# BANK MARKETING

*A report submitted in partial fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

**GANGU VARSHINI**

**Regd. No.: 20B91A0472**

**Under Supervision of Mr. Gundala Nagaraju**

**Henotic Technology Pvt Ltd, Hyderabad**

**(Duration: 7th July, 2022 to 6th September, 2022)**


**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**SAGI RAMA KRISHNAM RAJUENGINEERING COLLEGE**

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM,

ANDHRA PRADESH

# Table of Contents

# Abstract

**Key words**: Predicting Term Deposit Suscriptions,balance,loan,deposit, Duplicate values,unique values,missingvalues,variablecount,Targetvariables,DecisionTreeClassifier,RandomForest Classifier .

Find the best strategies to improve for the next marketing campaign.

How can the financial institution have a greater effectiveness for future marketing campaigns?

In order to answer this, we have to analyze the last marketing campaign the bank performed and identify the patterns that will help us find conclusions in order to develop future strategies.

This is the classic marketing bank dataset uploaded originally in the UCI Machine Learning Repository.

The dataset gives you information about a marketing campaign of a financial institution in which you will have to analyze in order to find ways to look for future strategies in order to improve future marketing campaigns for the bank.

Here in bank marketing data set , we are predicting the term deposit subscriptions.

# 1.0 Introduction

In this project we study different approachs to predict the sucess of bank telemarketing. As instrument we have a dataset related with direct marketing campaigns based on phone calls of a Portuguese banking institution. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (yes) or not (no) subscribed. The data under study here is called Bank Marketing Dataset (BMD) and he was found in the Machine Learning Repository (UCI). The data is public available in the is https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset. The size of the dataset is considerably large, especially if we consider its origin. Data from clients of financial institutions are usually difficult to find, and when found, are rarely available in this quantity. In the BMD data we have **11163** observations, with seventeen features. The eighteen features are briefly described in Table 1, were in the left column we have the original feature name in the dataset, and in the right column its description, mentioning also if the feature is numeric, categorial, and with how many levels (if categorical, of course). The first one called of y is the response, the desired target. The other features are presented in the same order that they appear in the dataset.

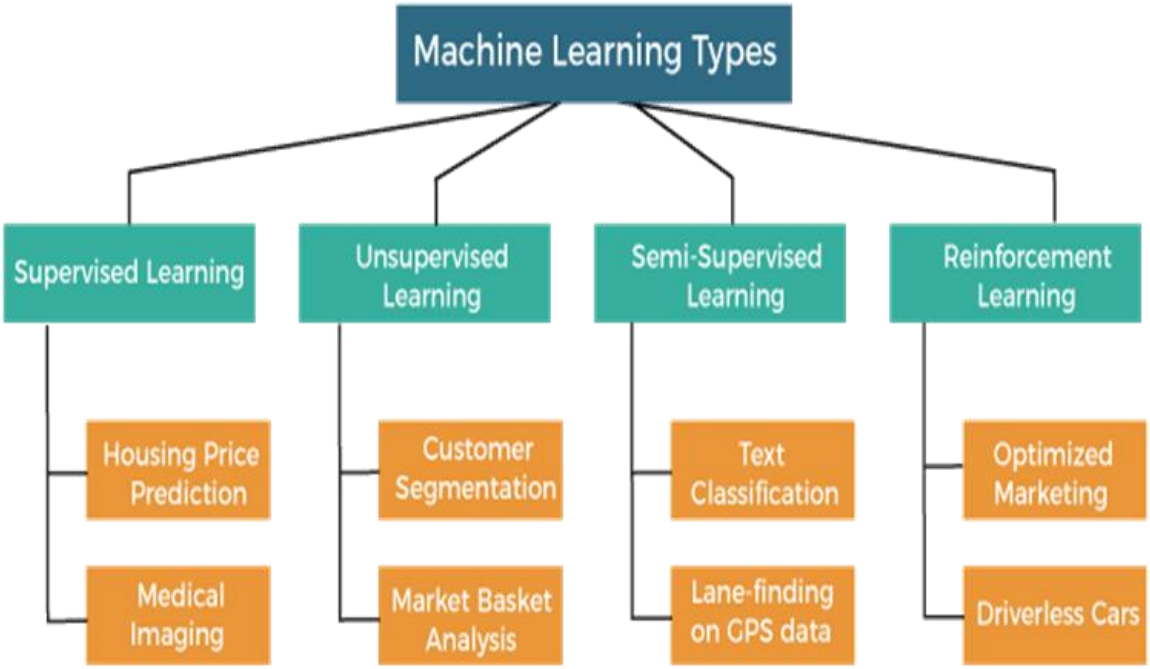
To predict, given the seventeen features, if the bank term deposit would be or not subscribed, ten algorithms are used. They are: four generalized linear models with a Bernoulli response and with different link functions (logit, probit, cauchit and complementary log-log); a standard linear regression model; naive Bayes classifier; three discriminant analysis algorithms (linear, quadratic and regularized); four support vector machines with different kernels (linear, polynomial, radial and sigmoid); a random forest; and a decision tree. As mentioned before, the BMD consists of **11163** observations. A random sample of 10% of this size, 4119 observations, was withdrawn to be used as a test dataset. The rest, 37069 observations, was used as a train dataset.

## 1.1. What are the different types of Machine Learning?

Machine Learning (ML) is a sub-category of artificial intelligence that refers to the process by which computers develop pattern recognition, or the ability to continuously learn from and make predictions based on data, then make adjustments without being specifically programmed to do so.

Machine learning algorithms have the ability to improve themselves through training. Today, ML algorithms are trained using three prominent methods. These are three types of machine learning**: supervised learning, unsupervised learning, and reinforcement learning.**

As with any method, there are different ways to train machine learning algorithms,each with their own advantges and disadvantages.To understand the pros and cons of each type of machine learning,we must first look at what kind of data they ingest. In ML,there are two kinds of data– labelled data and unlabelled data.




## SUPERVISED LEARNING:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem.

The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to

improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

## UNSUPERVISED LEARNING :

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

## REINFORCEMENT LEARNING:

Reinforcement learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness,

expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

## 1.2 Benefits of Using Machine Learning in Bank Marketing :

- According to a forecast by the research company Autonomous Next, banks around the world will be able to reduce costs by 22% by 2030 through using artificial intelligence technologies. Savings could reach $1 trillion.
- Financial companies employ 60% of all professionals who have the skills to create AI systems.
- It is expected that face recognition technology will be used in the banking sector to prevent credit card fraud. Face recognition technology will increase its annual revenue growth rate by over 20% in 2020.

