Bank Marketing Effectiveness Prediction

By

Ajinkya Morade

Nitesh Gajakosh

Data science trainees,

AlmaBetter,

Bangalore

Abstract

This project discusses the prediction model of Bank Marketing Effectiveness of a Portuguese Marketing institution. The marketing campaigns were based on phone calls.The classification goal is to predict if the client will subscribe to a term deposit.The project discusses data preparation, feature engineering techniques to remove non-predictive parameters, used for model training. Nine prediction models are trained and evaluated in the testing set: Logistic regression, Decision Tree Classifier, Support Vector Classifier(SVC), Random forest Classifier (RF), Gradient Boosting Classifier, XGB Classifier,Naive Bayes Classifier, Neural Network Classifier. Among them, the Gradient Boosting Classifier, which is a decision-tree based ensemble learning algorithm gives the best performance.As the dataset is an imbalanced one we are using different performance metrics for evaluation other than the accuracy score. The Gradient Boosting Classifier has the highest f1-score (0.50) in the testing set. The analysis results show that the Gradient Boosting method has advantages in the prediction of bank marketing effectiveness.

Problem Statement

The data is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to assess if the product (bank term deposit) would be ('yes') or not ('no') subscribed. The classification goal is to predict if the client will subscribe to a term deposit (variable y).

Introduction

Data-driven prediction of Bank Marketing Campaign.This dataset is related to direct marketing campaigns of a Portuguese banking institution which can be used to optimize marketing campaigns to attract more customers to term deposit subscription.The marketing campaigns were based on phone calls. The classification goal is to predict if the client will subscribe to a term deposit (target variable y).The dataset had 45, 211 instances with 11.7% positive responses and 17 features out of which 7 are numeric features and 10 are categorical features. This was in addition to the detection of outliers and extreme values.

Data Description

The list of features available to us are given below:-

Input variables:

Bank Client data:-

- age (numeric)
- job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
- marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
- education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
- default: has credit in default? (categorical: 'no','yes','unknown')
- housing: has a housing loan? (categorical: 'no','yes','unknown')
- loan: has a personal loan? (categorical: 'no','yes','unknown')

Related to the last contact of the current campaign:

- contact: contact communication type (categorical: 'cellular','telephone')
- month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
- day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
- duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is

obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Other attributes:

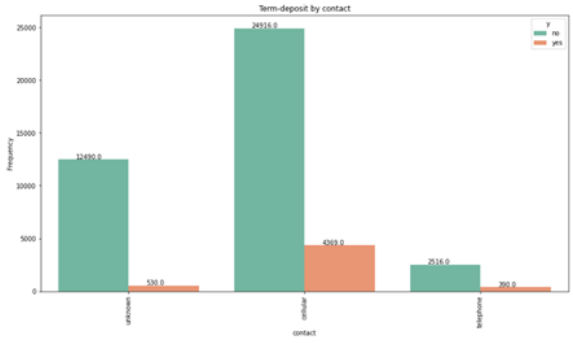
- campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- previous: number of contacts performed before this campaign and for this client (numeric)
- poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Output variable (desired target):

- y - has the client subscribed to a term deposit? (binary: 'yes','no').

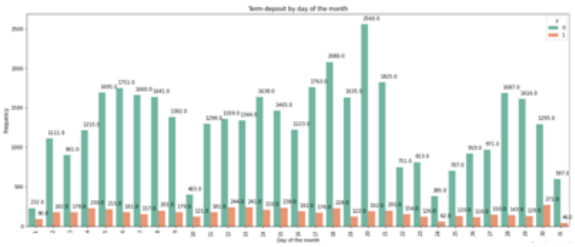
1.Exploratory Data Analysis

❖ Data Analysis with respect to mode of contact chosen



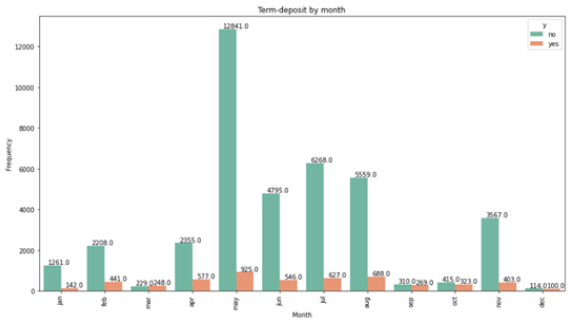
Most of the customers have been contacted by cellular rather than telephone.But the subscription rate is same for both the cellular and telephone mode of contacts. Thus we can say contact made to the customer through the mode of telephone are more likely to subscribe for the term deposit.

❖ Data Analysis with respect to day of the month.



