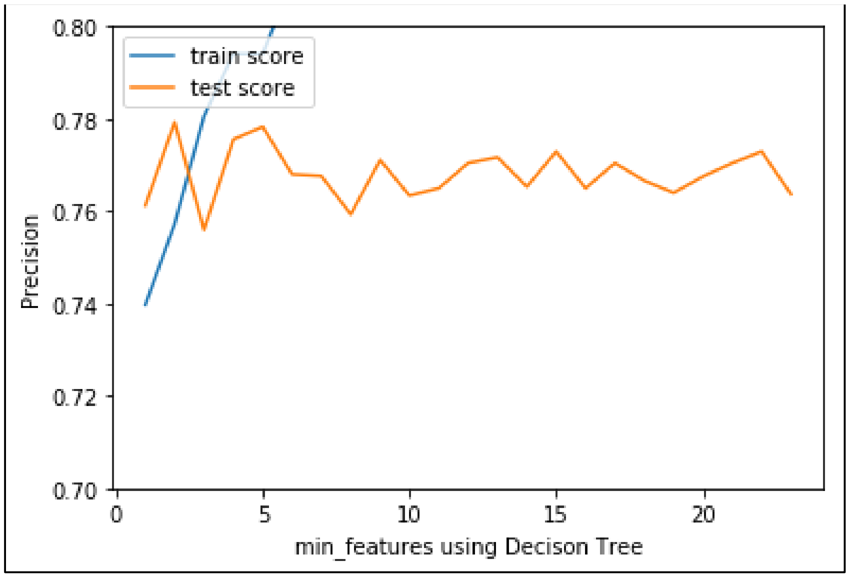
[illegible]