Artificial Intelligence and Machine Learning are able to provide unprecedented levels of automation, either by taking over the tasks of human experts, or by enhancing their performance while assisting them with routine, repetitive tasks. But what are the main benefits of Machine Learning in Banking? This question has many possible answers, and what is even more interesting, the number of answers will continue to expand as the newest technological solutions hit the market. Here is an attempt to highlight the most important ones:

### Greater Automation and Improved Productivity :

Artificial Intelligence and Machine Learning can easily handle mundane tasks, allowing managers more time to work on more sophisticated challenges than repetitive paperwork. Automation across the entire organization will ultimately lead to greater profits.

### Personalized Customer Service

Automated solutions with Big Data capabilities can track and store as much information about the bank's customers as needed, providing the most precise and personalized customer experience. Optimizing the customer footprint allows banks to leverage analytical capabilities of Artificial Intelligence and Machine Learning to detect even the most subtle tendencies in customer behavior, which helps create a more personalized experience for each individual client.

### More precise Risk Assessment

Having an accurate digital footprint of each customer also can help banks reduce uncertainty for managers working with individual clients. The automated system is more accurate than a human in such areas as analysis of loan underwriting, eliminating any possible human bias.

### Advanced Fraud Detection and Prevention

This is probably the top benefit of AI/ML for any financial institution because there has historically been, and will continue to be, criminals who are devising methods to commit financial fraud. Fortunately, there are currently a wide range of proven methods and techniques of ML-powered Fraud Detection on the market. We will talk about all of them in greater detail in this article, and you will find out how to make your bank even more secure thanks to these technological innovations!

## 1.3 About Industry (Banking Sector)

Banking Industry Report

The automobile insurance industry has been going through massive transformations in the past couple of years. With more emphasis on customized insurance plans and the increasing level of market competition. As of 2018, the automobile insurance industry has reached the $200 billion mark. And this means that now there is no room for those auto insurance organizations that are not serious about their revered clientele. With a new competent organization budding every other week, the market competition is getting more cut-throat than ever.

Banking comes under **tertiary industries** These are concerned with providing support services to primary and secondary industries as well as activities relating to trade. These industries provide service facilities.

### 1.3.1 AI/ML Role in Bank Marketing

Artificial Intelligence (AI) has been around for a long time. AI was first conceptualized in 1955 as a branch of Computer Science and focused on the science of making “intelligent machines” machines that could mimic the cognitive abilities of the human mind, such as learning and problem-solving. AI is expected to have a disruptive effect on most industry sectors, many-fold compared to what the internet did over the last couple of decades. Organizations and governments around the world are diverting billions of dollars to fund research and pilot programs of applications of AI in solving real-world problems that current technology is not capable of addressing.

- **Customer service/engagement (Chatbot)**

Chatbots deliver a very high ROI in cost savings, making them one of the most commonly used applications of AI across industries. Chatbots can effectively tackle most commonly accessed tasks, such as balance inquiry, accessing mini statements, fund transfers, etc. This helps reduce the load from other channels such as contact centres, internet banking, etc.

- **Robo Advice**

Automated advice is one of the most controversial topics in the financial services space. A robo-advisor attempts to understand a customer's financial health by analyzing data shared by them, as well as their financial history. Based on this analysis and goals set by the client, the robo-advisor will be able to give appropriate investment recommendations in a particular product class, even as specific as a specific product or equity.

- **General Purpose / Predictive Analytics**

One of AI's most common use cases includes general-purpose semantic and natural language applications and broadly applied predictive analytics. AI can detect specific patterns and correlations in the data, which legacy technology could not previously detect. These patterns could indicate untapped sales opportunities, cross-sell opportunities, or even metrics around operational data, leading to a direct revenue impact.

- **Cybersecurity**

AI can significantly improve the effectiveness of cybersecurity systems by leveraging data from previous threats and learning the patterns and indicators that might seem unrelated to predict and prevent attacks. In addition to preventing external threats, AI can also monitor internal threats or breaches and suggest corrective actions, resulting in the prevention of data theft or abuse.

- **Credit Scoring / Direct Lending**

AI is instrumental in helping alternate lenders determine the creditworthiness of clients by analyzing data from a wide range of traditional and non-traditional data sources. This helps lenders develop innovative lending systems backed by a robust credit scoring model, even for those individuals or entities with limited credit history. Notable companies include Affirm and GiniMachine.

## 2.0 Bank Marketing Data :

Find the best strategies to improve for the next marketing campaign.

How can the financial institution have a greater effectiveness for future marketing campaigns?

In order to answer this, we have to analyze the last marketing campaign the bank performed and identify the patterns that will help us find conclusions in order to develop future strategies.

Input variables:

- bank client data:

1 - age (numeric)

2 - job : type of job (categorical: “admin.”,“unknown”,“unemployed”,“management”,“housemaid”,“entrepreneur”, “student”,“blue-collar”,“self-employed”,“retired”,“technician”,“services”)

3 - marital : marital status (categorical: “married”,“divorced”,“single”; note: “divorced” means divorced or widowed)

4 - education (categorical: “unknown”,“secondary”,“primary”,“tertiary”)

5 - default: has credit in default? (binary: “yes”,“no”)

6 - balance: average yearly balance, in euros (numeric)

7 - housing: has housing loan? (binary: “yes”,“no”)

8 - loan: has personal loan? (binary: “yes”,“no”)

- related with the last contact of the current campaign:

9 - contact: contact communication type (categorical: “unknown”,“telephone”,“cellular”)

10 - day: last contact day of the month (numeric)

11 - month: last contact month of year (categorical: “jan”, “feb”, “mar”, …, “nov”, “dec”)

12 - duration: last contact duration, in seconds (numeric)

- other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)

15 - previous: number of contacts performed before this campaign and for this client (numeric)

16 - poutcome: outcome of the previous marketing campaign (categorical: “unknown”,“other”,“failure”,“success”)

- output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: “yes”,“no”)

The main factors for Bank Marketing are age, job, marital, education, default, balance, housing, loan, contact, day, month, duration, campaign, pdays, previous, poutcome, deposit.

## 2.1 Internship Project - Data Link

The internship project data has taken from Kaggle and the link is https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset .

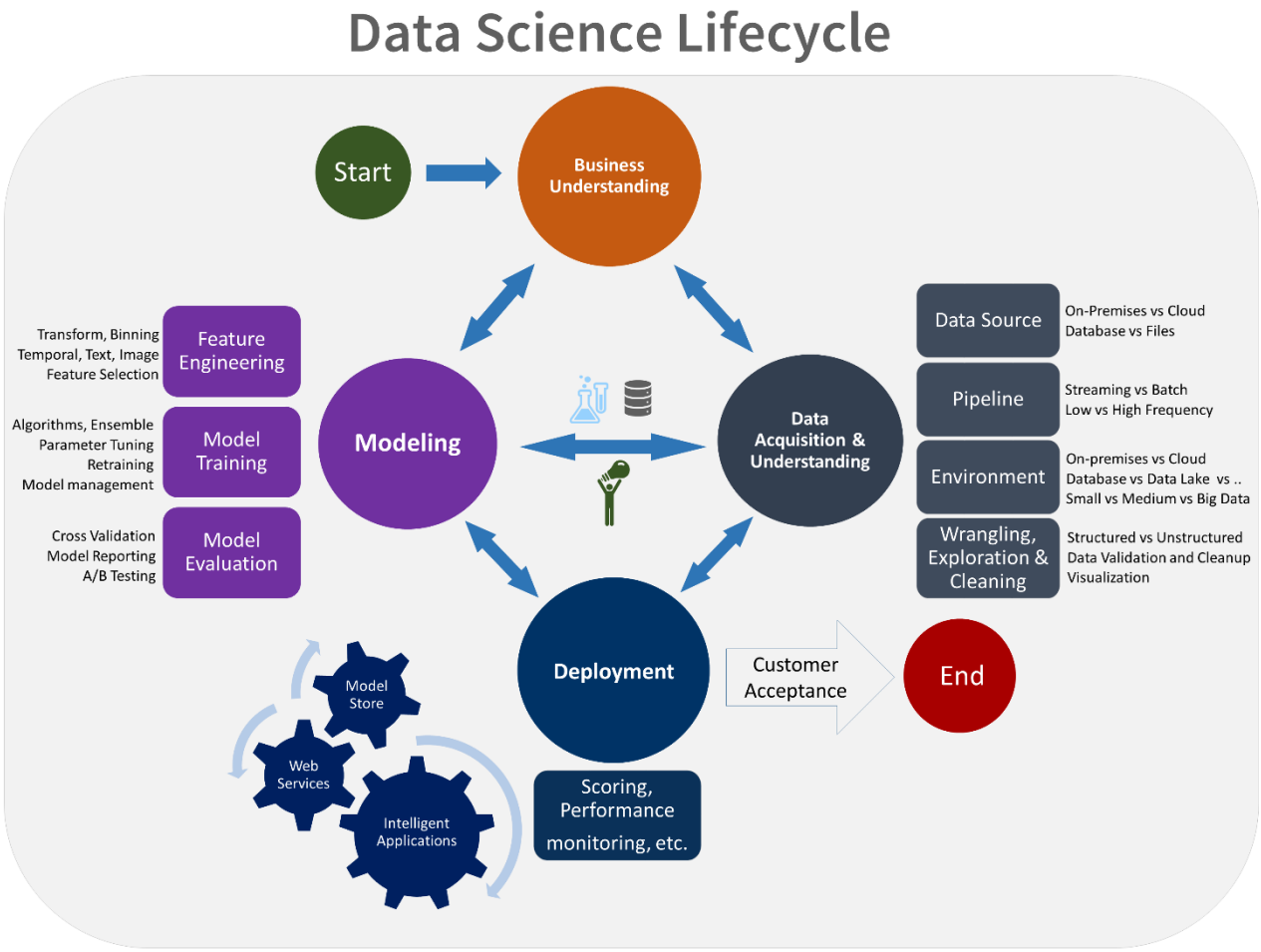
# 3 AI / ML Modelling and Results

## 3.1 Your Problem of Statement

**Problem Statement:** Propose a model to predict **Client Subscription Deposit in Bank Marketing** by using different AI/ML algorithms and suggest the best algorithm with evaluation metrics.

## 3.2 Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



### 3.2.1 Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.

The exploratory analysis is done for Auto Quote / Policy Conversionwith different parameters and all the charts are presented in **Appendices 6.2 - List of charts (6.2.1 to 6.2.9)**

### 3.2.2 Data Pre-processing

We removed variables which does not affect our target variable(Claimed_Target)as they mayadd noise and also increase our computation time,we checked the data for anomalous data points and outliers.We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

#### 3.2.2.1 Check the Duplicate and low variation data

Duplicate Values: When two features have the same set of values

Duplicate Index: When the value of two features are different, but they occur at the same index

Steps to delete duplicates:

1.Use the get duplicate features functions to get all the constant features.

2. Store all the duplicate features as a list for removing from the dataset.

3.Drop all such features from the dataset.

```
bank_dup = bank[bank.duplicated(keep='last')]
```

**OUTPUT:**

False

Since it doesn’t has any duplicate values no need for deleting them

#### 3.2.2.2 Identify and address the missing variables

How to Identify Missing Values?

We can check for null values in a derived dataset. But, sometimes, it might not be this simple to identify missing values. One needs to use the domain knowledge and look at the data description to understand the variables. There are variables that have a minimum value of zero. On some columns, a value of zero does not make sense and indicates an invalid or missing value.

Quick Classification of Missing Data

There are three types of missing data as below:

**Missing Completely At Random (MCAR):** It is the highest level of randomness. This means that the missing values in any features are not dependent on any other feature's values. This is the desirable scenario in case of missing data.

**Missing At Random (MAR):** This means that the missing values in any feature are dependent on the values of other features.

**Missing Not At Random (MNAR):** Missing not at random data is a more serious issue and, in this case, it might be wise to check the data gathering process further and try to understand why the information is missing.

## What to Do with the Missing Values?

We identified the missing values in a derived dataset, next we should decide the further Course of action.