All the days have nearly the similar distribution for people subscribing for the term deposit but the Bank contacts more people between 17-21 and least on 31st and 1st day of the month.

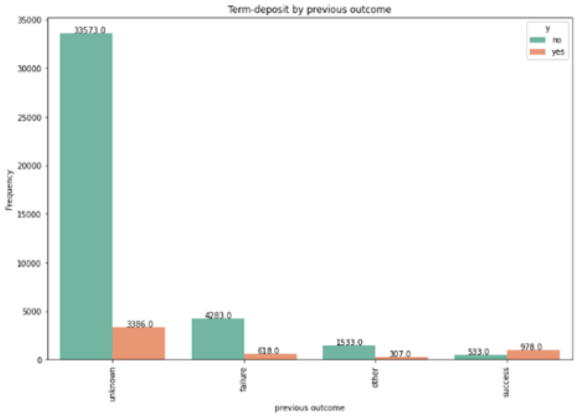
❖ Which month has the highest subscription?



People have been contacted more in the month of May, followed by June, July, August and November. Very few people have been contacted in the month of December. May has highest term deposits but also highest count of non-subscription to the long term deposits. Very few people have been contacted in the months of March,

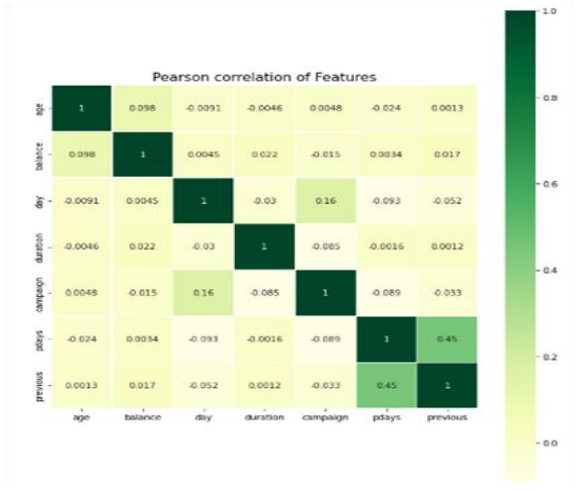
September,October,December and have almost equal chances for subscribing the deposit or not.

- Data analysis on the basis of outcome of the previous campaign.



Outcome of the previous campaign is unknown for the majority of the customers but of the customers who had a successful outcome from the previous campaign, 64% of those customers did subscribe for a term deposit. Which is 4 times of the people whose previous outcome of the campaign was failure.

