"PREDICTION OF TERM DEPOSIT SUBSCRIPTION"

A

MAJOR PROJECT REPORT

Submitted for the partial fulfilment of the requirement for the award of Degree

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

MEGHA SAHU (0101CS151065)

NEERAJ CHOUHAN (0101CS151072)

RANU BHARADWAJ (0101CS151091)

ARTI GURJAR (0101IT151017)

Guided By:

Prof. Raju Baraskar

Prof. Anjana Deen

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDALAYA

BHOPAL-462033

SESSION 2018-2019

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Megha Sahu, Neeraj Chouhan, Ranu Bharadwaj, Arti Gurjar** of B.E. Fourth Year, Computer Science & Engineering have completed their major Project titled **"Prediction Of Term Deposit Subscription"** during the year 2018-19 under our guidance and supervision.

We approve the project for the submission for the partial fulfillment of the requirement for the award of degree of B.E. in Computer Science & Engineering.

Prof. Raju Baraskar
Assistant Professor
(Project Guide)
Department of Computer
Science & Engineering

Prof. Anjana Deen
Assistant Professor
(Project Guide)
Department of Computer
Science & Engineering

Dr. R.S. Rajput
Director
UIT RGPV Bhopal

Dr. Piyush Shukla
HOD DoCSE
UIT RGPV Bhopal

DECLARATION BY CANDIDATE

We, hereby declare that the work which is presented in the major project, entitled **"Prediction Of Term Deposit Subscription"** submitted in partial fulfilment of the requirement for the award of Bachelor degree in Computer Science and Engineering has been carried out at University Institute of Technology RGPV , Bhopal and is an authentic record of our work carried out under the guidance of **Prof. Raju Baraskar** (Project Guide) and **Prof. Anjana Deen** (Project Guide), Department of Computer Science and Engineering, UIT RGPV, Bhopal.

The matter in this project has not been submitted by us for the award of any other degree.

MEGHA SAHU (0101CS151065)

NEERAJ CHOUHAN (0101CS151072)

RANU BHARADWAJ (0101CS151091)

ARTI GURJAR (0101IT151017)

ACKNOWLEDGEMENT

After the completion of major project work, words are not enough to express our gratitude to everyone helped us reach our goal, and above all we thank almighty for helping us reach this moment in life.

First and foremost, we take this opportunity to express our deepest regards and heartfelt gratitude to our project guide **Prof. Raju Baraskar** and **Prof. Anjana Deen** of Computer Science and Engineering Department, RGPV Bhopal for their inspiring guidance and timely suggestions in carrying out our project successfully. They have also been a constant source of inspiration for us.

We are grateful for our source of inspiration **Dr. Piyush Shukla,** Prof. And Head of DoCSE, UIT-RGPV, Bhopal for his unforgettable support and inspiration towards the project. We would also like to thank all the teachers of our department for providing invaluable support and motivation. We are also grateful to our teammates, friends and colleagues for their help and cooperation throughout this work.

MEGHA SAHU (0101CS151065)

NEERAJ CHOUHAN (0101CS151072)

RANU BHARADWAJ (0101CS151091)

ARTI GURJAR (0101IT151017)

ABSTRACT

This document sequentially applies a set of Data Science techniques to gain insights from the Direct Marketing campaign of a Banking Institution. Data Science analysis of this data will benefit the business processes of the Banking and Financial Management Industry. The Bank that initiated the telemarketing campaign, contacted potential savings account depositors. It covered information about clients, the bank's products, social factors and economics. Thereby, allowing for prediction of Term Deposit subscriptions by bank customers. Competition between banking institutions have moved from a local focus defined as the area of one city or one county, to a national focus defined as the area of one country.

Via the internet, banking service providers have expanded the range of services they have traditionally offered to customers. The expanded services now exist has separate business areas that provide deposit, loan, mortgage, credit, transaction card, vehicle loan, and business loan services. High risk short term loans, and investment brokerages, have become available with the same convenience of all other types of banking services. Thereby, a customer-based focus of analysis of banking services via Data Science, allows for understanding of the possible effects of the concentration of a wide variety of banking resources into a small group of national enterprises. Divergent demographic and economic characteristics of consumers are now examinable independent of geographic area, in order to determine the likelihood of procurement of financial services.

TABLE OF CONTENTS

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 DATA ANALYSIS WITH MACHINE LEARNING

Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning uses category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

Machine started leaning own there on from past experience and without any explicit programming is called machine learning. Machine learning is subset of Artificial intelligence. Machine learning is the process of developing, testing, and applying predictive algorithms to achieve this goal.

Example: tesla automated car, chat bot.

1.2 TYPES OF MACHINE LEARNING

1.2.1 **Supervised machine learning**:

- It consist of a target/outcome variable which is to be predicted from a given set of predictors.
- Using these set of variables, we generate a function that map inputs to desired outputs.
- The training process continue until the model achiever a desired level of accuracy on the training data.
- We give enough input to the algorithm and supervisor keep correcting result until it became to make decision by their own.
- Input have expected output associated with them.
- When algorithm is trained it will predict correct output of never seen input.

Application: hi cortana, ok google , siri etc.

Algorithm:

- Linear regression
- Random Forest
- Support Vector Machine

1.2.2 **Unsupervised machine learning**:

- In this we do not have any target or outcome variable to predict/estimate.
- It is used for clustering population in different groups, which is widely used for segmenting customers in different groups of specific intervention.
- Input don't have expected output associated with them instead it detect pattern of initial characteristic of input data.
- No previous knowledge
- Example clustering: similar data group together, if we are watching football match for first time then we make assumption on our own.

Algorithm:

- K-Mean algorithm
- Apriori algorithm
- Hierarchical clustering

1.2.3 **Reinforcement machine learning**:

- Using this, the machine is trained to make specific decisions.
- The machine is exposed to an environment where it trains itself continually using trial and error.
- This machine learn from past experience and tries to capture the best possible knowledge to make accurate business decisions.
- Automatically determine the ideal behavior
- Interaction between two elements environment and learning agent
- We give reward and penalty for the decision so that accuracy get increased.

Example: Markov decision process.

CHAPTER 2

LITERATURE SURVEY

2.1 MACHINE LEARNING ALGORITHMS

2.1.1 The Random Forests Algorithm

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

Advantages:

•Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.

•It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

•The algorithm can be used in both classification and regression problems.

•Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.

•You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages:

•Random forests is slow in generating predictions because it has multiple decision trees.

•The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Finding important features :

Random forests also offer a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

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

decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

Random Forests vs Decision Trees

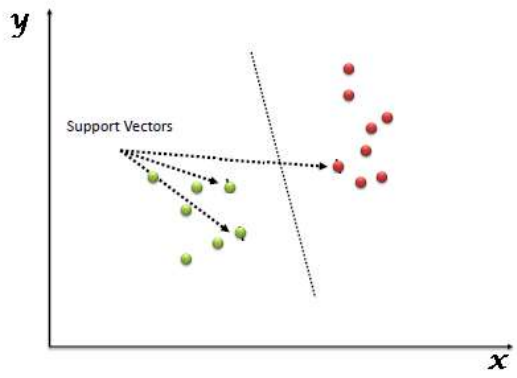
- •Random forests is a set of multiple decision trees.
- •Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- •Decision trees are computationally faster.
- •Random forests is difficult to interpret, while a decision tree is easily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

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

2.1.2 Support Vector Machine

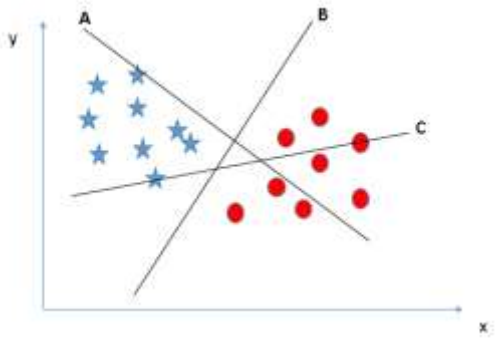
Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, perform classification by finding the hyper-plane that differentiate the two classes very well

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

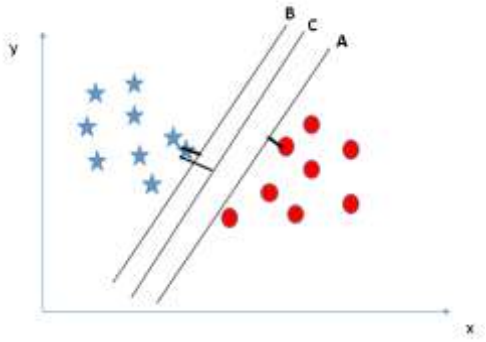


How does it work

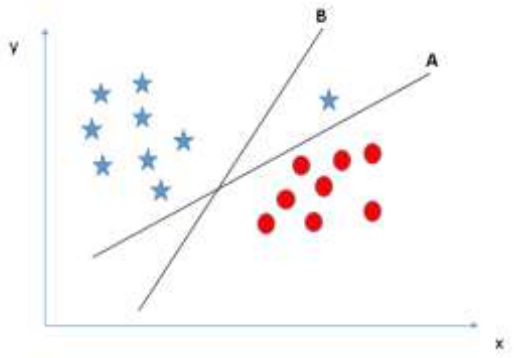
- **Identify the right hyper-plane:** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle. Select the hyper-plane which segregates the two classes better.