- Ignore the missing values
- Drop the missing values
- Case Deletion
- Imputation
- **None**: None is a Python singleton object that is often used for missing data in Python code.
- **NaN**: Nan (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

```
bank.isnull().sum()
```

## OUTPUT:

```
age         0
```

job 0
marital 0
education 0
default 0
balance 0
housing 0
loan 0
contact 0
day 0
month 0
duration 0
campaign 0
pdays 0
previous 0
poutcome 0
deposit 0
dtype: int64

**To handle missing values:**

```
#SimpleImputer
from sklearn.impute import SimpleImputer
imputer_str = SimpleImputer(missing_values=np.nan,strategy='most_frequent',fill_value=None,verbose=0,copy=True,add_indicator=False)
train ['Magnitude Type'] = imputer_str.fit_transform(train [['Magnitude Type']])
train ['Root Mean Square'] = imputer_str.fit_transform(train [['Root Mean Square']])
train ['Azimuthal Gap'] = imputer_str.fit_transform(train [['Azimuthal Gap']])
```

### 3.2.2.3 Handling of Outliers

An outlier is a data point in a data set that is distant from all other observations. A data point that lies outside the overall distribution of dataset (95% of customer behaviour / claims / spending nature of customer)

Detecting outliers or anomalies is one of the core problems in data mining. The emerging expansion and continued growth of data and the spread of IoT devices, make us rethink the

way we approach anomalies and the use cases that can be built by looking at those anomalies.

We now have smart watches and wristbands that can detect our heartbeats every few minutes. Detecting anomalies in the heartbeat data can help in predicting heart diseases. Anomalies in traffic patterns can help in predicting accidents. It can also be used to identify bottlenecks in network infrastructure and traffic between servers. Hence, the use cases and solution built on top of detecting anomalies are limitless.

Another reason why we need to detect anomalies is that when preparing datasets for machine learning models, it is really important to detect all the outliers and either get rid of them or analyse them to know why you had them there in the first place.

## Method 1 - Standard Deviation:

In statistics, if a data distribution is approximately normal then about 68% of the data values lie within one standard deviation of the mean and about 95% are within two standard deviations, and about 99.7% lie within three standard deviations. Therefore, if you have any data point that is more than 3 times the standard deviation, then those points are very likely to be anomalous or outliers.

### Z score

Z score indicates how many standard deviation away a data point

Calculate the **Z score = (X - m)/Sigma**, where m = mean, Sigma = standard deviation

### Method 2 - Boxplots:

Box plots are a graphical depiction of numerical data through their quantiles. It is a very simple but effective way to visualize outliers. Think about the lower and upper whiskers as the boundaries of the data distribution. Any data points that show above or below the whiskers, can be considered outliers or anomalous.

```
# Outlier Analysis of TotalPrice, Frequency and Recency

import seaborn as sns

variables =['age', 'job', 'marital', 'education',

    'default', 'balance', 'housing', 'loan', 'contact',
```

```
    'day', 'month', 'duration', 'campaign',
    'pdays', 'previous', 'poutcome', 'deposit']
plt.rcParams['figure.figsize'] = [25,15]
sns.boxplot( data=bank[variables],orient="v", palette="Set2", whis=1.5, saturation=1, width=0.7)
plt.title("Outliers Variable Distribution", fontsize = 14, fontweight = 'bold')
plt.ylabel("Range", fontweight = 'bold')
plt.xlabel("Attributes", fontweight = 'bold')
```

#### 3.2.2.4 Categorical data and Encoding Techniques

A categorical variable is one that has two or more categories (values). There are two types of categorical variable, **nominal** and **ordinal**. A nominal variable has no intrinsic ordering to its categories. For example, gender is a categorical variable having two categories (male and female) with no intrinsic ordering to the categories. An ordinal variable has a clear ordering.Many ML algorithms are unable to operate on categorical or label data directly. However, Decision tree can directly learn from such data. Hence, they require all input variables and output variables to be numeric. This means that categorical data must be converted to a numerical form. Few types of categorical variable encoding are:

**1.One hot encoding**: Encoding each categorical variable with different Boolean variables (also called dummy variables) which take values 0 or 1, indicating if a category is present in an observation.

**Integer Encoding / Label Encoding**: Replace the categories by a number from 1 to n (or 0 to n-1, depending the implementation), where n is the number of distinct categories of the variable.

3. **Count or frequency encoding**: Replace the categories by the count of the observations that show that category in the dataset. Similarly, we can replace the category by the frequency -or percentage- of observations in the dataset. That is, if 10 of our 100

observations show the colour blue, we would replace blue by 10 if doing count encoding, or by 0.1 if replacing by the frequency.

4. **Ordered Integer Encoding**: Categories are replaced by integer 1 to k, where k is the distinct categories in variable, but this numbering is decided by mean of target of each category.

bank.info()

**OUTPUT:**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in bank.columns:
    if bank[col].dtype=='object':
        bank[col]=le.fit_transform(bank[col])
```

### 3.2.2.5 Feature Scaling

Feature Scaling is done on the dataset to bring all the different types of data to a ***Single Format***. Done on **Independent Variable**.Types of Feature Scaling: Min Max Scaler, Standard Scaler.

Min Max Scalar:

- It scales and transforms the data in-between 0 and 1.
- ANN performs well when do scale the data using Min-max-Scalar.

Standard Scalar:

- It scales and transform the data with respect to *Mean = 0* and *Standard Deviation = 1***.**