- Heat Map

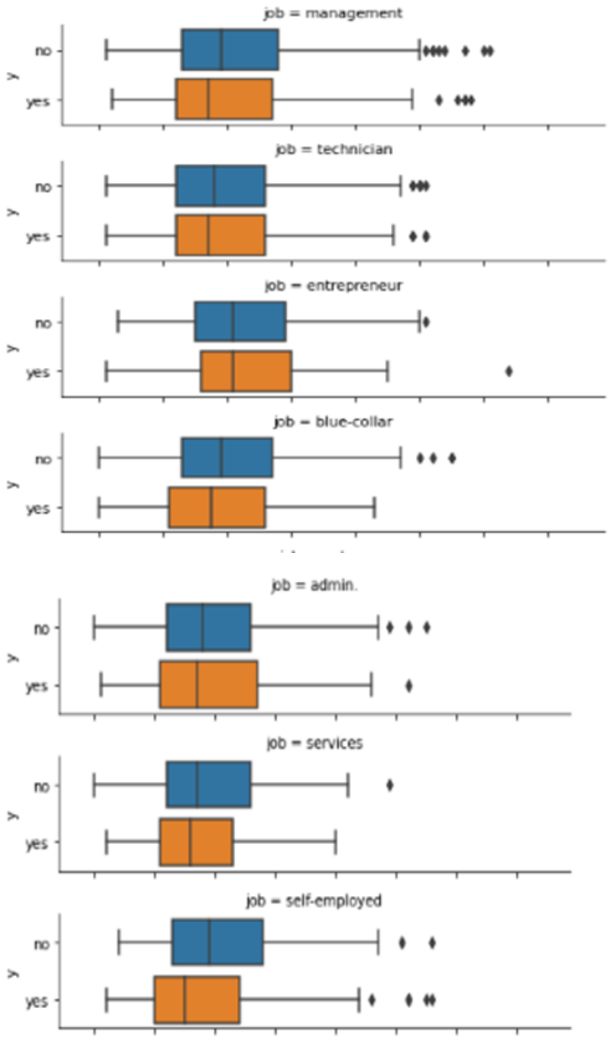


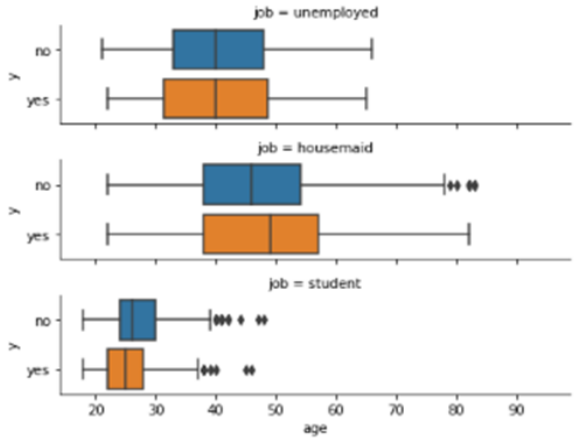
Pday and previous have high correlation of 0.45 and pday has negative correlation with every other variable except balance.

Subscription agewise Density Distribution

The distribution of the customers who subscribed for the term deposit is relatively higher in the age group of 30-60.

- Data Analysis with respect to Age group and Job type.





There is not much age wise difference in subscription to the term deposit if the job profile is entrepreneur, admin, unemployed.

People of profession- management, technician, blue-collar, unknown, services, self employed, student who subscribe to the term deposit are more likely to subscribe in their younger age.

People of profession- unknown, retired, housemaids who subscribe to the term deposit have their mean age higher than those who don't subscribe for the term deposit.

The Impact of Duration in Opening a Term Deposit

Customers whose duration status is above average have 25% chances of subscribing for term deposit.

Analysis to get hidden insights of data .. This process helped us figuring out various aspects and relationships among the target and the independent variables. It gave us a better idea of which feature behaves in which manner compared to the target variable.

The goal of EDA is to leverage visualization tools, summary tables, and hypothesis testing to:

- Provide summary level insight into a dataset.
- Uncover underlying patterns and structures from data.
- Identify outliers, missing data, class balance, and other data-related issues.

Scaling and Encoding

One Hot Encoding

One hot encoding is a process by which categorical features are refashioned into a form that could be used by machine learning algorithms for better prediction .

One hot encoding is applied to categorical data. It refers to variables that are made up of labels, for instance, a 'color' variable could have the values 'red', 'yellow', and 'green'.

Let us take a look at a chart for better understanding.

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Most machine learning algorithms require inputs or output variables to be of numeric type in order

for them to understand. This implies that any categorical data must be mapped to integers**.** With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector, like the one represented in the chart.

Label Encoding

This approach is very simple and it involves converting each value in a column to a number. Consider a dataset of bridges having a column names bridge-types having below values. Though there will be many more columns in the dataset, to understand label-encoding, we will focus on one categorical column only.

1 Algorithms

1. **Logistic regression**

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

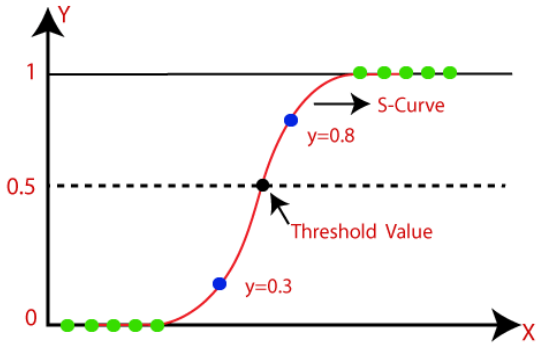
In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Logistic Regression Assumptions

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

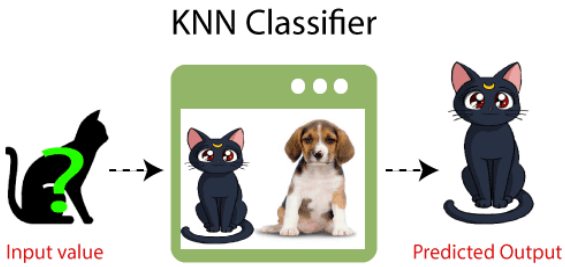
- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other .
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.