Globsyn Knowledge Foundation Kolkata

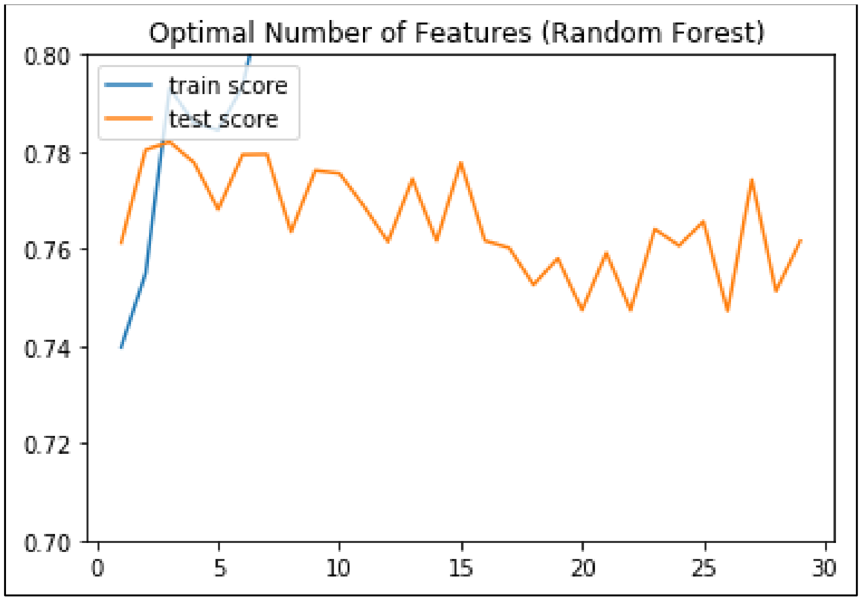
Document sign date :May 3, 2019

Related Graphs

- **Decision Tree with RFECV**



- **Random Forest with RFECV**

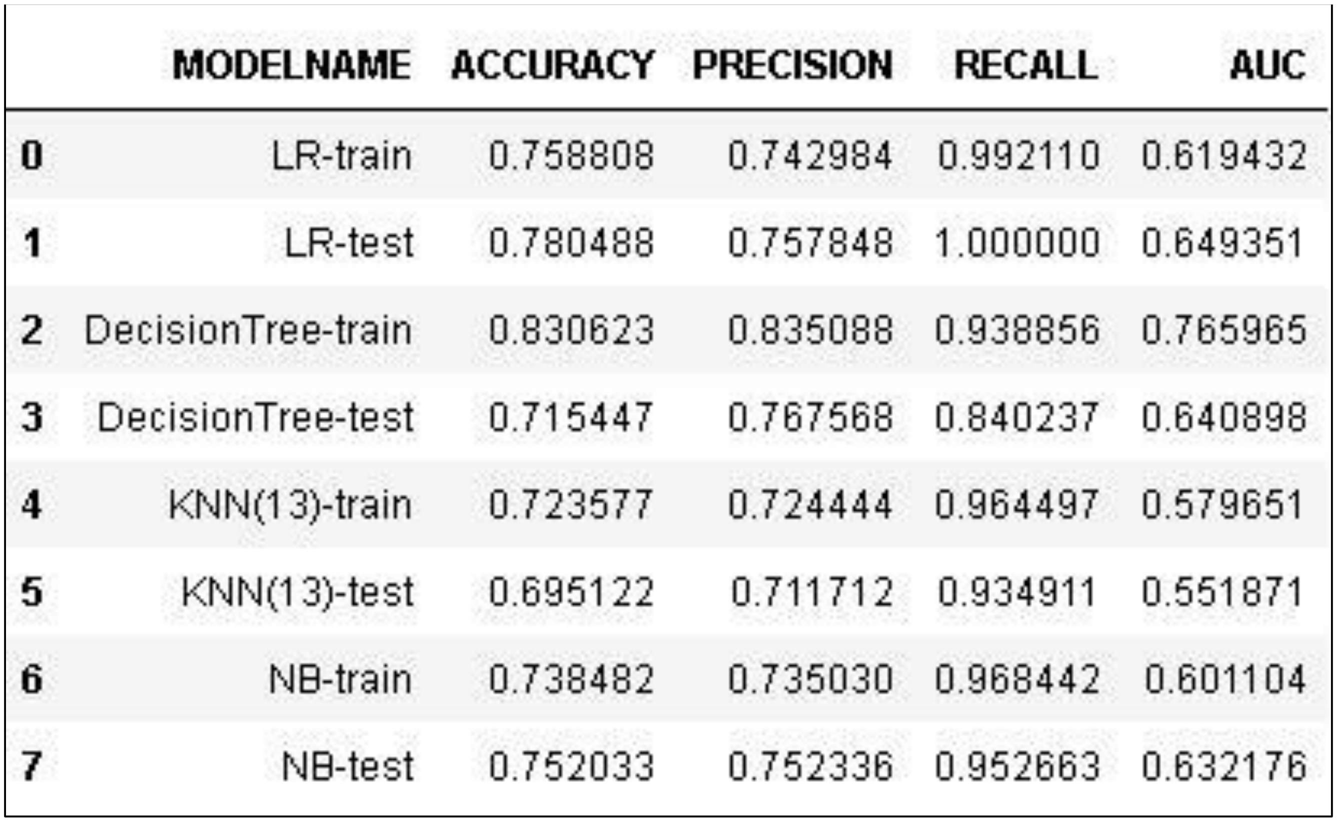


[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

- **Other Models**



	MODELNAME	ACCURACY	PRECISION	RECALL	AUC
0	LR-train	0.758808	0.742984	0.992110	0.619432
1	LR-test	0.780488	0.757848	1.000000	0.649351
2	DecisionTree-train	0.830623	0.835088	0.938856	0.765965
3	DecisionTree-test	0.715447	0.767568	0.840237	0.640898
4	KNN(13)-train	0.723577	0.724444	0.964497	0.579651
5	KNN(13)-test	0.695122	0.711712	0.934911	0.551871
6	NB-train	0.738482	0.735030	0.968442	0.601104
7	NB-test	0.752033	0.752336	0.952663	0.632176

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

- **Second Approach**

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn import feature_selection
from sklearn import tree
from sklearn import utils
from sklearn import naive_bayes
from sklearn import feature_selection
from sklearn import ensemble
from sklearn import linear_model
from sklearn import neighbors
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
import re
df=pd.read_csv("Interview.csv")
df.head()
df.drop(["Unnamed:     23","Unnamed:     24","Unnamed:     25","Unnamed:     26","Unnamed:
27"],axis=1,inplace=True)
df.columns=["date","client","industry","location","position","skills","inter_type","ID","gender",
"curr_location","job_location",

