Machine Learning Project Report
Title - Bank Marketing

Submitted by:
102003721- Palak
102003708- Sanya Sachdeva

BE Third Year, COE

Submitted to: Dr. Harpreet Singh
Designation of Faculty

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

B

Computer Science and Engineering Department, TIET, Patiala

November 2022

Introduction

This project is related to the telemarketing. Telemarketing is a domain in which companies directly contact customers and try to sell the products which can help customer to achieve their goals.

In telemarketing, customer can be contacted by any means like e-mails, calls, etc. As part of telemarketing process, customer is contacted and provided information about a product and it can sometimes be required to contact customer more than one time.

The problem in telemarketing domain is company have to call all of the customers be it a potential customer or not. This become a lot of logistical work and lot of efforts are wasted on the customers which don't fit in the potential buyer category for that product.

This problem requires solution so that company can target only the potential customers which have higher probability to buy that product instead of wasting efforts on the customers which have lower probability of buying that product.

So past data can be used for the data analysis and ultimately machine learning techniques can be applied to predict the potential buyers as per various attributes related to customers.

Machine Learning is one of the most popular sub-fields of Artificial Intelligence. Machine learning concepts are used almost everywhere, such as Healthcare, Finance, Infrastructure, Marketing, Self-driving cars, recommendation systems, chatbots, social sites, gaming, cyber security, and many more. Currently, Machine Learning is under the development phase, and many new technologies are continuously being added to Machine Learning. It helps us in many ways, such as analyzing large chunks of data, data extractions, interpretations, etc. Hence, there are unlimited numbers of uses of Machine Learning.

Machine learning is a powerful tool and if it's implemented in activities such as telemarketing we can improve pitches and staff, we can enhance personnel into well-trained professionals and provide better service to our clients that will go beyond their expectations. We have a bright future where machines and humans can work together in unison, optimize communication and make our economy grow. Thus, we should embrace a world that sooner or later will work as a singularity where machines and humans can

connect between them seamlessly, seeking for a more united, optimal and dynamic economical and sociological system.

A number of reviews pertaining to not only the diverse factors like personal, socio-economic, psychological and other environmental variables that influence the performance of students but also the models that have been used for the performance prediction are available in the literature Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Real-world raw data and images are often incomplete, inconsistent and lacking in certain behaviors or trends. They are also likely to contain many errors. So, once collected, they are pre-processed into a format the machine learning algorithm can use for the model.

Background

Now days, we have a lot of tools that make our job of data analysis very easy and we can apply different machine learning algorithms to get the required results. But as there are lots of tools and different machine learning algorithms like Nave Bayes, Linear Regression etc., we need to find a proper combination of both so that the analysis is efficient and correct.

To solve this problem, different tools and algorithms performance were compared on serial as well as parallel platform. Frameworks like SAS, SPSS, Weka, Scikit-Learn, and libraries for R allow deep analysis of smaller datasets. But these methods fall short when it comes to big data; they are inefficient and slow. Hence parallel implementations of machine learning algorithms were developed. One of the tools that can be used for this purpose is Sparks MLlib. This library in spark has parallel implementations of certain machine learning algorithms. The frameworks considered in this paper are Weka, Scikit-Learn and Sparks MLlib.

Apache spark is a fast, general engine for large-scale data processing. It runs programs up to 100 times faster than Hadoop, Map-Reduce in-memory, or 10 times faster on disk [3]. MLlib [4] is Sparks scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives.

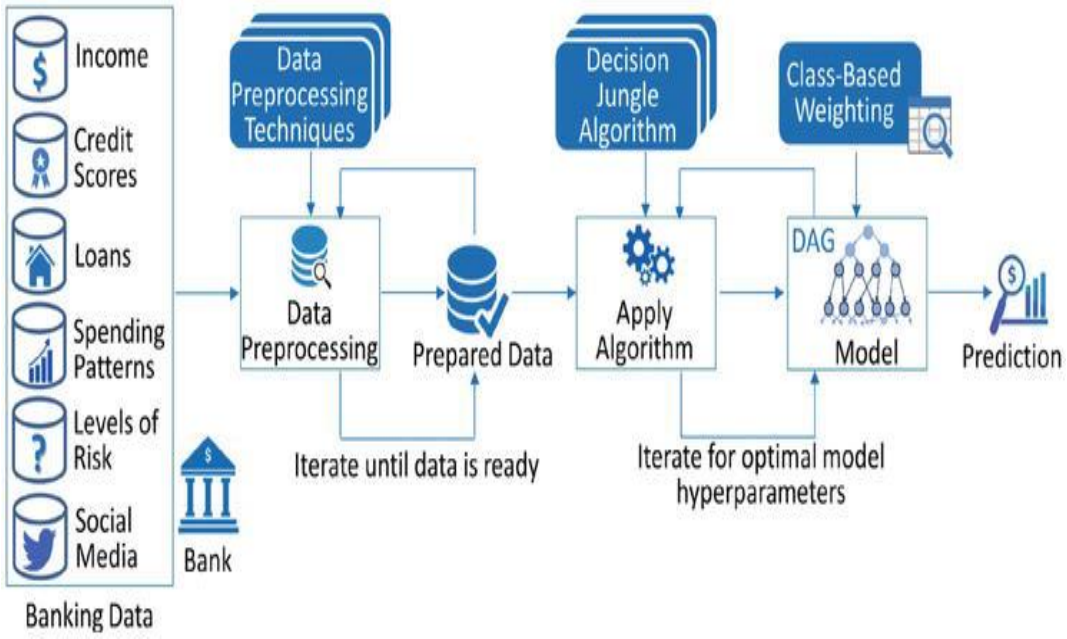
Comparison results vary with the dataset. David D. Lewis et al. in [5] have compared Bayesian classifier and a decision tree learning algorithm on two text categorization data sets. It was seen that both algorithms perform reasonably but had pre filtering support.

Andrade et al. have implemented Nave Bayes algorithm on Compute Unified Device Architecture (CUDA) platform to improve document classification on web. CUDA is a parallel computing platform and programming model manufactured by NVDIA. It utilizes Graphics Processing Units (GPUs). It was seen that parallelized Nave Bayes maintained the accuracy of classification but was 34 times faster than the sequential CPU based implementation of the same algorithm and 11 times faster when it ran on 4 threads on CPU .

Ismail Fahmi discusses the importance of representing information digitally to help users and how with the growth in amount of information, manual classification became difficult. He examines the performance of

Iterative Dichotomiser 3 (ID3), Instance-Based Learning and Nave Bayes algorithms for automatic text classification. All three algorithms were tested on various datasets feature size affected error rate. Error rate first decreased with increase in number of features but after a certain point, it started increasing again. Overall winner turned out to be ID3and Nave Bayes had the highest computation time among the considered classifiers.

Architecture



Data Set Description

This dataset has been taken from the UCI Machine learning repository. This data is related to direct marketing campaigns (phone calls) of Portuguese banking institution. The goal is to predict if client will subscribe a term deposit or not based on some attributes of the client so that potential clients can be targeted instead of wasting efforts on the clients which have very low probability of buying the term deposit.

