MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Final Report

UPLIFT MODEL FOR CUSTOMER PROPENSITY MODELING

SUBMITTED
BY

SAI RAGHU TEJA DAVULURI 170907024

Under the Guidance of:

Dr. Goutham Simha GD
Assistant Professor
Department of ECE
Manipal Institute of Technology

Mr. AAKASH SHARMA
Sr. Manager
AS Dept
Ugam Solutions Pvt. Ltd.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

MANIPAL-576104, KARNATAKA, INDIA

Manipal

09-08-2021.

CERTIFICATE

This is to certify that the project titled **UPLIFT MODEL FOR CUSTOMER PROPENSITY MODELING** is a record of the bonafide work done by **SAI RAGHU TEJA** (*Reg. No. 170907024*) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (BTech) in **ELECTRONICS AND COMMUNICATION ENGINEERING** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Unit of Manipal Academy of Higher Education), during the academic year 2020 - 2021.

Dr. Goutham Simha GD
Assistant Professor
Department of ECE
Manipal Institute of Technology

Mr. AAKASH SHARMA
Head of Department
AS Dept
Ugam Solutions Pvt. Ltd.

CERTIFICATE

ugam
POWERING ANALYTICS INTO ACTION
A Merkle Company

Date: **05 Aug, 2021**

To whomsoever it may concern

This is to certify that **Mr. Sai Raghu Teja** was employed with **Ugam Solutions SEZ Private Limited**, an analytics and the services company, from **02 March, 2021** to **20 Jul, 2021** as a **Consultant** in the **Analytics Services** department.

- He worked on a project titled - Uplift Model for Customer Propensity Modeling

His performance on the above mentioned project was found to be satisfactory.

This letter is being issued to him in order to help him provide confirmation of his internship to the college.

For **Ugam Solutions SEZ Pvt. Ltd,**

Vaishali Mehta
Authorized signatory for Ugam Solutions SEZ Pvt Ltd

Ugam Solutions SEZ Pvt. Ltd.
CHIL-SEZ, India Land Tech Park Pvt Ltd, Block A, 1st Floor, Keeranatham Village, Saravanampatty PO, Coimbatore 641 035. Tel: +91 42 2664 8400
Ground Floor, Wing-A, H2 Block, Mountain Ash Building, Manyata Embassy Business Park SEZ, Outer Ring Road, Nagavara, Bangalore 560 045. Tel : +91 80 6117 2900
Registered Office : 6th Floor, B-Wing, Prism Tower, Malad Link Road, Goregaon (West), Mumbai - 400 104. Tel : +91 22 6652 7300
Email : info@ugamsolutions.com **Website :** www.ugamsolutions.com
CIN : U64200MH2008PTC182064

ACKNOWLEDGEMENT

I have tried to put in sincere effort in developing the concept of the project & completing the same. It would not have been feasible, however, without the help and the constant support & guidance of Mr. Aakash Sharma, who supervised my work with extra care. I'd want to express my deepest appreciation to everyone who helped us finish the project on time. Heartfelt appreciation to everyone who contributed to completing the project on schedule.

The project would not have been completed without the excellent teaching and training imparted by the great faculty members of our department who helped me throughout.

Lastly, I would like to thank my parents for their constant impetus and support in completing the project.

ABSTRACT

Uplift modelling has become commonplace as a result of the need to estimate the real advantage from targeting an individual for marketing objectives. Statistical machine learning algorithms are effective approaches for calculating probabilities in uplift models. Random Forests and the conventional technique Logistic Regression were used as statistical machine learning methods.

The data was obtained from a well-known retail firm, and the goal of the research is to determine which uplift modelling technique and statistical machine learning method, given the data, produces the best results. The partition of control data in each data set, as well as the variable selection stage, were shown to be critical components in the modelling procedures. For the uplift algorithm to be successful, the method of choice should be either model it is directly using Random Forest & Xgboost, or 'class variable transformation' using Logistic Regression. Furthermore, the 'subtraction of two models' did not perform well since each model focused too much on modelling the class in both data sets separately instead of modelling the difference between the class probabilities.

Hence, the conclusion is to use an approach that models the uplift directly and to use a significant amount of control data in the data sets.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1 INTRODUCTION

This project begins with a general introduction to the area for the degree project, as presented in the following sub-sections.

1.1) General

Predictive modelling is a typical method in retail and marketing for targeting and assessing individual responses when an action is done. The action is typically referred to as a customer campaign/offer, and the response to the model is the chance that a given consumer would act on the offer. Putting it in a different way, the objective is to predict the conditional class probability:

$$P(Y = 1|X = x) \quad \text{(Eq 1.1)}$$

Where the response **Y 0, 1** indicates whether a client responded positively to an activity (i.e., made a purchase) or not. The customer's quantitative and qualitative qualities are represented by X = (X1,..., XP), and '**x**' represents a single observation.

The resultant classifier may then be used to decide which consumers should be targeted when marketing promotions are sent out using standard response modelling. In real-life situations, this isn't always the best method to take because the targeted consumers are more likely to accept the offer once it's been sent out. As a result, a second-order strategy has been developed.

1.2) Motivation

When dealing with uplift modelling, which involves utilizing one treatment group and one control group, a problem occurs. Only one outcome may be seen for each participant in the experiment. The person is either in the treatment group or the control group. A person can never be a member of both organizations. To put it another way, it is impossible to know for sure if the consumer in the treatment group responds as a result of the therapy because the same client cannot be in the control group at the same moment. As a result, unlike in classification issues where the person's class is known, it is not possible to assess judgments at the individual observational unit. As a result, assessing uplift models becomes more difficult.

Furthermore, uplift modelling has yet to be validated on the data utilized in this research, as well as analogous data owned by the data owner. As a result, it's uncertain if the uplift modelling approach can be used on this data and produce the required results.

As a result, the issue to be solved is how to use the uplift modelling technique to maximize customer targeting in the marketing domain while also being able to model the real benefit from targeting one specific individual.

1.3) Problem Definition

When a group of the highest-scoring consumers is targeted, a Direct Marketing campaign makes the premise that it will generate maximum incremental response. A Propensity/Response model will not inform marketers which consumers are most likely to contribute to incremental campaign response by itself. As a result, a different statistical model is required to target consumers whose response propensities are drastically influenced by "touching" them with an offer.

So, the question to be answered is how to optimize customer targeting in the marketing domain by using the uplift modelling approach, and at the same time, being able to model the actual gain from targeting one specific individual. Furthermore, how should the uplift modelling technique be implemented in the best way to obtain the most relevant results given this kind of data?

1.4) Organization of Report

In chapter 2, the idea behind uplift modeling is explained, along with some related work that has already been done in the area. The data is described in Chapter 3, which includes statistics from the various campaigns and the types of variables gathered in the various data sets. The variables are listed in a table in which no variable is left out, implying that the table provides a list of all the variables that are utilized before any variable selection is performed.

Following the data description comes Chapter 4, which covers all of the project's theories. A description of how to pre-process data, as well as various ideas of variable selection, are provided here. Additionally, the three distinct approaches to uplift modelling are discussed, as well as the statistical machine learning algorithms that are utilized to do uplift modelling. The uplift modeling approach used in this thesis is the Two-Model approach. Random Forest (without and with tuning), Xgboost (without and with tuning), Logistic Regression, Stacking Classifier, and Neural Networks are the statistical machine learning algorithms used for uplift modelling. There is also an explanation of the resampling approach, Cross-Validation. The assessment criteria utilized in this research, notably Receiver Operating Characteristic Curves and Qini
Curves are also discussed.

The experimental results are given in detail in Chapter 5. The outcomes of the data pre-processing are first shown. Second, each implementation is detailed, including tables and figures showing the best-performing model's outcomes (s). Chapter 6 of the study concludes with conclusions based on the findings.

1.5) Software Requirements

The model is developed using Python 3.6. In this, essential libraries used for classification & regression are NumPy, pandas, pylift, matplotlib (for visualization), sci-kit learn (for splitting and standardizing training & testing set). The IDE on which we will build the model is Jupyter Notebook.

CHAPTER-2 BACKGROUND THEORY

Machine learning is a field of computer science and statistics that tries to classify a single instance into a category or calculate the conditional probability that it belongs to each of the classes given specified attributes. This method may be used to a variety of fields, including marketing. In fact, this method of categorization isn't very well adapted to marketing. Consider a marketing effort in which a special offer is offered to a (randomly) chosen segment of potential consumers. Then, based on the outcomes of the consumers' activities, a classifier may be created on top of it. As a consequence, the resultant classifier is utilized to determine which consumers should get the campaign. As a consequence, the consumers who are most likely to respond favourably to the offer once the campaign is put out will be
targeted. For a marketer, this is unfavourable.

Information is retrieved from an existing dataset using predictive analytics in order to identify patterns and forecast future trends or events. It identifies the likelihood of future events based on previous data using data, statistical algorithms, and machine learning approaches.[3] There are several uses of predictive modelling in which the outcome is forecasted as advise solely to a human decision-maker, and no action is performed automatically as a result of the model. An example is workload prioritization. For example, we can predict which customers are most likely to churn (cancel their contracts) in the telecom industry. We can anticipate which people are most likely to recover in healthcare. We can estimate which future donors are more likely to contribute to colleges or charity organizations.[10]

Table-2.1: Treatment to be given to particular use case according to the target outcome.

Use Case	Targeted Outcome	Treatment
Phone Customer	Not Churn	*Offer upgrade*
Patient	Recover	*Treat*
Voter	Vote	*Message*
Donor	Donate	*Solicit*
Prospect	Join	*Invite*
Candidate	Accept Offer	*Pay Moving Bonus*
Inmate	Not Relapse	*Coach*
PSTD Veteran	Not self-harm	*Counsel*
Gas Well	Not Shut-in	*Insulate*
Retail Customer	Buy	*Offer sale*

Some consumers would have made a purchase regardless of whether they were targeted by the campaign or not, thus delivering the offer to this type of client wastes money. After that, some customers may have a negative reaction to receiving a marketing offer. Receiving campaign offers from the firm in question or ceasing to be a client for any other reason merely because they got the offer may be upsetting to some. Customer churn occurs when a customer ceases to conduct business with a firm. Customer turnover is something the firm in question wishes to avoid at all costs. In other words, this is not a particular consumer the marketer wants to target because sending out the campaign would be an unnecessary expenditure, and they would lose a customer.