- **Identify the right hyper-plane:** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now Here, maximizing the distances between nearest data point and hyper-plane will help to decide the right hyper-plane.
  This distance is called as **Margin**.

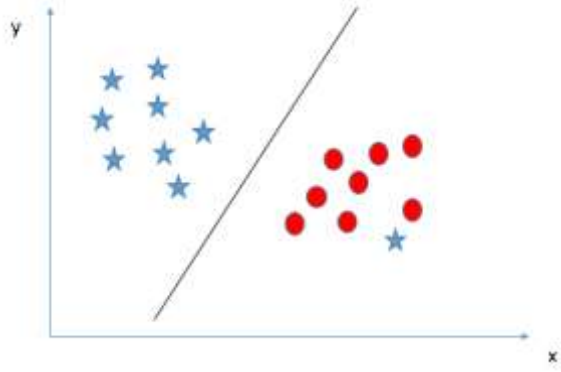


- **Identify the right hyper-plane:** Select the hyper-plane **B** as it has higher margin compared to **A.** But here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.Hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

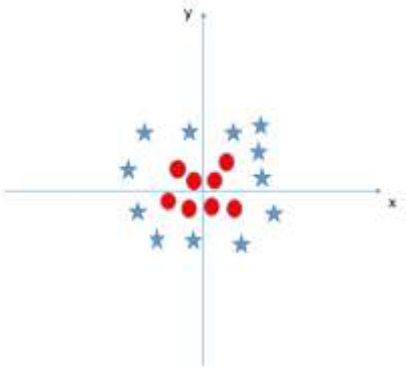


- **We classify two classes:** segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, SVM is robust to outliers.

- **Find the hyper-plane to segregate to classes:** We can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linearhyper-plane.



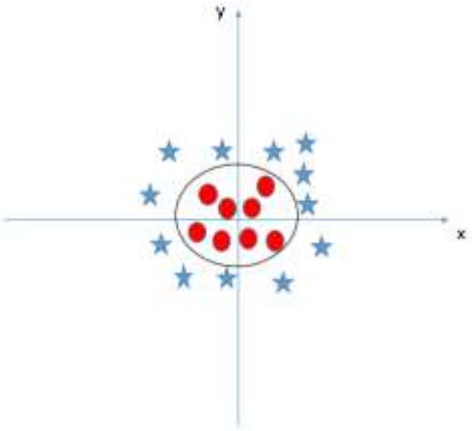
SVM can solve this problem. Easily! It solves this problem by Introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the kernel **trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.When we look at the hyper-plane in original input space it looks like a circle:



Pros and Cons associated with SVM

- **Pros:**
  - It works really well with clear margin of separation
  - It is effective in high dimensional spaces.
  - It is effective in cases where number of dimensions is greater than the number of samples.
  - It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- **Cons:**
  - It doesn't perform well, when we have large data set because the required training time is higher
  - It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

2.1.3 Linear Regression Algorithm:

Linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

Linear Regression Model Representation

It is an attractive model because the representation is so simple. The representation is a linear equation that combines a specific set of input values (x) the solution to which is the predicted output for that set of input values (y). As such, both the input values (x) and the output value are numeric. The linear equation assigns one scale factor to each input value or column, called a coefficient and represented by the capital Greek letter Beta (B). One additional coefficient is also added, giving the line an additional degree of freedom (e.g. moving up and down on a two-dimensional plot) and is often called the intercept or the bias coefficient.
For example, in a simple regression problem (a single x and a single y), the form of the model would be:

$y = B0 + B1*x$

In higher dimensions when we have more than one input (x), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. B0 and B1 in the above example).

When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model (0 * x = 0). This becomes relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

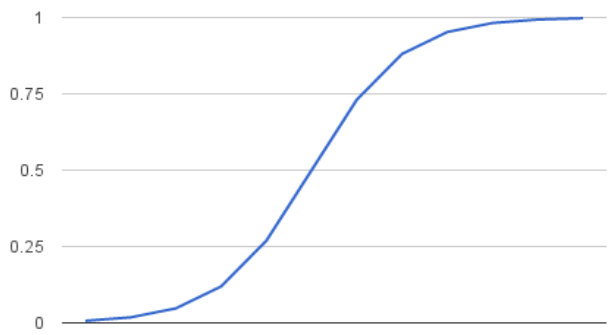
Linear Regression Learning the Model

1. Simple Linear Regression

Simple linear regression when we have a single input, we can use statistics to estimate the coefficients. This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics.

2. Ordinary Least Squares

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients. The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.



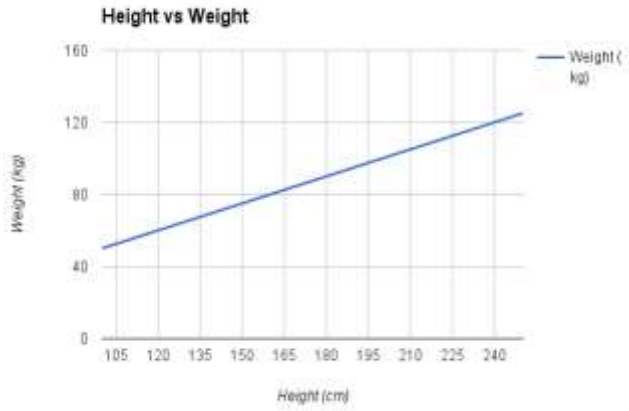
Making Predictions with Linear Regression

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs. Let's make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

y = B0 + B1 * x1 or weight =B0 +B1 * height

Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

Sample Height vs Weight Linear Regression



2.1.4 Logistic Regression Algorithm

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).

Logistic Function

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)

Where e is the base of the natural logarithms and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation. Input values (x) are combined linearly using weights or coefficient values to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary

values (0 or 1) rather than a numeric value. Below is an example logistic regression equation: y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)) Where,

y is the predicted output

b0 is the bias or intercept term

b1 is the coefficient for the single input value (x).

Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Logistic Regression Predicts Probabilities

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modelling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

P(sex=male|height)

Written another way, we are modelling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

P(X) = P(Y=1|X)

Note that the probability prediction must be transformed into a binary value (0 or 1) in order to actually make a probability prediction.

p(X) = e^(b0 + b1*X) / (1 + e^(b0 + b1*X))

ln(p(X) / 1 – p(X)) = b0 + b1 * X

ln(odds) = b0 + b1 * X

We can move the exponent back to the right and write it as:

odds = e^(b0 + b1 * X)

The linear combination relates to the log-odds of the default class.

Learning the Logistic Regression Model

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from training data. This is done using maximum-likelihood estimation.

Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data
The best coefficients would result in a model that would predict a value very close to 1 (e.g. male) for the default class and a value very close to 0 (e.g. female) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

Prepare Data for Logistic Regression

The assumptions made by logistic regression about the distribution and relationships in data are much the same as the assumptions made in linear regression.

- **Binary Output Variable**: The logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.
- **Remove Noise**: Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- **Gaussian Distribution**: Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model.
- **Remove Correlated Inputs**: Like linear regression, the model can overfit if you have multiple highly-correlated inputs.
- **Fail to Converge**: It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

2.2 MACHINE LEARNING TOOLS

The proliferation of free open source software has made machine learning easier to implement both on single machines and at scale, and in most popular programming languages. These open source tools include libraries for the likes of Python, R, C++, Java, Scala, JavaScript and Go.

2.2.1 Pandas:

It is a free library for modeling in Python. It converts CSV, JSON, and TSV data files or a SQL database into rows and columns to simplify analysis. *pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.*pandas* is a NumFOCUS sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to donate to the project. Pandas is hands down one of the best libraries of python. It supports reading and writing excel spreadsheets, CVS's and a whole lot of manipulation. It is more like a mandatory library you need to know if you're dealing with datasets from excel files and CSV files. i.e for Machine learning and data science.

Benefits of pandas

- A lot of functionality
- Active community
- Extensive documentation
- Plays well with other packages
- Built on top of NumPy
- Works well with scikit-learn

Python has long been great for data munging and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.Combined with the excellent IPython toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate.