Following steps will be used to solve this problem.

- Data will be collected.
- Data will be loaded.
- Various features will be analyzed to find if any strong relation between any of the feature and outcome or within feature themselves.
- Data will be pre-processed like numerical features will be transformed so that they should be on single scale and categorical features will be encoded to numerical by using one-hot encoding and feature engineering and feature selection will be performed.

- This is classification problem so various classification algorithms will be used like logistic regression, Support vector classifier, Random Forest Classifier, Ada-Boost Classifier, Gradient-Boost Classifier and K-Nearest Neigbours.
- Best classifier will be selected among mentioned classifiers on the basis of F score and various other parameters like training and testing times.
- Best classifier will be further tuned by tuning various hyper-parameters.
- Top features will be selected on the basis of feature importance and accuracy will be checked with and without feature selection and accordingly feature selection will be done.
- Stability of the classifier will be tested by using K-Fold.

Description

Dataset contains information related to direct marketing campaigns of Portuguese banking institution. The marketing campaigns were based on phone calls and often more than one contact to the same client was required, to access if the product would be yes or no for the subscription.

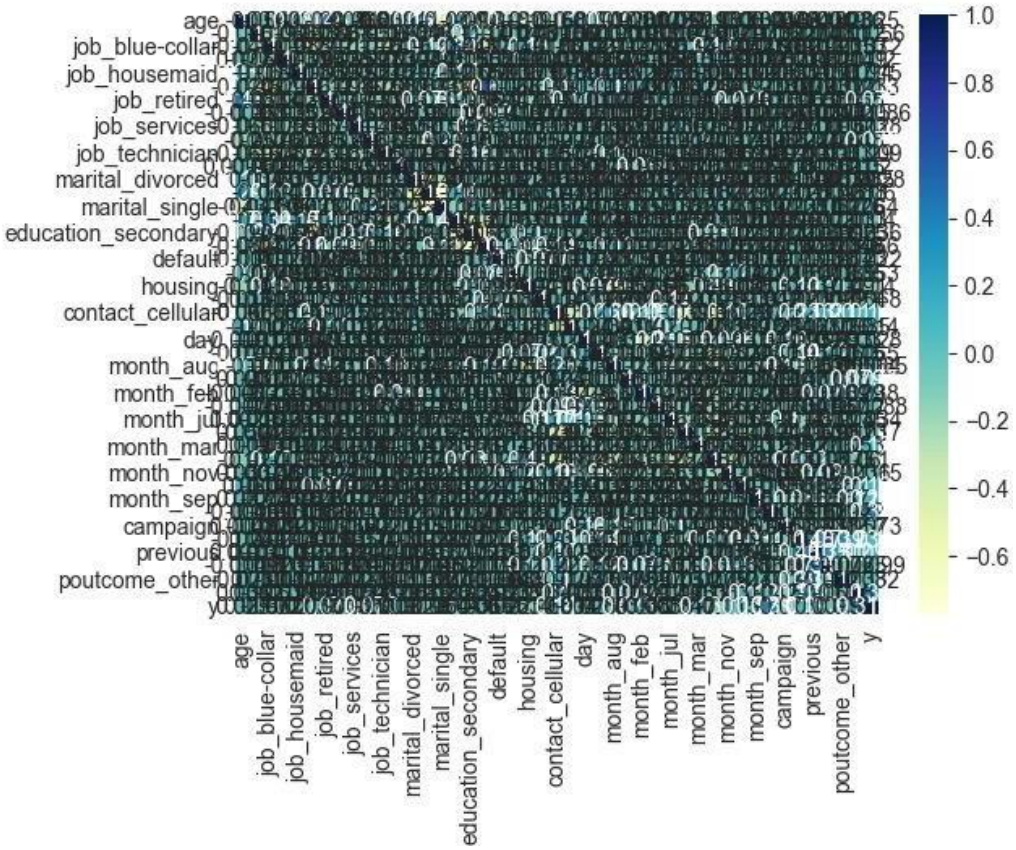
Dataset contains 41188 examples and 20 input variables. Following are the input variables.

1. **Age:** Age is the factor which can impact client interest in the term deposit.
2. **Job:** This is type of job client have.
3. **Marital:** This is marital status of the client.
4. **Education:** This gives educational background of the client.
5. **Default:** Whether client has credit in default.
6. **Housing:** Client has housing loan or not.
7. **Lona:** Client has personal loan or not.
8. **Contact:** contact communication type. Cellular or telephone.
9. **Month:** Last contact month of the year.
10. **Day_of_week:** Last contact day of the week.
11. **Duration:** last contact duration in seconds. This attribute highly affects the output target (if duration = 0 then y = “no”). This input will only be included for the benchmarking purposes and will be discarded for predictive modelling.
12. **Campaign:** Number of contacts performed during this campaign for this client including last contact.
13. **Pdays:** Number of days that passed by after the client was last contacted from a previous campaign. 999 means client was not previously contacted.
14. **Previous:** number of contacts performed before this campaign and for this client.
15. **Poutcome:** Outcome of the previous marketing campaign.
16. **Emp.var.rate:** Employment variation rate - quarterly indicator.
17. **Cons.price.idx:** Consumer price index - monthly indicator.
18. **Cons.conf.idx:** Consumer confidence index - monthly indicator.
19. **Euribor3m:** Euribor 3 month rate - daily indicator.
20. **Nr.employed:** Number of employees - quarterly indicator.
21. **Y:** Output variable. Has client subscribed to term deposit or not?

Preprocessing:

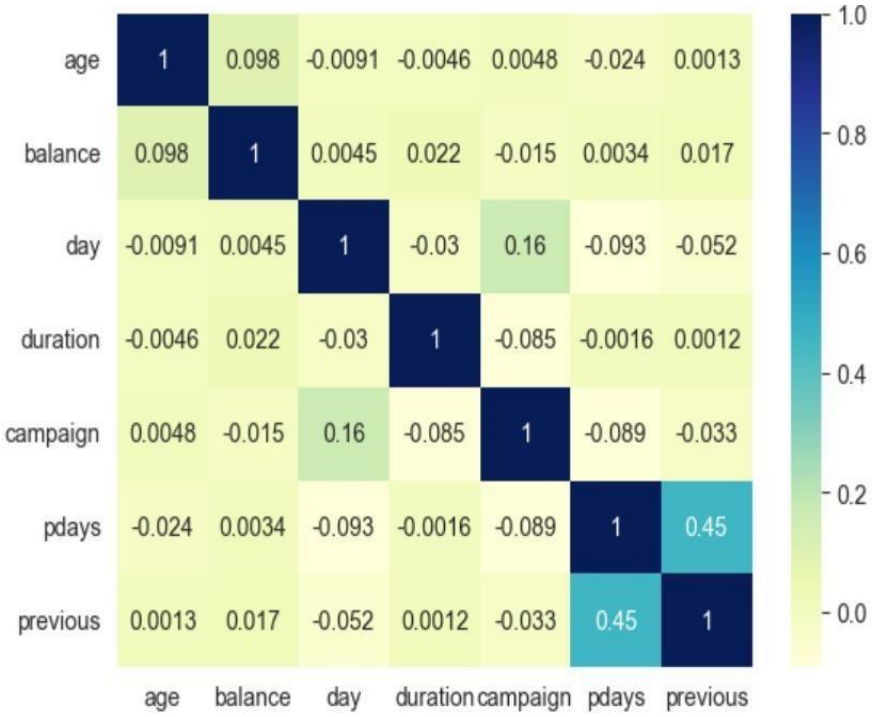
Preprocessing is applied on the dataset and correlation is found out, boxplot and count plot is also plotted.

