PREDICTING STORE REVENUE

A

Project Report

Submitted for the partial fulfilment

Of B.Tech Degree

in

Information Technology

by

Keshav Chandra [1505213025]

Kapil Chaudhary [1505213024]

Under the supervision of

Ms. Shipra Gautam

Department of Computer Science and Engineering

Institute of Engineering & Technology

Dr. APJ Abdul Kalam Technical University Uttar Pradesh, Lucknow

June, 2019

DECLARATION

We hereby declare that this project entitled **PREDICTING STORE REVENUE** submitted by us, in the partial fulfilment for the award of **Bachelor of Technology** in **Computer Science and Engineering** is a record of bona-fide work carried out by us under the supervision and guidance of **Ms. Shipra Gautam** at the department of **Computer Science and Engineering** at **Institute of Engineering and Technology, Lucknow**. This project has not been submitted by us at any other institute for requirement of any other degree.

Date: 04/06/2018 By:

Kapil Chaudhary(1505213024)
Keshav Chandra(1505213025)

CERTIFICATE

This is to certify that project report entitled "**PREDICTING STORE REVENUE**" submitted by Kapil Chaudhary(roll number 1505213024) and Keshav Chandra(roll number 1505213025) in the partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering is a record of work carried out by them under my supervision and guidance at department of Computer Science and Engineering at Institute of Engineering and Technology, Lucknow.

(Ms.Shipra Gautam)

IT Program

(Prof. Y.N. Singh)

Convenor IT Program

Department of Engineering and Technology

Institute of Engineering & Technology

Dr. A.P.J. Abdul Kalam Technical University

Lucknow, Uttar Pradesh

ACKNOWLEDGEMENT

The completion of any inter-disciplinary project depends upon cooperation,coordination and combined efforts of several sources of knowledge. We are grateful to our project supervisor, **Ms. Shipra Gautam**, for his even willingness to give us valuable advice and direction whenever We approached her with a problem. I am thankful to him for providing immense guidance for this project.

We owe special debt of gratitude to our Head of Department and project coordinator, **Dr. Y.N Singh**, for his constant support and guidance throughout the course of work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me.

We take this opportunity to thank all my lecturers who have directly or indirectly helped in the completion of this project.

By:

Keshav Chandra(1505213025)

Kapil Chaudhary(1505213024)

TABLE OF CONTENTS

CHAPTER **page**

Introduction

In which states/places of USA are the stores performing the best, and in which states, the stores are lagging behind? Can you suggest states/places where more stores may be opened, and places where stores may be reviewed for possible closure? Is there any state/place where you would plan on merging stores to improve performance? What are the main reasons/indicators behind a store performing better or worse in comparison with the others?

With this information the corporation hopes we can identify the stores which play a key role in their sales and use that information to take the correct measures to ensure success of their business.The importance matrix also play a key role in strategy building in a company which suggest which parameters have a strong correlation between store revenue and other features such as store location,number of workers working,product type etc.

Companies use Business Analytics to make data-driven decisions. The insight gained by Business Analytics enables these companies to automate and optimize their business processes. In fact, data-driven companies that utilize Business Analytics achieve a competitive advantage because they are able to use the insights.Business Analytics also provides support for companies in the process of making proactive tactical decisions, and Business Analytics makes it possible for those companies to automate decision making in order to support real-time responses.

PROBLEM DESCRIPTION

We have a dataset – **storeData.zip** – associated with the problem, containing three distinct files – **storeTrain.csv, storeTest.csv, and storeVars.txt**. The training dataset, given as storeTrain.csv, contains records of 1693 departmental stores, pertaining to a specific chain/brand of stores in USA. The columns/variables of this dataset are explained in storeVars.txt.

The test dataset, given as storeTest.csv, contains records of 1130 stores of the same chain/brand, but different from the ones in the training set. In each record of the test dataset, **the target/response variable REVENUE_2013** is missing, whereas it is present in each record of the training dataset.

The problem is to evaluate the estimated values for the target variable REVENUE_2013 for the test data set, using an algorithm which is deemed best at the needed job.

We have a dataset – storeData.zip – associated with the problem, containing three distinct files – storeTrain.csv, storeTest.csv, and storeVars.txt. The training dataset, given as storeTrain.csv, contains records of 1693 departmental stores, pertaining to a specific chain/brand of stores in USA. The columns/variables of this dataset are explained in storeVars.txt.

The test dataset, given as storeTest.csv, contains records of 1130 stores of the same chain/brand, but different from the ones in the training set. In each record of the test dataset, the target/ response variable REVENUE_2013 is missing, whereas it is present in each record of the training dataset.

The problem is to evaluate the estimated values for the target variable REVENUE_2013 for the test data set, using an algorithm which is deemed best at the needed job

storeVars.txt

The columns/variables of this dataset(here the storeTrain.csv) are explained in storeVars.txt.They are as follows:

STORE_NO	Sequential (unique) ID for the stores
REVENUE_2013	Target Variable (numeric/integer)
ZIP	Zip Code for the store location
SQUARE_FEET	Carpet area of the store
U_CITY	City of the store in USA
U_STATE	State of the store in USA
CENSUS_REGION	Census region of the store in USA
CENSUS_DIVISION	Census division of the store in USA
TOT_ATTRITION_2012	Total attrition at the store in 2012
TOT_ATTRITION_2013	Total attrition at the store in 2013
NUM_ASSISTANT_MANAGERS	Number of assistant managers at the store
NUM_CUST_ACC_REPS	Number of customer representatives at the store
NUM_STORE_MANAGERS	Number of store managers at the store
NUM_EMP_PAY_TYPE_H	Number of employees with hourly pay
AVG_PAY_RATE_PAY_TYPE_S	Average pay per employee on salary basis
AVG_PAY_RATE_PAY_TYPE_H	Average pay per employee on hourly basis
NAT_CURR_ROBBERY	Current account of robbery in the area
NAT_CURR_BURGLARY	Current account of burglary in the area
NAT_CURR_MOT_VEH_THEFT	Current account of motor vehicle theft in the area
NAT_PAST_ROBBERY	Past account of robbery in the area
NAT_PAST_BURGLARY	Past account of burglary in the area
NAT_PAST_MOT_VEH_THEFT	Past account of motor vehicle theft in the area
FRONTAGE_ROAD	Whether the store has a front road opening
NUM_PARKING_SPACES	Number of parking spaces available at the store

SIGNAGE_VISIBILITY_IND	Indicator of signage visibility for the store
AUTOZONE_IND	Indicator of "Autozone" store in the neighborhood
TARGET_IND	Indicator of "Target" store in the neighborhood
WALMART_IND	Indicator of "Walmart" store in the neighborhood
PAYLESS_IND	Indicator of "Payless" store in the neighborhood
COMP_PRESENCE_IND	Indicator of competing store in the neighborhood
STRIP_SHOP_CENTER_IND	Indicator of marketplace in the neighborhood
SINGLE_TENANT_IND	Indicator if the store is a single tenant property
PAD_IN_SHOP_CENTER_IND	Indicator if the store is a pad site in front of a mall
MARKETING_EXP_2013	Total marketing expenses incurred by the store in 2013
MARKETING_EXP_2012	Total marketing expenses incurred by the store in 2012
TOT_NUM_LEADS	Total number of leads pursued by the store
NUM_CONVERTED_TO_AGREEMENT	Total number of leads converted to agreement
PERC_CONVERTED_TO_AGREEMENT	Percentage of leads converted to agreement
URBANICITY	The nature of the area where the store is located
CYA01V001	Total population of the area
CYB02V001	Total number of households in the area
CYA12V001	"Never Married" people in the area
CYA12V002	"Now Married" people in the area
CYA12V003	"Now Married, Spouse Present" people in the area
CYA12V007	"Widowed" people in the area

