

Final Report

UPLIFT MODEL FOR CUSTOMER PROPENSITY MODELING

SUBMITTED

BY

SAI RAGHU TEJA DAVULURI 170907024

Shape, arrow

Description automatically generated Under the Guidance of:

Dr. Goutham Simha GD Mr. AAKASH SHARMA

Assistant Professor Sr. Manager

Department of ECE AS Dept

Manipal Institute of Technology Ugam Solutions Pvt. Ltd.

|  |
| --- |
| **A screenshot of a cell phone  Description generated with very high confidenceDEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**  MANIPAL-576104, KARNATAKA, INDIA |

Manipal

< Date >

**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 Mr. AAKASH SHARMA

Assistant Professor Head of Department

Department of ECE AS Dept

Manipal Institute of Technology Ugam Solutions Pvt. Ltd.

CERTIFICATE

ACKNOWLEDGEMENT

I have tried to put in sincere effort in developing the concept of the project & completing the same. However, it wouldn’t have been possible without the constant support & guidance by Mr. Aakash Sharma, who supervised my work with extra care. I would like to express my special thanks of gratitude to all the people who helped for finalizing the project within the time frame.

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 to complete the project.

**ABSTRACT**

The desire to model the true gain from targeting an individual in marketing purposes has led to the common use of uplift modelling. Efficient methods for estimating the probabilities in uplift models are statistical machine learning methods. The statistical machine learning methods applied are Random Forests along with the standard method Logistic Regression.

The data is collected from a well-established retail company and the purpose of the project is thus to investigate which uplift modelling approach and statistical machine learning method that yields in the best performance given the data used in this project. The variable selection step was shown to be a crucial component in the modelling processes as so was the amount of control data in each data set. For the uplift to be successful, the method of choice should be either model it 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 tended to focus too much on modelling the class in both data sets separately instead of modelling the difference between the class probabilities.

The conclusion is hence to use an approach that models the uplift directly and to use a great amount of control data in the data sets.

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Title** | **Page No.** |
| 2.1 | Treatment to be given to particular use case according to the target outcome. | 4 |
| 3.1 | Dictionary of the dataset provided by the client | 7 |
| 4.1 | Null values in the data set (before & after treatment |  |
| 4.2 | Categorical Columns Conversion |  |
| 4.3 | Description & output for Random Forest |  |
| 4.4 | Description & output for Xgboost |  |
| 4.5 | Description & output for Xgboost with Gradient Search CV |  |
| 4.6 | Description & output for Xgboost with Bayesian Optimization |  |
| 4.7 | Description & output for Logistic Regression |  |
| 4.8 | Description & output for Neural Networks |  |
| 4.9 | Description & output for Stacking Classifier |  |
| 4.10 | Quantile Division |  |
| 4.11 | Quantile wise probability distributions of true positives |  |
|  | (A) XGBoost (After Hyperparameter Tuning) |  |
|  | (B) XGBoost (After Bayesian Optimization) |  |
|  | (C) Neural Networks |  |
|  | (D) XGBoost (Before Tuning) |  |
| 5.1 | Quantile wise probability distributions of true positives in Stacking Classifier Algorithms |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Title** | **Page No.** |
| 2.1 | Outcomes in an Uplift Model | 5 |
| 3.1 | Detailed description of the dataset |  |
| 4.1 | Boxplot Distribution |  |
| 4.2 | (A) *Age* Boxplot (Before IQR) |  |
|  | (B) *Age* Boxplot (After IQR) |  |
| 4.3 | (A) *Vehicle\_cnt\_select* Boxplot (Before IQR) |  |
|  | (B) *Vehicle\_cnt\_select* Boxplot (After IQR) |  |
| 4.4 | (A) *Driver\_cnt\_select* Boxplot (Before IQR) |  |
|  | (B) *Driver\_cnt\_select* Boxplot (After IQR) |  |
| 4.5 | Correlation matrix between the variables.(Heatmap) |  |
| 4.6 | Churn Analysis |  |
| 4.7 | Random Forest |  |
| 4.8 | Cross Validation Technique |  |
| 4.9 | XGBoost |  |
| 4.10 | Logistic Regression |  |
| 4.11 | (A) Model Loss |  |
|  | (B) Model AUC Score |  |
| 4.12 | Stacking Classifier |  |

TABLE OF CONTENTS

Contents Page No.