The first type of customer is known as a **Sure Thing**, while the second is known as a **Do-Not-Disturb.** Then there's the **Lost Cause** and the **Persuadable**, which are two other types of clients. The lost cause, as the term implies, is someone who would have a negative reaction, i.e., would not make any purchases at all, regardless of whether they were targeted or not. Both the sure things and the lost causes are considered a waste of money to give treatment because the treatment will not affect their response. For Do-not-Disturb or the sleeping dogs, the treatment has the opposite effect than intended, and the customer is lost.
The persuadable, on the other hand, is the consumer that the marketer is looking for. This is the type of client who would not have made a purchase if they hadn't gotten the campaign offer, but would do so if they did. These are the kind of consumers a marketer may influence for the better. An overview of the different types of customers can be seen in Fig 2.1.

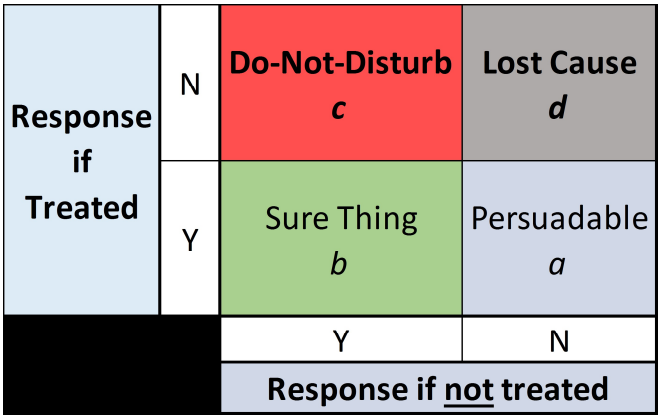

Fig 2.1: Outcomes in an Uplift Model

Uplift Modeling is a solution for this type of problem. Uplift modelling was designed with two different training sets in mind, one including the control group and the other containing the treatment group. Customers who were not targeted by the campaign are in the control group, while those who were targeted by the campaign are in the treatment group. The explicit goal in uplift modeling is to model the conditional average treatment effect. The conditional average treatment effect or uplift estimates the increase of purchase probability given that a customer receives treatment compared to if no

treatment is given. Identifying which customers are more likely to purchase before treatment would ideally let a company target a minor part of the sample and thus reduce marketing costs while maintaining or even increasing their earnings. The uplift model returns a score for each customer, where a higher score means a higher chance of a positive outcome. This score should be seen as a priority list of whom to give treatment first. The score is then used to partition the individuals into segments for the treatment group and control group, and the uplift is computed per segment. A more detailed and theoretical description of uplift modeling can be seen in further chapters. Uplift modelling is already widely used in marketing, despite the fact that it has gotten less attention in the literature than one might expect.

The decision tree-based models are also compared to more straightforward standard response-based models such that two uplift models and six standard response models are used in total [5]. The data that is being modelled on is that of a retail company's consumers. The objective is to categories customers as persuadable, based on whether or not they visit the retailer's website as a result of the campaign. According to the findings, using uplift modelling rather than response models to forecast the persuadable, i.e., which consumers are responding positively to ads, is both viable and successful. The conventional response models were effective at forecasting whether or not a consumer would visit the website, but they were terrible at predicting whether or not they would reply to the campaign(s). As a result, this research will primarily compare alternative techniques for uplift modelling, rather than standard response or purchase models, which have been shown to perform poorly in many cases.

CHAPTER-3 DATASET DESCRIPTION

The data for this uplift algorithm project was gathered from a well-known insurance firm with all its physical location's data as well as an online store where consumers may place purchases.

The marketplaces and campaigns utilized in the project will be provided in the following subsections, as well as a table with explanations of all the variables.

3.1) Markets & Campaigns

Depending on the customers' purchase behavior, the customer base can be segmented into different categories. The company is working actively to encourage frequent customers for more purchases. Because uplift modelling is employed here, the focus will be on this group of consumers, and ads will mostly be directed to customers of the type "Persuadable." Also, the campaigns differ depending on the stage of the customer, and thus, by focusing on frequent customers, there will be consistency when it comes to what kind of campaign is used in the methods of this project.

This project will focus on a single business market sector, and all of the data utilized in the uplift models will come from that market.

3.2) Variables

The table below lists all of the variables utilized in the data set before variable selection. Each row in the data set corresponds to a customer's answer, and each column in the data set contains all of the variables.

Table 3.1: Dictionary of the dataset provided by the client.

Treatment	HO= Control Group Promo = Test Group
zip5	zip code
unique_key	unique identifier of the record
age	age of the customer
channel	DM = Direct Mail DM_EM= Direct Mail & Email
state	state
inq_month	months since the last inquiry was made by the customer
resp	did the customer respond
conv	did the customer convert
region	region
division	division
cancel_reason_bucket	Policy cancelation reason
annual_premium_select	the annual premium on the policy
driver_cnt_select	drivers count in a household
vehicle_cnt_select	vehicle count in a household
polk_flag	presence of auto in a household
pif_own_rent_cd	O - homeowner R- Renter T - Refused information blank - no information available
internet_sale_ind	Policy purchased through the Internet
pif_risk_lvl	risk level of customer B - preferred C - Non-Preferred D – Reject

In fig 3.1, a detailed description of the dataset based on the data type for each column is present, along with the number of null values present in each column.

```
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   zip5                   10000 non-null  int64
 1   unique_key             10000 non-null  int64
 2   age                    6959 non-null   float64
 3   channel_cd             10000 non-null  object
 4   state                  10000 non-null  object
 5   cancels_month          10000 non-null  int64
 6   treatment              10000 non-null  object
 7   resp                   10000 non-null  int64
 8   conv                   10000 non-null  int64
 9   region                 10000 non-null  object
 10  division               10000 non-null  object
 11  cancel_reason_bucket   10000 non-null  object
 12  annual_premium_select  9581 non-null   float64
 13  driver_cnt_select      9581 non-null   float64
 14  vehicle_cnt_select     9581 non-null   float64
 15  polk_flag              10000 non-null  int64
 16  pif_own_rent_cd        9400 non-null   object
 17  internet_sale_ind      9535 non-null   object
 18  pif_risk_lvl           9535 non-null   object
```

Fig 3.1: Detailed description of the dataset

In further chapters, the missing values will be treated so that there is no data loss & will also be converting string type (object in DType in Fig 3.1) to int type so that they can be put into various Machine Learning models.

CHAPTER-4 METHODOLOGY

Uplift modelling is a data mining/predictive modelling approach for calculating the incremental influence of a therapy on a person's behavior. There are three overall approaches that exist for uplift modeling, and the first one is recognized as Subtraction of Two Models. The difference between the class probabilities of the output on test data is taken as the result. The second technique is to use a conditional divergence measure as a splitting criteria in a tree-based method to directly simulate the uplift. The third method is to employ a Class Variable Transformation, which allows an arbitrary probabilistic classification model to be converted into a model that directly predicts uplift. In this paper, we will be using the first approach. The first technique will be used in this study. This will be the starting point for creating statistical machine learning methods for categorization issues. Logistic Regression, Random Forests, and Multilayer Perceptron (Neural Networks) are ideal models in this scenario when employing a statistical machine learning approach with the goal of using it in an uplift modelling environment, as these do binary model classification

algorithm.

As a result, the theoretical foundation for data pre-processing, uplift modelling, classification, and assessment metrics will be covered in the following sections. Finally, the various programming environments are discussed, along with some justifications for their compatibility with the data and statistical machine learning approaches utilized in this research.

4.1) *Data Collection*:

The data is collected from the retail company's database and includes qualitative and quantitative attributes about the customers. The company that the data has collected is in the field of Auto and Mortgage Insurance. The data describes, among other things, the behavior of different customers in terms of demographics, description of the driver and their count, description of the vehicle, premium, and reason for their cancellation if it is. A unique code is given to all the customers, which differentiates the analysis. There are also two binary response variables that show whether a customer has made a responded or converted during a particular campaign period or not.

Each data collection utilized in this project is associated with a single campaign. As a result, a single client may appear in many data sets. There is only one variable that indicates whether a consumer is in the control or treatment group. Those observations in the control group did not get any campaign offers, but customers in the treatment group did receive the offer.

A sample of the whole data set is taken with a record of 10,000 and of 19 variables comprising 11 categorical variables and eight continuous variables on

which the model would be built. The sample data is divided into two sets in the ratio of 80: 20 as train data and test data just after the sample creation so that there are no similarities exist among the sets. If any pre-processing steps are performed on the data before the test-train separation, it will lead to data leakage. The data is loaded into the jupyter notebook from CSV, where the modelling and pre-processing are done.

4.2) Pre-processing:

The raw data that is being available is often very heavy in size and does usually has very high dimensionality. Also, the raw data is most likely to include a lot of errors such as missing values and outliers. Pre-processing data is all about removing and manipulating the raw data by specific methods so that the data that the model is going to be built on is a good representation of the results that are desired. The management and handling of raw data could be a very challenging task since pre-processing of raw data is often done manually and takes a lot of time [1]. The steps that are followed in our project in pre-processing are a)Missing value Treatment, b) Categorical Variable treatment, c) Outlier Treatment, d) Feature Engineering.