CYA12V008	"Divorced" people in the area
CYA21V001	Total land area in square miles
CYB07VBASE	Total number of households in the area
XCX03V069	Average intracity mass transit fares
PERC_CYEA07V001	"White" population (as % of total population)
PERC_CYEA07V004	"African American" population (as % of total population)
PERC_CYEA07V007	"American Indian" population (as % of total population)
PERC_CYEA07V010	"Asian" population (as % of total population)
PERC_CYB11V006	"6 Person" households (as % of total households)
PERC_CYB11V007	"7+ Person" households (as % of total households)
PERC_CYC13VV01	Per Capita Income based on total population (% of total population for whom per capita is calculated)

LITERATURE OVERVIEW

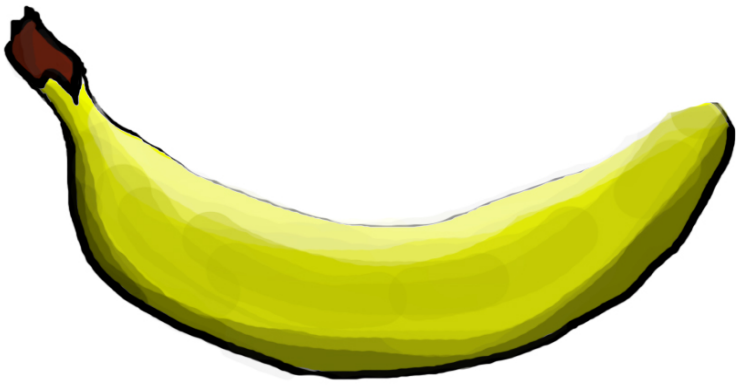
Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: ***The ability to learn***. Machine learning is actively being used today, perhaps in many more places than one would expect.Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data.

For instance, suppose you are given an basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this:

- If shape of object is rounded and depression at top having color Red then it will be labelled as –**Apple**.
- If shape of object is long curving cylinder having color Green-Yellow then it will be labelled as –**Banana**.

Now suppose after training the data, you have given a new separate fruit say Banana from basket and asked to identify it.



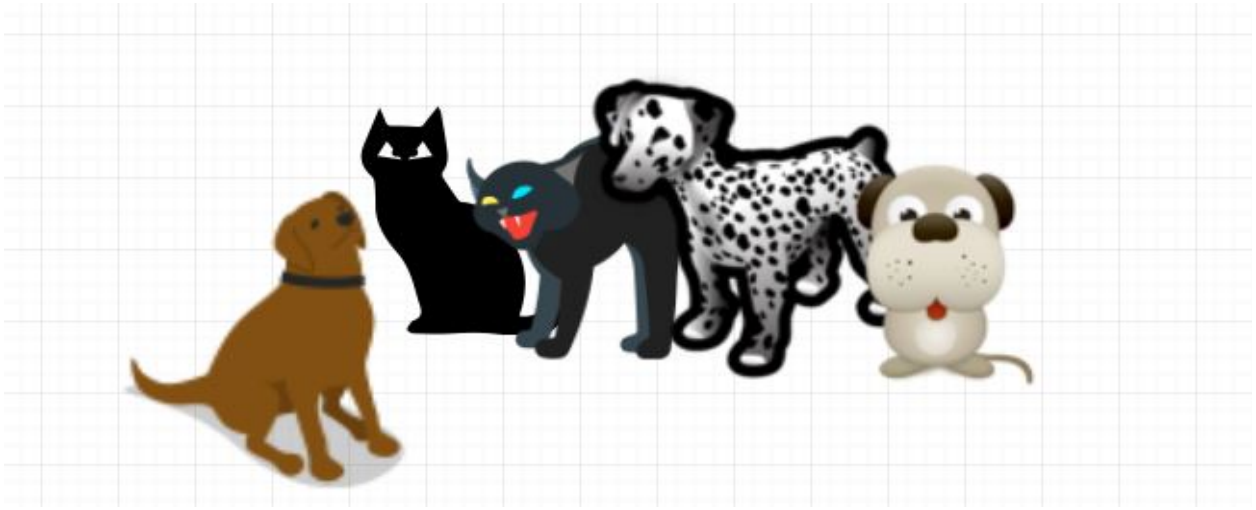
Since the machine has already learned the things from previous data and this time have to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in Banana category. Thus the machine learns the things from training data(basket containing fruits) and then apply the knowledge to test data(new fruit).

Supervised learning classified into two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore machine is restricted to find the hidden structure in unlabeled data by our-self.
For instance, suppose it is given an image having both dogs and cats which have not seen ever.



Classification of cat and dog through Unsupervised Learning

Thus the machine has no idea about the features of dogs and cat so we can't categorize it in dogs and cats. But it can categorize them according to their similarities, patterns, and differences i.e., we can easily categorize the above picture into two parts. First first may contain all pics having **dogs** in it and second part may contain all pics having **cats** in it. Here you didn't learn anything before, means no training data or examples.

Unsupervised learning classified into two categories of algorithms:

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Reinforcement learning is an area of Machine Learning. Reinforcement. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience.

Example : The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.

Reward steps of robot in Reinforcement Learning

The above image shows robot, diamond and fire. The goal of the robot is to get the reward that is the diamond and avoid the hurdles that is fire. The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles. Each right step will give the robot a reward and each wrong step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.

Cross Validate a Model

Once we are done with training our model, we just can't assume that it is going to work well on data that it has not seen before. In other words, we can't be sure that the model will have the desired accuracy and variance in production environment. We need some kind of assurance of

the accuracy of the predictions that our model is putting out. For this, we need to validate our model. This process of deciding whether the numerical results quantifying hypothesised relationships between variables, are acceptable as descriptions of the data, is known as validation.

To evaluate the performance of any machine learning model we need to test it on some unseen data. Based on the models performance on unseen data we can say weather our model is Under-fitting/Over-fitting/Well generalised. Cross validation (CV) is one of the technique used to test the effectiveness of a machine learning models, it is also a re-sampling procedure used to evaluate a model if we have a limited data. To perform CV we need to keep aside a sample/portion of the data on which is do not use to train the model, later us this sample for testing/validating. There are many methods

Below are the few common techniques used for CV.