"venue","native_location","permission","hope","three_hours","alt_number","resume","clear_a
bout_venue","letter","expected",
       "observed","married"]
df.date=df.date.str.replace(" ","")
df.date=df.date.str.replace(r'\d\d\d\d&\d\d\.\d\d\w\w',"2016")
df.date=df.date=df.date.str.replace("/",".")
df.date=df.date.str.replace(r'\WApr\W',".04.")
df.date=df.date.str.replace('Apr',".04.")
df.date=df.date.str.replace(r'\d\d\d\d&\d\.\d\d\w\w',"2016")
df.date=df.date.str.replace('-',".")
df.date=df.date.str.replace(r'\W16',".2016")
df.date=df.date.str.replace(r'\W15',".2015")
df.drop(df[df.date.isnull()].index,inplace=True)
import datetime
def conv(s):
  ar=str(s).split(".")
  try:
    d=datetime.date(int(ar[2]),int(ar[1]),int(ar[0]))
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
    except Exception:
        print(len(ar),s)
    return (d)
df["new_date"]=df.date.apply(conv)
df["new_date"]=pd.to_datetime(df.new_date)
df.drop(df[df.expected.isnull()].index,inplace=True)
df.drop("date",axis=1,inplace=True)
df.client.replace({"Standard Chartered Bank Chennai":"Standard Chartered Bank","Aon hewitt
Gurgaon":"Aon Hewitt","Hewitt":"Aon Hewitt"},inplace=True)
df.industry.replace({"IT Products and Services":"IT","IT Services":"IT"},inplace=True)
df.location.replace({"CHENNAI":"Chennai","chennai":"Chennai","Gurgaonr":"Gurgaon","-
Cochin- ":"Cochin","chennai ":"Chennai"},inplace=True)
df=df[df.new_date<'2019-01-01']
df.drop(df[(df.hope.isnull()) & (df.three_hours.isnull()) & (df.alt_number.isnull()) &
(df.resume.isnull()) & (df.clear_about_venue.isnull())& (df.letter.isnull())].index,inplace=True)
df.observed.replace({"yes":"Yes","yes ":"Yes","no":"No","No ":"No","NO":"No","no
":"No"},inplace=True)
df.expected.replace({"yes":"Yes","11:00 AM":"Yes","10.30 Am":"Yes"},inplace=True)
df.letter.replace({"Havent Checked":"No","Need To Check":"No","Not sure":"No","Yet to
Check":"No","Not Sure":"No","Not yet":"No",
            "no":"No","na":"No","yes":"Yes","Na":"No"},inplace=True)
df.clear_about_venue.replace({"No- I need to
check":"No","na":"No","yes":"Yes","Na":"No","no":"No"},inplace=True)
df.resume.replace({"No- will take it soon":"No","Not
yet":"No","na":"No","yes":"Yes","Na":"No","Not Yet":"No"},inplace=True)
df.alt_number.replace({"No I have only thi
number":"No","na":"No","yes":"Yes","Na":"No"},inplace=True)
df.three_hours.replace({"No Dont":"No","Na":"No","yes":"Yes"},inplace=True)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
df.hope.replace({"Na":"No","yes":"Yes","Not Sure":"No","cant Say":"No","Not sure":"No"},inplace=True)
df.permission.replace({"Not yet":"No","Yet to confirm":"No","yes":"Yes","Na":"No"},inplace=True)
for i,s in df.iterrows():
    if (df["expected"][i]=="Uncertain"):
        df["expected"][i]=df["observed"][i]
df.permission.replace({"Yes":1,"No":0},inplace=True)
df.hope.replace({"Yes":1,"No":0},inplace=True)
df.three_hours.replace({"Yes":1,"No":0},inplace=True)
df.alt_number.replace({"Yes":1,"No":0},inplace=True)
df.resume.replace({"Yes":1,"No":0},inplace=True)
df.clear_about_venue.replace({"Yes":1,"No":0},inplace=True)
df.letter.replace({"Yes":1,"No":0},inplace=True)
df.expected.replace({"Yes":1,"No":0},inplace=True)
df.observed.replace({"Yes":1,"No":0},inplace=True)
df.married.replace({"Married":1,"Single":0},inplace=True)
df.drop(df[(df.permission.isnull()) | (df.hope.isnull()) | (df.three_hours.isnull()) |
         (df.alt_number.isnull()) | (df.resume.isnull()) | (df.clear_about_venue.isnull()) |
        (df.letter.isnull()) | (df.expected.isnull()) | (df.observed.isnull())].index,inplace=True)
df.inter_type.replace({"Scheduled Walkin":"Scheduled","Scheduled ":"Scheduled","Scheduled Walk In":"Scheduled","Walkin ":"Walkin"},inplace=True)
df.curr_location.replace({"chennai":"Chennai","chennai ":"Chennai","CHENNAI":"Chennai"},inplace=True)
dum_cl=pd.get_dummies(df2.client,prefix="cl",drop_first=True)
dum_ind=pd.get_dummies(df2.industry,prefix="ind",drop_first=True)
dum_jbl=pd.get_dummies(df2.job_location,prefix="jbl",drop_first=True)
#dum_loc=pd.get_dummies(df2.location,prefix="loc",drop_first=True)
dum_pos=pd.get_dummies(df2.position,prefix="pos",drop_first=True)
```

[illegible]

Globsyn Knowledge Foundation Kolkata