4.2.1) Missing value Treatment:

It is often an issue faced by all the engineers who deal with data. The data is collected from many sources and surveys, and in the end, all the sources of data get amalgamated and formed a database. The database engineer makes sure that all the data has been correctly fit into the pre-defined variables or features. So, since the variables are pre-defined, there exits a missing or unfilled value at certain blocks, which are called missing values; thus, it is of high importance to handle the missing data somehow. Overall, the missing values can be divided into three different categories according to [2], namely missing at random (MAR), missing completely at random (MCAR), and missing not at random (NMAR).

The missing data is MAR if, for example, respondents in a certain profession are less likely to report their income in a survey. The missing value thus depends on other variables than the one that is missing. If the data is said to be MCAR, then the missing value does not depend on the rest of the data. This can, for example be if some questionnaires in a survey accidentally get deleted. If the missing data depends on the variable that is missing, the data is said to med NMAR. An example of this can be if respondents with high income are less likely to report their income in a survey.

In our data, we have a case of MCAR. Having this kind of missing data causes the observed training data to give a corrupted picture of the actual population. There are many different ways to treat missing data. One of the

majorly used methods is to delete rows or columns, imputing with mean, median, mode. Imputation methods are in these conditions dangerous. However, if the quantity of missing data is significant enough in comparison to the total dataset, a complex method such as the EM-algorithm or Multiple Imputations can be used to manage missing values. KNN Imputer was chosen for our project. kNN Imputer's goal is to find 'k' samples in a dataset that are similar or close in space. The value of the missing data points is then estimated using these 'k' samples. The mean value of the 'k'-neighbors discovered in the dataset

is used to impute each sample's missing values.

The Euclidean distance is calculated in the presence of missing coordinates by disregarding the missing values and by scaling up the weight of the non-missing data.

$$d_{xy} = \sqrt{weight * squared\ distance\ from\ present\ coordinates} \quad \text{(Eq. 4.1)}$$

$$weight = \frac{Total\ number\ of\ coordinates}{Number\ of\ present\ coordinates} \quad \text{(Eq. 4.2)}$$

Table 4.1: Null values in the data set (before & after treatment)

col_name	null values	After Imputation
zip5	0	0
unique_key	0	0
age	3041	0
channel_cd	0	0
state	0	0
cancels_month	0	0
treatment	0	0
resp	0	0
conv	0	0
region	0	0
division	0	0
cancel_reason_bucket	0	0
annual_premium_select	419	0
driver_cnt_select	419	0
vehicle_cnt_select	419	0
polk_flag	0	0
pif_own_rent_cd	600	0
internet_sale_ind	465	0
pif_risk_lvl	465	0

```
Code:

from sklearn. impute import KNNImputer

imputer = KNNImputer (n_neighbors= 5)

imputed_X_train = pd. DataFrame(imputer.fit_transform(data_X_train))
```

4.2.2) Categorical Variable treatment:

Carrying Exploratory data analysis or Model building on categorical variables is next to impossible due to its mathematics. So, the categorical variables in the data have to be treated by certain methods and should be converted into numerical values. Most used are Label Encoding and Dummy Coding.

- Label Encoder: It converts non-numerical labels into numerical ones (or nominal categorical variables). The numerical designations are always in the range of 0 to n classes.
- Dummy Coding: Dummy coding is a method for turning a categorical input variable into a continuous variable that is widely utilized. The term 'dummy' refers to a variable that reflects one level of a category variable. A level's presence is denoted by 1, and its absence is indicated by 0. One fake variable will be created for each level present.

Table 4.2: Categorical Columns Conversion

variable	Categorical Treatment
'division'	Dummy Encoding
'region'	Dummy Encoding
'Cancel_reason_bucket'	Dummy Encoding
'Pif_own_rent_cd'	Dummy Encoding
'Internet_sale_ind'	Dummy Encoding
'Pif_risk_lvl'	Dummy Encoding
treatment'	Label Encoding
channel_cd'	Label Encoding
state'	Dummy Encoding

4.2.3) Outlier Treatment:

For each feature, there may be values that fall outside of the Least and Highest Values for a normal distribution graph. Outliers enhance your data's variability, lowering statistical power. As a consequence, eliminating outliers might make your findings statistically significant. Generally, the values above 1.5*IQR + 0.75 quantile and below 0.25 quantile-1.5*IQR of the feature are considered outliers.

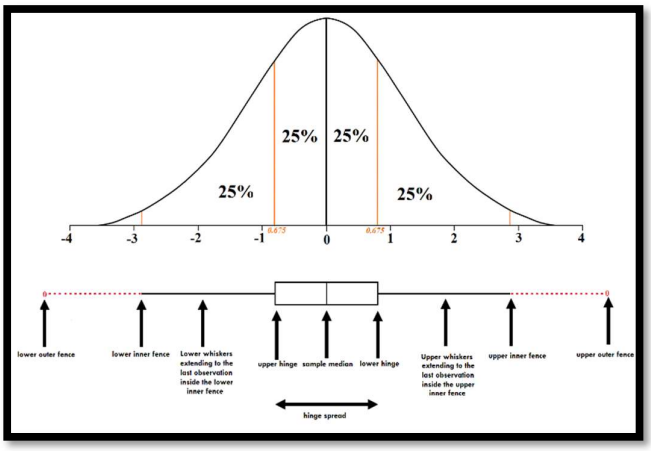

Fig 4.1: Boxplot Distribution

```
outliers = ["age", "annual_premium_select", "driver_cnt_select",
"vehicle_cnt_select"]

for col in outliers:

    out_age =
data_X_train[col].quantile(0.75)+1.5*iqr(data_X_train[col])


data_X_train[col]=data_X_train[col].mask(data_X_train[col]>out_age,out
_age)
```

The columns AGE, VEHICLE_CNT_SELECT, DRIVER_CNT_SELECT have outliers, and the process of the IQR method removes these. Below are the boxplots of the features before and after the outlier treatment and the circular dots are the outliers, and those are removed after the outlier treatment.

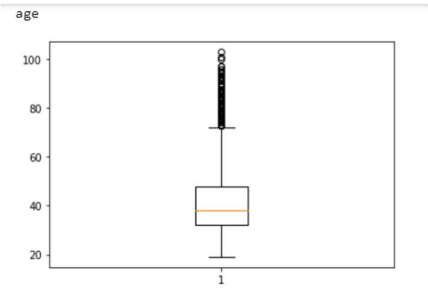

Fig 4.2 (A) : *Age* Boxplot (Before IQR)

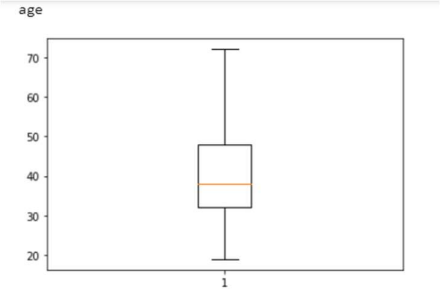

Fig 4.2 (B) : *Age* Boxplot (After IQR)

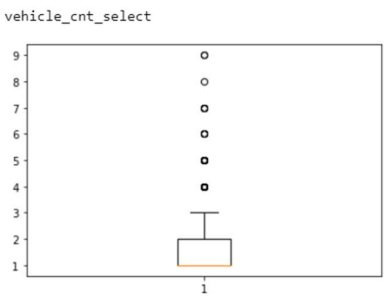

Fig 4.3 (A) :
Vehicle_cnt_select Boxplot
(Before IQR)

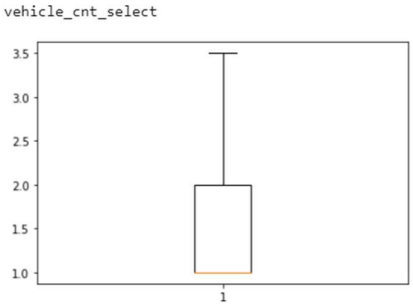

Fig 4.3 (B) :
Vehicle_cnt_select Boxplot
(After IQR)

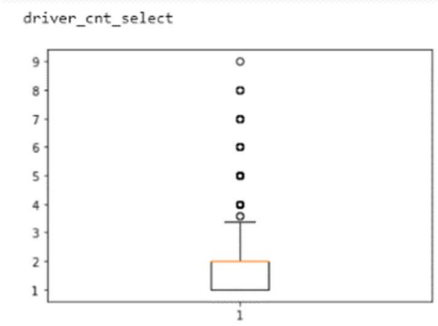

Fig 4.4 (A) :
Driver_cnt_select Boxplot
(Before IQR)

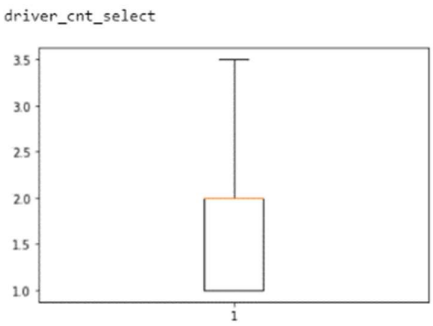

Fig 4.4 (B) :
Driver_cnt_select Boxplot
(After IQR)

4.2.4) Feature Engineering

i. High variability in the data: There are certain situations where the variables don't give any data information. Those variables should be excluded from the data analysis, or else the data might misguide the correlation and give worse results after the model building. Our data has two variables of this kind. The first one is unique_key. The reason to discard the unique_key feature from the data is that it is unique for every observation in the data, and it does not say much anything about the patterns or details regarding the data. The second one is the zip5 variable. The reason to remove this variable is that there are 4000+ values in the set of 10000 observations which, if loaded into the model, would be very highly misleading. Since this is the sample set of population, there exists a high variability.

ii. Invariability in the data: These are the cases where there is no change in the value over any of the observations. In our data set, there is no such kind of invariable features in our data set.

iii. Multicollinearity: In a multivariate regression model, multicollinearity occurs when there are substantial intercorrelations between two or more independent variables. When a researcher or analyst tries to figure out how well each independent variable can be utilized to predict or explain the dependent variable in a statistical model, multicollinearity can lead to skewed or misleading conclusions. Multicollinearity, in general, can result in larger confidence intervals, resulting in less accurate probability when it comes to the influence of independent variables in a model. That is, statistical conclusions drawn from a multicollinear model may not be reliable. In our data set, we do have multicollinear data which will be described below.