A fast and efficient DataFrame object for data manipulation with integrated indexing;Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format. Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form. Flexible reshaping and pivoting of data sets. Intelligent label-based slicing, fancy indexing, and sub setting of large data sets. Columns can be inserted and deleted from data structures for size mutability.

Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.

High performance merging and joining of data sets. Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

2.2.2 Matplotlib:

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

It is a plotting library, meaning it's used for visualization of machine learning data. It's also flexible and easily integrated with third-party tools such as ggplot, NumPy, and HoloViews.

A Matplotlib figure can be categorized into several parts as below:

- Figure: It is a whole figure which may contain one or more than one axes (plots). You can think of a Figure as a canvas which contains plots.
- Axes: It is what we generally think of as a plot. A Figure can contain many Axes. It contains two or three (in the case of 3D) Axis objects. Each Axes has a title, an x-label and a y-label.
- Axis: They are the number line like objects and take care of generating the graph limits.
- Artist: Everything which one can see on the figure is an artist like Text objects, Line2D objects, collection objects. Most Artists are tied to Axes.

Creating different types of graphs with Pyplot

1) Bar Graphs

2) Pie Charts

3) Histogram

4) Scatter Plots and 3-D plotting

2.2.3 NumPy:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. It is open source, which is an added advantage of NumPy. NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Features:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

NumPy fully supports an object-oriented approach, starting, once again, with *ndarray*. For example, *ndarray* is a class, possessing numerous methods and attributes. Many of its methods mirror functions in the outer-most NumPy namespace, giving the programmer complete freedom to code in whichever paradigm she prefers and/or which seems most appropriate to the task at hand.

NumPy gives us the best of both worlds: element-by-element operations are the "default mode" when an ndarray is involved, but the element-by-element operation is speedily executed by pre-compiled C code. In NumPy

```
c = a * b
```

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

2.3 MACHINE LEARNING FRAMEWORK

2.3.1 Scikit-learn:

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction. Scikit-learn comes loaded with a lot of features. scikit-learn is used to build models. It should not be used for reading the data, manipulating and summarizing it.

Components of scikit-learn:

- Supervised learning algorithms: Think of any supervised learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of algorithms is one of the big reasons for high usage of scikit-learn.
- Cross-validation: There are various methods to check the accuracy of supervised models on unseen data
- Unsupervised learning algorithms: Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- Various toy datasets: This came in handy while learning scikit-learn. I had learnt SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.
- Feature extraction: Useful for extracting features from images and text (e.g. Bag of words)

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

Implementation: Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

2.3.2 Anaconda:

Anaconda is a package manager, an environment manager, a Python/R data science distribution, and a collection of over 1,500+ open source packages. Anaconda is free and easy to install, and it offers free community support.

Download anaconda from the official website and Install it. After installing Anaconda or Miniconda, for a desktop graphical user interface (GUI) use Navigator. For Anaconda prompt (or Terminal on Linux or macOS), use that and conda.

Packages available in Anaconda

- Over 200 packages are automatically installed with Anaconda.
- Over 2000 additional open source packages (including R) can be individually installed from the Anaconda repository with the condainstall command.
- Thousands of other packages are available from Anaconda Cloud.
- Other packages can be downloaded using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in some cases they can work together. However, the preference should be to install the conda package if it is available.

2.3.3 Jupyter notebook:

Project Jupyter is a comprehensive software suite for interactive computing, that includes various packages such as Jupyter Notebook, QtConsole, nbviewer, JupyterLab. This tutorial gives you an exhaustive knowledge on Project Jupyter. By the end of this tutorial, you will be able to apply its concepts into your software coding.

"Jupyter" is a loose acronym meaning Julia, Python, and R.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

The notebook interface

2.3.4 Spyder:

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package.n Furthermore, Spyder offers built-in integration with many popular scientific packages, including NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy, and more.n Beyond its many built-in features, Spyder can be extended even further via third-party plugins.n Spyder can also be used as a PyQt5 extension library, allowing you to build upon its functionality and embed its components, such as the interactive console or advanced editor, in your own software.

conda install -c anaconda spyder

Core components

1. Editor: Work efficiently in a multi-language editor with a function/class browser, real-time code analysis tools (pyflakes, pylint, and pycodestyle), automatic code completion (jedi and rope), horizontal/vertical splitting, and go-to-definition.
2. Interactive console: Harness the power of as many IPython consoles as you like with full workspace and debugging support, all within the flexibility of a full GUI interface. Instantly run your code by line, cell, or file, and render plots right inline with the output or in interactive windows.
3. Documentation viewer: Render documentation in real-time with Sphinx for any class or function, whether external or user-created, from either the Editor or a Console.
4. Variable explorer: Inspect any variables, functions or objects created during your session. Editing and interaction is supported with many common types, including numeric/strings/bools, Python lists/tuples/dictionaries, dates/timedeltas, Numpy arrays, Pandas index/series/dataframes, PIL/Pillow images, and more.
5. Development tools: Examine your code with the static analyzer, trace its execution with the interactive debugger, and unleash its performance with the profiler. Keep things organized with project support and a built-in file explorer, and use find in files to search across entire projects with full regex support.

2.4 DATA

It is useful to characterize learning problems according to the type of data they use. This is a great help when encountering new challenges, since quite often problems on similar data types can be solved with very similar techniques. For instance natural language processing and bioinformatics use very similar tools for strings of natural language text and for DNA sequences. Vectors constitute the most basic entity we might encounter in our work. For instance, a life insurance company might be interesting in obtaining the vector of variables (blood pressure, heart rate, height, weight, cholesterol level, smoker, gender) to infer the life expectancy of a potential customer. A farmer might be interested in determining the ripeness of fruit based on (size, weight, spectral data). An engineer might want to find dependencies in (voltage, current) pairs. Likewise one might want to represent documents by a vector of counts which describe the occurrence of words. The latter is commonly referred to as bag of words features.

2.4.1 TYPES OF DATA

One of the challenges in dealing with vectors is that the scales and units of different coordinates may vary widely.

Lists: In some cases the vectors we obtain may contain a variable number of features. For instance, a physician might not necessarily decide to perform a full battery of diagnostic tests if the patient appears to be healthy.

Sets: Sets may appear in learning problems whenever there is a large number of potential causes of an effect, which are not well determined. Consequently we need to infer the properties of an object given a set of features, whose composition and number may vary considerably.

Matrices: Matrices are a convenient means of representing pairwise relationships. For instance, in collaborative filtering applications the rows of the matrix may represent users whereas the columns correspond to products. Only in some cases we will have knowledge about a given (user, product) combination, such as the rating of the product by a user.

Images: Images could be thought of as two dimensional arrays of numbers, that is, matrices. This representation is very crude, though, since they exhibit spatial coherence (lines, shapes) and (natural images exhibit) a multiresolution structure.

Video: Video adds a temporal dimension to images. Again, we could represent them as a three dimensional array.

Trees and Graphs: Trees and Graphs are often used to describe relations between collections of objects. In the case of gene ontology the relationships form a directed acyclic graph.

Strings: Strings occur frequently, mainly in the area of bioinformatics and natural language processing. They may be the input to our estimation problems, e.g. when

classifying an e-mail as spam, when attempting to locate all names of persons and organizations in a text, or when modeling the topic structure of a document.

Compound structures: Compound structures are the most commonly occurring object. That is, in most situations we will have a structured mix of different data types. For instance, a webpage might contain images, text, tables, which in turn contain numbers, and lists, all of which might constitute nodes on a graph of webpages linked among each other.

CHAPTER-3

PROBLEM DESCRIPTION

3.1 PROBLEM STATEMENT

Our main objective in this project is to Design a prediction model to analyse the data set to find out the target customers for the Term Deposit Subscription.

Applying various machine learning tools to find best probability of data set

In order to save costs and time, it is important to filter the dataset to keep a certain success rate.

We are trying to develop a framework that can help business sector to decide and choose which customer to call and which one to leave alone. This data was obtained from the UC Irvine Machine Learning Repository, and relates to the direct marketing campaigns of a banking sector attempting to get its clients to subscribe to a term deposit. The marketing campaigns were conducted by making multiple phone calls to the clients. The client responses and predictor variable information are used to assess whether the subject will subscribe to the bank term deposit or not. The dataset has 20 predictor variables (categorical or numeric) and a (binary) response variable consisting of either “yes” or “no” to the term deposit subscription. Some of the factors included in this study are demographic (age, job, education, marital status); others concern information on customer financial history (whether they have defaulted on a loan before) as well as their current financial status (whether they have a home loan or personal loan, and the average annual balance of their account). There is also information about the marketing campaign process (how long since the bank contacted the customer, whether it was by mobile phone or landline, and the duration of the conversation). All of this information is compiled and used to predict the response of the customer to the offer of a term deposit with this bank. In preparation for the analysis, we converted all factor predictor variables to dummy variables with the corresponding number of levels.