Acknowledgement i

Abstract ii

List of Tables iii

List of Figures iv

1. Introduction
   1. General 1
   2. Motivation 1
   3. Problem Definition 2
   4. Organization of Report 2
   5. Software Requirements 3
2. Background Theory 4
3. Dataset Description
   1. Markets and Campaigns 7
   2. Variables 7
4. Methodology
   1. Data Collection
   2. Pre-processing

4.2.1. Missing value Treatment

4.2.2. Categorical Value Treatment

4.2.3. Outlier Treatment

4.2.4. Feature Engineering

* 1. Exploratory Data Analysis
  2. Modelling

4.4.1. Random Forest

4.4.2. Grid Search

4.4.3. XGBoost

4.4.4. XGBoost with Gradient Search CV

4.4.5. XGBoost with Bayesian Optimization

4.4.6. Logistic Regression

4.4.7. Neural Net

4.4.8. Stacking Classifier

* 1. Modelling to Results

1. Result
2. Conclusion and Future Scope
   1. Work Conclusion 4
   2. Future Scope of Work 4

References

Project Detail

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. *General*

In retail and marketing, predictive modeling is a common tool used for targeting and evaluating the response from individuals when an action is taken on. The action is normally referred to a campaign/offer that is sent out to the customers and the response to model is the likelihood that a specific customer will act on the offer. Putting it in a different way, the objective is to predict the conditional class probability:

P(Y = 1|X = x) (Eq 1.1)

where the response Y ∈ {0, 1} reflects whether a customer gave a positive response (i.e., made a purchase) to an action or not. X = (X1, ..., Xp) are the quantitative and qualitative attributes of the customer and ‘x’ is one observation.

Using traditional response modeling, the resulting classifier can then be used to select what customers to be targeted when sending out campaigns/offers in a marketing purpose. In real life scenario, this is not always the desirable approach to use since the targeted customers are those who are most likely to react positively to the offer after the offer has been sent out. As a result, a second order approach is to be used which is recognized as uplift modeling.

* 1. *Motivation*

There is a problem that arises when working with uplift modeling, i.e., using one treatment group and one control group. For every individual in the experiment, only one outcome can be observed. Either the individual belongs to the treatment group or the control group. One individual can never belong to both groups. Put differently, it is not possible to know for sure that there is a causal connection that the customer in treatment group responds because of the treatment since the same customer cannot be in the control group at the same time. Thus, it is not possible to evaluate the decisions at the individual observational unit as is possible in for example classification problems where the class of the individual is known. This in turn makes it a bit trickier when evaluating uplift models.

Furthermore, uplift modeling has not yet been tested on the data used in this project, or on similar data belonging to the company owning the data. Thus, it isn’t clear if it is even possible to apply the uplift modeling technique on this data and to obtain desired results.

As a result, the question to be answered is how to optimize customer targeting in the marketing domain by using the uplift modeling approach, and at the same time being able to model the true gain from targeting one specific individual.

* 1. *Problem Definition*

An assumption that Direct Marketing campaign has that it will achieve maximum incremental Response when a group of the highest scored customers is targeted. A Propensity/Response model itself is not going to tell marketers which customers are most likely to contribute to the incremental campaign response. An alternative statistical model is therefore needed to target the customers whose propensities of response are dramatically driven by “touching” customers with a promotion.

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 true gain from targeting one specific individual. Furthermore, how should the uplift modelling technique be implemented in the best way to obtain the most applicable results given this kind of data?

* 1. *Organization of Report*

In chapter 2, the idea behind uplift modeling is explained along with some related work that has already been made in the area. Chapter 3 contains description of the data, i.e., statistics of the different campaigns that are used and the type of variables collected in the different data sets. The variables are listed in a table where no variable is excluded, meaning that the table contains the list of all the variables that are used before any kind of variable selection is made.