K nearest neighbor classification

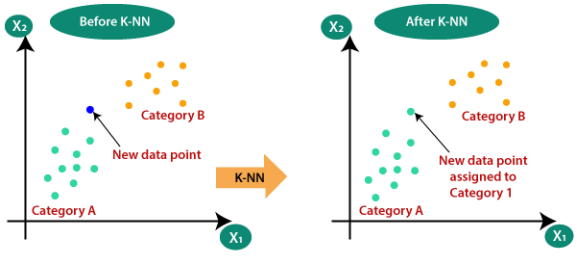
- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

- The K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.
- The K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- The KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

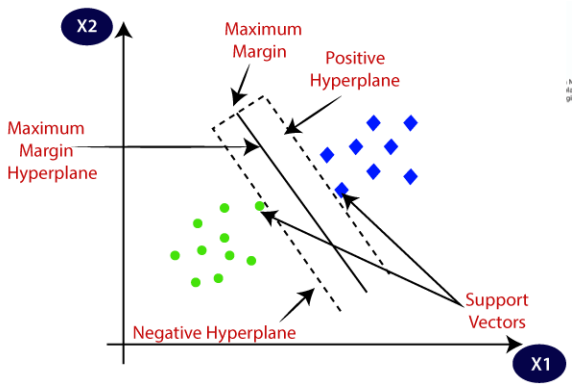


- **Support vector machine classification**

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.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the 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:



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.

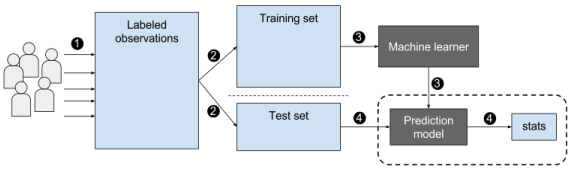
Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

- **Gradient boosting classifier**

  Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets,

  Steps to Gradient Boosting

In order to implement a gradient boosting classifier, we'll need to carry out a number of different steps. We'll need to:



- Fit the model
- Tune the model's parameters and Hyperparameters
- Make predictions
- Interpret the results

Fitting models with Scikit-Learn is fairly easy, as we typically just have to call the fit() command after setting up the model.

However, tuning the model's hyperparameters requires some active decision making on our part. There are various arguments/hyperparameters we can tune to try and get the best accuracy for the model. One of the ways we can do this is by altering the learning rate of the model. We'll want to check the performance of the model on the training set at different learning rates, and then use the best learning rate to make predictions.

Predictions can be made in Scikit-Learn very simply by using the predict() function after fitting the classifier. You'll want to predict on the features of the testing dataset, and then compare the predictions to the actual labels. The process of evaluating a classifier typically involves checking the accuracy of the classifier and then tweaking the parameters/hyperparameters of the model until the classifier has an accuracy that the user is satisfied with.

- **Decision tree classifier**

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

  In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

  The decisions or the test are performed on the basis of features of the given dataset.

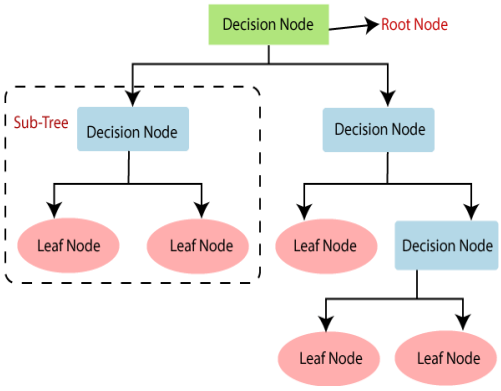
  It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

  It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

  In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

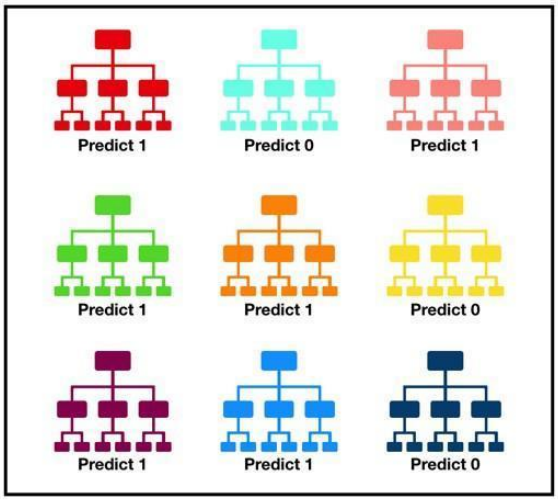
  A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.

  Below diagram explains the general structure of a decision tree:



- Random forest classifier

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).