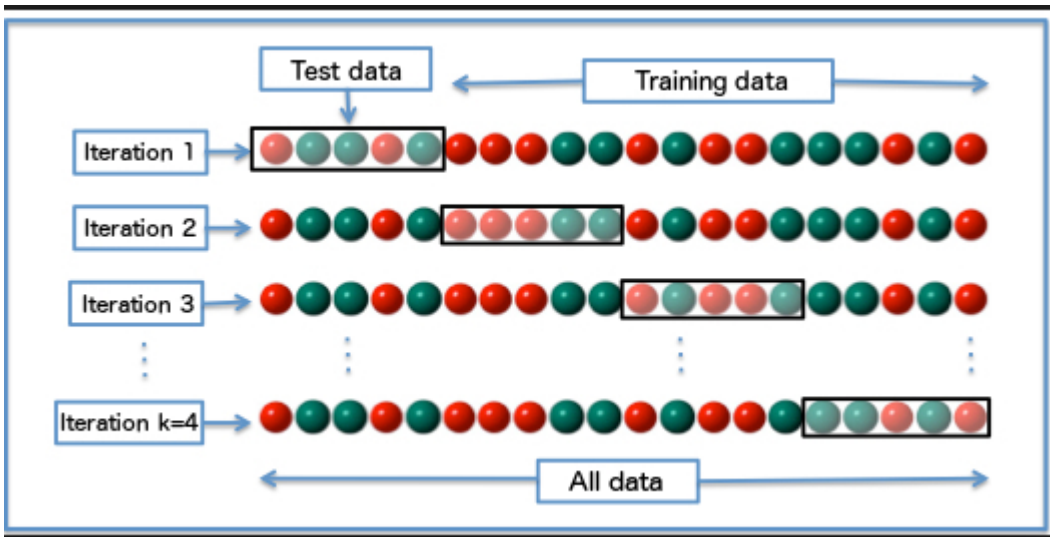
1)**Train_Test Split approach**

In this approach we randomly split the complete data into training and test sets. Then Perform the model training on the training set and use the test set for validation purpose, ideally split the data into 70:30 or 80:20. With this approach there is a possibility of high bias if we have limited data, because we would miss some information about the data which we have not used for training. If our data is huge and our test sample and train sample has the same distribution then this approach is acceptable.

2)**K-Folds Cross Validation**

K-Fold is a popular and easy to understand, it generally results in a less biased model compare to

other methods. Because it ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one among the best approach if we have a limited input data. This method follows the below steps.



K Fold **C**ross Validation Steps

1. Split the entire data randomly into k folds (value of k shouldn't be too small or too high, ideally we choose 5 to 10 depending on the data size). The higher value of K leads to less biased model (but large variance might lead to overfit), where as the lower value of K is similar to the train-test split approach we saw before.

2. Then fit the model using the K—1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.

3. Repeat this process until every K-fold serve as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

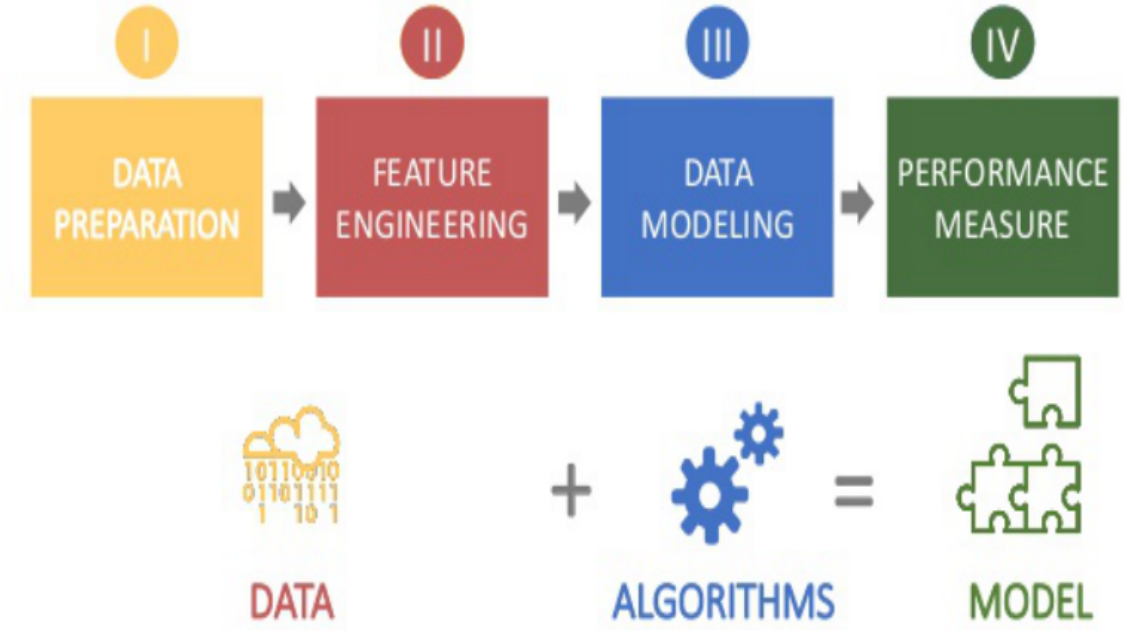
WORK STRATEGY

come to desired result the work must be divided in the following sections:

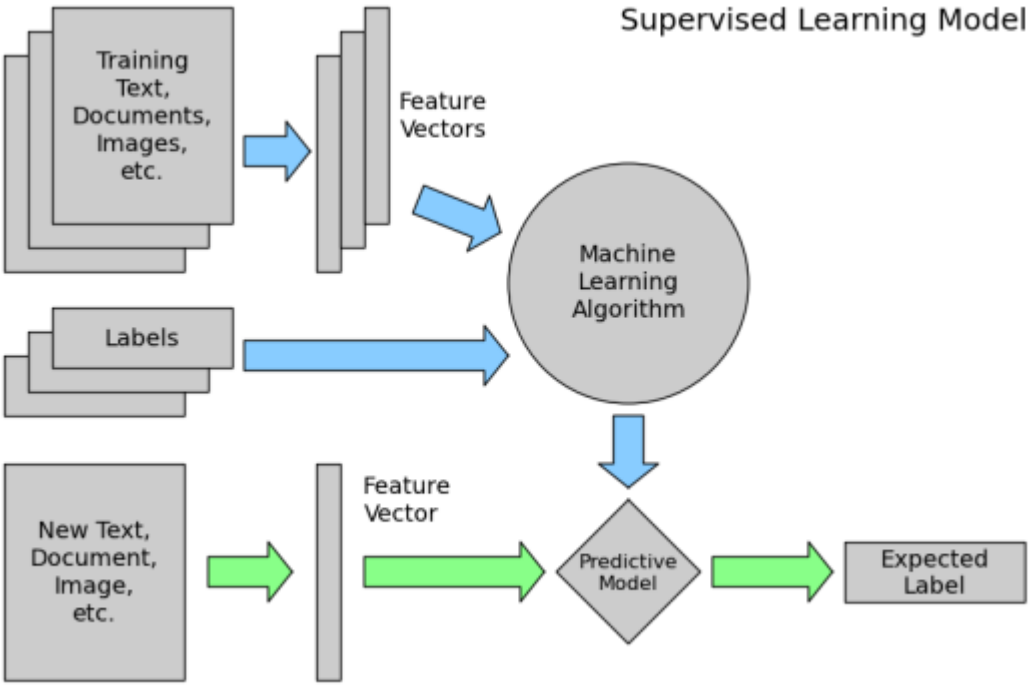
1. Data Summarization and Handling Missing Data
2. Data Cleaning
3. Attribute Selection
4. Modelling
5. Predicting for the test data

WHAT IS A MACHINE LEARNING **MODEL**?

There are **4 steps** to build a machine learning model...