3.2 RELATED WORK

Several studies have been delved into by many researchers and data miners in the phenomenon of Bank Direct Marketing via Telemarketing Campaigns. Decision Support Systems as well other data driven approaches were extensively explored. The primary objective of targeting customers through direct marketing (telemarketing) to outsource long-term deposits. Logistic Regression, Decision Tree, Neural Network and Support Vector Machine algorithms were employed Logistic Regression and Decision Tree were applied to a dataset regarding Bank. While some interesting results were obtained, a comparison of the error rate using unpruned and pruned trees as well as a bagging doesn't yield a strong ground for prediction. It was discovered that the most productive months for embarking on direct marketing fell within the last quarter. focused on defining the relevant features for increasing Bank Telemarketing effectiveness of customer subscription to term deposits. Regarding evaluation, Classification Accuracy, Sensitivity as well as Specificity measures were employed. Notwithstanding the extensive nature of their study, no mention was made about class balancing and transformation. Since the binary values of the class attribute are highly unbalanced and result obtained without balancing the class would lead to overfitting.

CHAPTER 4

PROPOSED WORK

4.1 OVERVIEW

We are preparing a prediction model for data set analysis.

That data set comes from the UCI machine learning Repository and it is related to direct marketing campaigns phone calls of the banking sector.

The goal is to predict whether the client will subscribe to a term deposit or not.

In the data set based on some parameter the model will predict probabilities of client to subscribe to a term deposit.

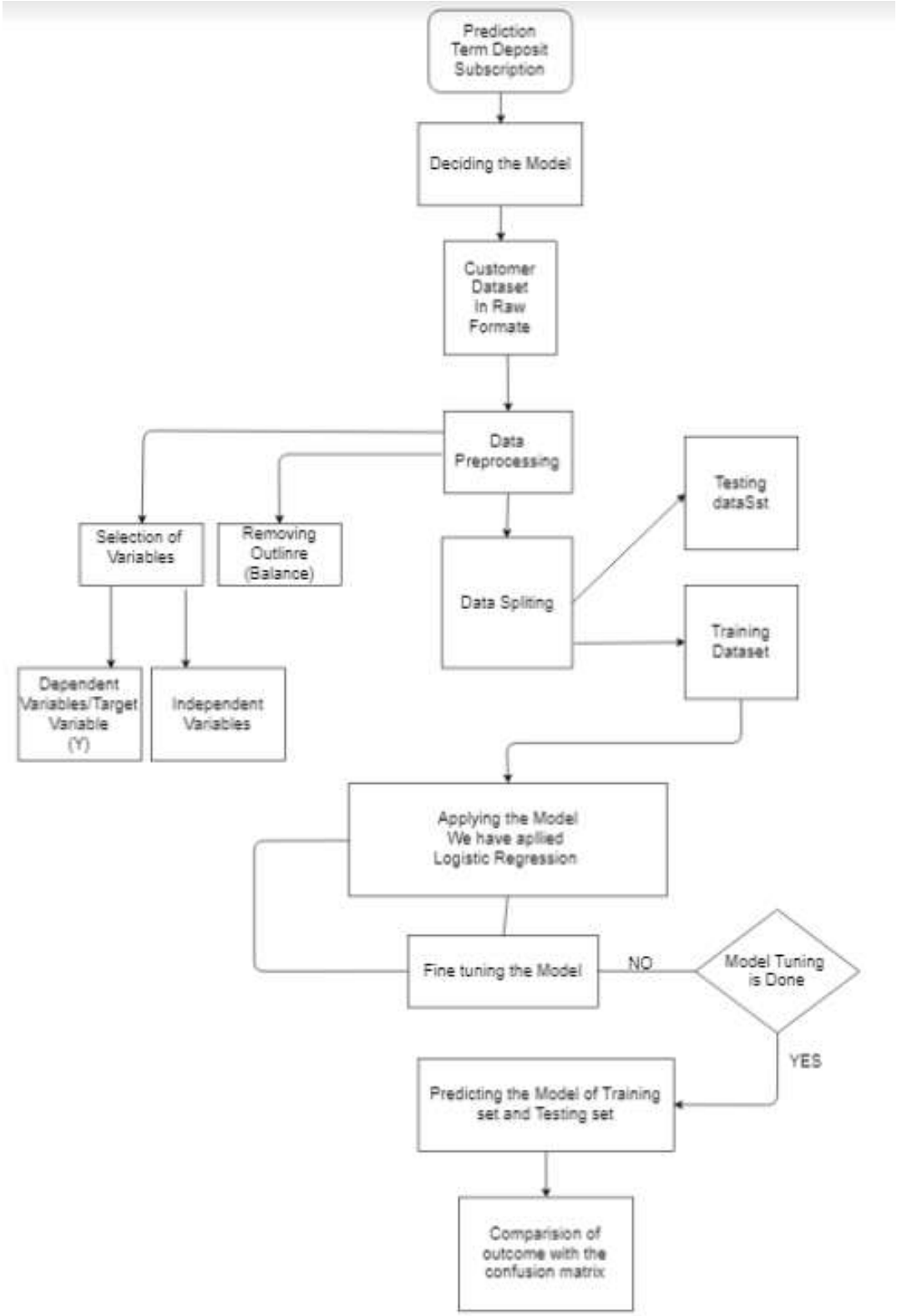
We have applied a machine learning algorithm to prepare a prediction model and shown a comparison between machine learning algorithms.

For example the phone call is made by a marketing campaign to a client if the phone call duration will be more the probability of client to taking interest in the term deposit will be more and this is how the model will able to predict whether the client will subscribe to a term deposit.

BENEFIT OF PROPOSED WORK:-

- Quality improvement as number of call dropouts will be less.
- Reduces time complexity.
- Minimize business calls/ messages/ email for customers.

4.2 FLOWCHART



CHAPTER-5

DESIGN AND DEVELOPMENT

5.1 TOOL DESCRIPTION

5.1.1 HARDWARE REQUIREMENTS

- CPU (Central Processing Unit) –Intel Core i3
- GPU (Graphics Processing Unit)
- RAM -8Gb

5.1.2 SOFTWARE REQUIREMENTS

- Python
- Anaconda
- Jupyter notebook
- Spyder

5.2 PRELIMINARY INSTALLATION

5.2.1 INSTALLING DEPENDENCIES

Following things are needed to execute the code we will be writing.

- Python3

  Download Python 3.7.1 from official website (https://www.python.org/downloads/) and install it.

- Anaconda Distribution

  Download Anaconda Distribution with Python 3.7.1 version from the official website (https://www.anaconda.com/distribution/) and Install it. After installing Anaconda, for a desktop graphical user interface (GUI) use Navigator.

- Jupyter notebook

  First install Python and Anaconda Distribution. When these are installed Jupyter notebook is also installed. To run the notebook:

```
jupyter notebook
```

CHAPTER 6

IMPLEMENTATION

6.1 DATASET

The dataset comes from the website of UCI Machine Learning repository, and it is related to direct marketing campaigns (phone calls) of a banking.

S.no	Input Variables	Description
1	age	Numeric
2	job	Type of Job (Categorical)
3	marital	Marital Status (Categorical)
4	education	Level of education (Categorical)
5	housing	Has Housing Loan (Categorical)
6	loan	Has personal Loan (Categorical)
7	contact	Contact communication type (Categorical)
8	day	Day of month on which customer was contacted(numeric)
9	month	Last contact month of year (Categorical)
11	duration	Last contact duration, in seconds (numeric)
12	campaign	Number of contact performed during the campaign and for this client (Numeric)
13	pdays	Number of days that passed by after the client was contacted from a previous campaign (numeric)
14	previous	Number of contact performed before this campaign and for this client (numeric)
15	y	Outcome of the call(categorical)

6.2 DATA PRE-PROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Whenever the data is gathered from different online sources it is collected in raw format which is not suitable for the analysis. For achieving better results from the applied model in Machine Learning the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

Steps in data preprocessing are:

- First the raw data is collected from the online data repositories for our project data set is downloaded from https://archive.ics.uci.edu/ml/datasets
- The downloaded data is in raw format so for making the data more usable first convert the dataset into comma separated values, for this purpose Microsoft Excel has text to column option available which can change the delimiter of a data set.
- Some cells which contain missing data field have to be either removed or replaced with the respective columns mean values.
- For analysis purpose the data has to be in numeric form so the string value has to be replaced with suitable numeric data .Thus the data columns which have yes and no value can be exchanged with 1 and 0.
- The columns which have categorical value can be converted to numeric value using one hot encoding. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Suppose a field “contact” contains 2 value cellular and telephone now we cannot directly assign 0 and 1 or any other numeric value to it so we will split it into two columns cellular and telephone now for the row whose value was cellular we will add 1 in cellular column and 0 in telephone column, similarly for the row whose value was telephone we will add 0 in cellular column and 1 in telephone column.
- For doing one hot encoding there is a python library called
- Another problem associated with raw data is that sometimes it has some values which appear to be numeric but are actually mistakenly printed text then that whole , those values also needs to be carefully examined and removed because if there is any value other that numeric the model will not work.