The description of the data is followed by chapter 4 which contains all the theory related to the project. Here, a theoretical description of how to pre-process data is presented along with some theory of variable selection. Furthermore, the three different approaches for uplift modeling, as well as the statistical machine learning methods that are used to perform uplift modeling is described. The uplift modeling approach used in this thesis is Two-Model approach. The statistical machine learning methods used for uplift modeling are Random Forest (without & with Tuning), Xgboost (without & with Tuning), Logistic Regression, Stacking Classifier and Neural Networks. Moreover, a description of the resampling method Cross Validation is presented. The evaluation metrics that are used in this project are also presented, namely Receiver Operating Characteristic Curves and Qini curves.

In chapter 5, all the experimental results are presented. Firstly, the results from the pre-processing of the data are presented. Secondly, each implementation is described along with tables and figures of the results of the best performing model(s). The report ends with chapter 6 where we reach to conclusions based on the results obtained.

* 1. *Software Requirements*

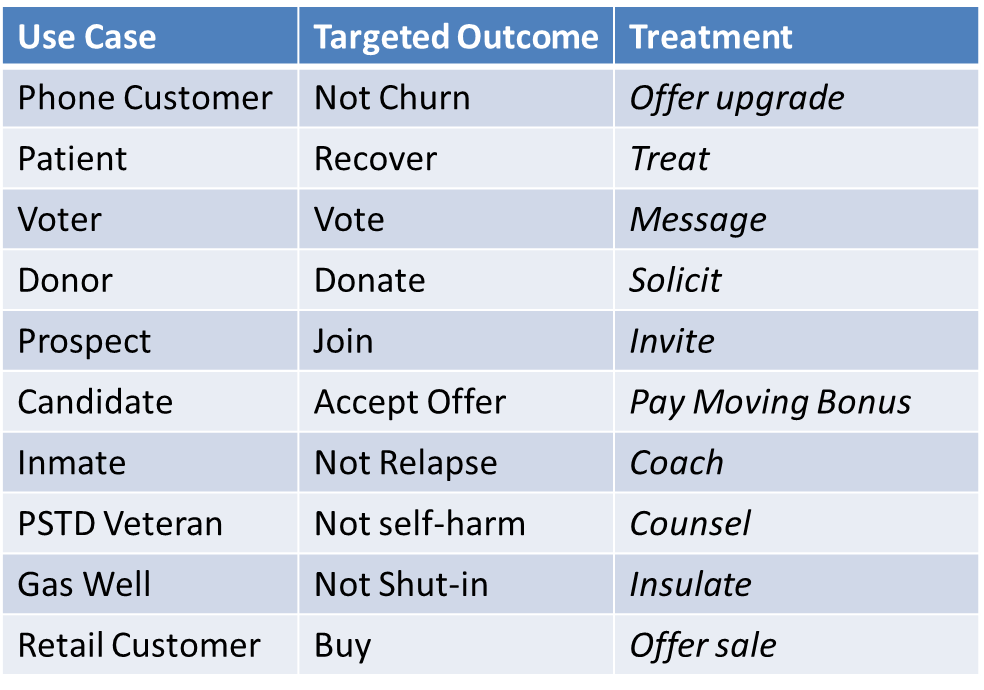
The model is developed using Python 3.6. In this, essential libraries used for classification & regression are NumPy, pandas, pylift, matplotlib (for visualization), scikit learn (for splitting and standardizing training & testing set). The IDE on which we will build the model is Jupyter Notebook.

CHAPTER-2 BACKGROUND THEORY

Machine learning is a branch in computer science and statistics which often aims to, given some attributes, classify a specific instance into some category, or the conditional probability that it belongs to each of the classes. This technique can be used in a lot of areas, with one of them being marketing. In reality, this classification technique is not really well suited for marketing. For instance, consider a marketing campaign where an offer is sent out to a (randomly) selected subgroup of potential customers. Then using the results of the actions taken from the customers, a classifier can be built on top of it. Thus, the resulting classifier is used to select which customers to send the campaign to. The result will be that the customers who are most likely to react positively to the offer after the campaign has been sent out, will be used as targets. This is not desirable for the marketer.