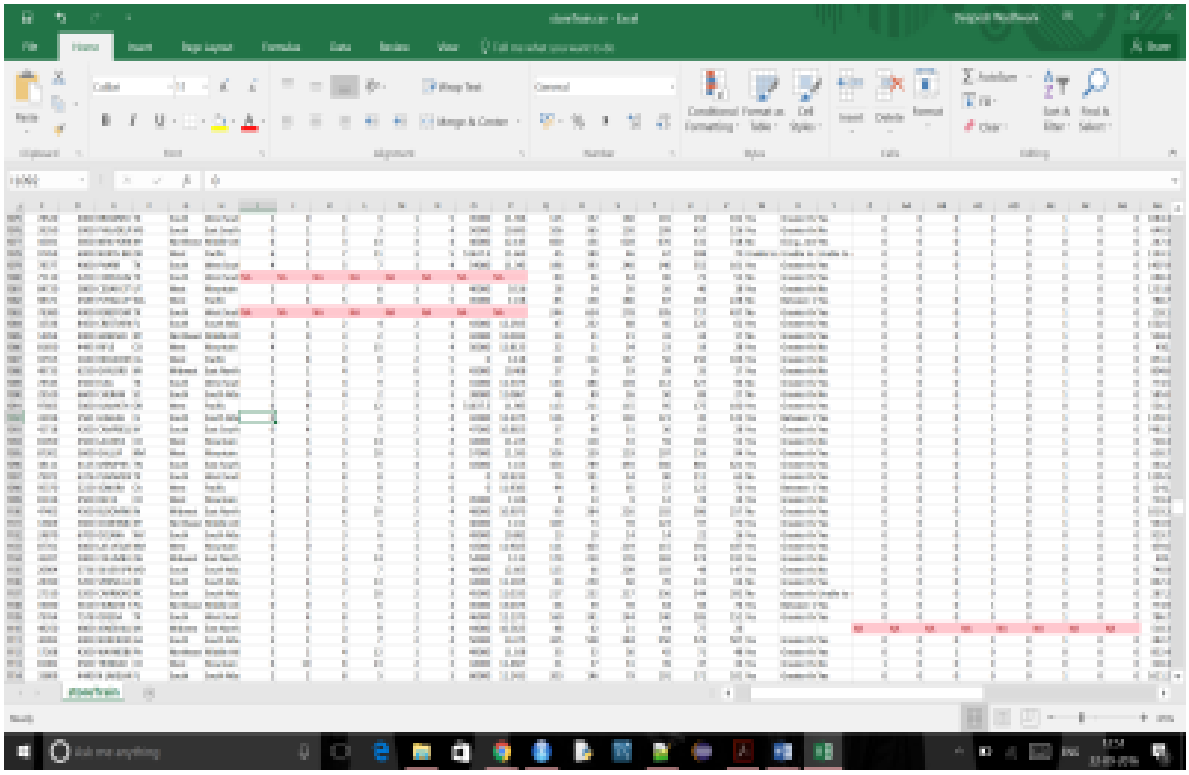
DATA FLOW DIAGRAM



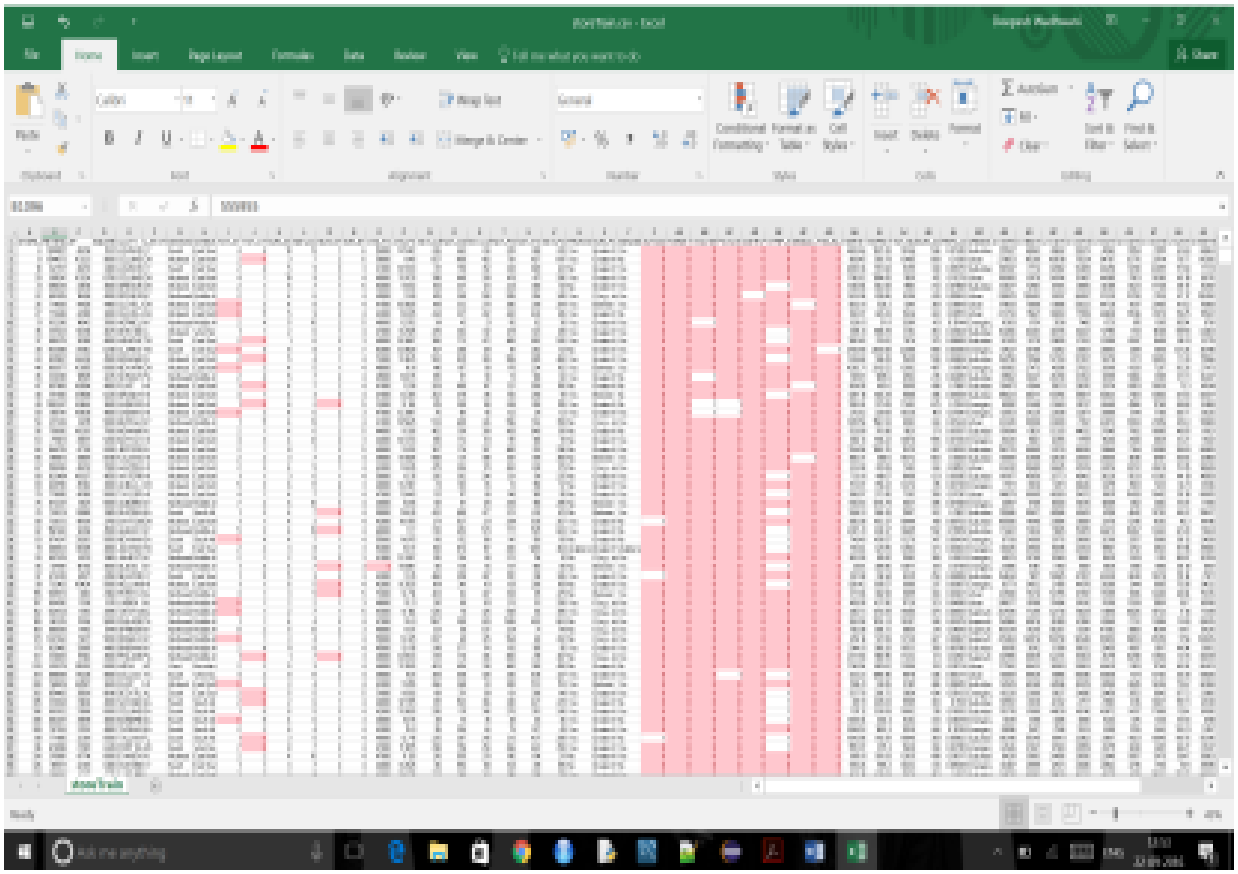
DATA FLOW DIAGRAM

Data Summarization and Handling Missing Data

We can observe in the problem statement that the data size is not very large (just 1169 records for Train data and 1130 records for test data) Thus, preliminary data analysis and visualisation can be done in MS Excel. And the developments can be then analysed in R.



The first observation was the Missing data, for that ‘NA’ values were highlighted red using conditional formatting. It was observed that only 13 entries had some data missing from them. These 13 entries were then discarded. This point on, the data had no missing values.



The next observation is the zero entries, if one attribute has a lot of zero entries, it most probably will have no correlation with the output variable. Thus we can use conditional formatting to highlight all 0 values, and can use those attribute to plot correlation plot later in R.