6.3 ALGORITHM

6.3.1 Logistic Function

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).

Logistic Function

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)

Where e is the base of the natural logarithms and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation. Input values (x) are combined linearly using weights or coefficient values to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary values (0 or 1) rather than a numeric value. Below is an example logistic regression equation: y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)) Where,

y is the predicted output

b0 is the bias or intercept term

b1 is the coefficient for the single input value (x).

Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

Logistic Regression Predicts Probabilities

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modelling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

P(sex=male|height)

Written another way, we are modelling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

P(X) = P(Y=1|X)

Note that the probability prediction must be transformed into a binary value (0 or 1) in order to actually make a probability prediction.

p(X) = e^(b0 + b1*X) / (1 + e^(b0 + b1*X))

ln(p(X) / 1 – p(X)) = b0 + b1 * X

ln(odds) = b0 + b1 * X

We can move the exponent back to the right and write it as:

odds = e^(b0 + b1 * X)

The linear combination relates to the log-odds of the default class.

6.4 CODE AND CORRESPONDING RESULTS

Applying Logistic Regression Algorithm over our Dataset.

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import preprocessing
        import matplotlib.pyplot as plt
        plt.rc("font",size=14)
        from sklearn.linear_model import logistic_regression_path
        from sklearn.model_selection import train_test_split
        import seaborn as sns
        sns.set(style="white")
        sns.set(style="whitegrid",color_codes=True)
```

We have pandas and numpy for data pre-processing part. Matplotlib for plots .Logistic Regression to build the model for logistic regression.train_test_split to split to data between train and testing data. seaborn used to visualize things

```
In [2]: data=pd.read_csv('bank.csv',header=0)
        print(data.shape)
        data.dropna(inplace=True)
        print(data.shape)
        print(list(data.columns))
```

```
(41188, 16)
(41188, 16)
['age', 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'duration', 'campaign',
'pdays', 'previous', 'poutcome', 'y']
```

The dataset provides the bank customers information. It includes 41,188 records and 21 fields.y – has the client subscribed a term deposit?(binary: “1”,means “Yes”, “0” means “No”)

```
In [3]: print(data.head())
```

```
   age       job  marital    education  default housing loan  \
0   44  blue-collar  married     basic.4y  unknown     yes   no
1   53   technician  married      unknown       no      no   no
2   28   management   single  university.degree       no     yes   no
3   39     services  married      high.school       no      no   no
4   55      retired  married     basic.4y       no     yes   no

    contact month day_of_week  duration  campaign  pdays  previous \
0  cellular   aug         thu       210         1    999         0
1  cellular   nov         fri       138         1    999         0
2  cellular   jun         thu       339         3      6         2
3  cellular   apr         fri       185         2    999         0
4  cellular   aug         fri       137         1      3         1

     poutcome  y
0  nonexistent  0
1  nonexistent  0
2      success  1
3  nonexistent  0
4      success  1
```

Showing first five entry of the datset.

```
In [4]: data['education'].unique()

Out[4]: array(['basic.4y', 'unknown', 'university.degree', 'high.school',
               'basic.9y', 'professional.course', 'basic.6y', 'illiterate'],
              dtype=object)
```

The education column of the dataset has many categories and we need to reduce the categories for a better modelling. The education column has the above categories. After grouping, this is the columns:

```
In [5]: data['education']=np.where(data['education']=='basic.9y','Basic',data['education'])
        data['education']=np.where(data['education']=='basic.6y','Basic',data['education'])
        data['education']=np.where(data['education']=='basic.4y','Basic',data['education'])
        data['education'].unique()

Out[5]: array(['Basic', 'unknown', 'university.degree', 'high.school',
               'professional.course', 'illiterate'], dtype=object)
```

```
In [6]: data['y'].value_counts()

Out[6]: 0    36548
        1     4640
        Name: y, dtype: int64
```

Data Exploration: There are 36548 no's and 4640 yes's in the outcome variables, get a sense of the numbers across the two classes.

```
In [7]: print("respose rate--")
        print(data['y'].value_counts()/data.shape[0]*100)

        respose rate--
        0   88.734583
        1   11.265417
        Name: y, dtype: float64
```

```
In [8]: sns.countplot(x='y',data=data,palette='hls')
        plt.show()
        plt.savefig('count_plot')
```

```
In [9]: print(data.groupby('y').mean())

             age   duration campaign     pdays previous emp_var_rate \
        y
        0 39.911185 220.844807 2.633085 984.113878 0.132374    0.248875
        1 40.913147 553.191164 2.051724 792.035560 0.492672   -1.233448

          cons_price_idx cons_conf_idx euribor3m nr_employed
        y
        0      93.603757    -40.593097  3.811491  5176.166600
        1      93.354386    -39.789784  2.123135  5095.115991
```

The average age of customers who bought the term deposit is higher than that of the customers who didn't. The pdays (days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.

Campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit.

```
In [10]: print(data.groupby('job').mean())
```

```
                 age   duration campaign     pdays previous \
job
admin.       38.187296 254.312128 2.623489 954.319229 0.189023
blue-collar  39.555760 264.542360 2.558461 985.160363 0.122542
entrepreneur 41.723214 263.267857 2.535714 981.267170 0.138736
housemaid    45.500000 250.454717 2.639623 960.579245 0.137736
management   42.362859 257.058140 2.476060 962.647059 0.185021
retired      62.027326 273.712209 2.476744 897.936047 0.327326
self-employed 39.949331 264.142153 2.660802 976.621393 0.143561
services     37.926430 258.398085 2.587805 979.974049 0.154951
student      25.894857 283.683429 2.104000 840.217143 0.524571
technician   38.507638 250.232241 2.577339 964.408127 0.153789
unemployed   39.733728 249.451677 2.564103 935.316568 0.199211
unknown      45.563636 239.675758 2.648485 938.727273 0.154545

                 emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m \
job
admin.           0.015563      93.534054      -40.245433     3.550274
blue-collar      0.248995      93.656656      -41.375816     3.771996
entrepreneur     0.158723      93.605372      -41.283654     3.791120
housemaid        0.433396      93.676576      -39.495283     4.009645
management      -0.012688      93.522755      -40.489466     3.611316
retired         -0.698314      93.430786      -38.573081     2.770066
self-employed    0.094159      93.559982      -40.488107     3.689376
services         0.175359      93.634659      -41.290048     3.699187
student         -1.408000      93.331613      -40.187543     1.884224
technician       0.274566      93.561471      -39.927569     3.820401
unemployed      -0.111736      93.563781      -40.007594     3.466583
unknown          0.357879      93.718942      -38.797879     3.949033

                 nr_employed       y
job
admin.       5164.125350  0.129726
blue-collar  5175.615150  0.068943
entrepreneur 5176.313530  0.085165
housemaid    5179.529623  0.100000
management   5166.650513  0.112175
retired      5122.262151  0.252326
self-employed 5170.674384  0.104856
services     5171.600126  0.081381
student      5085.939086  0.314286
technician   5175.648391  0.108260
unemployed   5157.156509  0.142012
unknown      5172.931818  0.112121
```

We can calculate categorical means for other categorical variables such as, education and marital status to get a more detailed sense of our data.