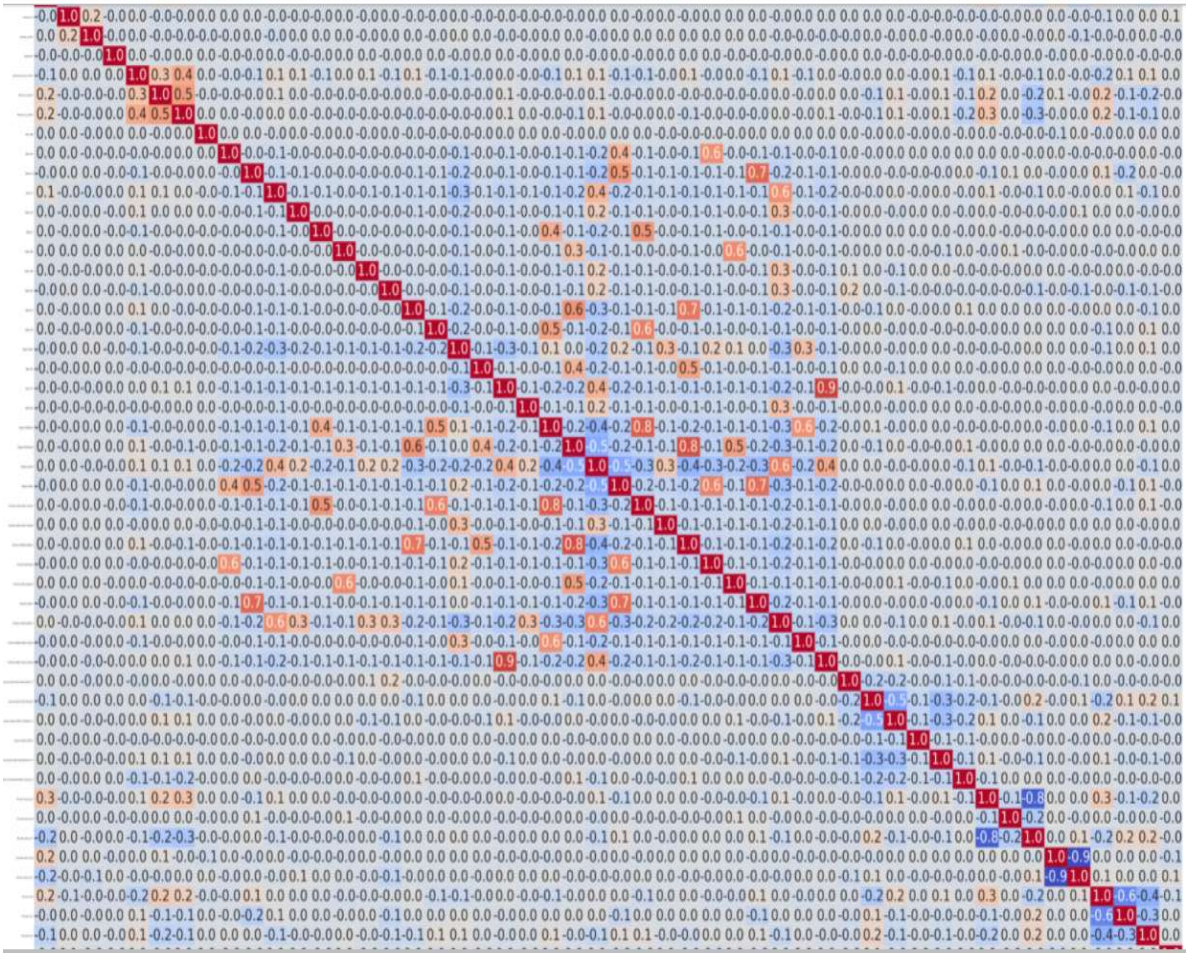

Fig 4.5: Correlation matrix between the variables (Heatmap).

After the data has been expanded, we have plotted a heat map (Fig 4.5) that is the correlation between every two individual variables comprising its Pearson's r coefficient. According to Multicollinearity, the variables have a correlation coefficient of value greater than 0.80 is to be assumed as a multi collinear

variable corresponding to the individual. In our data, we have three sets of features which are listed here as:
#State_TX & Division_West_South_Central ,,,
Region_Midwest & Division_Middle_Atlantic,
Region_Northeast& Division_East_North_Central.
From the above three sets, the variables Division_West_South_Central, Region_Midwest, and Region_Northeast were taken from each set.

4.3) Exploratory Data Analysis:

Exploratory Data Analysis (EDA) is a type of data analysis that uses visual approaches to summaries a dataset's major properties. Before beginning the modelling process, EDA is used to examine what the data can tell us. It is difficult to discern the main properties of data from a column of numbers or a complete spreadsheet. Deriving insights from simple statistics may be laborious, dull, and daunting. In this circumstance, exploratory data analysis approaches have been dev

eloped as a help.

There are two methods to categorize exploratory data analysis. The first distinction is that each approach is either non-graphical or graphical. Moreover, second, each method is either univariate or multivariate (usually just bivariate).

- Target-Churn Analysis: Target or churn analysis analyzes the outcome or target variable concerning the other features among the data and the univariate analysis of the feature itself. In the data, the target variable is binary and consists of 7999 1's and 2001 0's, making it 20% of responses from the data, which is more than an industry standard to go for modelling.

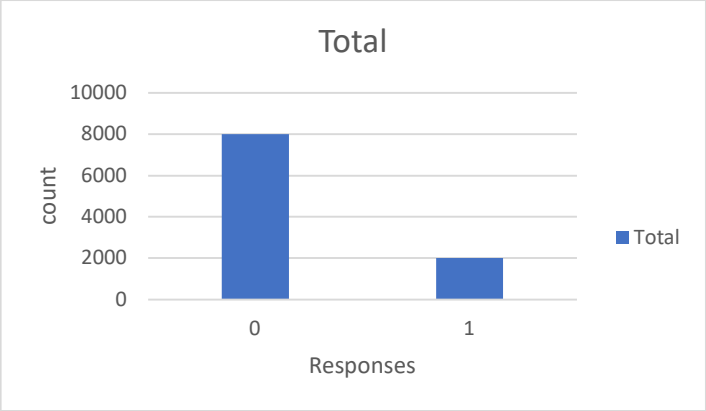

Fig 4.6: Churn Analysis

4.4) Modelling:

Creating a descriptive diagram of links between various types of information to be stored in a database is known as data modelling. Till now, in our project, the data has been prepared and furnished so that it can be speculated into our model. A model is nothing but a set of statistical models which on stacked over each other gives us a helpful output which can be used a classification or regression problem. In our project, we have used multiple statistical approaches to aim at a binary classification. In our binary classification, the output is either 1 or 0, which is a customer is responding to us, or the customer does not respond respectively. Here on we discuss each model in detail.

4.4.1) Random Forest:

A random forest algorithm's building pieces are decision trees. A decision tree is a decision-making tool with a tree-like structure. An introduction to decision trees will assist us in comprehending how random forest algorithms function.
There are three parts in a decision tree structure: decision nodes, leaf nodes, and root nodes. A decision tree method splits a training dataset into branches, which are subsequently divided into additional branches by the algorithm. This pattern repeats until a leaf node is reached. There is no way to separate the leaf node any farther.
The qualities utilized to forecast the result are represented by the nodes in the decision tree. The leaves are connected to the decision nodes. The random forest's structure is seen in the diagram below.

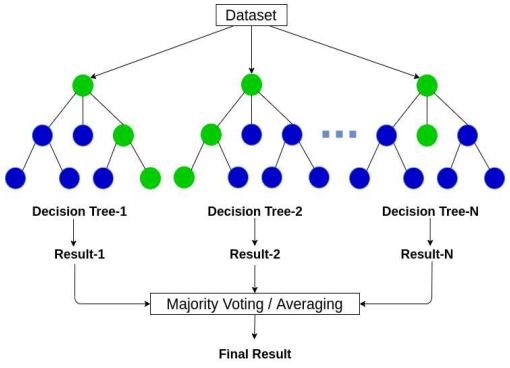

Fig 4.7: Random Forest

Code :

```
model_up = RandomForestClassifier(random_state=100)
model_up.fit(X_train_up, y_train_up)

#predict the results for test
test_pred_up = model_up.predict(X_train_up)
test_pred_val_up = model_up.predict(X_val_fea)

#cross val
scores_up = cross_val_score(model_up, X_train_up, y_train_up,
```

```
                                cv=5,
                                scoring='accuracy')
print('CV_accuracy:',scores_up)
print("Average accuracy score (across experiments):")
print(scores_up.mean())

#test the accuracy
#accuracies_up['RandomForest Classifier'] = accuracy_score(test_pred_up,
y_train_up)
#recall_up['RandomForest Classifier'] =
metrics.recall_score(y_train_up,test_pred_up)

print('Accuracy of RandomForest Classifier is: ',
accuracy_score(y_train_up,test_pred_up))
print('Recall Score of RndomForest Classifier is: ',
metrics.recall_score(y_train_up, test_pred_up))

# confusion matrix
matrix_up = confusion_matrix(y_train_up,test_pred_up, labels=[1,0])
print('Confusion matrix : \n',matrix_up)

# outcome values order in sklearn
tp_up, fn_up, fp_up, tn_up =
confusion_matrix(y_train_up,test_pred_up,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp_up, fn_up, fp_up, tn_up)

# classification report for precision, recall f1-score and accuracy
matrix_up = classification_report(y_train_up,test_pred_up,labels=[1,0])
print('Classification report : \n',matrix_up)

# auc_score_up['RandomForest Classifier'] = roc_auc_score(y_train_up,
test_pred_up)
print('Auc_Score:',roc_auc_score(y_train_up, test_pred_up))

#test the accuracy
#val_accuracies_up['RandomForest Classifier'] = accuracy_score(test_pred_val_up,
y_val)
#val_recall_up['RandomForest Classifier'] =
metrics.recall_score(y_val,test_pred_val_up)

print('Accuracy of RandomForest Classifier is: ',
accuracy_score(y_val,test_pred_val_up))
print('Recall Score of RndomForest Classifier is: ', metrics.recall_score(y_val,
test_pred_val_up))

# confusion matrix
matrix_val_up = confusion_matrix(y_val,test_pred_val_up, labels=[1,0])
print('Confusion matrix : \n',matrix_val_up)

# outcome values order in sklearn
tp_val_up, fn_val_up, fp_val_up, tn_val_up =
confusion_matrix(y_val,test_pred_val_up,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp_val_up, fn_val_up, fp_val_up, tn_val_up)

# classification report for precision, recall f1-score and accuracy
matrix_val_up = classification_report(y_val,test_pred_val_up,labels=[1,0])
print('Classification report : \n',matrix_val_up)

#val_auc_score_up['RandomForest Classifier'] = roc_auc_score(y_val,
test_pred_val_up)
print('Auc_Score:',roc_auc_score(y_val, test_pred_val_up))
```