Using predictive analytics, information is extracted from existing dataset for determining patterns and predicting the forthcoming trends or outcomes. It uses data, statistical algorithms, and machine learning techniques to identify the likelihood of future outcomes based on historical data.[3] There are many applications of predictive modeling where the outcome is predicted as advice only to a human decision maker, and no action is directly taken automatically from the model result.  An example is workload prioritization.  For example, in the telecom industry we can predict which customers are most likely to churn (cancel their contracts).  In healthcare we can predict which patients are most likely to recover. For universities or charitable organizations, we can predict which prospective benefactors are most likely to donate. [10]

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



Some customers would have made a purchase whether they were targeted with the campaign or not, and thus unnecessary expenses are wasted in the case of sending the offer to this kind of customer. Then there are customers who actually react in a negative way by getting a campaign offer. Some might find it disturbing to receive campaign offers from the company in question, or stop being a customer out of some other reason just because they received the offer. When a customer stops doing business with a company it is called customer churn. Customer churn is something that the company in question really wants to avoid. In other words, this is not a customer the marketer wants to target since it is an unnecessary expense to send out the campaign in this case, and they needlessly lose a customer.

The first kind of customer is called a Sure Thing and the second one is commonly mentioned as a Do-Not-Disturb. Then there are two more categories of customers, namely the Lost Cause and the Persuadable. You can tell by the name that the lost cause is someone who would react negatively, i.e., would not make any purchase at all, whether they were targeted or not. Both the sure things and the lost causes are considered as a waste of money to give treatment to, 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 customer that the marketer wants to find and target. This kind of customer is a person who would not have made any purchase if they would not have received the campaign offer, but who would make a purchase if they did. These are the kind of customers the marketer can affect in a positive direction. An overview of the different type of customers can be seen in Fig 2.1.

Diagram

Description automatically generated

Fig 2.1: Outcomes in an Uplift Model

The solution to this kind of problem is called Uplift Modeling. The original idea behind uplift modeling is to use two separate training sets, namely one data set containing the control group and one containing the treatment group. The control group contains the customers who were not targeted by the campaign, and the treatment group contains the customers who received the campaign. The explicit goal in uplift modeling is to model the conditional average treatment effect. The conditional average treatment effect or uplift, estimate the increase of purchase probability given that a customer receives treatment compared to if no treatment is given. By being able to identify which customers who are more likely to purchase before treatment is given, would ideally let a company target a smaller part of the sample and thus reduce marketing costs meanwhile they maintain or even increase their earnings. The uplift model returns a score for each customer, where a higher score means a higher chance of 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 modeling is already applied frequently in the marketing domain, although it has not received as much attention in the literature as one might believe.

The decision tree-based models are also compared to simpler standard response-based models such that three uplift models and three standard response models are used in total [5]. The data that is modeled on reflects the costumers of a retail company. The goal is thereby to classify costumers as persuadable where the response reflects whether they go to the retail company’s website or not because of the campaign. The result of the study is that they find it possible and more effective to use uplift modeling than response models to predict the persuadable, i.e., which costumers are having a positive response to the campaigns. The standard response models were good at predicting if a costumer were going to the website or not but performed very badly in predicting if they responded to the campaign(s) or not. Hence, this is the reason why this project will mainly focus on comparing different approaches for uplift modeling, and not to include traditional response or purchase models since they have been proven to perform worse in many cases.

CHAPTER-3 DATASET DESCRIPTION

The data used in this project is collected from a well-established retail company that has physical stores as well as a website where the customers can make orders online.

In the following subsections, the markets and campaigns used in the project will be presented along with a table of descriptions 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 with encouraging frequent customers for more purchases. The focus will be on this category of customers since uplift modeling is used here, so that campaigns will mainly be sent 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 a consistency when it comes to what kind of campaign that is used in the methods of this project.

One specific market sector of the company is chosen for this project, and all the data used in the uplift models will be generated from this market.

*3.2) Variables*

Following is a table of all the variables used in the data set before variable selection is made. Each row is related to response of a customer and each column in the data set contains all the different 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 customer |
| resp | did the customer respond |
| conv | did the customer convert |
| region | region |
| division | division |
| cancel\_reason\_bucket | Policy cancelation reason |
| annual\_premium\_select | 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 Internet |
| pif\_risk\_lvl | risk level of customer B - preferred C - Non-Preferred D – Reject |

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

A picture containing text, newspaper, document, receipt

Description automatically generated

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 modeling is a data mining/predictive modeling technique that directly models the incremental impact of a treatment on an individual’s behavior. There are three overall approaches that exists 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 approach is to model the uplift directly by using a conditional divergence measure as splitting criterion in a tree-based method. The third approach is to use a Class Variable Transformation that allows for a conversion of an arbitrary probabilistic classification model into a model that predicts uplift directly. In this paper we will be using first approach. This will be the underlying model for constructing the statistical machine learning methods that can be used as a classification problem. In this case when using a statistical machine learning method w kith, the purpose to apply it in an uplift modeling setting, suitable models are Logistic Regression, Random Forests and Multilayer Perceptron (Neural Networks) as these performs binary classification.