```
In [11]: print(data.groupby('marital').mean())
```

```
                age   duration  campaign     pdays  previous  emp_var_rate \
marital
divorced  44.899393  253.790330  2.61340  968.639853  0.168690     0.163985
married   42.307165  257.438623  2.57281  967.247673  0.155608     0.183625
single    33.158714  261.524378  2.53380  949.909578  0.211359    -0.167989
unknown   40.275000  312.725000  3.18750  937.100000  0.275000    -0.221250

          cons_price_idx  cons_conf_idx  euribor3m  nr_employed         y
marital
divorced       93.606563     -40.707069   3.715603  5170.878643  0.103209
married        93.597367     -40.270659   3.745832  5171.848772  0.101573
single         93.517300     -40.918698   3.317447  5155.199265  0.140041
unknown        93.471250     -40.820000   3.313038  5157.393750  0.150000
```

```
In [12]: print(data.groupby('education').mean())
```

```
                      age   duration  campaign     pdays  previous \
education
Basic           42.163910  263.043874  2.559498  974.877967  0.141053
high.school     37.998213  260.886810  2.568576  964.358382  0.185917
illiterate      48.500000  276.777778  2.277778  943.833333  0.111111
professional.course  40.080107  252.533855  2.586115  960.765974  0.163075
university.degree    38.879191  253.223373  2.563527  951.807692  0.192390
unknown         43.481225  262.390526  2.596187  942.830734  0.226459

                    emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m \
education
Basic                   0.191329       93.639933     -40.927595   3.729654
high.school             0.032937       93.584857     -40.940641   3.556157
illiterate             -0.133333       93.317333     -39.950000   3.516556
professional.course     0.173012       93.569864     -40.124108   3.710457
university.degree      -0.028090       93.493466     -39.975805   3.529663
unknown                 0.059099       93.658615     -39.877816   3.571098

                    nr_employed         y
education
Basic               5172.014113  0.087029
high.school         5164.994735  0.108355
illiterate          5171.777778  0.222222
professional.course 5170.155979  0.113485
university.degree   5163.226298  0.137245
unknown             5159.549509  0.145003
```

```
In [13]: %matplotlib inline
pd.crosstab(data.job,data.y).plot(kind="bar")
plt.title('Purchase Frequency for job Title')
plt.xlabel('job')
plt.ylabel('Freqeuncy of Prchase')
plt.savefig('purchase_fre_job')
```

In [14]:
```
table=pd.crosstab(data.marital,data.y)
table.div(table.sum(1).astype(float),axis=0).plot(kind='bar',stacked=True)
plt.title('stacked bar chart for marital status vs purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of customer')
plt.savefig('marital_vs_pur_stack')
```

In [15]:
```
table=pd.crosstab(data.education,data.y)
table.div(table.sum(1).astype(float),axis=0).plot(kind='bar',stacked=True)
plt.title('stacked bar chart for education vs purchase')
plt.xlabel('education')
plt.ylabel('Proportion of customer')
plt.savefig('edu_vs_pur_stack')
```

In [16]:
```
pd.crosstab(data.day_of_week,data.y).plot(kind='bar')
 plt.title('purchase frequency for day of weel')
 plt.xlabel('day of week')
 plt.ylabel('Proportion of customer')
 plt.savefig('pur_dayofweek_bar')
```

In [17]:
```
pd.crosstab(data.month,data.y).plot(kind='bar')
plt.title('purchase frequency for month')
plt.xlabel('month')
plt.ylabel('Proportion of customer')
plt.savefig('pur_month_free_bar')
```

In [18]:
```
data.age.hist()
plt.title('historagram for age')
plt.xlabel('age')
plt.ylabel('frequency')
plt.savefig('hist_age')
```

In [19]:
```
pd.crosstab(data.poutcome,data.y).plot(kind='bar')
plt.title('purchase frequency for outcome')
plt.xlabel('poutcome')
plt.ylabel('Proportion of customer')
plt.savefig('pur_fre_pout_bar')
```

In [22]:
```
data_final_vars = data_final.columns.values.tolist()
y=['y']
X=[i for i in data_final_vars if i not in y]
print(X)
```

```
['age', 'duration', 'campaign', 'pdays', 'previous', 'emp_var_rate', 'cons_price_idx', 'cons_conf_idx', 'euribor3m',
 'nr_employed', 'job_admin.', 'job_blue-collar', 'job_entrepreneur', 'job_housemaid', 'job_management', 'job_retired',
 'job_self-employed', 'job_services', 'job_student', 'job_technician', 'job_unemployed', 'job_unknown',
 'marital_divorced', 'marital_married', 'marital_single', 'marital_unknown', 'education_Basic', 'education_high.school',
 'education_illiterate', 'education_professional.course', 'education_university.degree', 'education_unknown',
 'default_no', 'default_unknown', 'default_yes', 'housing_no', 'housing_unknown', 'housing_yes', 'loan_no',
 'loan_unknown', 'loan_yes', 'contact_cellular', 'contact_telephone', 'month_apr', 'month_aug', 'month_dec',
 'month_jul', 'month_jun', 'month_mar', 'month_may', 'month_nov', 'month_oct', 'month_sep', 'day_of_week_fri',
 'day_of_week_mon', 'day_of_week_thu', 'day_of_week_tue', 'day_of_week_wed', 'poutcome_failure',
 'poutcome_nonexistent', 'poutcome_success']
```

In [23]:
```
X=data.iloc[:,0:20]
Y=data.iloc[:,20:21]
```

In [24]:
```
X=X.drop('default',axis=1)
X=X.drop('month',axis=1)
X=X.drop('day_of_week',axis=1)
```

In [25]:
```
X=np.array(X)
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:,1] = labelencoder_X.fit_transform(X[:,1])
X[:, 2] = labelencoder_X.fit_transform(X[:, 2])
X[:, 3] = labelencoder_X.fit_transform(X[:, 3])
X[:, 4] = labelencoder_X.fit_transform(X[:, 4])
X[:, 5] = labelencoder_X.fit_transform(X[:, 5])
X[:, 6] = labelencoder_X.fit_transform(X[:, 6])
X[:, 11] = labelencoder_X.fit_transform(X[:, 11])
```

In [26]:
```
onehotencoder = OneHotEncoder(categorical_features = [1,2,3,4,5,6,11])
X = onehotencoder.fit_transform(X).toarray()
```

In [27]:
```
from sklearn.model_selection import train_test_split
X_train, x_test, y_train, y_test=train_test_split(X,Y, test_size=0.2)
```

In [29]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[29]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=0, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

```
In [30]: y_out=classifier.predict(x_test)
         print('accuracy of lgc on test set : {:.2f}'.format(classifier.score(x_test,y_test)))
```

accuracy of lgc on test set : 0.91

```
In [32]: from sklearn import model_selection
         from sklearn.model_selection import cross_val_score
         kfold = model_selection.KFold(n_splits=10,random_state=7)
         modelCV = LogisticRegression()
         scoring='accuracy'
         results= model_selection.cross_val_score(modelCV,X_train,y_train, cv=kfold ,
                                                  scoring=scoring)
         print("10-fold cv avg accuracy: %.3f" %(results.mean()))
```

10-fold cv avg accuracy: 0.909

```
In [33]: from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test,y_out)
```

```
Out[33]: array([[7147,  172],
                [ 555,  364]], dtype=int64)
```

```
In [34]: from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve
         logit_roc_auc = roc_auc_score(y_test, classifier.predict(x_test))
         fpr,tpr, thresholds = roc_curve(y_test,classifier.predict_proba(x_test)[:,1])
         plt.figure()
         plt.plot(fpr,tpr,label='Logistic Regression (area=%0.2f)'%logit_roc_auc)
         plt.plot([0,1],[0,1],'r--')
         plt.xlim([0.0,1.0])
         plt.ylim([0.0,1.05])
         plt.ylabel('true positive rate')
         plt.title('receive operating characterstic')
         plt.legend(loc='lower right')
         plt.savefig('log_roc')
         plt.show()
```

Random Forest Implementation: -

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import preprocessing
        import matplotlib.pyplot as plt
```

```
In [2]: data=pd.read_csv('bank.csv',header=0)
        print(data.shape)
        data.dropna(inplace=True)
        print(data.shape)
        print(list(data.columns))
```

(41188, 16)
(41188, 16)
['age', 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays', 'previous', 'poutcome', 'y']

In [23]:
```
X=data.iloc[:,0:20]
Y=data.iloc[:,20:21]
```

In [25]:
```
X=np.array(X)
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:,1] = labelencoder_X.fit_transform(X[:,1])
X[:, 2] = labelencoder_X.fit_transform(X[:, 2])
X[:, 3] = labelencoder_X.fit_transform(X[:, 3])
X[:, 4] = labelencoder_X.fit_transform(X[:, 4])
X[:, 5] = labelencoder_X.fit_transform(X[:, 5])
X[:, 6] = labelencoder_X.fit_transform(X[:, 6])
X[:, 11] = labelencoder_X.fit_transform(X[:, 11])
```

In [26]:
```
onehotencoder = OneHotEncoder(categorical_features = [1,2,3,4,5,6,11])
X = onehotencoder.fit_transform(X).toarray()
```

In [27]:
```
from sklearn.model_selection import train_test_split
X_train, x_test, y_train, y_test=train_test_split(X,Y, test_size=0.2)
```

In [37]:
```
from sklearn.ensemble import RandomForestRegressor
classifier = RandomForestRegressor(n_estimators = 10, random_state = 0)
classifier.fit(X_train, y_train)
```

```
D:\python + ML\Practice\venv\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[37]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', 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, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [38]:
```
y_out=classifier.predict(x_test)
print('accuracy of RF on test set : {:.2f}'.format(classifier.score(x_test,y_test)))
```