We observed that there were total of 8 such variables, which are:

- STRIP_SHOP_CENTRE_IND
- AUTOZONE_IND
- SINGLE_TENANT_IND
- TARGET_IND
- PAID_IN_SHOP_CENTRE_IND
- WALMART_IND
- PAYLESS_IND
- COMP_PRESENCE_IND

To observe the data in R we can use str() function to see all the data types as

```
> str(storeTrain,vec.len=0)
'data.frame':    1693 obs. of  56 variables:
 $ STORE_NO                : int  NULL ...
 $ REVENUE_2013            : int  NULL ...
 $ ZIP                     : int  NULL ...
 $ SQUARE_FEET             : int  NULL ...
 $ U_CITY                  : Factor w/ 1110 levels
 $ U_STATE                 : Factor w/ 49 levels
 $ CENSUS_REGION           : Factor w/ 4 levels
 $ CENSUS_DIVISION         : Factor w/ 9 levels
 $ TOT_ATTRITION_2012      : int  NULL ...
 $ TOT_ATTRITION_2013      : int  NULL ...
 $ NUM_ASSISTANT_MANAGERS  : int  NULL ...
 $ NUM_CUST_ACC_REPS       : int  NULL ...
 $ NUM_STORE_MANAGERS      : int  NULL ...
 $ NUM_EMP_PAY_TYPE_H      : int  NULL ...
 $ AVG_PAY_RATE_PAY_TYPE_S : num  NULL ...
 $ AVG_PAY_RATE_PAY_TYPE_H : num  NULL ...
 $ NAT_CURR_ROBBERY        : int  NULL ...
 $ NAT_CURR_BURGLARY       : int  NULL ..
 $ NAT_CURR_MOT_VEH_THEFT  : int  NULL
 $ NAT_PAST_ROBBERY        : int  NULL ...
 $ NAT_PAST_BURGLARY       : int  NULL ...
 $ NAT_PAST_MOT_VEH_THEFT  : int  NULL ...
 $ FRONTAGE_ROAD           : Factor w/ 4 levels
```

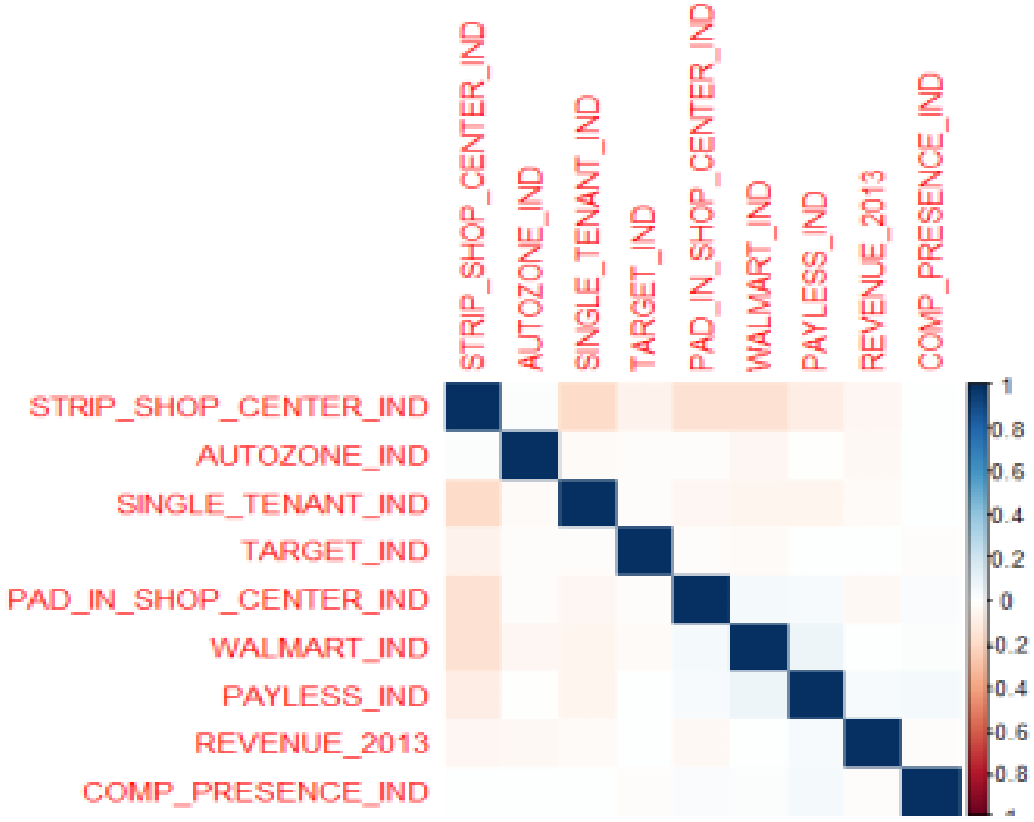
$ NUM_PARKING_SPACES : Factor w/ 5 levels
$ SIGNAGE_VISIBILITY_IND : Factor w/ 5 levels
$ AUTOZONE_IND : int NULL ...
$ TARGET_IND : int NULL ...
$ WALMART_IND : int NULL ...
$ PAYLESS_IND : int NULL ...
$ COMP_PRESENCE_IND : int NULL ...
$ STRIP_SHOP_CENTER_IND : int NULL ...
$ SINGLE_TENANT_IND : int NULL ...
$ PAD_IN_SHOP_CENTER_IND : int NULL ...
$ MARKETING_EXP_2013 : num NULL ...
$ MARKETING_EXP_2012 : num NULL ...
$ TOT_NUM_LEADS : int NULL ...
$ NUM_CONVERTED_TO_AGREEMENT : int NULL ...
$ PERC_CONVERTED_TO_AGREEMENT: num NULL ...
$ URBANICITY : Factor w/ 5 levels
$ CYA01V001 : int NULL ...
$ CYB02V001 : int NULL ...
$ CYA12V001 : int NULL ...
$ CYA12V002 : int NULL ...
$ CYA12V003 : int NULL ...
$ CYA12V007 : int NULL ...
$ CYA12V008 : int NULL ...
$ CYA21V001 : num NULL ...
$ CYB07VBASE : int NULL ...
$ XCX03V069 : num NULL ...
$ PERC_CYEA07V001 : num NULL ...
$ PERC_CYEA07V004 : num NULL ...
$ PERC_CYEA07V007 : num NULL …
$ PERC_CYEA07V010 : num NULL ...
$ PERC_CYB11V006 : num NULL ...
$ PERC_CYB11V007 : num NULL ...
$ PERC_CYC13VV01 : num NULL ...

This shows us that in 56 attributes, it has all int, num and factor data types. Hence a cleaning algorithm will be needed at some point. But before that we should analyse the 7 variables we observed in excel.

These variable can now be analysed with R for correlation plot. The code for doing this can referred to in annexure A. The correlation plot is as shown.

From the corrplot we can observe that the all the variable has negative correlation with the target variable Revenue_2013. And, all except STRIP_SHOP_CENTRE_IND has neglect able correlation. Hence they can be neglected.

At this point we can delete these columns from the data and continue our analysis with the remaining data containing 49 attributes.



Correlation Plot

Data cleaning

As we have already observed that the data contains different categorical variables having factors and thus we need a data cleaning algorithm, the code of the algorithm can be observed in annexure A.

The clean data will have all numerical values and these can be used to draw various plots which can be used to make an educated guess of variables which will have most effect of the target variable. This will be done in attribute selection.