```
# Scaling the features by using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
mmscaler = MinMaxScaler(feature_range=(0, 1))
x_train = mmscaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train)
x_test = mmscaler.fit_transform(x_test)
x_test = pd.DataFrame(x_test)
```

## 3.2.3 Selection of Dependent and Independent variables

The dependent or target variable here isClaimed Target which tells us a particular policy holder has filed a claim or not the target variable is selected based on our business problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

### 3.2.4 Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods

#### 3.2.4.1 Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.It can be performed using strata function from the library sampling.

#### 3.2.4.2 Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

### 3.2.5 Models Used for Development

We built our predictive models by using the following ten algorithms

#### 3.2.5.1 Model 01

Logistic uses logit link function to convert the likelihood values to probabilities so we can get a good estimate on the probability of a particular observation to be positive class or negative class.The also gives us p-value of the variables which tells us about significance of each independent variable.

#### 3.2.5.2 Model 02

Random forest is an algorithm that consists of many decision trees. It was first developedby Leo Breiman and Adele Cutler. The idea behind it is to build several trees,to have the

instance classified by each tree, and to give a "vote" at each class.The model uses a "bagging" approach and the random selection of features to build acollection of decision trees with controlled variance. The instance's class is to the classwith the highest number of votes, the class that occurs the most within the leaf in whichthe instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.
- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

### 3.2.5.3 Model 03(Decision tree)

**Decision Tree** is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

DecisionTreeRepresentation**:**
Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is the repeated for the subtree rooted at the new node.
The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf.(in this case Yes or No).

### 3.2.5.4 Model 04(ExtraTreesClassifier)

**Extremely Randomized Trees Classifier (Extra Trees Classifier)** is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output its classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

### 3.2.5.5 Model 05(KNeighborsClassifier)

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

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 suite category by using K- NN algorithm. 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. 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.

### 3.2.5.6 Model 06(GaussianNB)

Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naïve Bayes is the extension of naïve Bayes. While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data.

### 3.2.5.7 Model 07(SVC)

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

### 3.2.5.8 Model 08(XGB Classifier)

XGB is an implementation of Gradient Boosted decision trees. This library was written in C++. It is a type of Software library that was designed basically to improve speed and model performance. It

has recently been dominating in applied machine learning. XGB models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGB. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and the variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

#### 3.2.5.9 Model 09(LGBM Classifier)

LGBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage.

It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of LGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks Gradient-based One Side Sampling Technique for LGBM: Different data instances have varied roles in the computation of information gain. The instances with larger gradients (i.e., under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g., larger than a predefined threshold, or among the top percentiles), and only randomly drop those instances with small gradients to retain the accuracy of information gain estimation. This treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range.

#### 3.2.5.10 Model 10(Bagging Classifier)

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can

typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each

other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

#### 3.2.5.11 Model 11(GradientBoostingClassifier)

**Gradient boosting classifier** is a set of machine learning algorithms that include several weaker models to combine them into a strong big one with highly predictive output. Models of a kind are popular due to their ability to classify datasets effectively.

Using **gradient boost for classification** we discover the initial prediction for every patient in the **log (odds)**.

## 3.3 AI/ML Model Analysis And Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto dataset of size ~ **11163** deposits.

### 3.3.1 Different Model codes

- The Python code for models with stratified sampling technique as follows:

```
# Build the Classification models and compare the results
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
```

```
from sklearn.ensemble import GradientBoostingClassifier
import lightgbm as lgb
# Create objects of classification algorithm with default hyper-parameters
ModelLR = LogisticRegression()
ModelDC = DecisionTreeClassifier()
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=5)
ModelGNB = GaussianNB()
ModelXGB = XGBClassifier(n_estimators=1, max_depth=3,eval_metric='mlogloss')
ModelSVM = SVC(probability=True)
ModelBAG = BaggingClassifier(base_estimator=None, n_estimators=100, max_samples=1.0, max_features=1.0,
                             bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False,
                             n_jobs=None, random_state=None, verbose=0)
ModelGB = GradientBoostingClassifier()
ModelLGB = lgb.LGBMClassifier()
# Evalution matrix for all the algorithms
#MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN, ModelGNB, ModelSVM, ModelXGB, ModelLGB,ModelBAG,ModelGB,ModelLGB]
MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN, ModelGNB, ModelSVM, ModelXGB, ModelLGB,ModelBAG,ModelGB]
for models in MM:
    # Fit the model
    models.fit(x_train, y_train)
```

```
    # Prediction
    y_pred = models.predict(x_test)
    y_pred_prob = models.predict_proba(x_test)
    # Print the model name
    print('Model Name: ', models)
    # confusion matrix in sklearn
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    # actual values
    actual = y_test
    # predicted values
    predicted = y_pred
    # confusion matrix
    matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None,
normalize=None)
    print('Confusion matrix : \n', matrix)
    # outcome values order in sklearn
    tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
    print('Outcome values : \n', tp, fn, fp, tn)
    # classification report for precision, recall f1-score and accuracy
    C_Report = classification_report(actual,predicted,labels=[1,0])
    print('Classification report : \n', C_Report)
    # calculating the metrics
    sensitivity = round(tp/(tp+fn), 3);
    specificity = round(tn/(tn+fp), 3);
    accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
```

```
    balanced_accuracy = round((sensitivity+specificity)/2, 3);
    precision = round(tp/(tp+fp), 3);
    f1Score = round((2*tp/(2*tp + fp + fn)), 3);
    # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between
-1 to +1.
    # A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
    mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
    MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
    print('Accuracy :', round(accuracy*100, 2),'%')
    print('Precision :', round(precision*100, 2),'%')
    print('Recall :', round(sensitivity*100,2), '%')
    print('F1 Score :', f1Score)
    print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
    print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
    print('MCC :', MCC)
    # Area under ROC curve
    from sklearn.metrics import roc_curve, roc_auc_score
    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
    # ROC Curve
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import roc_curve
    logit_roc_auc = roc_auc_score(y_test, y_pred)
    fpr, tpr, thresholds = roc_curve(y_test, models.predict_proba(x_test)[:,1])
    plt.figure()
    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
```

```
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
    print('----------------------------------------------------------------------------------------------------')
    #----------------------------------------------------------------------------------------------------
    new_row = {'Model Name' : models,
               'True_Positive' : tp,
               'False_Negative' : fn,
               'False_Positive' : fp,
               'True_Negative' : tn,
               'Accuracy' : accuracy,
               'Precision' : precision,
               'Recall' : sensitivity,
               'F1 Score' : f1Score,
               'Specificity' : specificity,
               'MCC':MCC,
               'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
```

'Balanced Accuracy':balanced_accuracy}

CSResults = CSResults.append(new_row, ignore_index=True)

### 3.3.2 LightGBM Python code

- The Python code for models with stratified sampling technique as follows:

```
# Training the lightgbm model on the Training set
import lightgbm as lgb
# Build the model
modelLGB = lgb.LGBMClassifier()
# Fit the model with train data
modelLGB.fit(x_train,y_train)
# Predict the model with test data set
y_pred = modelLGB.predict(x_test)
y_pred_prob = modelLGB.predict_proba(x_test)
# Confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
```

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
```

```
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test,modelLGB.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print('----------------------------------------------------------------------------------------------------
-
```

### 3.3.3 Decision Tree Python Code

- The Python code for models with stratified sampling technique as follows:

```
# To build the 'Decision Tree' model with random sampling
from sklearn.tree import DecisionTreeClassifier
```

```
ModelDT = DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=None,
                                 max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,min_weight_fraction_leaf=0.0,
                                 random_state=None, splitter='best')
# Train the model with train data
ModelDT.fit(x_train,y_train)
# Predict the model with test data set
y_pred = ModelDT.predict(x_test)
y_pred_prob = ModelDT.predict_proba(x_test)
# Confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
```

```
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
```

```
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test,ModelDT.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print('--------------------------------------------------------------------------------------------
-------')
#--------------------------------------------------------------------------------------------------
------
new_row = {'Model Name' : ModelDT,
        'True_Positive': tp,
        'False_Negative': fn,
        'False_Positive': fp,
        'True_Negative': tn,
        'Accuracy' : accuracy,
        'Precision' : precision,
```