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn. heatmap() function.



This figure shows Heatmap showing correlations among all the features of the dataset.
More the temperature on the scale more the similarities between the features.

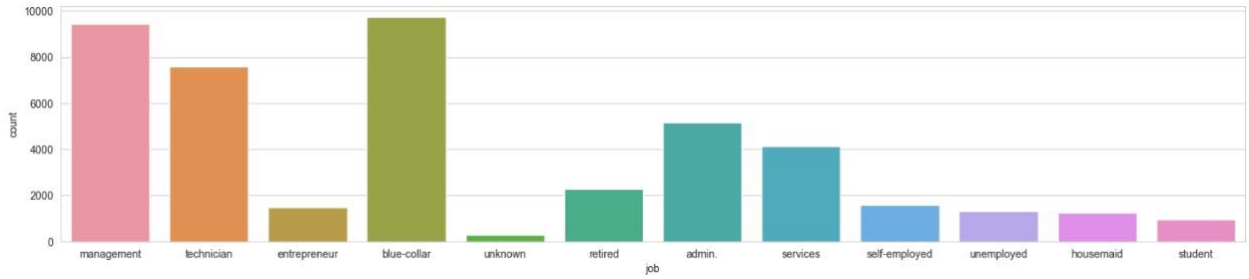
CORELATION MATRIX



This figure is showing a correlation matrix between the features of the dataset showing us similarities between different features.

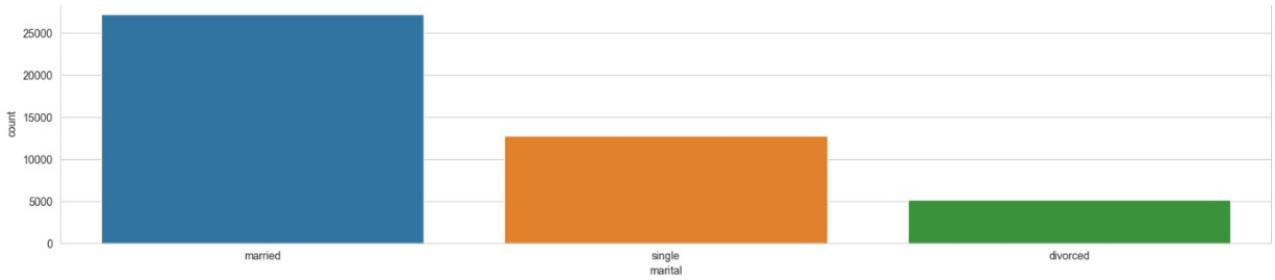
COUNT PLOTS

seaborn.countplot() method is used to Show the counts of observations in each categorical bin using bars.



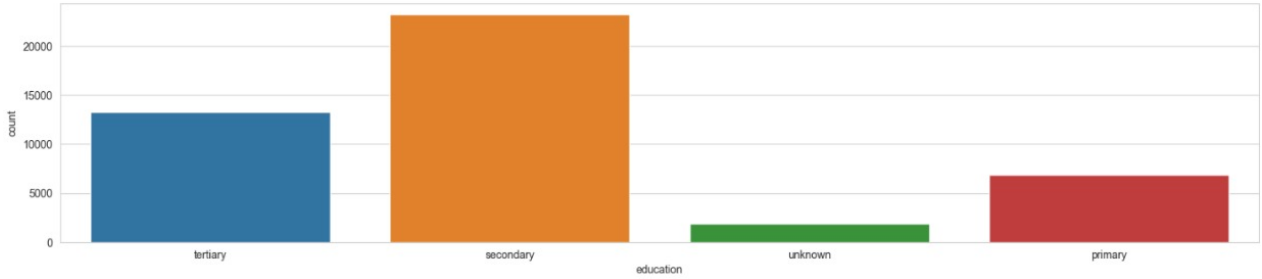
Count Plot-Job

This figure shows a plot between no. of customers and type of job customers have i.e. management, technician,entrepreneur,blue-collor,retired,admin,services,housemaid,student etc.



Count Plot-Marital Status

This figure shows a plot between no. of customers and their martial status i.e. married,single,divorced etc.



Count Plot-Education

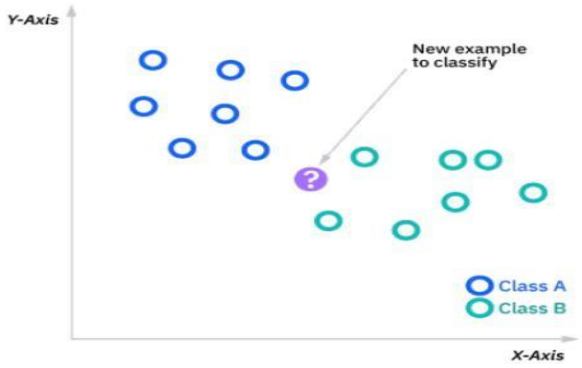
This figure shows a plot between no. of customers and their education qualifications i.e. primary, secondary, tertiary etc.

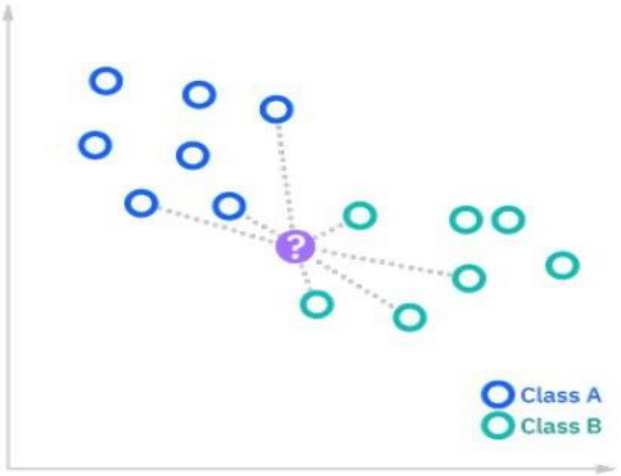
Algorithms Used:

KNN CLASSIFIER

In this technique outcome is found based on the nearest neighbors of the data points. K is the number of neighbors and outcome will be decided based on the number of neighbors belong to that class. Let's say any data point have more nearest neighbors belong to class "no" then outcome will be made as "no".

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered "plurality voting", the term, "majority vote" is more commonly used in literature. The distinction between these terminologies is that "majority voting" technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don't necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%.