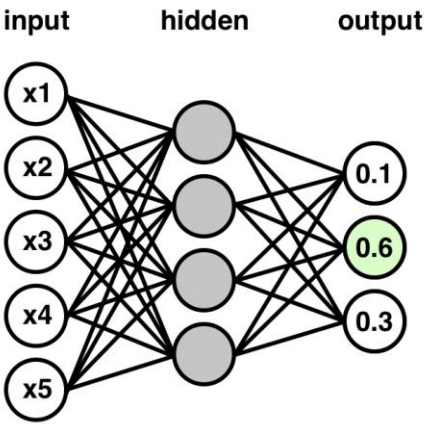


Tally: Six 1s and Three 0s
Prediction: 1

- **Neural network classifier**

A neural network consists of units (neurons), arranged in layers, which convert an input vector into some output. Each unit takes an input, applies a (often nonlinear) function to it and then passes the output on to the next layer. Generally the networks are defined to be feed-forward: a unit feeds its output to all the units on the next layer, but there is no feedback to the previous layer. Weightings are applied to the signals passing from one unit to another, and it is these weightings which are tuned in the training phase to adapt a neural network to the particular problem at hand. This is the learning phase.Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to

Train on.



- **Naive bayes classifier**

The Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

The Naïve Bayes Classifier is one of the simplest and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Bayes' Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

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

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Stratified Sampling:

Stratified random sampling is a type of probability sampling using which a research organization can branch off the entire population into multiple non-overlapping, homogeneous groups (strata) and randomly choose final members from the various strata for research which reduces cost and improves efficiency.

Stratified Sampling: Is an important concept that is often missed when developing a model either for regression or classification. Remember, that in order to avoid overfitting of our data we must implement a cross validation however, we must make sure that at least the features that have the greatest influence on our label (whether a potential client will open a term deposit or not) is equally distributed. What do I mean by this?

Personal Loans:

For instance, having a personal loan is an important feature that determines whether a potential client will open a term deposit or not. To confirm it has a heavy weight on the final output you can check the correlation matrix above and you can see it has a -11% correlation with opening

a deposit. What steps should we take before implementing stratified sampling in our train and test data?

1) We need to see how our data is distributed.

2) After noticing that the column of loan contains 87% of "no" (Does not have personal loans) and 13% of "yes" (Have personal loans.)

3) We want to make sure that our training and test set contains the same ratio of 87% "no" and 13% "yes"." Stratified Sampling: Is an important concept that is often missed when developing a model either for regression or classification. Remember, that in order to avoid overfitting of our data we must implement a cross validation; however, we must make sure that at least the features that have the greatest influence on our label (whether a potential client will open a term deposit or not) is equally distributed.

Handling imbalance target variable using SMOTE

SMOTE: Synthetic Minority Oversampling Technique

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

SMOTE first selects a minority class instance at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.

Python code for SMOTE Algorithm :-

```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))

print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))


# import SMOTE module from imblearn library

# pip install imblearn (if you don't have imblearn in your system)

from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state = 2)

x_train_smote, y_train_smote = sm.fit_resample(x_train, y_train.ravel())


print('After OverSampling, the shape of train_X: {}'.format(x_train.shape))

print('After OverSampling, the shape of train_y: {} \n'.format(y_train.shape))
```

```
print("After OverSampling, counts of label '1': {}".format(sum(y_train == 1)))

print("After OverSampling, counts of label '0': {}".format(sum(y_train == 0)))
```

Though this algorithm is quite useful, it has few drawbacks associated with it.

i) The synthetic instances generated are in the same direction i.e. connected by an artificial line to its diagonal instances. This in turn complicates the decision surface generated by few classifier algorithms.

ii) SMOTE tends to create a large no. of noisy data points in feature space.

among those is chosen. When max_features is set 1, this amounts to building a totally random decision tree

Avoiding Overfitting:

Brief Description of Overfitting?

This is an error in the modeling algorithm that takes into consideration random noise in the fitting process rather than the pattern itself. You can see that this occurs when the model gets an awesome score in the training set but when we use the test set (Unknown data for the model) we get an awful score. This is likely to happen because of overfitting of the data (taking into consideration random noise in our pattern). What we want our model to do is to take the overall pattern of the data in order to correctly classify whether a potential client will subscribe to a term deposit or not. In the examples above, it is most likely that the Decision Tree Classifier and Random Forest classifiers are overfitting since they both give us nearly perfect scores (100% and 99%) accuracy scores.

How can we avoid Overfitting?

The best alternative to avoid overfitting is to use cross validation. Taking the training test and splitting it. For instance, if we split it by 3, 2/3 of the data or 66% will be used for training and 1/3 33% will be used for testing and we will do the testing process three times. This algorithm will iterate through all the training and test sets and the main purpose of this is to grab the overall pattern of the data.

Hyperparameter Tuning:

We are using Gradient Boosting Classifiers as our final model.

The overall parameters of the Gradient Boosting Model can be divided into three categories:

1. ***Tree-Specific Parameters***: These affect each individual tree in the model.
2. ***Boosting Parameters***: These affect the boosting operation in the model.
3. ***Miscellaneous Parameters***: Other parameters for overall functioning.