The process of data cleaning is as follows:

As the data provided in the train dataset contains certain NA values that will hinder the prediction of the final value of the response variable. So we have to do data cleaning of the train dataset.

Step1) We have used the following code to find out the number of the attributes against which NA values are coming. The following code will find out against which attribute we have what count of NA values.

```
for(i in c(1:56))
{
 count=0;
 count<-sum(is.na(train1[,i]))/1693;
 if(count>.8)
 {   print(i) ;}}
```

As the count is very less , so the percentage of NA is very less. So we need not delete the attributes for the same.

Step2)Next we saw certain attributes against which the values are either o and 1. We have counted and we found the values were almost zero, so we have discarded those attributes in our prediction

Step 3)We have used the following code to set all other values to NA.

```
#train data cleaning
train1[is.na(train1)]<-0
#convert categorical data into the numeric data
for(i in 1:49){
 if(is.factor(train1[,i]))
  train1[,i]<-as.integer(train1[,i])
}
```

FEATURE SELECTION

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing.

In this post, you will discover feature selection techniques that you can use in Machine Learning.

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.

· **Improves Accuracy**: Less misleading data means modeling accuracy improves.

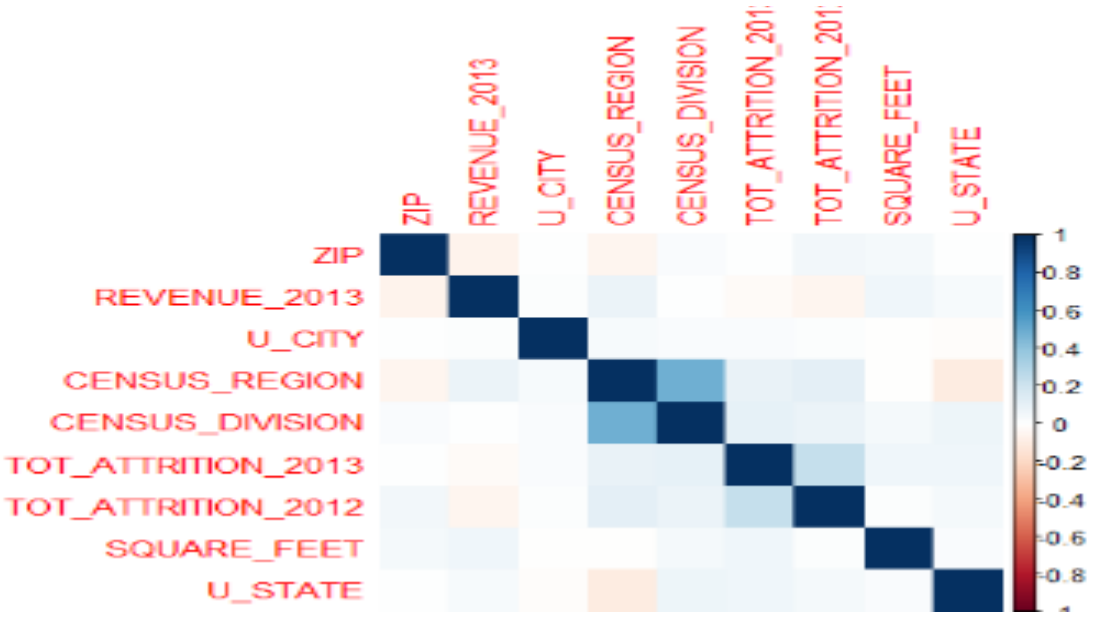
· **Reduces Training Time**: fewer data points reduce algorithm complexity and algorithms train faster.

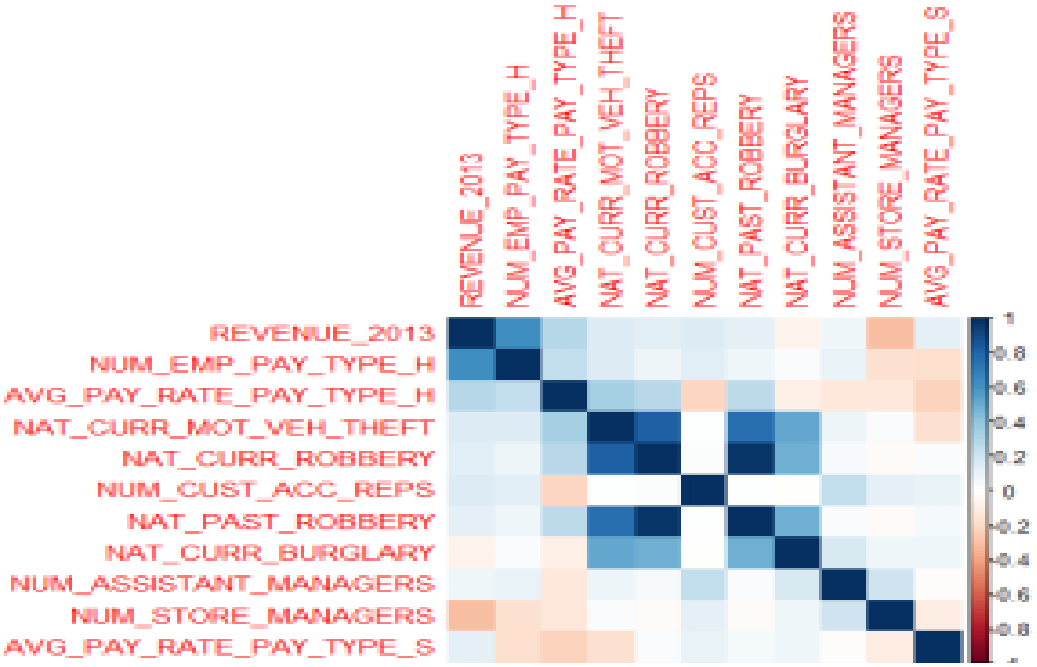
Feature selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples.

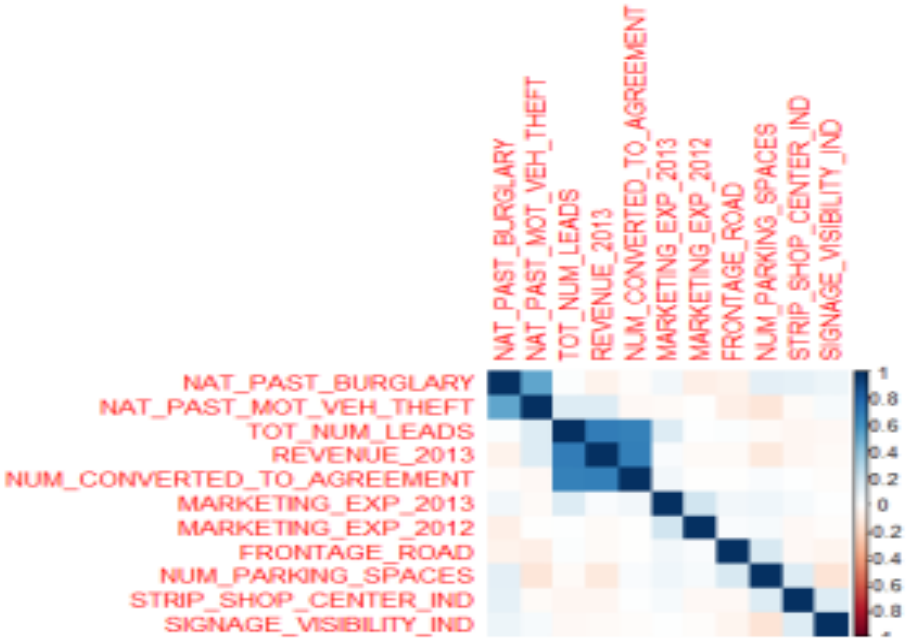
The Feature selection is done automatically by XGBoost. But to verify that the parameters selected are logically sound, we must draw some corrplots to see how all variables correlate with the target variable.