The goal of the k-nearest neighbor algorithm is to identify the nearest neighbors of a given query point, so that we can assign a class label to that point. In order to do this, KNN has a few requirements:

Determine distance metrics-

Euclidean distance (p=2):

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Manhattan distance (p=1):

$$d(x,y) = \left(\sum_{i=1}^{m} \left| x_i - y_i \right| \right)$$

Minkowski distance:

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} \left| x_i - y_i \right| \right)^{1/p}$$

Hamming distance:

$$\text{Hamming Distance} = D_H = \left(\sum_{i=1}^{k} |x_i - y_i|\right)$$

$$x=y \quad D=0$$
$$x \neq y \quad D \neq 1$$

The following code is an example of how to create and predict with a KNN model:

```
from sklearn.neighbors import KNeighborsClassifier
model_name = 'K-Nearest Neighbor Classifier'
knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p=2)
knn_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures), ('classifier' , knnClassifier)])
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

Advantages and disadvantages of the KNN algorithm:

Advantages:

-Easy to implement: Given the algorithm's simplicity and accuracy, it is one of the first classifiers that a new data scientist will learn.

- Adapts easily: As new training samples are added, the algorithm adjusts to account for any new data since all training data is stored into memory.

- Few hyperparameters: KNN only requires a k value and a distance metric, which is low when compared to other machine learning algorithms.

Disadvantages:

- Does not scale well: Since KNN is a lazy algorithm, it takes up more memory and data storage compared to other classifiers. This can be costly from both a time and money perspective. More memory and storage will drive up business expenses and more data can take longer to compute. While different data structures, such as Ball-Tree, have been created to address the computational inefficiencies, a different classifier may be ideal depending on the business problem.

- Curse of dimensionality: The KNN algorithm tends to fall victim to the curse of dimensionality, which means that it doesn't perform well with high-dimensional data inputs. This is sometimes also referred to as the peaking phenomenon (PDF, 340 MB) (link resides outside of ibm.com), where after the algorithm attains the optimal number of features, additional features increases the amount of classification errors, especially when the sample size is smaller.

- Prone to overfitting: Due to the "curse of dimensionality", KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to prevent this from occurring, the value of k can also impact the model's behavior. Lower values of k can overfit the data, whereas higher

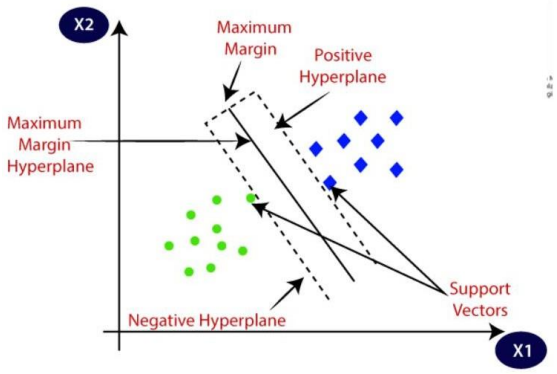
values of k tend to “smooth out” the prediction values since it is averaging the values over a greater area, or neighborhood. However, if the value of k is too high, then it can underfit the data.

LINEAR SVM

Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

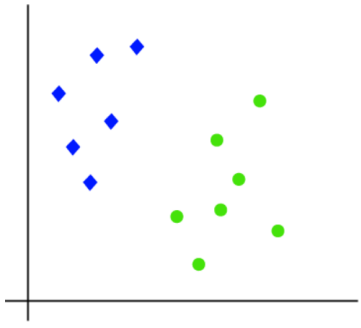
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

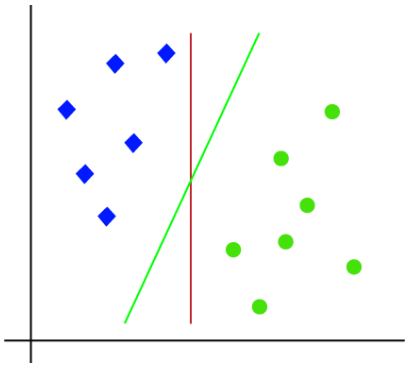
The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Linear SVM:

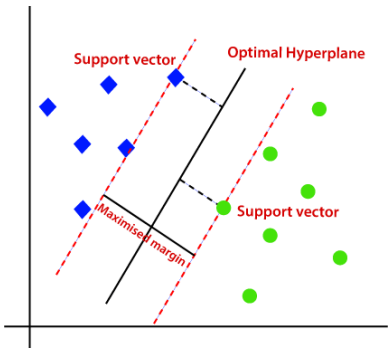
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



NAÏVE BAYES CLASSIFIER

This model is easy to build and is mostly used for large datasets. It is a probabilistic machine learning model that is used for classification problems. The core of the classifier depends on the Bayes theorem with an assumption of independence among predictors. That means changing the value of a feature doesn't change the value of another feature.

Why is it called Naive?

It is called Naive because of the assumption that 2 variables are independent when they may not be. In a real-world scenario, there is hardly any situation where the features are independent. Since it is a probabilistic approach, the predictions can be made really quick. It can be used for both binary and multi-class classification problems.

Conditional Probability for Naive Bayes:

Conditional probability is defined as the likelihood of an event or outcome occurring, based on the occurrence of a previous event or outcome. Conditional probability is calculated by multiplying the probability of the preceding event by the updated probability of the succeeding, or conditional, event.