'Recall' : sensitivity,

'F1 Score' : f1Score,

'Specificity' : specificity,

'MCC':MCC,

'ROC_AUC_Score':roc_auc_score(y_test, y_pred),

'Balanced Accuracy':balanced_accuracy}

HTResults = HTResults.append(new_row, ignore_index=True)

#----------

### 3.3.4 Extra Trees Python code

- The Python code for models with stratified sampling technique as follows:

  # To build the 'ExtraTreesClassifier' model with random sampling along with default hyper parameters values

  from sklearn.ensemble import ExtraTreesClassifier

  ModelET=ExtraTreesClassifier(n_estimators=100,criterion='gini', max_depth=None, min_samples_split=2,

  min_samples_leaf=1,min_weight_fraction_leaf=0.0, max_features='sqrt',

  max_leaf_nodes=None,min_impurity_decrease=0.0, bootstrap=False, oob_score=False,

  n_jobs=None,random_state=None,verbose=0, warm_start=False, class_weight=None,

  ccp_alpha=0.0, max_samples=None)

  # Train the model with train data

  ModelET.fit(x_train,y_train)

  # Predict the model with test data set

  y_pred = ModelET.predict(x_test)

```
y_pred_prob = ModelET.predict_proba(x_test)
# Confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
```

```
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test,ModelET.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print('-------------------------------------------------------------------------------------------------------')
#-------------------------------------------------------------------------------------------------------
new_row = {'Model Name' : ModelET,
        'True_Positive': tp,
        'False_Negative': fn,
        'False_Positive': fp,
        'True_Negative': tn,
        'Accuracy' : accuracy,
        'Precision' : precision,
        'Recall' : sensitivity,
        'F1 Score' : f1Score,
        'Specificity' : specificity,
        'MCC':MCC,
        'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
        'Balanced Accuracy':balanced_accuracy}
HTResults = HTResults.append(new_row, ignore_index=True)
```

#----------

------:

**Stratified Sampling**: LGBMClassifier model performance is good, by considering the confused matrix, highest accuracy (0.837) &good F1 score (0.847).This is because random forest uses bootstrap aggregation which can reduce bias and variance in the data and can leadsto good predictions withclaims dataset.

**Simple Random Sampling**: Artificial Neural Networks / LGBMClassifier are out performed Logistic Regression model, by considering the confused matrix, highest accuracy (0.837) &good F1 score (0.847).This is because Artificial Neural Networks havehidden and complex patterns between different variables and can leads to good predictions with claims dataset.

## 4.0 Coclusions And Future Work

The model results in the following order by considering the model accuracy, F1 score and RoC AUC

| | Model Name | True Positive | False Negative | False Positive | True Negative | Accuracy | Precision | Recall | F1 Score | Specificity | MCC | ROC_AUC_Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LogisticRegression() | NaN | NaN | NaN | NaN | 0.781 | 0.767 | 0.779 | 0.773 | 0.782 | 0.561 | 0.780476 |
| 1 | DecisionTreeClassifier() | NaN | NaN | NaN | NaN | 0.747 | 0.717 | 0.782 | 0.748 | 0.715 | 0.498 | 0.748736 |
| 2 | (DecisionTreeClassifier(max_features='sqrt', r... | NaN | NaN | NaN | NaN | 0.826 | 0.764 | 0.921 | 0.835 | 0.738 | 0.666 | 0.829314 |
| 3 | (ExtraTreeClassifier(random_state=988455154), ... | NaN | NaN | NaN | NaN | 0.833 | 0.782 | 0.904 | 0.839 | 0.768 | 0.675 | 0.835815 |
| 4 | KNeighborsClassifier() | NaN | NaN | NaN | NaN | 0.722 | 0.748 | 0.634 | 0.686 | 0.803 | 0.444 | 0.718313 |
| 5 | SVC(probability=True) | NaN | NaN | NaN | NaN | 0.804 | 0.779 | 0.828 | 0.802 | 0.783 | 0.61 | 0.805319 |
| 6 | (DecisionTreeClassifier(random_state=916821151... | NaN | NaN | NaN | NaN | 0.822 | 0.76 | 0.918 | 0.832 | 0.733 | 0.659 | 0.825462 |
| 7 | ([DecisionTreeRegressor(criterion='friedman_ms... | NaN | NaN | NaN | NaN | 0.825 | 0.769 | 0.907 | 0.832 | 0.749 | 0.661 | 0.827923 |
| 8 | LGBMClassifier() | NaN | NaN | NaN | NaN | 0.837 | 0.773 | 0.937 | 0.847 | 0.746 | 0.691 | 0.841111 |
| 9 | GaussianNB() | NaN | NaN | NaN | NaN | 0.755 | 0.71 | 0.83 | 0.765 | 0.687 | 0.52 | 0.758343 |

1) **LGBMClassifier** with Stratified and Random Sampling
2) **Decision Tree** with Simple Random Sampling
3) **Extra Trees** with Simple Random Sampling

We recommend model – **LBGMClassifier** with Stratified and Random Sampling technique as a best fit for the give n BI claims dataset. We considered Random Forest .

# 5.0References

https://www.kaggle.com/datasets

https://www.businessinsider.in/finance/news/the-impact-of-artificial-intelligence-in-the-banking-sector-how-ai-is-being-used-in-2020/articleshow/72860899.cms

https://ibsintelligence.com/ibsi-news/5-applications-of-artificial-intelligence-in-banking/