Table 4.3: Description and Output for Random Forest

Measure	Train data	Test data
Accuracy	100%	78%
Recall	100%	4.6%
AUC Score	100%	50.8%

The above model performs well on the training data, but it fails to perform well on the test data set. So we consider these types of cases as overfitting. We have to perform hyperparameter tuning on the model.

On the training set, an overfit model may appear spectacular, but in real life, he or she will be worthless. As a result, cross-validation is included in the usual process for hyperparameter optimization to account for overfitting. Random Search or Grid Search can be used for cross-validation.

4.4.2) Grid search:

Because hyperparameter tuning is based on experimental findings rather than theory, the best way for determining the ideal settings is to evaluate the performance of each model using a variety of different combinations. However, judging each model only on the training data can lead to overfitting, one of the most common issues in machine learning.

If we optimize the model for the training data, it will perform well on the training set but will not generalize to new data, such as that found in the test set. When a model performs very well on the training set but not so well on the test set, overfitting occurs, resulting in a model that knows the training set but is unable to adapt to new conditions. As a result, to account for overfitting, cross-validation
is incorporated in the standard hyperparameter tuning procedure.

As a model performs very well on the training set but poorly on the test set, overfitting occurs, resulting in a model that knows the training set but is incapable of adapting to new conditions. As a result, to account for overfitting, cross-validation is incorporated in the standard hyperparameter tuning procedure. The data was then trained on K-1 of the folds and assessed on the Kth fold each time the model was iterated (called the validation data). Consider fitting a model with K = 5 as an example. In the first iteration, we will be training on the first four folds and assess on the fifth. The second time around, we practice the first, second, third, and fifth folds and assess the fourth. This procedure is performed three more times, each time evaluating a different fold. The model's final validation metrics are calculated by averaging the performance on each of the folds at the end of the training.

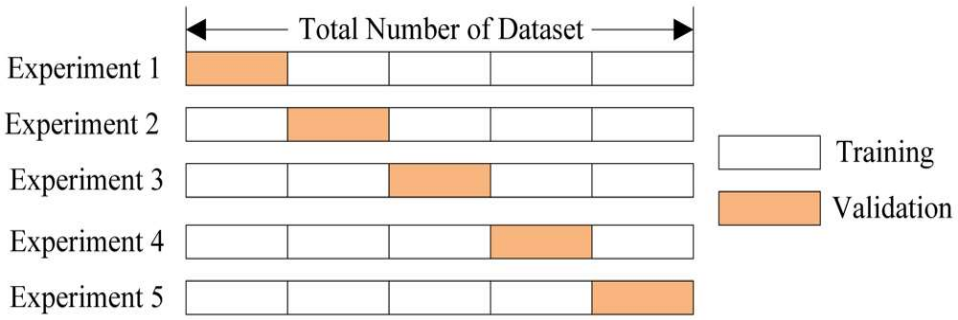

Fig.4.8 Cross-Validation Technique

For hyperparameter tuning, we run many rounds of the entire K-Fold CV process, each time with different model parameters. Then we compare all of the models, choose the best, train it on the whole training set, then put it through its paces on the testing set. We must divide our training data into K folds and train and evaluate K times each time we wish to analyses a fresh set of hyperparameters. We have 50 training loops if we have 10 sets of hyperparameters and employ a 5-fold CV. Fortunately, as with most machine learning issues, someone has already solved ours, and model tweaking using K-Fold CV can now be done automatically in Scikit-Learn.

Code :

```
from sklearn.model_selection import GridSearchCV

# Creating the parameter grid code description based on the results of random
search algorithm

param_grid = {

    'bootstrap': [True],

    'max_depth': [70,80, 90, 100, 110],

    'max_features': [2, 3],

    'min_samples_leaf': [2,3, 4, 5,6],

    'min_samples_split': [8, 10, 12],

    'n_estimators': [100, 200, 300, 1000]

}

rf = RandomForestClassifier()

rf_fea_grid = GridSearchCV(estimator = rf, param_grid = param_grid,

                           cv = 5, n_jobs = -1, verbose = 2)

# Fit the grid search to the data

rf_fea_grid.fit(X_train_up, y_train_up)

rf_fea_grid.best_params_

best_grid_fea = rf_fea_grid.best_estimator_

pred_best_grid_fea = best_grid_fea.predict(X_val_fea)

train_pred_best_grid_fea = best_grid_fea.predict(X_train_up)
```

```
print('Accuracy of RandomForest Classifier is: ',
accuracy_score(y_train_up,train_pred_best_grid_fea))

print('Recall Score of RndomForest Classifier is: ',
metrics.recall_score(y_train_up, train_pred_best_grid_fea))

print('Accuracy of RandomForest Classifier is: ',
accuracy_score(y_val,pred_best_grid_fea))

print('Recall Score of RndomForest Classifier is: ', metrics.recall_score(y_val,
pred_best_grid_fea))

# confusion matrix

matrix_val_best_grid_fea = confusion_matrix(y_val,pred_best_grid_fea,
labels=[1,0])

print('Confusion matrix : \n',matrix_val_best_grid_fea)

    # outcome values order in sklearn

tp_val_best_grid_fea, fn_val_best_grid_fea, fp_val_best_grid_fea,
tn_val_best_grid_fea =
confusion_matrix(y_val,pred_best_grid_fea,labels=[1,0]).reshape(-1)

print('Outcome values : \n',tp_val_best_grid_fea, fn_val_best_grid_fea,
fp_val_best_grid_fea, tn_val_best_grid_fea )

# classification report for precision, recall f1-score and accuracy

matrix_val_cl_best_grid_fea =
classification_report(y_val,pred_best_grid_fea,labels=[1,0])

print('Classification report : \n',matrix_val_cl_best_grid_fea)

# auc scores

#val_auc_score['RandomForest Classifier'] = roc_auc_score(y_val,
pred_best_random)

print('Auc_Score:',roc_auc_score(y_val, pred_best_grid_fea))

#randomaccuracy = evaluate(best_random, test_features, test_labels)
```

4.4.3) XGBoost:

It's a technique for learning in groups. It may not always be enough to depend on the findings of a single machine learning model. Ensemble learning is a method for combining the predictive capacity of several learners in a systematic way. The end result is a single model that combines the outputs of many single rf models.

The foundation learners, or models that make up the ensemble, might be from the same learning algorithm or from separate learning algorithms. Bagging and boosting are two types of ensemble learners that are commonly employed. Though these two approaches may be used to a variety of statistical models, decision trees have been the most popular.