The parameters used for tuning **tree specific features** are explained below:

1. ***min_samples_split***
    - Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
    - Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
    - Too high values can lead to under-fitting; hence, it should be tuned using CV.
2. ***min_samples_leaf***
    - Defines the minimum samples (or observations) required in a terminal node or leaf.
    - Used to control over-fitting similar to min_samples_split.
    - Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.
3. ***min_weight_fraction_leaf***
    - Similar to min_samples_leaf but defined as a fraction of the total number of observations instead of an integer.
    - Only one of #2 and #3 should be defined.
4. ***max_depth***
    - The maximum depth of a tree.
    - Used to control over-fitting as higher depth will allow models to learn relations very specific to a particular sample.
    - Should be tuned using CV.
5. ***max_leaf_nodes***
    - The maximum number of terminal nodes or leaves in a tree.
    - Can be defined in place of max_depth. Since binary trees are created, a depth of ‘n’ would produce a maximum of 2^n leaves.
    - If this is defined, the Gradient Boosting Model will ignore max_depth.
6. ***max_features***
    - The number of features to consider while searching for a best split. These will be randomly selected.
    - As a thumb-rule, square root of the total number of features works great but we should check upto 30-40% of the total number of features.
    - Higher values can lead to overfitting but it depends on case to case.

Now, let’s talk about the parameters for managing boosting.

1. ***learning_rate***
    - This determines the impact of each tree on the final outcome Gradient Boosting works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates.
    - Lower values are generally preferred as they make the model robust to the specific characteristics of the tree and thus allow it to generalize well.
    - Lower values would require a higher number of trees to model all the relations and will be computationally expensive.
2. ***n_estimators***
    - The number of sequential trees to be modeled
    - Although the Gradient Boosting Model is fairly robust at higher numbers of trees, it can still overfit at a point. Hence, this should be tuned using CV for a particular learning rate.
3. ***subsample***
    - The fraction of observations to be selected for each tree. Selection is done by random sampling.
    - Values slightly less than 1 make the model robust by reducing the variance.
    - Typical values ~0.8 generally work fine but can be fine-tuned further.

Further, there are miscellaneous parameters which help in increasing the performance of the model. They are as follows,

1.loss

- It refers to the loss function to be minimized in each split.
- It can have various values for classification and regression. Generally the default values work fine. Other values should be chosen only if you understand their impact on the model.

2. random_state

- The random number seed so that the same random numbers are generated every time.
- This is important for parameter tuning. If we don't fix the random number, then we'll have different outcomes for subsequent runs on the same parameters and it becomes difficult to compare models.
- It can potentially result in overfitting to a particular random sample selected. We can try running models for different random samples, which is computationally expensive and generally not used.

3.verbose

- The type of output to be printed when the model fits. The different values can be:
  - 0: no output generated (default)
  - 1: output generated for trees in certain intervals
  - >1: output generated for all trees

4.warm_start

- This parameter has an interesting application and can help a lot if used judicially.
- Using this, we can fit additional trees on previous fits of a model. It can save a lot of time and you should explore this option for advanced applications

5.presort

- Select whether to presort data for faster splits.
- It makes the selection automatically by default but it can be changed if needed.

Model Evaluation matrix:-

It is necessary to obtain the accuracy on training data, But it is also important to get a genuine and approximate result on unseen data, otherwise the Model is of no use.

So to build and deploy a generalized model we require to Evaluate the model on different metrics which helps us to better optimize the performance, fine-tune it, and obtain a better result.

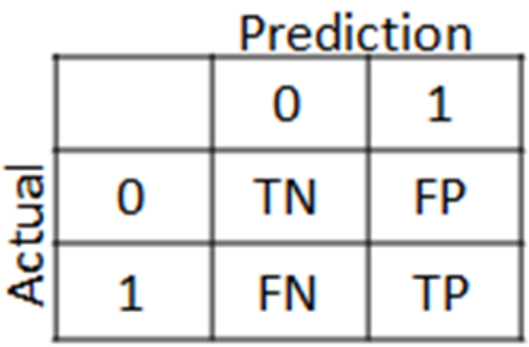
If one metric is perfect, there is no need for multiple metrics. To understand the benefits and disadvantages of Evaluation metrics because different evaluation metrics fit on a different set of a dataset.

Let's start Exploring various Evaluation metrics.

Confusion Matrix:-

Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.

A confusion matrix is defined as the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.
The main purpose of a confusion matrix is to see how our model is performing when it comes to classifying potential clients that are likely to subscribe to a term deposit. We will see in the confusion matrix four terms the True Positives, False Positives, True Negatives and False Negatives.