# 6.0Appendices

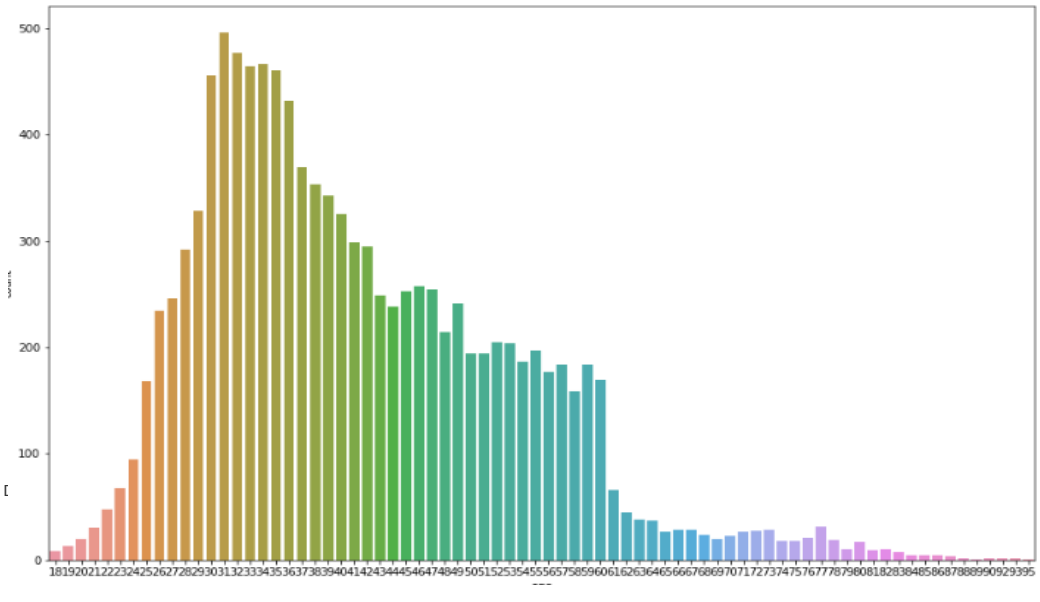
## 6.1 Python code Results

| | Model Name | True Positive | False Negative | False Positive | True Negative | Accuracy | Precision | Recall | F1 Score | Specificity | MCC | ROC_AUC_Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LogisticRegression() | NaN | NaN | NaN | NaN | 0.781 | 0.767 | 0.779 | 0.773 | 0.782 | 0.561 | 0.780476 |
| 1 | DecisionTreeClassifier() | NaN | NaN | NaN | NaN | 0.747 | 0.717 | 0.782 | 0.748 | 0.715 | 0.498 | 0.748736 |
| 2 | (DecisionTreeClassifier(max_features='sqrt', r... | NaN | NaN | NaN | NaN | 0.826 | 0.764 | 0.921 | 0.835 | 0.738 | 0.666 | 0.829314 |
| 3 | (ExtraTreeClassifier(random_state=988455154), ... | NaN | NaN | NaN | NaN | 0.833 | 0.782 | 0.904 | 0.839 | 0.768 | 0.675 | 0.835815 |
| 4 | KNeighborsClassifier() | NaN | NaN | NaN | NaN | 0.722 | 0.748 | 0.634 | 0.686 | 0.803 | 0.444 | 0.718313 |
| 5 | SVC(probability=True) | NaN | NaN | NaN | NaN | 0.804 | 0.779 | 0.828 | 0.802 | 0.783 | 0.61 | 0.805319 |
| 6 | (DecisionTreeClassifier(random_state=916821151... | NaN | NaN | NaN | NaN | 0.822 | 0.76 | 0.918 | 0.832 | 0.733 | 0.659 | 0.825462 |
| 7 | ([DecisionTreeRegressor(criterion='friedman_ms... | NaN | NaN | NaN | NaN | 0.825 | 0.769 | 0.907 | 0.832 | 0.749 | 0.661 | 0.827923 |
| 8 | LGBMClassifier() | NaN | NaN | NaN | NaN | 0.837 | 0.773 | 0.937 | 0.847 | 0.746 | 0.691 | 0.841111 |
| 9 | GaussianNB() | NaN | NaN | NaN | NaN | 0.755 | 0.71 | 0.83 | 0.765 | 0.687 | 0.52 | 0.758343 |

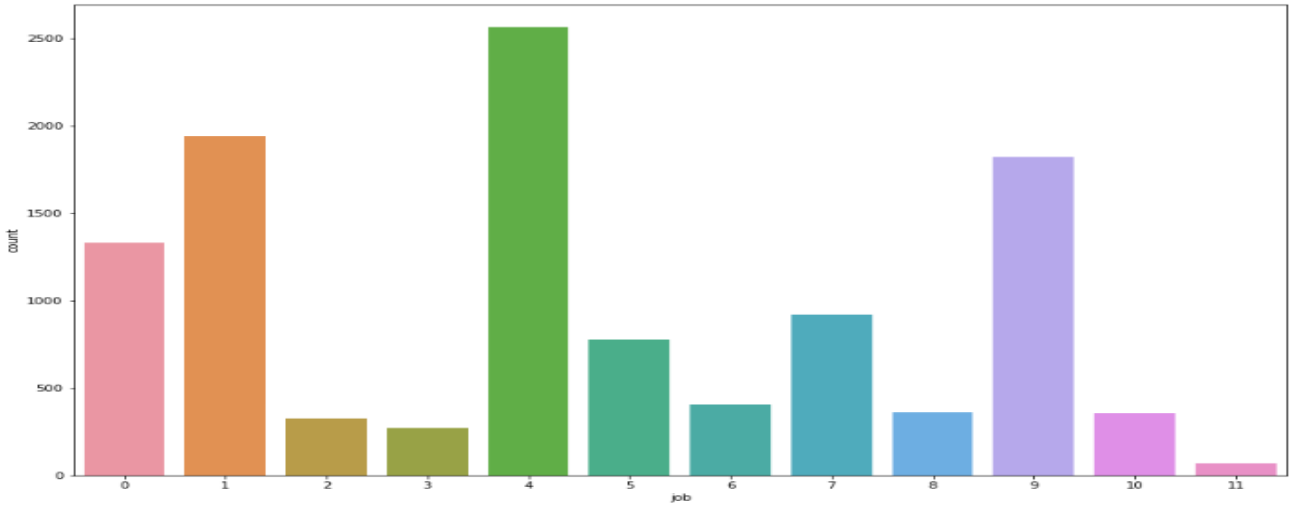
| Accuracy | Precision | Recall | F1 Score | Specificity | MCC | ROC_AUC_Score | Balanced Accuracy | False_Negative | False_Positive | True_Negative | True_Positive |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.781 | 0.767 | 0.779 | 0.773 | 0.782 | 0.561 | 0.780476 | 0.78 | 355.0 | 380.0 | 1362.0 | 1252.0 |
| 0.747 | 0.717 | 0.782 | 0.748 | 0.715 | 0.498 | 0.748736 | 0.748 | 350.0 | 496.0 | 1246.0 | 1257.0 |
| 0.826 | 0.764 | 0.921 | 0.835 | 0.738 | 0.666 | 0.829314 | 0.83 | 127.0 | 457.0 | 1285.0 | 1480.0 |
| 0.833 | 0.782 | 0.904 | 0.839 | 0.768 | 0.675 | 0.835815 | 0.836 | 155.0 | 404.0 | 1338.0 | 1452.0 |
| 0.722 | 0.748 | 0.634 | 0.686 | 0.803 | 0.444 | 0.718313 | 0.718 | 588.0 | 344.0 | 1398.0 | 1019.0 |
| 0.804 | 0.779 | 0.828 | 0.802 | 0.783 | 0.61 | 0.805319 | 0.806 | 277.0 | 378.0 | 1364.0 | 1330.0 |
| 0.822 | 0.76 | 0.918 | 0.832 | 0.733 | 0.659 | 0.825462 | 0.826 | 132.0 | 465.0 | 1277.0 | 1475.0 |
| 0.825 | 0.769 | 0.907 | 0.832 | 0.749 | 0.661 | 0.827923 | 0.828 | 149.0 | 438.0 | 1304.0 | 1458.0 |
| 0.837 | 0.773 | 0.937 | 0.847 | 0.746 | 0.691 | 0.841111 | 0.842 | 102.0 | 443.0 | 1299.0 | 1505.0 |
| 0.755 | 0.71 | 0.83 | 0.765 | 0.687 | 0.52 | 0.758343 | 0.758 | 273.0 | 546.0 | 1196.0 | 1334.0 |