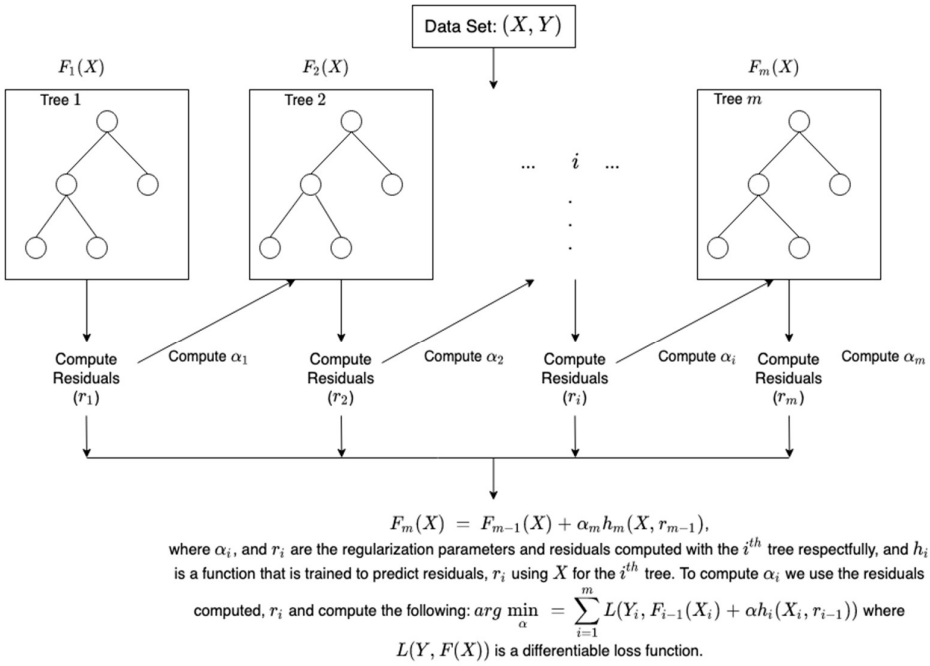

Fig 4.9: XgBoost Technique

Code :

```
from sklearn.model_selection import GridSearchCV   #Perforing grid search
import matplotlib.pylab as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 12, 4
model_xgb =
XGBClassifier(random_state=10,use_label_encoder=False,objective='binary:logistic
')
model_xgb.fit(data_X_train, data_y_train,eval_metric="auc", verbose=True)
#predicting x_test
test_pred_xgb = model_xgb.predict(data_X_train)
test_pred_val_xgb=model_xgb.predict(X_val)
#cross val
scores_xgb = cross_val_score(model_xgb, features, target,
                             cv=5,
                             scoring='accuracy')
print('CV_accuracy:',scores_xgb)
print("Average accuracy score (across experiments):")
print(scores_xgb.mean())
#appending accuracy score to accuracies dict
```

```
#accuracies['XGB Classifier'] = accuracy_score(test_pred_xgb, target)

#recall['XGB Classifier'] = metrics.recall_score(test_pred_xgb,target)

print('Accuracy Score of XGB Classifier is: ', accuracy_score(test_pred_xgb,
data_y_train))

print('Recall Score of XGB Classifier is: ',
metrics.recall_score(test_pred_xgb,data_y_train))

# confusion matrix

matrix_xgb = confusion_matrix(data_y_train,test_pred_xgb, labels=[1,0])

print('Confusion matrix : \n',matrix_xgb)


# outcome values order in sklearn

tp_xgb, fn_xgb, fp_xgb, tn_xgb =
confusion_matrix(data_y_train,test_pred_xgb,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp_xgb, fn_xgb, fp_xgb, tn_xgb )


# classification report for precision, recall f1-score and accuracy

matrix_cl_xgb = classification_report(data_y_train,test_pred_xgb,labels=[1,0])

print('Classification report : \n',matrix_cl_xgb)

#auc_score['XGB Classifier'] = roc_auc_score(target, test_pred_xgb)

print('Auc_Score:',roc_auc_score(data_y_train, test_pred_xgb))


#eval on val data

print('------eval on val data----')

#appending accuracy score to accuracies dict


#val_accuracies['XGB Classifier'] = accuracy_score(test_pred_val_xgb, y_val)

#val_recall['XGB Classifier'] = metrics.recall_score(test_pred_val_xgb,y_val)


print('Accuracy Score of XGB Classifier is: ', accuracy_score(test_pred_val_xgb,
y_val))

print('Recall Score of XGB Classifier is: ',
metrics.recall_score(test_pred_val_xgb,y_val))

# confusion matrix

matrix_val_xgb = confusion_matrix(y_val,test_pred_val_xgb, labels=[1,0])

print('Confusion matrix : \n',matrix_val_xgb)


# outcome values order in sklearn

tp_val_xgb, fn_val_xgb, fp_val_xgb, tn_val_xgb =
confusion_matrix(y_val,test_pred_val_xgb,labels=[1,0]).reshape(-1)
```

```
print('Outcome values : \n', tp_val_xgb, fn_val_xgb, fp_val_xgb, tn_val_xgb )


# classification report for precision, recall f1-score and accuracy
matrix_cl_val_xgb = classification_report(y_val,test_pred_val_xgb,labels=[1,0])
print('Classification report : \n',matrix_cl_val_xgb)
#val_auc_score['XGB Classifier'] = roc_auc_score(y_val, test_pred_val_xgb)
print('Auc_Score:',roc_auc_score(y_val, test_pred_val_xgb))
xgb_param = model_xgb.get_xgb_params()
print(xgb_param)
#print(n_estimators)
```

Table 4.4: Description and Output for Xgboost

Measure	Train data	Test data
Accuracy	100%	82%
Recall	100%	9.6%
AUC Score	100%	52.03%

4.4.4) XGBoost with Gradient Search CV:

It is similar to grid search in random forest, here we perform a series of steps updating each parameter at a particular step. Here we provided a table of parameter update at each step.

Table 4.5: Description and Output for Xgboost with Gradient Search CV

	Train accuracy	Test accuracy	train recall	test recall	train auc	test auc	update
XG_basic	91.21	82.32	90.9	9.6	91.21	52.03	-
xg_fea_grid_step1	91.64	82.99	92.04	49.42	91.64	70.33	-
xg_fea_grid_step2	90.87	81.47	91.29	49.98	90.84	70.78	max_deapth=2, min_child_weight=0
xg_fea_grid_step3	85.89	82.31	85.41	65.51	85.62	75.97	gamma=0
xg_fea_grid_step4	85.6	82.32	85.7	66.66	85.41	76.41	colsample_bytree=0.75, subsample=0.7, reg_alpha=0.0001
xg_fea_grid_step5	85.6	82.31	85.7	66.66	85.41	76.41	reg_lamba=0.05
xg_fea_grid_step6	100	74.14	100	13.69	100	55.72	learning_rate=0.5, n_estimators=3000

There is an increase in the output values, but it is not up to the industrial standards. So we have chosen to increase the probability of finding a particular set of output instead of getting just better results. One of the methods is by Bayesian optimization of the model.

4.4.5) GB with Bayesian Optimisation:

We'll use Bayesian optimization to try to discover the best values for max depth, learning rate, n estimators, and gamma, and compare the results to a model created with default settings.

Code :

xgb parameter tuning

xgb with Bayesian optmisation

```
import numpy as np
from xgboost import XGBClassifier
from bayes_opt import BayesianOptimization
from sklearn.model_selection import cross_val_score


pounds = {
    'learning_rate': (0.01, 1.0),
    'n_estimators': (200, 1000),
    'max_depth': (5,10),
    'subsample': (0.75, 1.0),  # Change for big datasets
    'colsample': (1.0, 2.0),  # Change for datasets with lots of features
    'gamma': (0, 5)}


def xgboost_hyper_param(learning_rate,
                        n_estimators,
                        max_depth,
                        subsample,
                        colsample,
                        gamma):


    max_depth = int(max_depth)
    n_estimators = int(n_estimators)


    clf = XGBClassifier(
        max_depth=max_depth,
        learning_rate=learning_rate,
        n_estimators=n_estimators,
        gamma=gamma)
    return np.mean(cross_val_score(clf, X_train_up, y_train_up, cv=3,
scoring='roc_auc'))
```

```
xgb_bo = BayesianOptimization(
    xgboost_hyper_param,
    {'learning_rate': (0.01, 1.0),
    'n_estimators': (100, 1000),
    'max_depth': (4,10),
    'subsample': (0.2, 1.0),  # Change for big datasets
    'colsample': (0.80, 1.0),  # Change for datasets with lots of features
    'gamma': (0,8)})
xgb_bo.maximize(init_points=3, n_iter=5, acq='ei')
```

Table 4.6: Description and Output for Xgboost with Bayesian Optimization

Measure	Train data	Test data
Accuracy	92%	84.09%
Recall	91.47%	6.2%
AUC Score	89.01%	51.8%

4.4.6) Logistic regression

The logistic function which is also known as the sigmoid function, was created by statisticians to characterize the characteristics of population increase in ecology, such as how it rises fast and eventually reaches the environment's carrying capacity. It's an S-shaped curve that can translate any real-valued integer to a value between 0 and 1, but never exactly between those two points.

f(x)=1 / (1 + e^-value) (Eq. 4.3)

Where e is the natural logarithms' base (Euler's number or the *exp()* function in your spreadsheet) and value is the numerical value to be transformed. The values between -5 and 5 have been converted into the range 0 and 1 using the logistic function, as shown below.

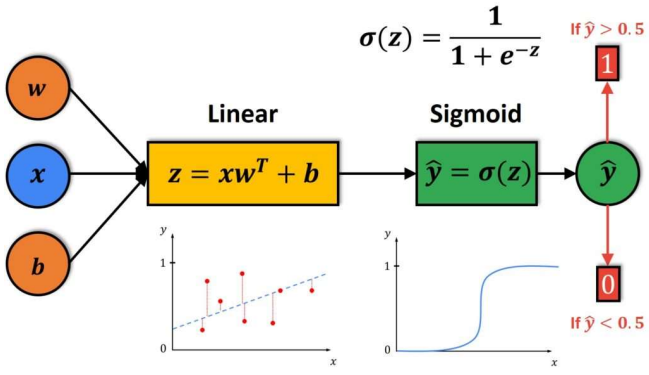

Fig 4.10: Logistic Regression

Code :

```
from sklearn.linear_model import LogisticRegression

#class sklearn.exceptions.FitFailedWarning

model_lgr = LogisticRegression(max_iter=100000,C=1, intercept_scaling=1,solver=
'newton-cg',

               dual=False, fit_intercept=True, penalty='l2', tol=0.0001)

model_lgr.fit(X_train_up, y_train_up)


test_pred_lgr = model_lgr.predict(X_train_up)

test_pred_val_lgr=model_lgr.predict(X_val_fea)

#cross val

scores_lgr = cross_val_score(model_lgr, X_train_up, y_train_up,

                             cv=5,

                             scoring='accuracy')

print('CV_accuracy:',scores_lgr)

print("Average accuracy score (across experiments):")

print(scores_lgr.mean())


#accuracies['Logistic Regression'] = accuracy_score(y_train_up, test_pred_lgr)

#recall['Logistic Regression'] = metrics.recall_score(y_train_up,test_pred_lgr)


print('Accuracy Score of Logistic Regression is: ', accuracy_score(y_train_up,
test_pred_lgr))

print('Recall Score of Logistic Regression Model is: ',
metrics.recall_score(y_train_up, test_pred_lgr))

# confusion matrix

matrix_lgr = confusion_matrix(y_train_up,test_pred_lgr, labels=[1,0])

print('Confusion matrix : \n',matrix_lgr)


# outcome values order in sklearn

tp_lgr, fn_lgr, fp_lgr, tn_lgr =
confusion_matrix(y_train_up,test_pred_lgr,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp_lgr, fn_lgr, fp_lgr, tn_lgr  )


# classification report for precision, recall f1-score and accuracy

matrix_cl_lgr = classification_report(y_train_up,test_pred_lgr,labels=[1,0])

print('Classification report : \n',matrix_cl_lgr)
```

```
#auc_score['Logistic Regression'] = roc_auc_score(y_train_up, test_pred_lgr)

print('Auc_Score:',roc_auc_score(y_train_up, test_pred_lgr))
```

Table 4.7: Description and Output for Logistic Regression

Measure	Train data	Test data
Accuracy	59.53%	55.17%
Recall	64.07%	58.19%
AUC Score	59.01%	56.29%

4.4.7) Neural net

A neural network is the group of algorithms or layers of algorithms that attempts to detect underlying relationships in a piece of data using a method that mimics how the human brain works. To reduce overfitting, we ran specific trails and numerous iterations to obtain the optimum combination of layers and the dropout layer.