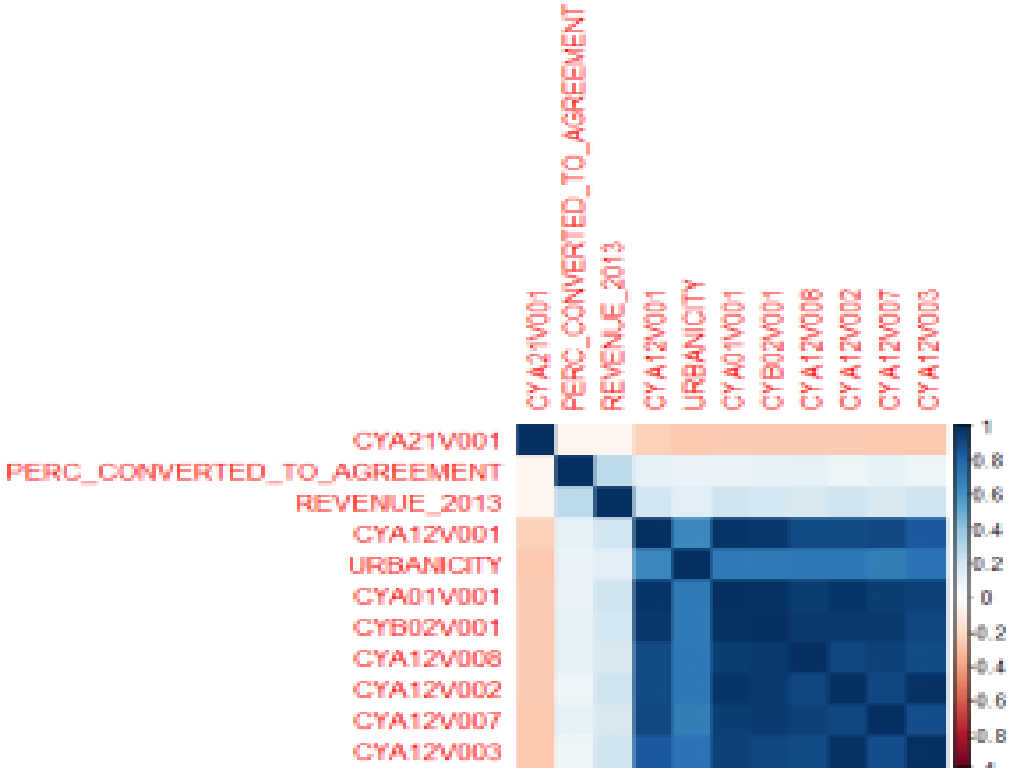
The code of this can be found in the annexure A.

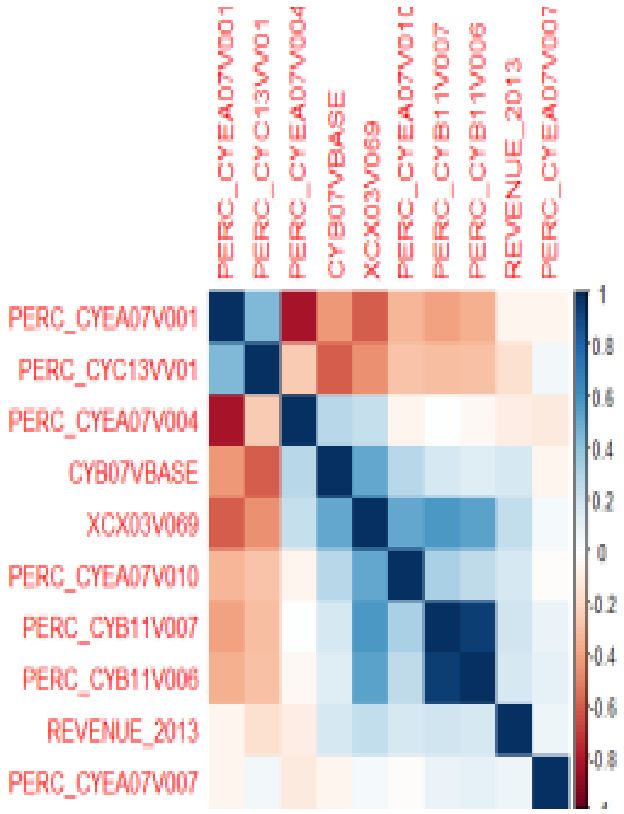
To check 47 variables plus 1 target variable (neglecting store number) we should draw multiple corrplots for the ease of understanding. The plots are as follows:


ZIP
REVENUE_2013
U_CITY
CENSUS_REGION
CENSUS_DIVISION
TOT_ATTRITION_201
TOT_ATTRITION_201
SQUARE_FEET
U_STATE
ZIP
REVENUE_2013
U_CITY
CENSUS_REGION
CENSUS_DIVISION
TOT_ATTRITION_2013
TOT_ATTRITION_2012
SQUARE_FEET
U_STATE
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8


REVENUE_2013
NUM_EMP_PAY_TYPE_H
AVG_PAY_RATE_PAY_TYPE_H
NAT_CURR_MOT_VEH_THEFT
NAT_CURR_ROBBERY
NUM_CUST_ACC_REPS
NAT_PAST_ROBBERY
NAT_CURR_BURGLARY
NUM_ASSISTANT_MANAGERS
NUM_STORE_MANAGERS
AVG_PAY_RATE_PAY_TYPE_S
REVENUE_2013
NUM_EMP_PAY_TYPE_H
AVG_PAY_RATE_PAY_TYPE_H
NAT_CURR_MOT_VEH_THEFT
NAT_CURR_ROBBERY
NUM_CUST_ACC_REPS
NAT_PAST_ROBBERY
NAT_CURR_BURGLARY
NUM_ASSISTANT_MANAGERS
NUM_STORE_MANAGERS
AVG_PAY_RATE_PAY_TYPE_S
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8


NAT_PAST_BURGLARY
NAT_PAST_MOT_VEH_THEFT
TOT_NUM_LEADS
REVENUE_2013
NUM_CONVERTED_TO_AGREEMENT
MARKETING_EXP_2013
MARKETING_EXP_2012
FRONTAGE_ROAD
NUM_PARKING_SPACES
STRIP_SHOP_CENTER_IND
SIGNAGE_VISIBILITY_IND
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8