Mathematically, the conditional probability of event A given event B has already happened is given by:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Probability of event A given B has occured

Probability of event A occured and event B occured

Probability of event B

Bayes Rule

Now we are prepared to state one of the most useful results in conditional probability: Bayes' Rule.

Bayes' theorem which was given by Thomas Bayes, a British Mathematician, in 1763 provides a means for calculating the probability of an event given some information.

Mathematically Bayes theorem can be stated as:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

Basically, we are trying to find the probability of event A, given event B is true.

Here P(B) is called prior probability which means it is the probability of an event before the evidence P(B|A) is called the posterior probability i.e., Probability of an event after the evidence is seen. Bayes' rule provides us with the formula for the probability of Y given some feature X. In real-world problems, we hardly find any case where there is only one feature.

When the features are independent, we can extend Bayes' rule to what is called Naive Bayes which assumes that the features are independent that means changing the value of one feature doesn't influence the values of other variables and this is why we call this algorithm "*NAIVE*"

Naive Bayes can be used for various things like face recognition, weather prediction, Medical Diagnosis, News classification, Sentiment Analysis, and a lot more.

When there are multiple X variables, we simplify it by assuming that X's are independent, so

$$P(Y=k|X) = \frac{P(X|Y=k)*P(Y=k)}{P(X)}$$

For n number of X, the formula becomes **Naive Bayes**:

$$P(Y=k|X1,X2....Xn) = \frac{P(X1|Y=k)*P(X2|Y=k).....*P(Xn|Y=k)*P(Y=k)}{P(X1)*P(X2)....*P(Xn)}$$

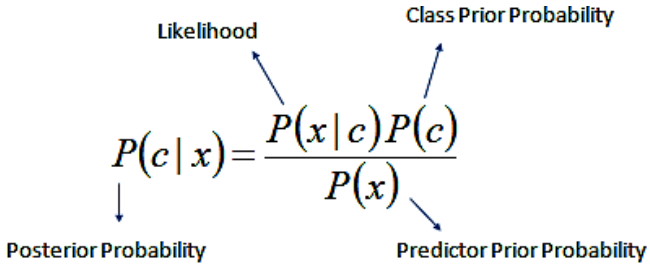
Which can be expressed as:

$$P(Y=k|X1,X2....Xn) = \frac{P(Y)\prod_{i=1}^{n} P(Xi|Y)}{P(X1)*P(X2)....*P(Xn)}$$

Since the denominator is constant here so we can remove it. It's purely your choice if you want to remove it or not. Removing the denominator will help you save time and calculations.

$$P(Y = k|X1, X2. \ldots Xn) \propto \mathrm{P(Y)} \prod_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{P(Xi|Y)}$$

This formula can also be understood as:



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

There are a whole lot of formulas mentioned here but worry not we will try to understand all this with the help of an example.

Decision Tree Using Information Gain

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

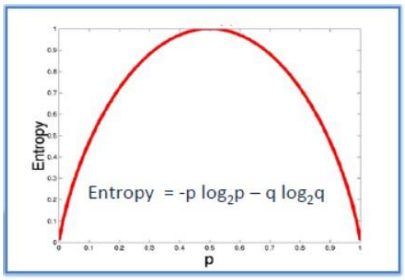
Algorithm

The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and

Information Gain to construct a decision tree. In ZeroR model there is no predictor, in OneR model we try to find the single best predictor, naive Bayesian includes all predictors using Bayes' rule and the independence assumptions between predictors but decision tree includes all predictors with the dependence assumptions between predictors.

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



Entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ = 1

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

Step 4a: A branch with entropy of 0 is a leaf node.

Step 4b: A branch with entropy more than 0 needs further splitting.

Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

Decision Trees - Issues

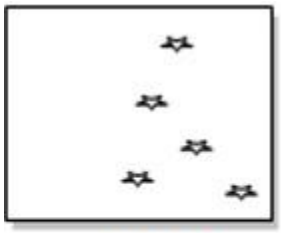
- Working with continuous attributes (binning)
- Avoiding overfitting
- Super Attributes (attributes with many unique values)
- Working with missing values

Decision tree using Gini Index

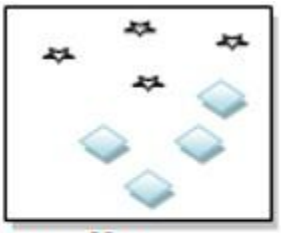
In Decision Tree, the major challenge is to identify the attribute of the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:
The Gini impurity measure is one of the methods used in decision tree algorithms to decide the optimal split from a root node and subsequent splits.Gini index is also known as Gini impurity.

What is the Gini Index?

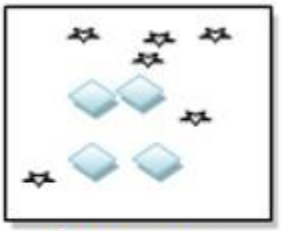
Gini index calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.
If all the elements are linked with a single class then it is called pure.
It ranges from 0-1
0 = all elements
1 = Randomly distributed