Positive/Negative: Type of Class (label) ["No", "Yes"] True/False: Correctly or Incorrectly classified by the model.

True Negatives (Top-Left Square): This is the number of correct classifications of the "No" class or potential clients that are not willing to subscribe to a term deposit.

False Negatives (Top-Right Square): This is the number of incorrect classifications of the "No" class or potential clients that are not willing to subscribe to a term deposit.

False Positives (Bottom-Left Square): This is the number of incorrect classifications of the "Yes" class or potential clients that are willing to subscribe to a term deposit.

True Positives (Bottom-Right Square): This is the number of correct classifications of the "Yes" class or potential clients that are willing to subscribe to a term deposit.

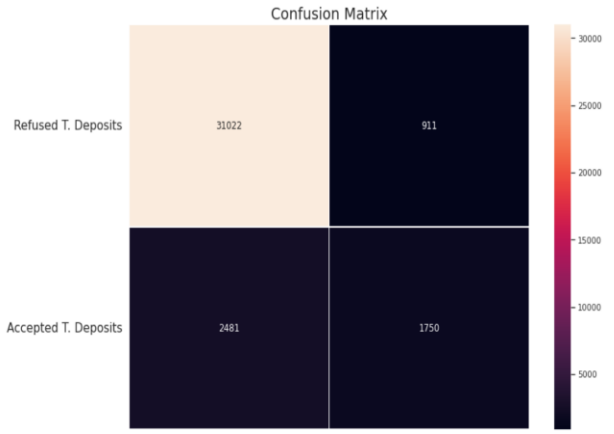


Fig:- Confusion matrix before smote on training dataset

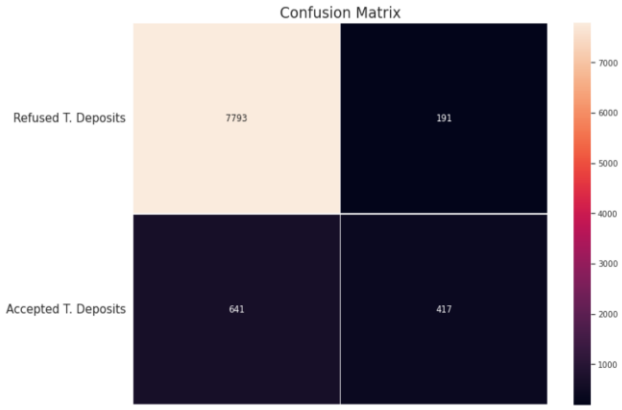


Fig:-Confusion matrix before smote on testing dataset

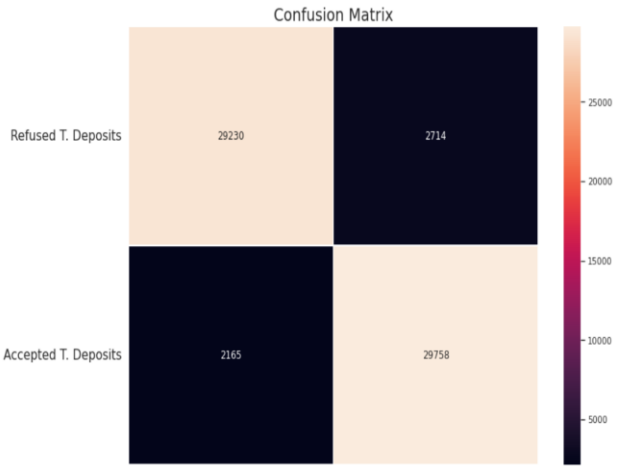


Fig:- Confusion matrix after SMOTE on training dataset

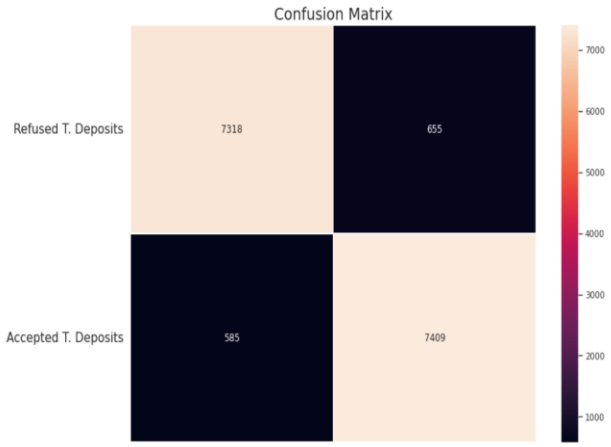


Fig:- Confusion matrix after SMOTE on testing dataset

Accuracy :-

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is useful when the target class is well balanced but is not a good choice for the unbalanced classes.