The following sections will hence include the theoretical background for data pre-processing, uplift modeling, classification, and evaluation metrics. Finally, the different programming environments of choice are presented along with some arguments of their compatibility with the data and statistical machine learning methods used in this project.

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 shows whether a customer has made a responded or converted during a particular campaign period or not.

Each data set used in this project corresponds to one specific campaign each, hence one customer can occur in several data sets. There is one variable that describes whether a customer belongs to the control group or the treatment group. Customers belonging to the control group are customers who did not receive any campaign offer, while customers belonging to the treatment group did receive the offer.

A sample of the whole data set is taken with a records of 10,000 and of 19 variables comprising 11 categorical variables and 8 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 would 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 have a 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 certain 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 data bases. The data base 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 values at certain blocks which are called as missing values thus it is of a 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 true population. There are several ways to treat missing data. Some of the majorly used method is to delete rows or columns, imputing with mean, median, mode. Imputation methods are in these conditions dangerous.

On the other hand, if the amount of missing data is large enough in size compared to the entire dataset, a way to handle missing values is to use some sophisticated algorithm such as EM-algorithm or Multiple Imputations. In our project we have chosen is KNN Imputer. The idea in kNN Imputer is to identify ‘k’ samples in the dataset that are similar or close in the space. Then we use these ‘k’ samples to estimate the value of the missing data points. Each sample’s missing values are imputed using the mean value of the ‘k’-neighbors found in the dataset.

In the presence of missing coordinates, the Euclidean distance is calculated by ignoring the missing values and scaling up the weight of the non-missing coordinates.

(Eq. 4.1)

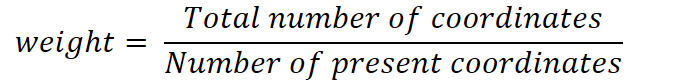
 (Eq. 4.2)

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 the mathematics involved in it. So, the categorical variables in the data has to be treated by certain methods and should be converted into numerical values. Most used are Label Encoding and Dummy Coding.

* Label Encoder: It is used to transform non-numerical labels to numerical labels (or nominal categorical variables). Numerical labels are always between 0 and n classes .
* Dummy Coding: Dummy coding is a commonly used method for converting a categorical input variable into continuous variable. ‘Dummy’, as the name suggests is a duplicate variable which represents one level of a categorical variable. Presence of a level is representing by 1 and absence is represented by 0. For every level present, one dummy variable will be created.

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 |

* + 1. *Outlier Treatment:*

There can be values for each feature which might fall beyond the Least and highest Value possible for a normal distribution graph. Outliers increase the variability in your data, which decreases statistical power. Consequently, excluding outliers can cause your results to become statistically significant. In general, the values above 1.5\*iqr + 0.75 quantile and below 0.25 quantile-1.5\*iqr of the feature is taken as outliers.

Chart, histogram

Description automatically generated with medium confidence

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 these are removed by the process of IQR method. 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.

Chart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

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

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

Chart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

Fig 4.3 (A) : *Vehicle\_cnt\_select* Boxplot (Before IQR)

Fig 4.3 (B) : *Vehicle\_cnt\_select* Boxplot (After IQR)

Chart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

Fig 4.4 (B) : *Driver\_cnt\_select* Boxplot (After IQR)

Fig 4.4 (A) : *Driver\_cnt\_select* Boxplot (Before IQR)