0.5 = equally distributed
It means an attribute with a lower Gini index should be preferred.



pure



Equally distributed



Impure

Equation of Gini index:

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

where *pi is* the probability of an object being classified to a specific class.

How the tree is spilted?
1. Target is the decision node.
2. It is subdivided into the parent node
3. Parent node is divided into child node basing on the value of how many 1 or 0 in parent node . These are again divided into leaf node based on target=1 & target=0
(leaf node is end node, it cannot be further divided)
Now calculate the Gini index and will find the one which factor decision is made.
The factor which gives the least Gini index is the winner, i.e,.based on that the decision tree is built.

RANDOM FOREST CLASSIFIER

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

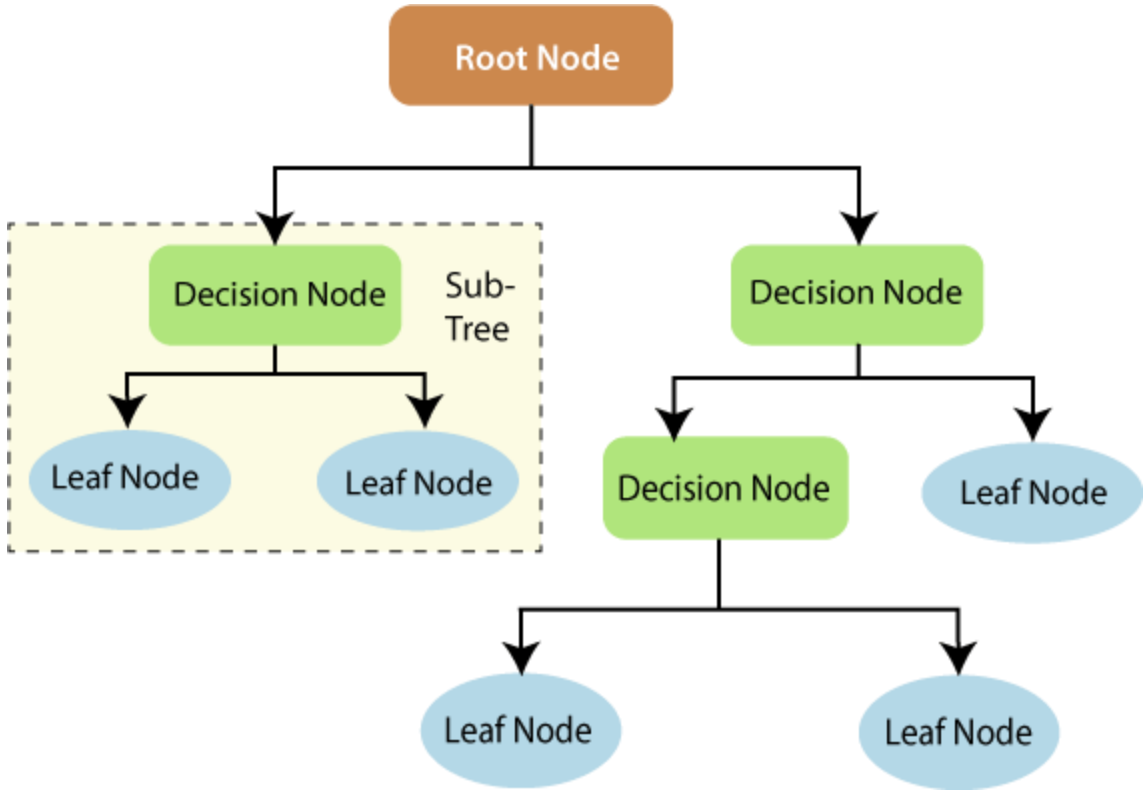
How random forest algorithm works

Understanding decision trees

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.



The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees. An overview of these fundamental concepts will improve our understanding of how decision trees are built.

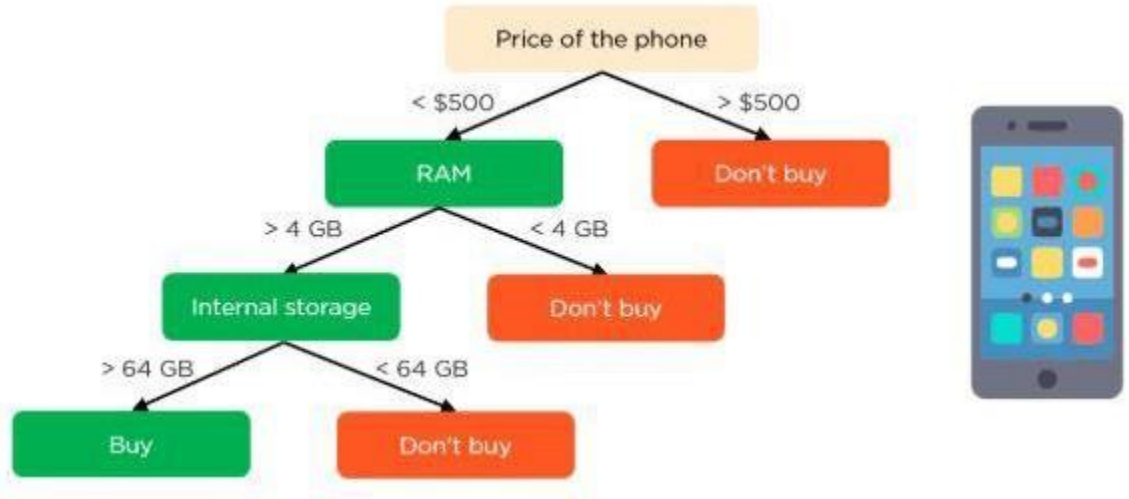
Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the conditional entropy of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Let’s take a simple example of how a decision tree works. Suppose we want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.

The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either *buying* or *not buying*. The main features that determine the choice include the price, internal storage, and Random Access Memory (RAM). The decision tree will appear as follows.



Applying decision trees in random forest

The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction.

Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

Our first example can still be used to explain how random forests work. Instead of having a single decision tree, the random forest will have many decision trees. Let’s assume we have only four decision trees. In this case, the training data comprising the phone’s observations and features will be divided into four root nodes.

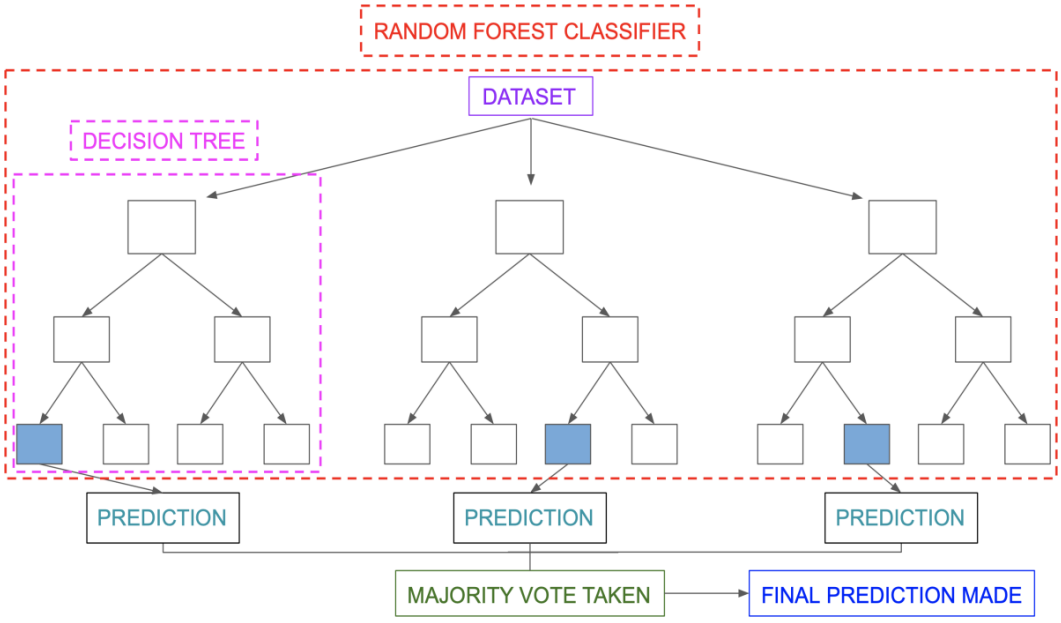
The root nodes could represent four features that could influence the customer's choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees.

The outcome chosen by most decision trees will be the final choice. If three trees predict *buying*, and one tree predicts *not buying*, then the final prediction will be *buying*. In this case, it's predicted that the customer will buy the phone.

Classification in random forests

Classification in random forests employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes.

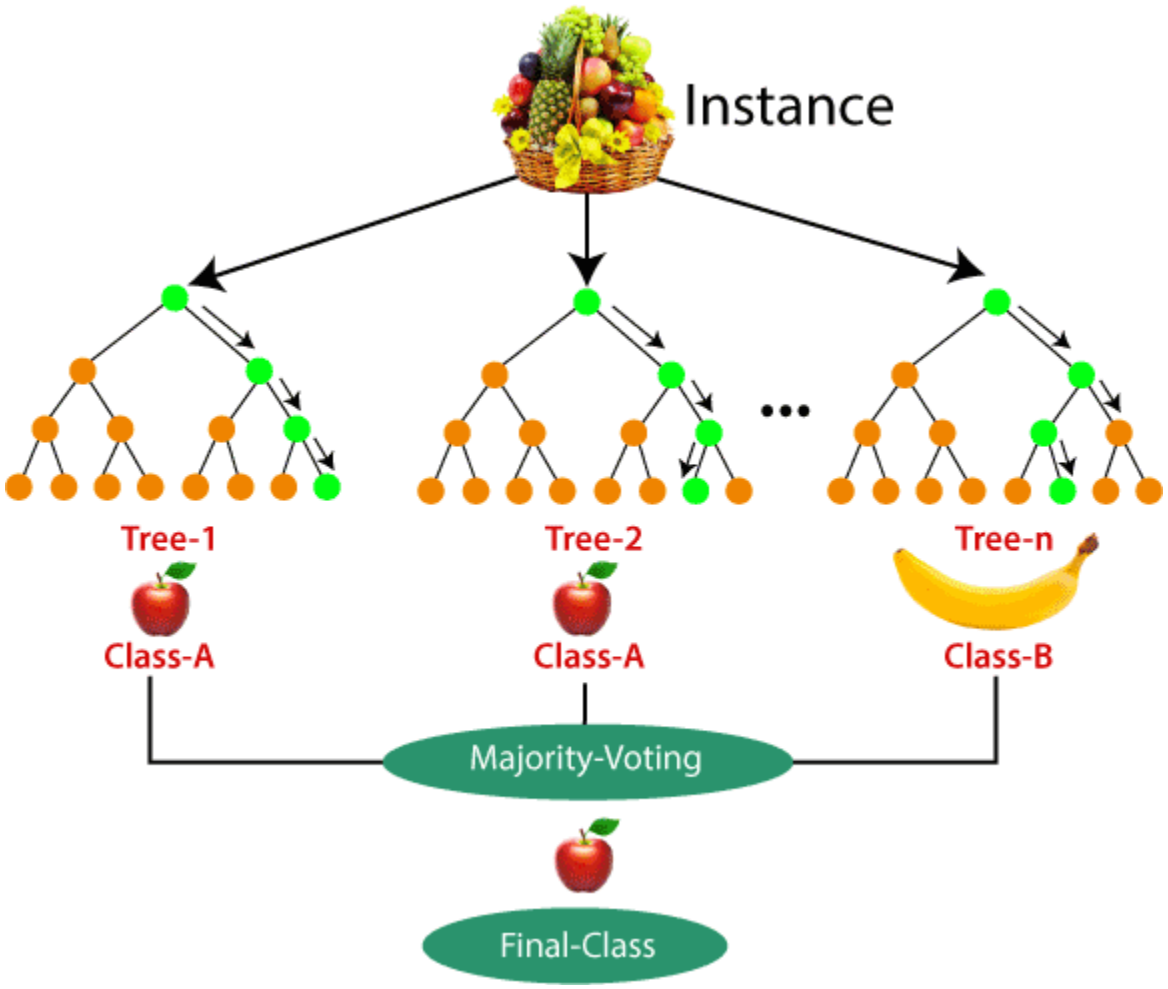
A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system. The diagram below shows a simple random forest classifier.



Let's take an example of a training dataset consisting of various fruits such as bananas, apples, pineapples, and mangoes. The random forest classifier divides this dataset into subsets. These subsets are given to every

decision tree in the random forest system. Each decision tree produces its specific output. For example, the prediction for trees 1 and 2 is *apple*.

Another decision tree (n) has predicted *banana* as the outcome. The random forest classifier collects the majority voting to provide the final prediction. The majority of the decision trees have chosen *apple* as their prediction. This makes the classifier choose *apple* as the final prediction.



Result

In results section, we have created this following table by running the below mentioned classifiers.

		Accuracy	Precision	Recall	F1 score	Sensitivity	Specificity	Error rate	TPR	FPR
1	Logistic regression with regularization	0.9658	0.9658	1	0.9826	1	0	0.0341	1	1
2	KNeighborsClassifier	0.8601	0.8874	0.9642	0.9242	0.9642	0.0559	0.1398	0.9642	0.9440
3	Linear SVM	0.9681	0.9681	1	0.9838	1	0	0.0318	1	1
4	Naive Bayes	0.8496	0.8887	0.9489	0.9178	0.9489	0	0.1503	0.9489	0.9170
5	Decision Tree(information gain)	0.8496	0.8887	0.9489	0.2808	0.9489	0.0829	0.1503	0.9489	0.9170
6	Decision Tree(gini index)	0.9669	0.9669	1	0.9831	1	0	0.0330	1	1
7	Random Forest	0.8601	0.8874	0.9642	0.9242	0.9642	0.0559	0.1146	0.9642	0.944

Confusion Matrix:

KNN:

```
array([[7720,  286],
       [ 979,   58]],
```

Random Forest

```
array([[8006,    0],
       [1037,    0]],
```

Naïve Bayes

```
array([[7597,  409],
       [ 951,   86]],
```

Decision Tree using Information Gain

```
array([[7597,  409],
       [ 951,   86]],
```

Linear SVM

```
array([[8755,    0],
       [ 288,    0]],
```

Logistic Regression with Regularisation

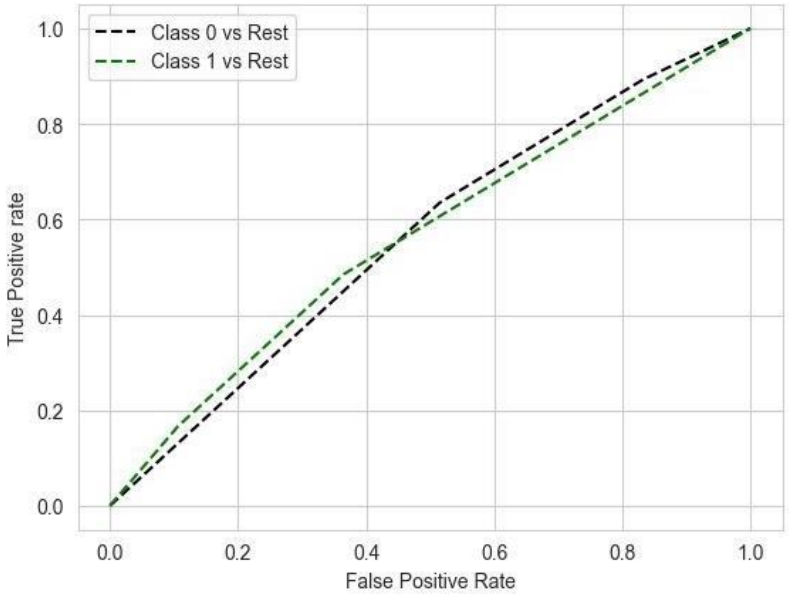
```
array([[8734,    0],
       [ 309,    0]],
```

Decision Tree using Gini Index

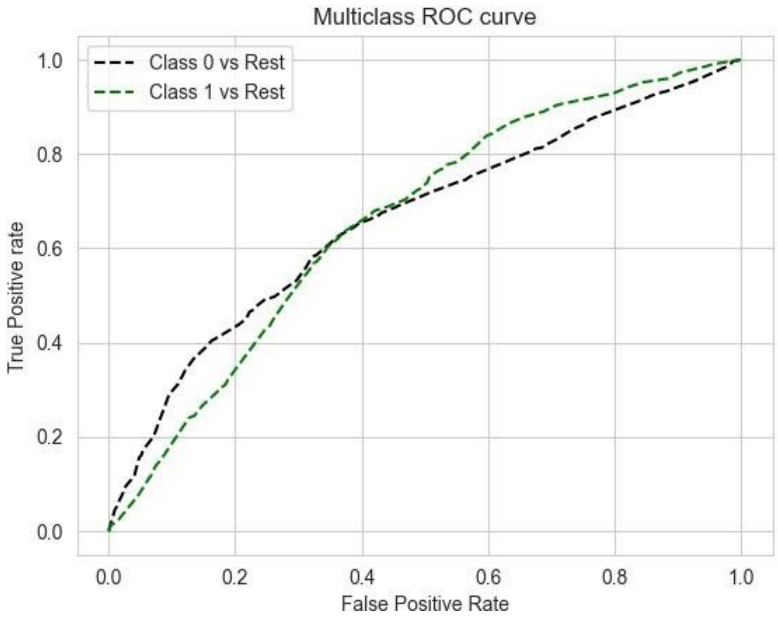
```
array([[32787,     0],
       [ 1121,     0]],
```

ROC Curves:

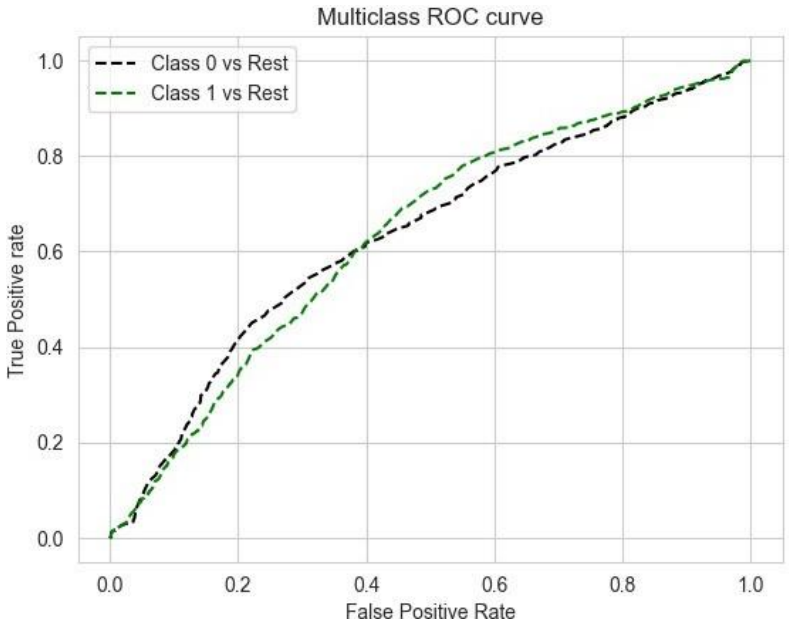
KNN:



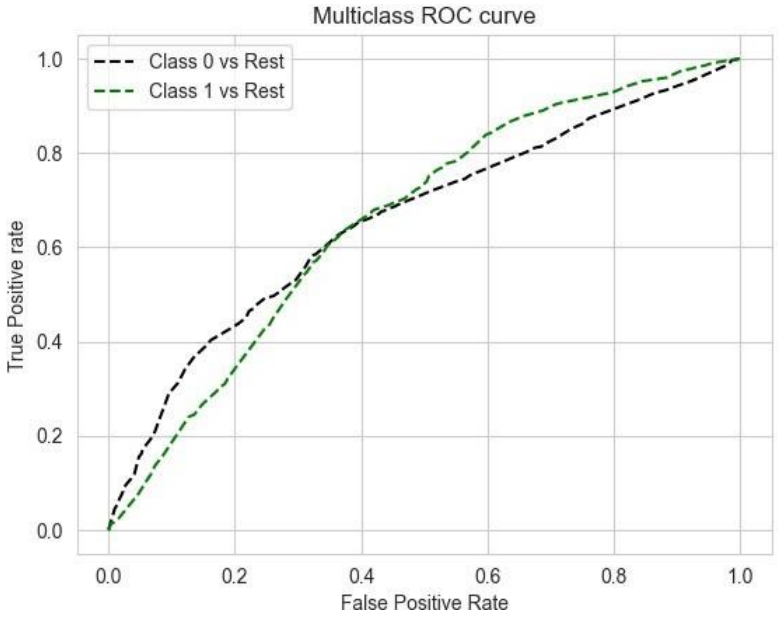
Random Forest:



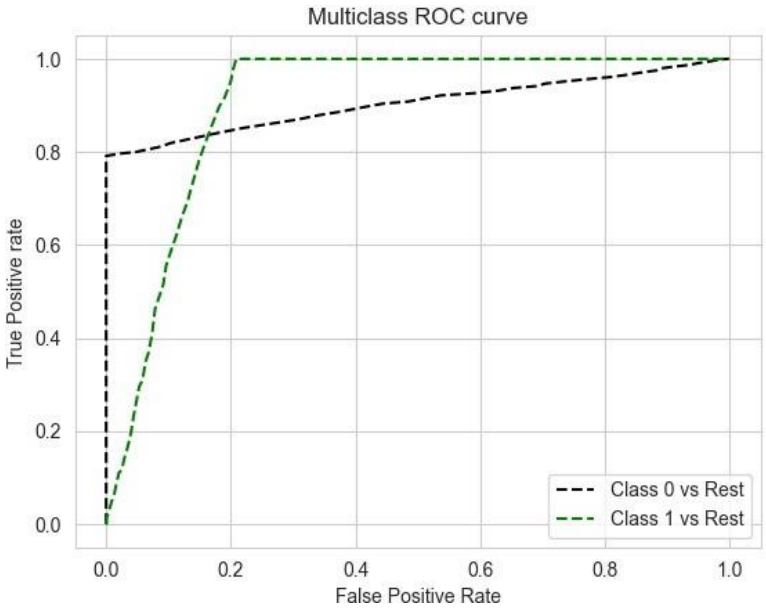
Naïve Bayes:



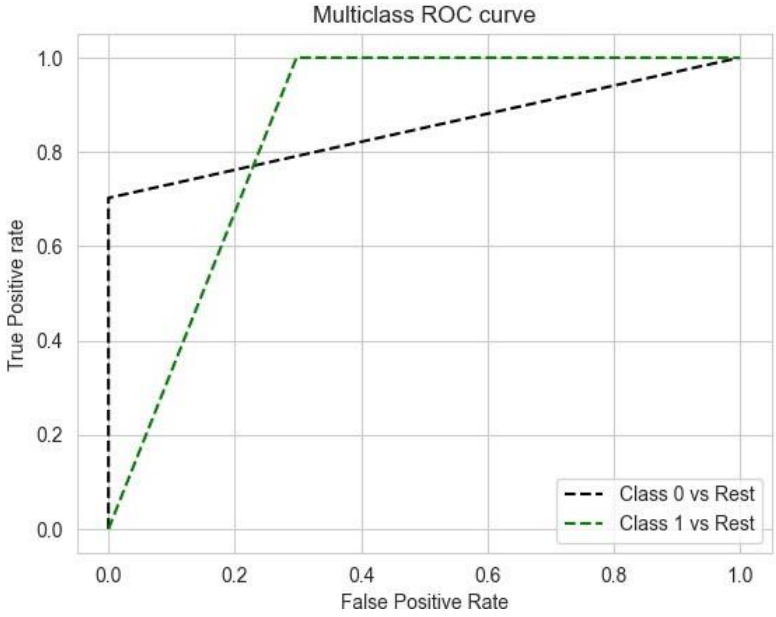
Decision Tree using Information Gain:



Logistic Regression with Regularisation:



Decision Tree using Gini Index:



Conclusion

As discussed during the exploratory analysis, outcome is not conclusively related to any of the feature accept the duration feature which we didn't take into consideration because that shouldn't be taken into

consideration as duration will be known only after calling to customer and if duration = 0 then surely outcome will be “no”.

So, nothing conclusive very high dependency on any of the feature is found which could impact the outcome significantly so all features were required to be used to train the model.

Time required for training and predicting was checked for reduced features data sets as well and it was found that time has reduced drastically with feature selection.

Following is the summary of the entire project:

- Data was acquired and loaded in the python.
- Exploratory data analysis was done, and features were analyzed thoroughly.
- Data pre-processing and feature engineering was also done.
- Multiple models were tried, and accuracy and F score was checked on the validation and test set.
- Finally, best model among tried models was selected.
- Grid search was used to further improve the model by selecting the best hyper-parameter.
- Features were reduced by selecting top features and accuracy was checked on reduced feature models.
- K-fold cross-validation was used to check the model consistency and it was found that model generalized well.
- Final model solves the problem and can be used for these kinds of issue provided features and distribution of features remains same. We can’t fit same model to another problem be it similar type of issue or not.