## 6.2 List of Charts

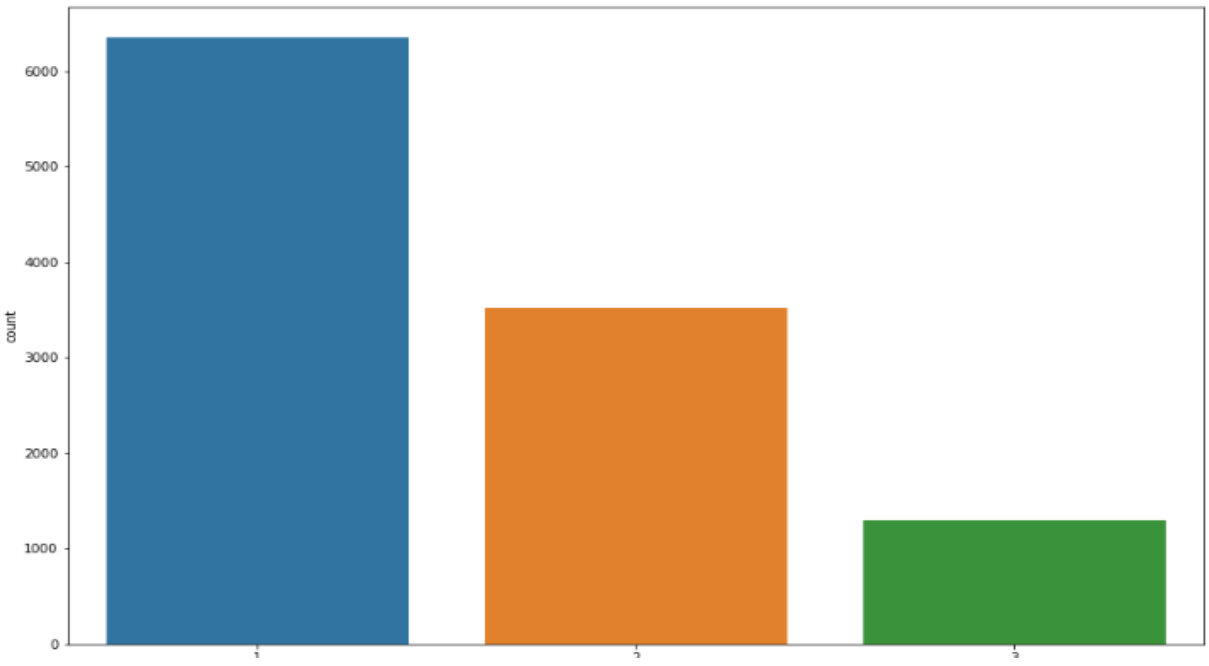
### 6.2.1 Chart 01: Count of Ages



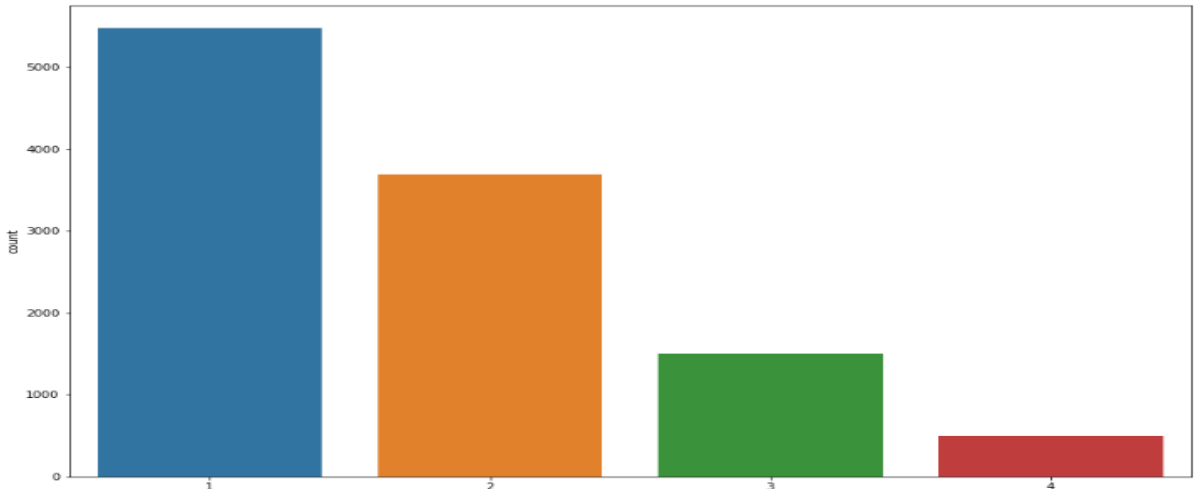
### 6.2.2 Chart 02: Count of job




### 6.2.3 Chart 03: Count of Marital status



### 6.2.4 Chart 04: Count of Education




### 6.2.5 Count of Job Pie chart