* + 1. *Feature Engineering*

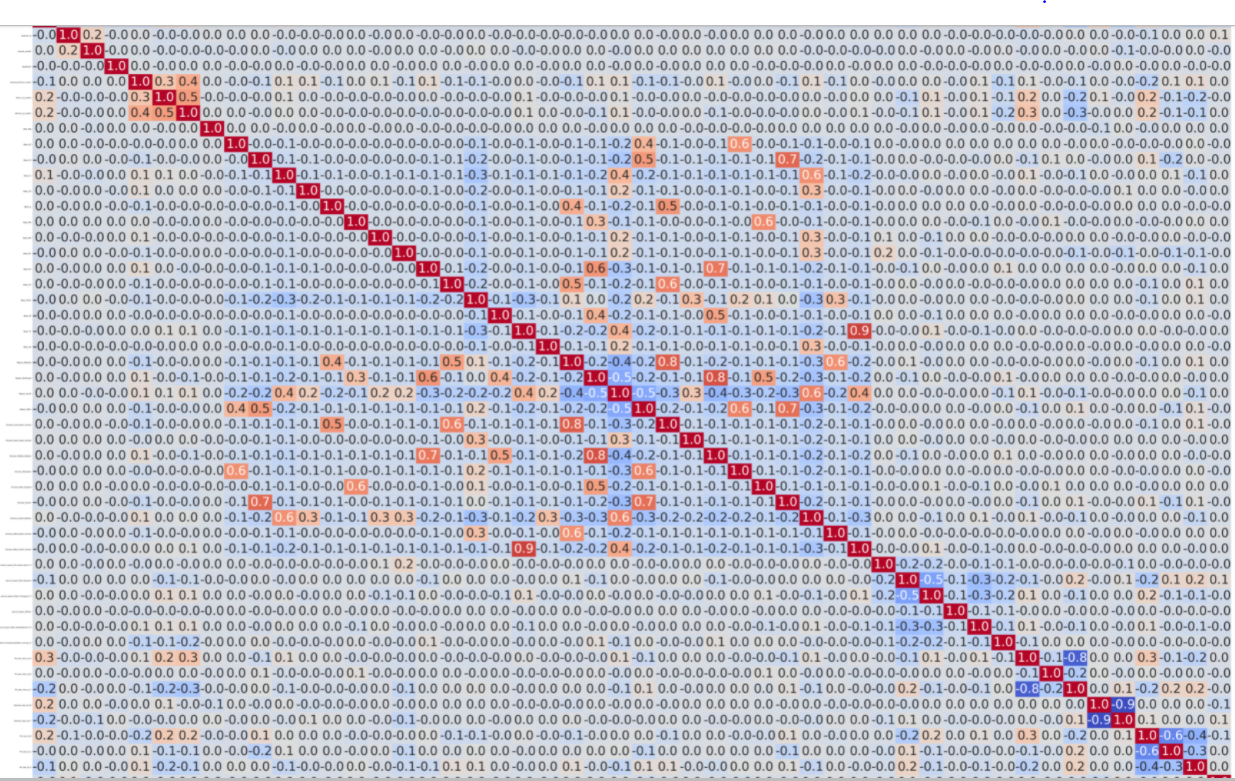
1. High variability in the data: There exists certain situations where the variables doesn’t give any information about data. Those variables should be excluded from the data analysis or else the data might misguide the correlation and give worse results after the model building. Our data has two variables of this kind. The first one is unique\_key. The reason to discard unique\_key feature from the data is because it is unique for every observation in the data, and it does not say much anything about the patterns or details regarding the data. Second one is the zip5 variable, the reason to remove this variable is that there is 4000+ values in the set of 10000 observations which if loaded into the model would be a very highly misleading. Since this is the sample set of population there exists a high variability.
2. 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.
3. Multicollinearity: Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model. Multicollinearity can lead to skewed or misleading results when a researcher or analyst attempts to determine how well each independent variable can be used most effectively to predict or understand the dependent variable in a statistical model. In general, multicollinearity can lead to wider confidence intervals that produce less reliable probabilities in terms of the effect of independent variables in a model. That is, the statistical inferences from a model with multicollinearity may not be dependable. 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 correlation between each two individual variables comprising its pearsons 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 was taken from each set.

*4.3) Exploratory Data Analysis:*

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modeling task. It is not easy to look at a column of numbers or a whole spreadsheet and determine important characteristics of the data. It may be tedious, boring, and/or overwhelming to derive insights by looking at plain numbers. Exploratory data analysis techniques have been devised as an aid in this situation.

Exploratory data analysis is generally cross classified in two ways. First, each method is either non-graphical or graphical. And second, each method is either univariate or multivariate (usually just bivariate).