```
#dum_skills=pd.get_dummies(df2.skills,prefix="skills",drop_first=True)
dum_int_type=pd.get_dummies(df2.inter_type,prefix="int_type",drop_first=True)
dum_gender=pd.get_dummies(df2.gender,prefix="gender",drop_first=True)
#dum_curr=pd.get_dummies(df2.curr_location,prefix="curr",drop_first=True)
dum_venue=pd.get_dummies(df2.venue,prefix="venue",drop_first=True)

dfn["year"]=dfn.new_date.dt.year
dfn["month"]=dfn.new_date.dt.month
dfn["day"]=dfn.new_date.dt.day
dfn["day_of_week"]=dfn.new_date.dt.dayofweek
dfn["week"]=dfn.new_date.dt.weekofyear
dfn.head()
dum_year=pd.get_dummies(dfn.year,prefix="year",drop_first=True)
dum_month=pd.get_dummies(dfn.month,prefix="month",drop_first=True)
dum_dayofweek=pd.get_dummies(dfn.day_of_week,prefix="dayofweek",drop_first=True)
dum_week=pd.get_dummies(dfn.week,prefix="week",drop_first=True)

dfn=pd.concat([dfn,dum_year,dum_month,dum_dayofweek,dum_week],axis=1)
#df
dfn.drop(["new_date","year","month","day","day_of_week","week"],axis=1,inplace=True)
get_feat_dt(dfn.drop("observed",axis=1),dfn.observed)
get_feat_rf(dfn.drop("observed",axis=1),dfn.observed)
run_dt(dfn.drop("observed",axis=1),dfn.observed,15)
run_rf(dfn.drop("observed",axis=1),dfn.observed,15)
modelstats1(dfn[['gender_3', 'ind_4', 'ind_6', 'jbl_8', 'pos_7', 'pos_8', 'p_1', 'e_1',
    'm_1', 'month_2', 'dayofweek_5', 'week_7', 'week_11', 'week_14',
    'week_17']],dfn.observed)
modelstats1(dfn[['gender_3', 'ind_4', 'ind_6', 'jbl_8', 'pos_7', 'pos_8', 'p_1', 'e_1',
    'm_1', 'month_2', 'dayofweek_5', 'week_7', 'week_11', 'week_14',
```

[illegible]

Globsyn Knowledge Foundation Kolkata