Code :

```
METRICS = [

      keras.metrics.AUC(name='auc'),

]

model = Sequential()

model.add(Dense(64, input_dim=46, activation='relu'))

model.add(Dense(32, activation='relu'))

#model.add(Dropout(0.2))

model.add(Dense(16, activation='relu',kernel_regularizer=regularizers.l1_l2(l1=1e-
5, l2=1e-4),bias_regularizer=regularizers.l2(1e-
4),activity_regularizer=regularizers.l2(1e-5)))

model.add(Dense(16, activation='relu',kernel_regularizer=regularizers.l1_l2(l1=1e-
5, l2=1e-4),bias_regularizer=regularizers.l2(1e-
4),activity_regularizer=regularizers.l2(1e-5)))

#model.add(Dropout(0.2))

model.add(Dense(8, activation='relu',kernel_regularizer=regularizers.l1_l2(l1=1e-5,
l2=1e-4),bias_regularizer=regularizers.l2(1e-
4),activity_regularizer=regularizers.l2(1e-5)))

model.add(Dense(4, activation='relu',kernel_regularizer=regularizers.l1_l2(l1=1e-5,
l2=1e-4),bias_regularizer=regularizers.l2(1e-
4),activity_regularizer=regularizers.l2(1e-5)))

model.add(Dense(1, activation='sigmoid'))

# compile the keras model

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=METRICS)

model.summary()
```

```
loss, auc = model.evaluate(X_train_up, y_train_up)
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['auc'])
plt.plot(history.history['val_auc'])
plt.title('model auc')
plt.ylabel('auc')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# evaluate the keras model
```

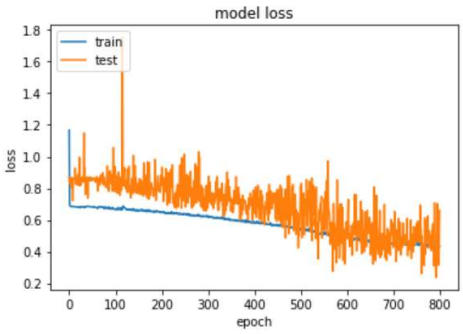

Fig 4.11 (A): Model Loss

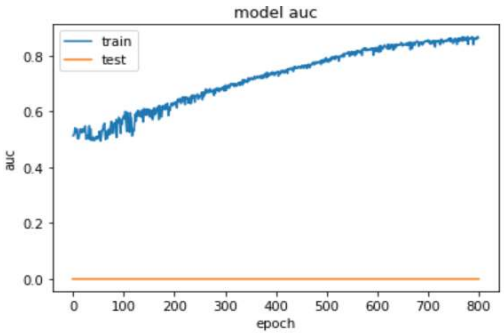

Fig 4.11 (B): Model AUC Score

Table 4.8: Description and Output for Neural Networks

Measure	Train data	Test data
AUC Score	87.25%	59.1%
Loss	49.75%	53.6%

4.4.8) Stacking Classifier

The simplest form of stacking can be described as an ensemble learning technique where the predictions of multiple classifiers (referred to as level-one classifiers) are used as new features to train a meta-classifier. The meta-classifier can be any classifier of your choice. The figure shows how three different classifiers get trained. Their predictions get stacked and are used as features to train the meta-classifier which makes the final prediction.

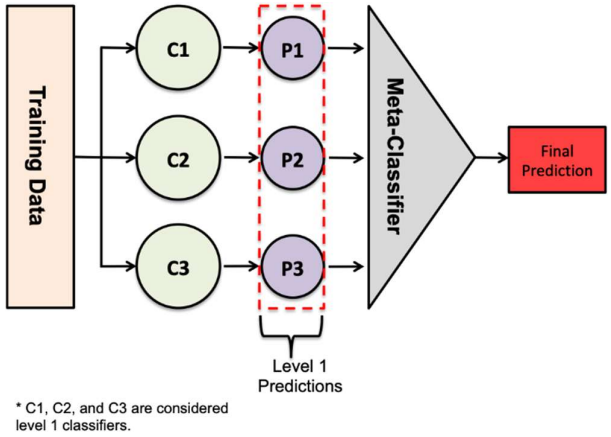

Fig 4.12: Stacking Classifier

In the data, the level one classifiers are random forest and logistic regression, and the meta classifier is the XGBosst algorithm which neutralizes the effect of both underfitting and overfitting.

Code:

```
estimators = [
    ('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
    #('xb',)
    ('lr',LogisticRegression(max_iter=100000,C=1, intercept_scaling=1,solver='newton-cg',
                dual=False, fit_intercept=True, penalty='l2', tol=0.0001))
    #('svc', make_pipeline(StandardScaler(),SVC(random_state=42)))
]
clf = StackingClassifier(
    estimators=estimators, final_estimator= XGBClassifier(learning_rate =0.3,
 n_estimators=2000,
 max_depth=9,
 min_child_weight=5,
 gamma=0.4,
 subsample=0.75,
 colsample_bytree=0.75,
```

```
 reg_alpha= 0.0001,
 reg_lambda=1,
 objective= 'binary:logistic')
)

clf.fit(X_train_up, y_train_up)
clf_pred=clf.predict(X_val_fea)
clf_pred_train = clf.predict(X_train_up)
print('Final prediction score: [%.8f]' % accuracy_score(y_train_up,
clf_pred_train))
print('Final prediction score: [%.8f]' % accuracy_score(y_val, clf_pred))
matrix_val_clf = confusion_matrix(y_val,clf_pred, labels=[1,0])
print('Confusion matrix : \n',matrix_val_clf)
```

Table 4.9: Description and Output for Stacking Classifier

Measure	Train data	Test data
AUC Score	85.17%	71.98%

4.5) Modeling to Results:

Accuracy, Recall, and Area under Curve(AUC) are the prominent and reliable metrics to estimate the performance of the given model in the academic standards. Nevertheless, for the project of Uplift Modelling, we have to choose the pattern of probabilities of the quantiles as the metric for the estimation of any given model. So in our work, after attaining the model results apart relying from on standard metrics, we also convert the given results as quantile of probabilities and check for the results. We can divide the process into two essential steps, mainly finding the probabilities and converting them into quantiles. The below is the code for finding probabilities after obtaining the result by standard metrics.

Step1: Predict Probabilities

```
prob01_nn_total=model.predict(X_val_fea)
predictions = model.predict_classes(X_val_fea)
flatten_predictions =np.ndarray.flatten(predictions)
flatten_prob =np.ndarray.flatten(prob01_nn_total)
```

```
#flatten_prob =np.asarray([item for subl in prob01_nn_total for item
in subl])

print(len(flatten_prob))

print(flatten_prob)

#flatten_prob

print(flatten_predictions)

print(len(flatten_predictions))

df1 = pd.DataFrame({'ar1':flatten_prob})

df2 = pd.DataFrame({'ar2':flatten_predictions})

pd.concat([df1.ar1, df2.ar2], axis=1)
```

Step2: Quantile division

After obtaining the desired binary classification probabilities, we divide them into 5 quantiles in the order of their respective true and false probabilities.

Table 4.10: Quantile Division (A>B>C>D>E)

Quantile	Probability Range	Lift and RR
1	>0.80	A
2	0.80>x>0.60	B
3	0.60>x>0.40	C
4	0.40>x>0.20	D
5	x<0.20	E

```
pred_nn_1 = pred_nn[pred_nn.treatment == 1]

pred_nn_0 = pred_nn[pred_nn.treatment == 0]

pred_nn['pred_resp'].value_counts()

pred_nn_sort=pred_nn.sort_values(by=['prob_scores'], ascending=False)

pred_nn_1_sort=pred_nn_1.sort_values(by=['prob_scores'], ascending=False)

pred_nn_0_sort=pred_nn_0.sort_values(by=['prob_scores'], ascending=False)

pred_nn_sort['QuantileRank']= pd.qcut(pred_nn_sort['prob_scores'],
                              q = 5, labels = False)

pred_nn_1_sort['QuantileRank']= pd.qcut(pred_nn_1_sort['prob_scores'],
                              q = 5, labels = False)

pred_nn_0_sort['QuantileRank']= pd.qcut(pred_nn_0_sort['prob_scores'],
                              q = 5, labels = False)

pred_nn_sort.head()

total = pred_nn_sort[['pred_resp','QuantileRank']]

promo = pred_nn_1_sort[['pred_resp','QuantileRank']]
```

```
ho = pred_nn_0_sort[['pred_resp','QuantileRank']]

total_count_by_quantile =
total.groupby('QuantileRank').agg({'pred_resp':('sum','count')}).iloc[::-1]

promo_count_by_quantile =
promo.groupby('QuantileRank').agg({'pred_resp':['sum','count']}).iloc[::-1]

ho_count_by_quantile =
ho.groupby('QuantileRank').agg({'pred_resp':['sum','count']}).iloc[::-1]

uplift_table = pd.concat([total_count_by_quantile, promo_count_by_quantile,
ho_count_by_quantile], axis=1)

uplift_table
```

Below tables (4.11 - A, B, C, D) are the quantile wise probability distributions of true positives in all the algorithms:

Table 4.11 (A): XgBoost (After Hyperparameter Tuning)