Precision :-

Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives. The importance of *Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn and this could be harmful to the business.*

Precision for a label is defined as the number of true positives divided by the number of predicted positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall (Sensitivity) :-

Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive. It is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!

Recall for a label is defined as the number of true positives divided by the total number of actual positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 Score:-

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

F1 Score is the harmonic mean of precision and recall.

$$F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

The F1 score punishes extreme values more. F1 Score could be an effective evaluation metric in the following cases:

- When FP and FN are equally costly.
- Adding more data doesn't effectively change the outcome
- True Negative is high

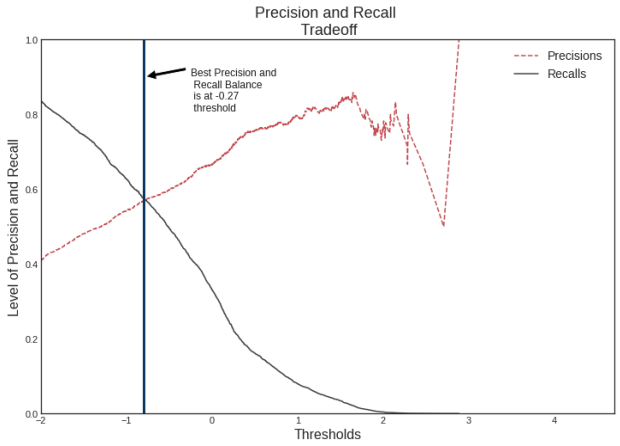


Fig:- Precision And Recall Tradeoff

AUC-ROC:- The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes.

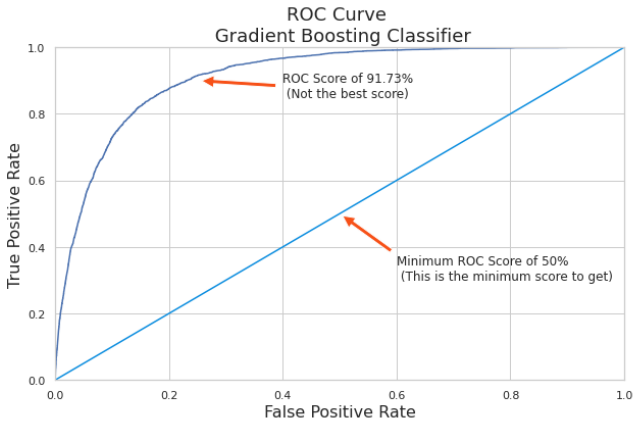


Fig :- ROC Curve (Gradient Boosting Classifier)

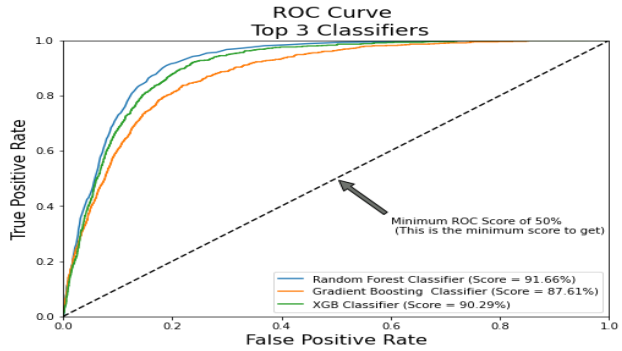


Fig :- ROC Curve (Top 3 Classifiers)

Conclusion :-

- **Months of Marketing Activity:** We saw that the month of highest level of marketing activity was the month of **May**. However, this was the month that potential clients tended to reject term deposits offers For the next marketing campaign, it will be wise for the bank to focus the marketing campaign during the months of **March, September, October and December.**
- **Campaign Calls**: A policy should be implemented that states that no more than 3 calls should be applied to the same potential client in order to save time and effort in getting new potential clients. Remember, the more we call the same potential client, the more likely he or she will decline to open a term deposit.
- **Age Category**: The next marketing campaign of the bank should target potential clients in their 20s or younger and 60s or older. The youngest category had a 60% chance of subscribing to a term deposit while the eldest category had a 76% chance of subscribing to a term deposit. It will be great if for the next campaign the bank addressed these two categories and therefore, increased the likelihood of more term deposits subscriptions.
- **Occupation:** Not surprisingly, potential clients that were students or retired were the most likely to subscribe to a term deposit. Retired individuals tend to have more term deposits in order to gain some cash through interest payments. Remember, term deposits are short-term loans in which the individual (in this case the retired person) agrees not to withdraw the cash from the bank until a certain date agreed between the individual and the financial institution.