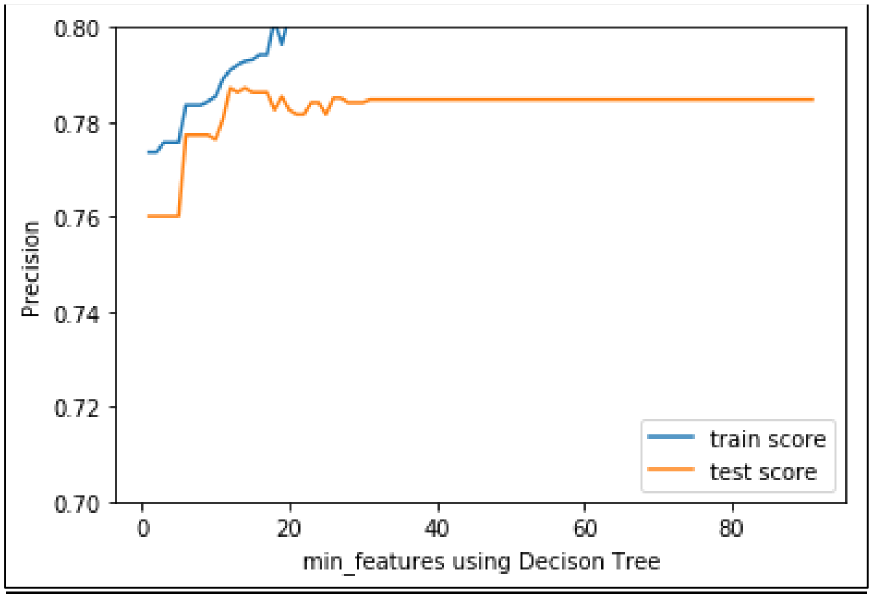
```
'week_17']],dfn.observed)
```

[illegible]

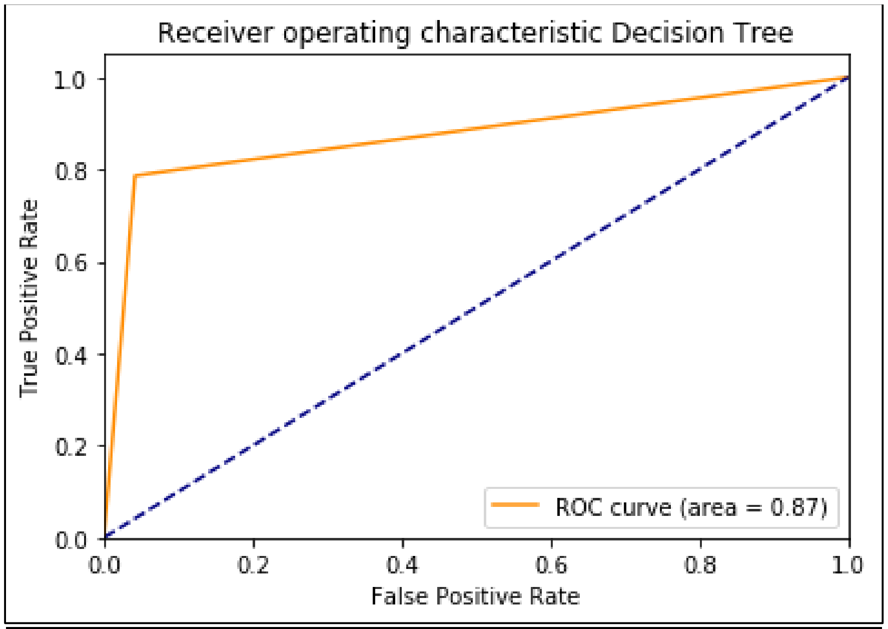
Globsyn Knowledge Foundation
Kolkata

Related Graphs

- **Decision Tree with RFECV**



ROC-AUC Curve

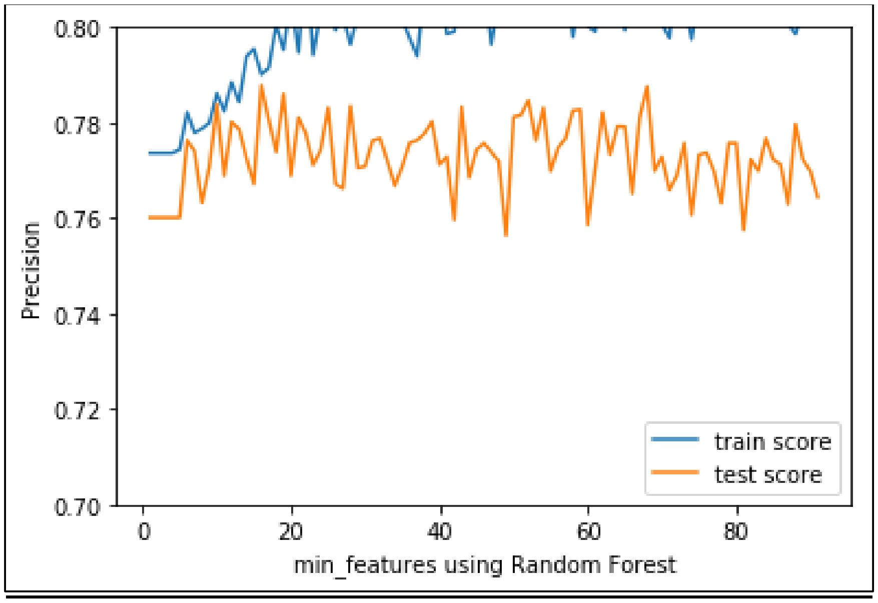


[illegible]

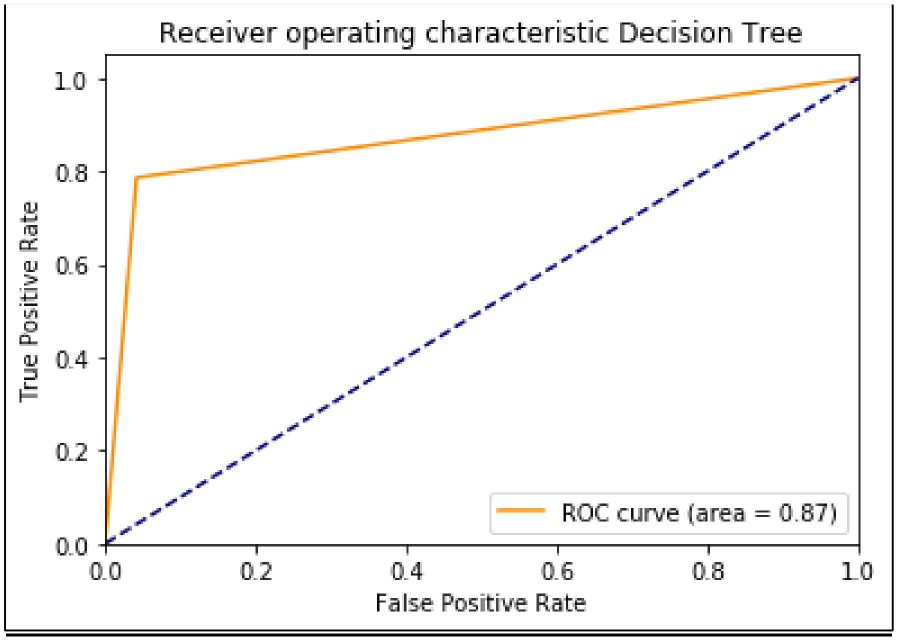
Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

- **Random Forest with RFECV**



<u>ROC-AUC Curve</u>



[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Conclusion

After thorough analysis of the dataset and application of various models through varied approaches, we arrived at our final model. Our aim was to maximize the precision while taking care of other performance metrics as well.
We got two final models :

- **Model of First Approach**

Train Set
precision 81.48148148148148
recall 95.46351084812623
accuracy 81.97831978319783
log loss 0.36358515520145684
Test Set
precision 77.27272727272727
recall 90.53254437869822
accuracy 75.20325203252033
log loss 1.600756778581371

- **Model of Second Approach**

Train Set
precision 79.04109589041096
recall 97.96264855687606
accuracy 82.00654307524536
log loss 0.3729174499199026
Test Set
precision 78.01724137931035
recall 96.79144385026738
accuracy 81.37254901960785
log loss 0.9561643505977544

The model can further be improved :

- **SkillSet column was unused and can be incorporated in the model.**
- **Boosting Algorithms could be used to improve the performance of the models.**
- **SVM and Neural Networks would also increase the performance metrics of the models.**

[illegible]

Globsyn Knowledge Foundation
Kolkata

Certificate

This is to certify that Mr. PRAVASH RANJAN NAYAK, KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 161020110024,has successfully completed a project on 'Interview Prediction' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

____

(Mr. Titas Roy Chowdhury)

Globsyn Finishing School

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Certificate

This is to certify that Mr. SYEDA ZARIF AFRIN, KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 161020110047, as successfully completed a project on 'Interview Prediction' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

____

(Mr. Titas Roy Chowdhury)

Globsyn Finishing School

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

Certificate

This is to certify that Mr. AMAN KUMAR GUPTA, KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 161020110004,has successfully completed a project on 'Interview Prediction' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

(Mr. Titas Roy Chowdhury)

Globsyn Finishing School

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

<u>Certificate</u>

This is to certify that Mr. AKASH RAY, KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 161020110051,has successfully completed a project on 'Interview Prediction' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

____

(Mr. Titas Roy Chowdhury)

Globsyn Finishing School

[illegible]

Globsyn Knowledge Foundation Kolkata

Document sign date :May 3, 2019

THANK YOU

[illegible]

Globsyn Knowledge Foundation
Kolkata

Document sign date :May 3, 2019