```
accuracy of RF on test set : 0.39
```

CHAPTER-7

RESULT

7.1 RESULT

Graph between Purchase Frequency VS Job Title

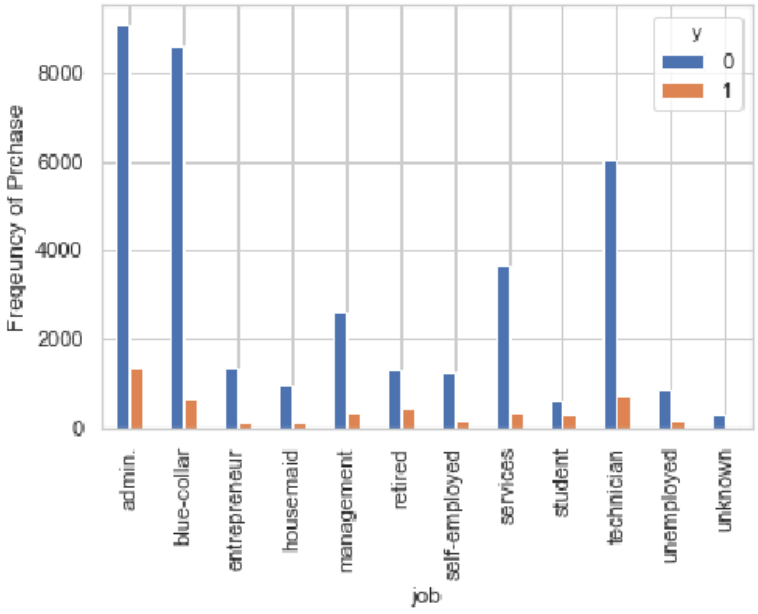


Figure -7.1

Graph between Marital Status VS Purchase Frequency

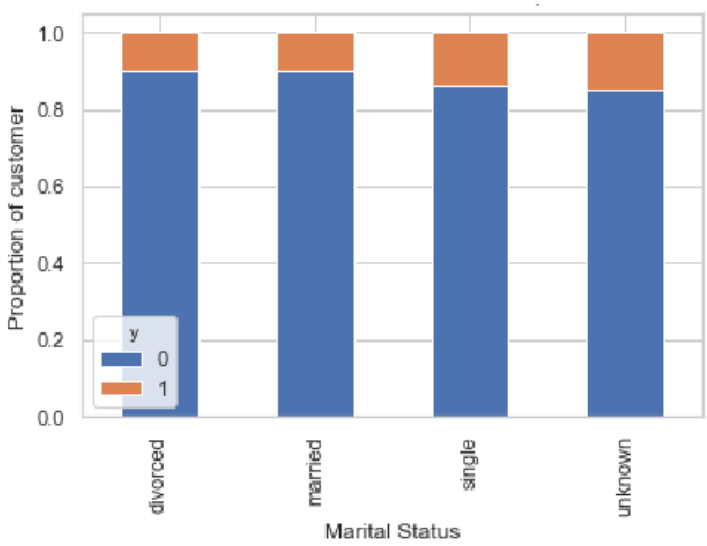


Figure - 7.2

Graph between Education VS Purchase Frequency

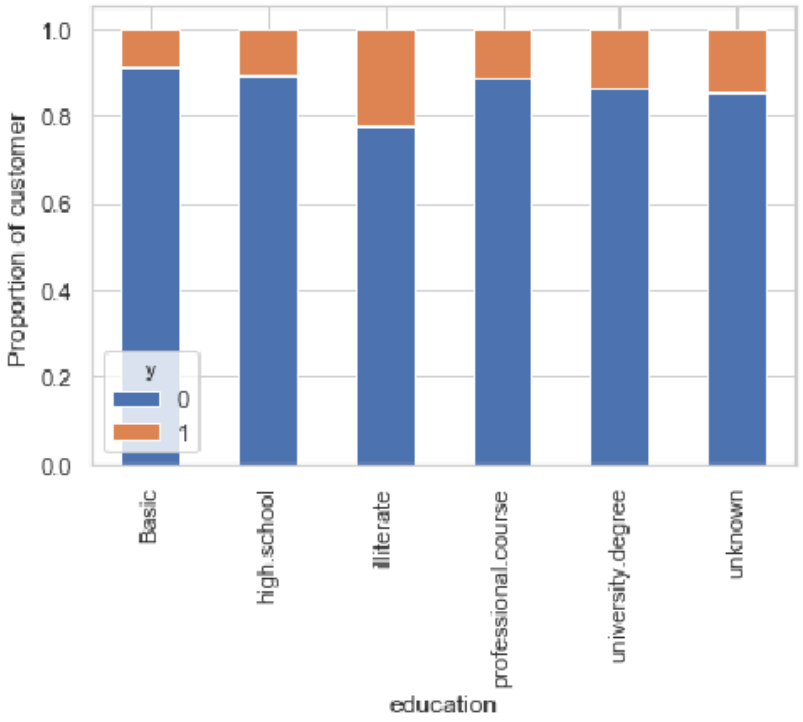


Figure -7.3

Graph between Purchase Frequency VS day_of_week

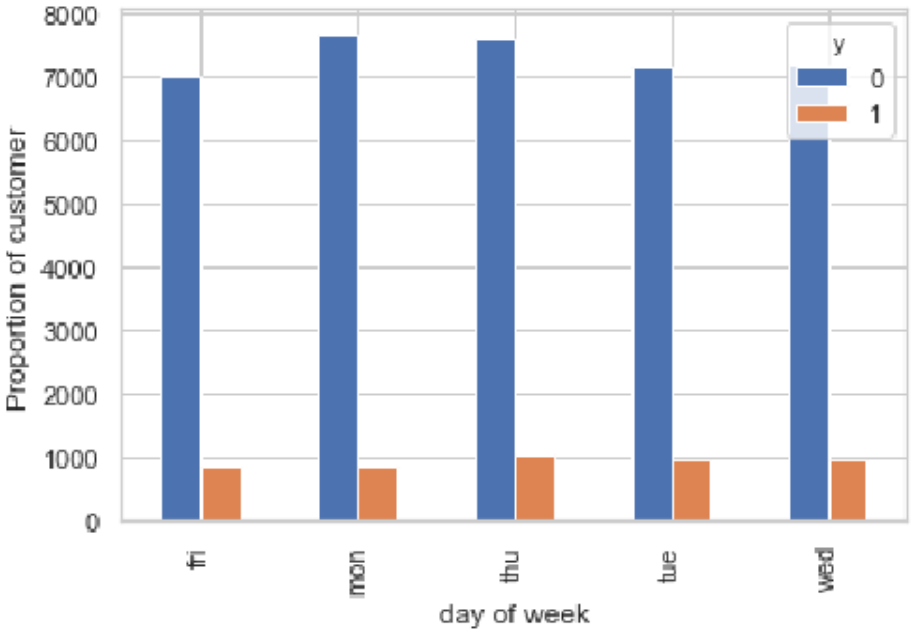


Figure -7.4

Graph between Purchase Frequency VS Month

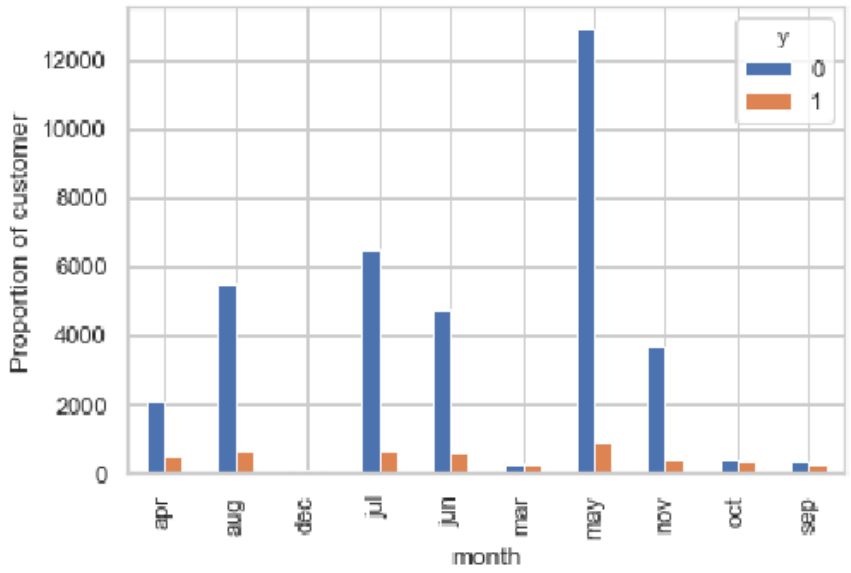


Figure -7.5

Histogram of age

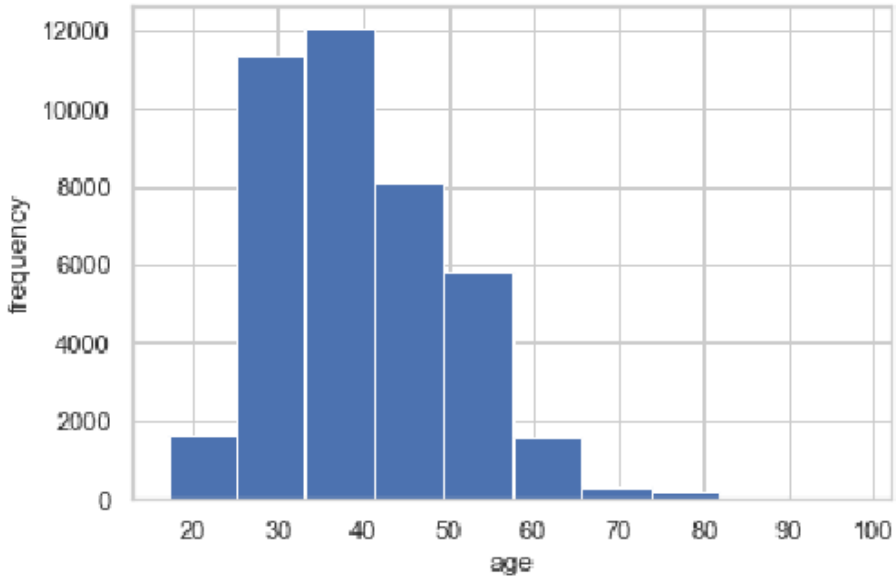


Figure - 7.6

Graph between Purchase Frequency VS Outcome

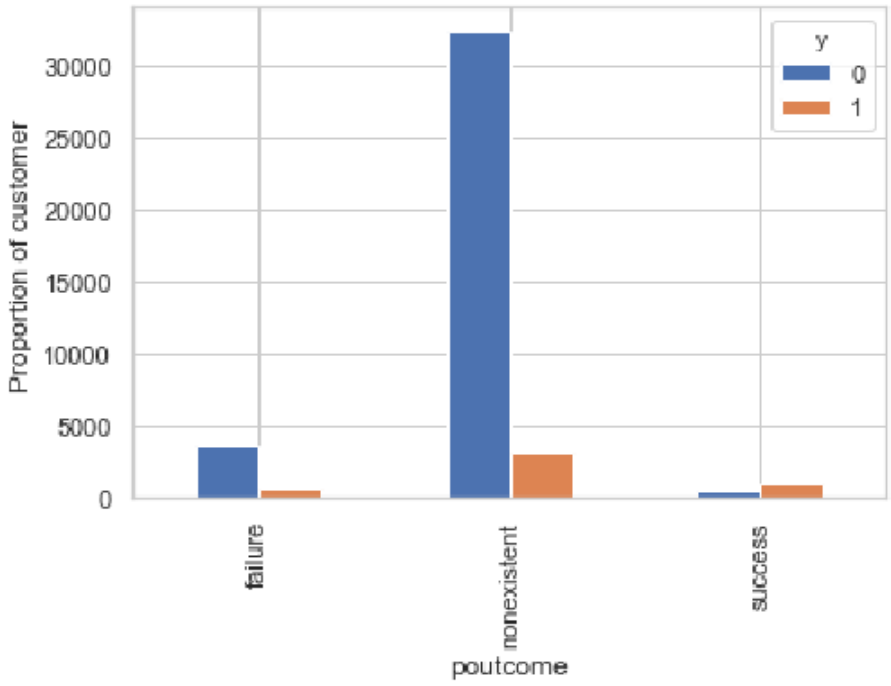


Figure -7.7

Final output chart Graph

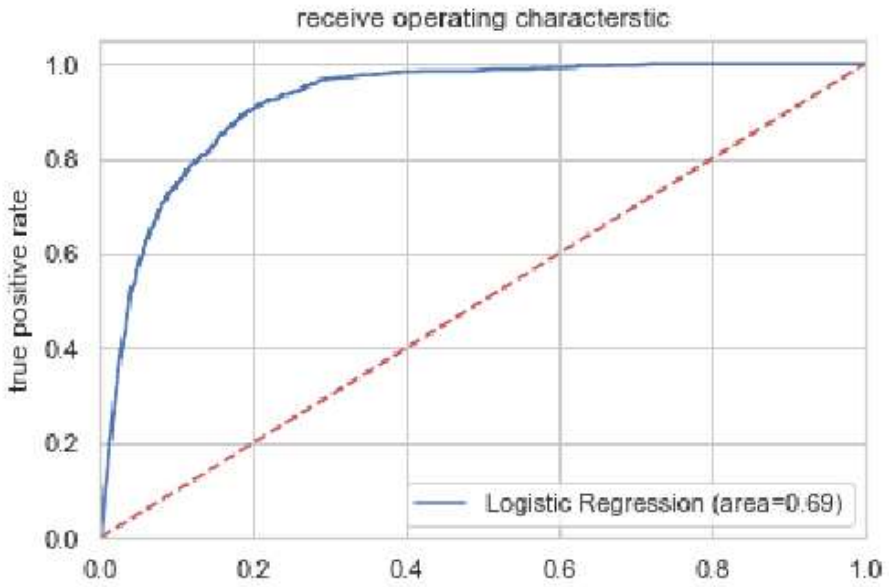


Figure - 7.8

The study applies four machine learning algorithms which comprise the Multilayer Perceptron Neural Network (MLPNN), Decision Tree (C4.5), Logistic Regression and finally Random Forest.The study applies four machine learning algorithms which comprise the Multilayer Perceptron Neural Network (MLPNN), Decision Tree (C4.5), Logistic Regression and finally Random Forest.

7.2 COMPARISION

The study applies two machine learning algorithms which comprise the Logistic Regression and Random Forest.

Logistic Regression as the first algorithm for this study has the advantage of discovering associations between nominal (binary) dependent variables and one or more continuous explanatory variables. In addition, Logistic Regression employs maximum probability approximation. With it, the initial values of the predicted parameters are used and the probability of the sample coming from a population with those parameters is calculated. The values of the estimated parameters are then attuned repeatedly until the greatest probability value is attained. That is, the maximum probability approaches strife to find estimations of parameters that make the data observed "most probable". Since this study uses a data set that has continuous independent variables and a binary dependent variable, applying Logistic Regression as a classifier to determining customer subscription to the term deposit of the bank telemarketing campaign under study is suitable.

As the Second classifier, Random Forests (RF), which comprise a mixture of tree predictors to the extent that each tree relies on the values of a random vector partitioned independently together with a similar spread for all trees in the forest. It employs a function of internal estimation to monitor inaccuracies and measure the strength of similar/dissimilar relationships . In this study, RF is used a predictive modeling classifier in determining customers who really subscribed to a term deposit.