* Target-Churn Analysis: Target or churn analysis is analyzing the outcome or target variable with respect to 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 a 20% of responses from the data which is more than an industrial standard to go for modelling.

Fig 4.6: Churn Analysis

*4.4) Modelling:*

The act of 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 useful 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 respond to us or the customer does not respond respectively. Here on we discuss each model in detail.

*4.4.1) Random Forest:*

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

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

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

Diagram

Description automatically generated

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 tunning on the model.

An overfit model may look impressive on the training set but will be useless in a real application. Therefore, the standard procedure for hyperparameter optimization accounts for overfitting through cross validation. The Cross Validation can be done either by Random Search or Grid Search.

*4.4.2) Grid search:*

Hyperparameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model. However, evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning: overfitting.

If we optimize the model for the training data, then our model will score very well on the training set but will not be able to generalize to new data, such as in a test set. When a model performs highly on the training set but poorly on the test set, this is known as overfitting, or essentially creating a model that knows the training set very well but cannot be applied to new problems. Therefore, the standard procedure for hyperparameter optimization accounts for overfitting through cross validation.

The technique of cross validation (CV) is best explained by example using the most common method, K-Fold CV. When we approach a machine learning problem, we make sure to split our data into a training and a testing set. In K-Fold CV, we further split our training set into K number of subsets, called folds. We then iteratively fit the model K times, each time training the data on K-1 of the folds and evaluating on the Kth fold (called the validation data). As an example, consider fitting a model with K = 5. The first iteration we train on the first four folds and evaluate on the fifth. The second time we train on the first, second, third, and fifth fold and evaluate on the fourth. We repeat this procedure 3 more times, each time evaluating on a different fold. At the very end of training, we average the performance on each of the folds to come up with final validation metrics for the model.

Chart, bar chart, box and whisker chart

Description automatically generated

Fig.4.8 Cross Validation Technique

For hyperparameter tuning, we perform many iterations of the entire K-Fold CV process, each time using different model settings. We then compare all of the models, select the best one, train it on the full training set, and then evaluate on the testing set. Each time we want to assess a different set of hyperparameters, we have to split our training data into K fold and train and evaluate K times. If we have 10 sets of hyperparameters and are using 5-Fold CV, that represents 50 training loops. Fortunately, as with most problems in machine learning, someone has solved our problem and model tuning with K-Fold CV can be automatically implemented in Scikit-Learn.

Code:

from sklearn.model\_selection import GridSearchCV

# Create the parameter grid based on the results of random search

param\_grid = {

'bootstrap': [True],

'max\_depth': [80, 90, 100, 110],

'max\_features': [2, 3],

'min\_samples\_leaf': [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))

#random\_accuracy = evaluate(best\_random, test\_features, test\_labels)

* + 1. *XGBoost:*

It is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.

The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.

Diagram

Description automatically generated

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% |

* + 1. *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 a provided a table of parameter update at each step.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ­ | 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 |

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

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.

* + 1. *Xgb with Bayesian Optimisation:*

We will try to find optimal values for max\_depth, learning\_rate, n\_estimators and gamma using Bayesian optimization and we will compare the effects on a model built with default parameters.

**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

pbounds = {

'learning\_rate': (0.01, 1.0),

'n\_estimators': (100, 1000),

'max\_depth': (3,10),

'subsample': (1.0, 1.0), # Change for big datasets

'colsample': (1.0, 1.0), # 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': (3,10),

'subsample': (1.0, 1.0), # Change for big datasets

'colsample': (1.0, 1.0), # Change for datasets with lots of features

'gamma': (0, 5)})

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% |

* + 1. *Logistic regression*

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.

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

Where e is the base of the natural logarithms (Euler’s number or the *exp()* function in your spreadsheet) 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.

Chart, diagram

Description automatically generated

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% |

* + 1. *Neural net*

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. We have performed certain trails and multiple iterations to get the best set of layers along with dropout layer to minimize the effect of overfitting.