Quantile Rank	Total			Promo			HO			IRR	Lift
	count	resp	RR	count	resp	RR	count	resp	RR		
1	399	1	0.0025	199	1	0.005	199	0	0	0.503	#DIV/0!
2	398	6	0.0151	199	3	0.0151	199	3	0.015	0	100
3	398	26	0.0653	199	11	0.0553	199	15	0.075	-2.01	73.333
4	398	68	0.1709	199	34	0.1709	199	33	0.166	0.503	103.03
5	399	131	0.3283	200	72	0.36	200	60	0.3	6	120

Table 4.11 (B): XgBoost (After Bayesian Optimization)

Quantile Rank	Total			Promo			HO			IRR	Lift
	count	resp	RR	count	resp	RR	count	resp	RR		
1	399	1	0.0025	199	1	0.005	199	0	0	0.503	#DIV/0!
2	398	5	0.0126	199	3	0.0151	199	2	0.01	0.503	150
3	398	19	0.0477	199	11	0.0553	199	8	0.04	1.508	137.5
4	398	71	0.1784	199	38	0.191	199	34	0.171	2.01	111.76
5	399	135	0.3383	200	72	0.36	200	62	0.31	5	118.13

Table 4.11 (C): Neural Network

Quantile Rank	Total			Promo			HO			IRR	Lift
	count	resp	RR	count	resp	RR	count	resp	RR		
1	399	47	0.117794	199	27	0.135678	199	20	0.100503	3.517588	135
2	398	56	0.140704	199	21	0.105528	199	36	0.180905	-7.53769	58.33333
3	398	207	0.520101	199	92	0.462312	199	116	0.582915	-12.0603	79.31034
4	398	238	0.59799	199	124	0.623116	199	110	0.552764	7.035176	112.7273
5	399	205	0.513784	200	99	0.495	200	108	0.54	-4.5	91.66667

Table 4.11 (D): XgBoost (Before Tuning)											
Quantile Rank	Total			Promo			HO			IRR	Lift
	count	resp	RR	count	resp	RR	count	resp	RR		
1	399	2	0.005013	199	1	0.005025	199	1	0.005025	0	100
2	398	2	0.005025	199	2	0.01005	199	0	0	1.005025	#DIV/0!
3	398	4	0.01005	199	1	0.005025	199	3	0.015075	-1.00503	33.33333
4	398	15	0.037688	199	4	0.020101	199	9	0.045226	-2.51256	44.44444
5	399	97	0.243108	200	46	0.23	200	53	0.265	-3.5	86.79245

The Lift is decreasing from the first quantile to $^{\text{the fifth}}$ quantile, but there wasn't stability over the distribution. There should not be any lift less than 100, which signifies the importance of HO performance than Promo performance. But we see instances as HO>Promo. So the above models were rejected on this basis

CHAPTER-5 RESULTS

In this section, the practical implementation of the project is presented and described along the Stacking Classifier and Data implementation on the business. Here the Model has performed well on the test data set. Quantile one has a lift of 183, and quantile 5 has a lift of 100, which is with the least, and the quantiles show a decreasing trend with the second quantile has the highest Lift, which benefits by targeting the responders of quantile 2 and 1 by the business.

Table 5.1: Stacking Classifier

Quantile Rank	Total			Promo			HO			IRR	Lift
	count	resp	RR	count	resp	RR	count	resp	RR		
1	399	17	0.042607	199	11	0.055276	199	6	0.030151	2.512563	183.3333
2	398	11	0.027638	199	8	0.040201	198	3	0.015152	2.504949	265.3266
3	398	19	0.047739	199	12	0.060302	200	7	0.035	2.530151	172.29
4	398	67	0.168342	199	33	0.165829	199	32	0.160804	0.502513	103.125
5	399	136	0.340852	200	69	0.345	200	69	0.345	0	100

This data is for a particular group of months and combined with the data source of the Ugam of US, and a Model has been built by us to select the best responders from the whole population of US. This a data of 10,000 as a sample from the population. This model that has been built will be used in the third quarter by the onboarded team.

CHAPTER-6 CONCLUSIONS & FUTURE SCOPE

The general conclusion is that each model required several trials for all data sets in order to capture the model parameters that gave the best model results. The models performed badly in general, however some were able to produce satisfactory results. The results and ideas for further study will be addressed in the sections that follow.

6.1) Discussion

When utilizing Random Forests to create uplift models, it wasn't always feasible to get models that outperformed a random classifier, which indicates that in certain situations, taking action is worse than doing nothing. One of the parameters in the upliftRF() function in the uplift [6] package that sets the minimum number of control observations that must exist in every terminal node is one of the reasons for this. As a result, the tree is required to contain control data in every area during the construction process.

Both Logistic Regression and Neural Networks were successful in capturing excellent models for both treatment and control data. This is evident in the Roc curve, which has curves that are high above the diagonal line and AUC values that are excellent, i.e., more than or equal to 0.5. This indicates that the models work well with data that hasn't been seen before (test data). The main problem for Neural Networks is that it needs balanced classes to perform well. When doing the Class Variable Transformation, it is not easy to obtain training data with balanced classes.

We've seen the results of all the models that have been tested in the previous sections. We notice that the tree algorithms have overfitted the data while the bagging algorithms and probability modelling. We could not exceed the recall by 15% on the test set. It has been the same with the mathematical logistic function. However, we have noticed that Logistic regression has under fitted on the data set. Sci-kit learn has a function called Stacking classifier, which can club multiple models to build a singular model. We have taken the three most well-performed models into the stacking classifier. The meta-model is the stacking classifier is the most important as it is the final model set to parameters on. So XgBoost has been selected after multiple trails, and logistic regression is chosen as a primary model, and Random forest is used to build it on to boost the values of the parameters to meet industrial standards.

6.2) Future Enhancement

The market evaluated in this project was for only one country, i.e., the USA. It would have been interesting if we were able to evaluate whether different markets differ from each other. It might be that customers in different markets react differently to marketing campaigns and even differently to different kinds of campaigns. Thus, future work could investigate if the marketing campaign should be of a different kind depending on what market is targeted. This could lead to happier and more loyal customers and an uplift for the company in terms of a more significant gain in selling.

The overall conclusion is that it is possible to perform uplift modeling to obtain models that make it possible to comprehend how to target only a subgroup of the entire customer base instead of targeting the whole customer base with campaign offers, given the data related to the different campaigns in this project. By doing this, the retail company still receives an incremental gain. For the uplift to be successful, the method of choice should be either the Modeling Uplift Directly approach using Random Forests, or the Class Variable Transformation using Logistic Regression. This is because Neural Networks are sensitive to uneven class distributions and are thus unable to obtain stable models given the data in this project.

REFERENCES

[1] Jiliang Tang Salem Alelyani, Huan Liu. “Feature Selection for Classification: A Review." In: Arizona state university 2014 (Jan. 2014), p. 1.

[2] Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. The Elements of Statistical Learning. Data Mining, Inference, and Prediction. 2nd ed. Springer Series in Statistics. Springer, 2008.

[3] Montgomery, Douglas C., Peck, Elizabeth A., and Vining, G. Geoffrey. Introduction to Linear Regression Analysis. Fifth Edition. Wiley Series in Probability and Statistics. Wiley, 2012.

[4] https://www.steveklosterman.com/uplift-modeling/

[5] Rzepakowski, Piotr and Jaroszewicz, Szymon. "Uplift modeling in direct marketing." In: Journal of Telecommunications and Information Technology 2012 (Jan. 2012), pp. 43–50

[6] Guelman, Leo. "uplift: Uplift Modeling." In: (2014). URL: https://CRAN.R-project.org/package=uplift.

[7] Liaw, Andy and Wiener, Matthew. "Classification and Regression by randomForest." In: R News 2.3 (2002), pp. 18–22. URL: https://CRAN.R-project.org/doc/Rnews/.

[8] https://www.kaggle.com/btyuhas/bayesian-optimization-with-xgboost

[9] https://www.kaggle.com/huntermcgushion/bayesian-hyperparameter-optimization-and-xgboost

[10] https://aiinpractice.com/xgboost-hyperparameter-tuning-with-bayesian-optimization/

PLAGIARISM CERTIFICATE

grammarly

Report: UPLIFT MODEL FOR CUSTOMER PROPENSITY MODELING

UPLIFT MODEL FOR CUSTOMER PROPENSITY MODELING

by Sai Raghu Teja Davuluri

General metrics

68,576	9,484	1069	37 min 56 sec	1 hr 12 min
characters	words	sentences	reading time	speaking time

Writing Issues

No issues found

Plagiarism

This text seems 100% original. Grammarly found no matching text on the Internet or in ProQuest's databases.

Report was generated on Monday, Aug 9, 2021, 08:34 PM

Page 1 of 90

PROJECT DETAILS

Student Details			
Student Name	Sai Raghu Teja Davuluri		
Register Number	170907024	Section / Roll No	D / 11
Email Address	teja12899@gmail.com	Phone No (M)	8639123779
Project Details			
Project Title	Uplift Model		
Project Duration	6 months	Date of reporting	02 March 2021
Organization Details			
Organization Name	Ugam Solutions Pvt. Ltd.		
Full postal address with pin code	Ground floor, H2 Block, Mountain Ash, Manayata Tech Park, Nagavara, Bengaluru, Karnataka 560045, India		
Website address	https://www.ugamsolutions.com/		
External Guide Details			
Name of Guide	Aakash Sharma		
Designation	Senior Manager, AS Dept		
Full contact address with pin code	Ugam Solutions Private Limited | Registered Office: Ground Floor, H2 Block, Mountain Ash, Manyata Tech Park, Nagawara, Bangalore 560045, India		
Email address	aakash.sharma@ugamsolutions.com	Phone No (M)	

Internal Guide Details	
Faculty Name	Dr. Goutham Simha GD
Full contact address with pin code	Dept of Electronics & Communication Engg, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA
Email address	goutham.simha@manipal.edu