Conversely, evaluation of the models was carried out using the prediction accuracies of these classifiers and the Receiver Operations Characteristic (ROC) curve.

Before modeling was done, an initial exploration of the dataset (41,188 observations and 20 variables) was conducted. The exploration revealed that the dependent variable having a binary class value of yes and no was high imbalanced with 36,548 of the instances being ‘No’ (did not subscribe to the term deposit) and the remaining 4,640 being ‘YES’ (subscribed to the term deposit).

The logistic regression produces the better accuracy among the two classifiers with 91.0 % followed by Random Forest with 39.0% percentage accuracy.

CHAPTER-8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

Thus the target no. of customers to be focused upon for term deposits by the bank are predicted successfully using logistic regression model with an accuracy of 89.8%

The average age of customers who bought the term deposit is higher than that of the customers who didn't.

The pdays(days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.

Surprisingly, campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit. We can calculate categorical means for other categorical variables such as, education and marital status to get a more detailed sense of our data.

The frequency of purchase of the deposit depends a great deal on the job title. Thus, the job title can be a good predictor of the outcome variable.

The marital status does not seem a strong predictor for the outcome variable. Education seems a good predictor of the outcome variable.

Day of week may not be a good predictor of the outcome. Month might be a good predictor of the outcome variable.

8.2 FUTURE SCOPE

The project can be further improved by using more data entries, recent data that is being generated can also we added to the existing data set to continuously increase the accuracy of the model.

As of now the accuracy is 89.8% by training the model against much larger data set with more number of entries the accuracy will increase. The accuracy of outcome might also increase by using other algorithms such as random forest, K mean, support vector machine.

BIBLIOGRAPHY AND REFRENCES

- https://www.udemy.com/machinelearning/
- https://archive.ics.uci.edu/ml/index.php
- https://www.geeksforgeeks.org/machine-learning/
- https://machinelearning-blog.com
- https://stackoverflow.com/search?q=machine+learning
- https://www.anaconda.com/distribution/
- https://docs.anaconda.com/anaconda/
- https://scikit-learn.org/stable/