**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

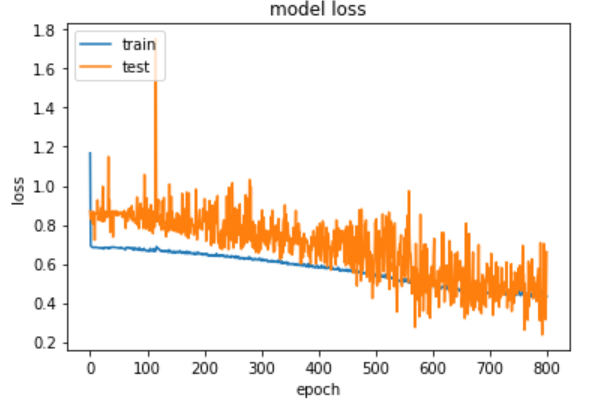
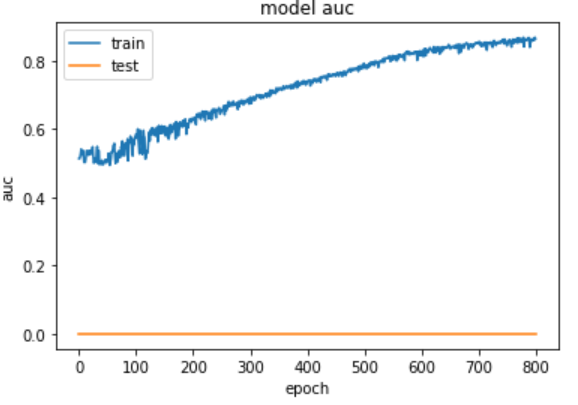
 

Fig 4.11 (A): Model Loss Fig 4.11 (B): Model AUC Score

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% |

* + 1. *Stacking Classifier*

The simplest form of stacking can be described as an ensemble learning technique where the predictions of multiple classifiers (referred as level-one classifiers) are used as new features to train a meta-classifier. The meta-classifier can be any classifier of your choice. 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.

Text

Description automatically generated with medium confidence

Fig 4.12: Stacking Classifier

In the data the level one classifiers are random forest and logistic regression and the meta classifier is XGBosst algorithm which neutralize 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 realiable metrics to estimate the performance of the given model in the academic standards. But for the project of Uplift Modelling we have to choose the pattern of probabilties of the quantiles as the metric for the estimation of any given model. So in our work after attaining the model results apart relaying 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 important steps mainly finding the probalities 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 probabilities of the desired binary classification 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** | **Proabability 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 first quantile to 5th 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

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

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 well performed on the test data set. Quantile one has 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.

This data is for a particular group of months and combined with 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 third quarter by the onboarded team.

CHAPTER-6 CONCLUSIONS & FUTURE SCOPE

The overall conclusion is that for all data sets, each model required plenty of tries for being able to capture the model parameters that yielded in the best model performances. In general, the models performed poorly although some were able to obtain satisfying results. In the following sections, the results & suggestions for future work will be discussed.

* 1. *Discussion*

When building the uplift models using Random Forests, it was not always possible to obtain models that performed better than a random classifier which means that the result of an action is worse than doing nothing in those cases. One reason for this is due to one of the parameters in the upliftRF() method in the uplift [6] package that defines the minimum number of control observations that must exist in any terminal node. Hence, during the building process, the tree is forced to contain control data in every region.

Both Logistic Regression and Neural Networks were able to capture good models for treatment data and control data separately. This can be seen in the Roc curve, where the curves are high above the diagonal line and the values of AUC are very good, i.e., greater than or equal to 0.5. This means that the models perform well on unseen data (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.

In the previous sections we have seen performance of all the models that have been tried. We can notice that the tree algorithms have overfit on the data whilst the bagging algorithms and probability modelling. We could not exceed the recall by 15% on the test set. It has been same with the mathematical logistic function. But we have noticed that Logistic regression has underfit on the data set. Sci-kit learn has a function called Stacking classifier which can be used to club multiple models to build a singular model. We have taken the three most well performed models into the stacking classifier. The meta model in 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 parameters values to meet industrial standards.

* 1. *Future Enhancement*

The market evaluated in this project was for only one country i.e., 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 kind of campaigns. Thus, possible future work could be to 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 as well as an uplift for the company in terms of greater gain in the selling.

The overall conclusion is that it is possible to perform uplift modeling to obtain models that makes 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. 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 due to the fact that Neural Networks are sensitive to uneven class distributions and are thus not able 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/>

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 | @manipal.edu | | |