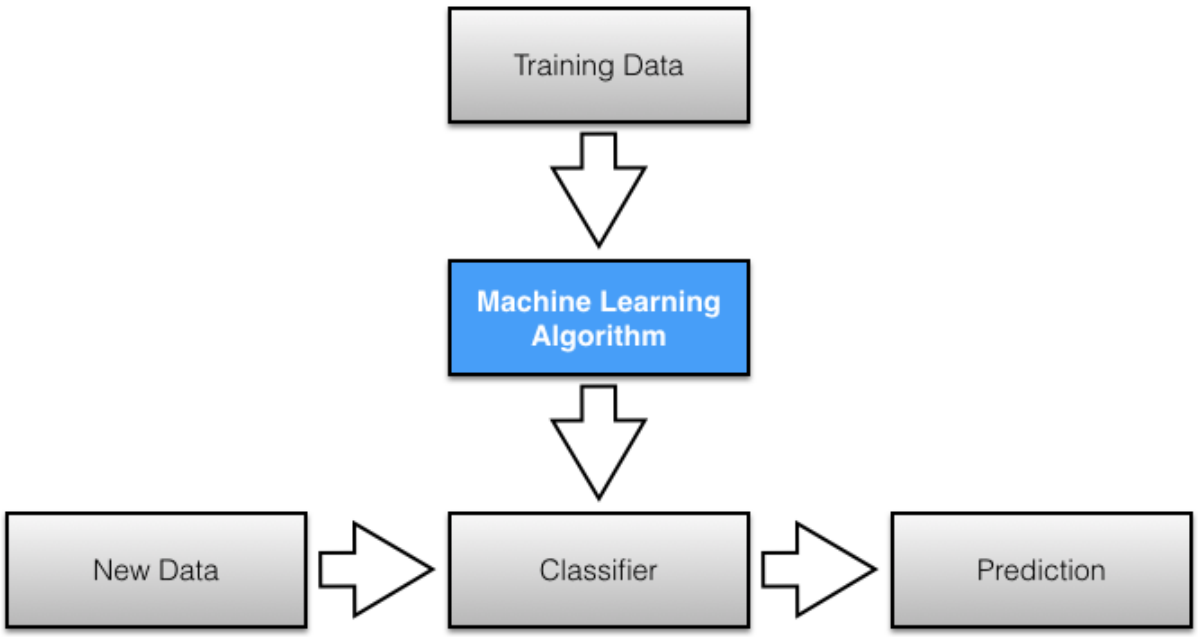

CYA21V001
PERC_CONVERTED_TO_AGREEMENT
REVENUE_2013
CYA12V001
URBANICITY
CYA01V001
CYB02V001
CYA12V008
CYA12V002
CYA12V007
CYA12V003
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8


PERC_CYEA07V001
PERC_CYC13VV01
PERC_CYEA07V004
CYB07VBASE
XCX03V069
PERC_CYEA07V010
PERC_CYB11V007
PERC_CYB11V006
REVENUE_2013
PERC_CYEA07V007
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8

Modelling

A model is something that is created by the training process. These are basically our machine learning algorithms, example are Decision tree model, Random forest. ... The process of training and ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learning.

Modeling is a process that uses data mining and probability to forecast outcomes. Each model is made up of a number of predictors, which are variables that are likely to influence future results. Once data has been collected for relevant predictors, a statistical model is formulated. The model may employ a simple linear equation, or it may be a complex neural network, mapped out by sophisticated software. As additional data becomes available, the statistical analysis model is validated or revised.



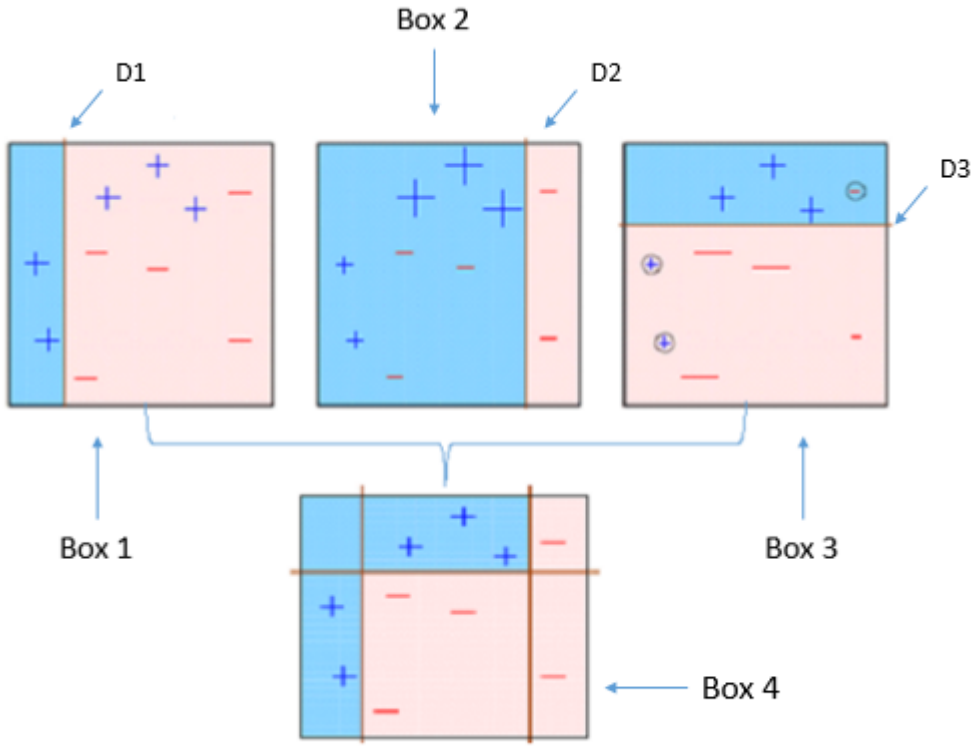
XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library. Yes, it uses gradient boosting (GBM) framework at core. Yet, does better than GBM framework alone. XGBoost was created by Tianqi Chen, PhD Student, University of Washington. It is used for supervised ML problems. Let's look at what makes it so good:

1. Parallel Computing: It is enabled with parallel processing (using OpenMP); i.e., when you run xgboost, by default, it would use all the cores of your laptop/machine.
2. Regularization: I believe this is the biggest advantage of xgboost. GBM has no provision for regularization. Regularization is a technique used to avoid overfitting in linear and tree-based models.
3. Enabled Cross Validation: In R, we usually use external packages such as caret and mlr to obtain CV results. But, xgboost is enabled with internal CV function (we'll see below).
4. Missing Values: XGBoost is designed to handle missing values internally. The missing values are treated in such a manner that if there exists any trend in missing values, it is captured by the model.
5. Flexibility: In addition to regression, classification, and ranking problems, it supports user-defined objective functions also. An objective function is used to measure the performance of the model given a certain set of parameters. Furthermore, it supports user defined evaluation metrics as well.
6. Availability: Currently, it is available for programming languages such as R, Python, Java, Julia, and Scala.
7. Save and Reload: XGBoost gives us a feature to save our data matrix and model and reload it later. Suppose, we have a large data set, we can simply save the model and use it in future instead of wasting time redoing the computation.
8. Tree Pruning: Unlike GBM, where tree pruning stops once a negative loss is encountered, XGBoost grows the tree upto max_depth and then prune backward until the improvement in loss function is below a threshold.

How does XGBoost work ?

XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing. Let's understand boosting first (in general).
Boosting is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations. Let's look at a classic classification example:



Four classifiers (in 4 boxes)

Four classifiers (in 4 boxes), shown above, are trying hard to classify + and - classes as homogeneously as possible. Let's understand this picture well.

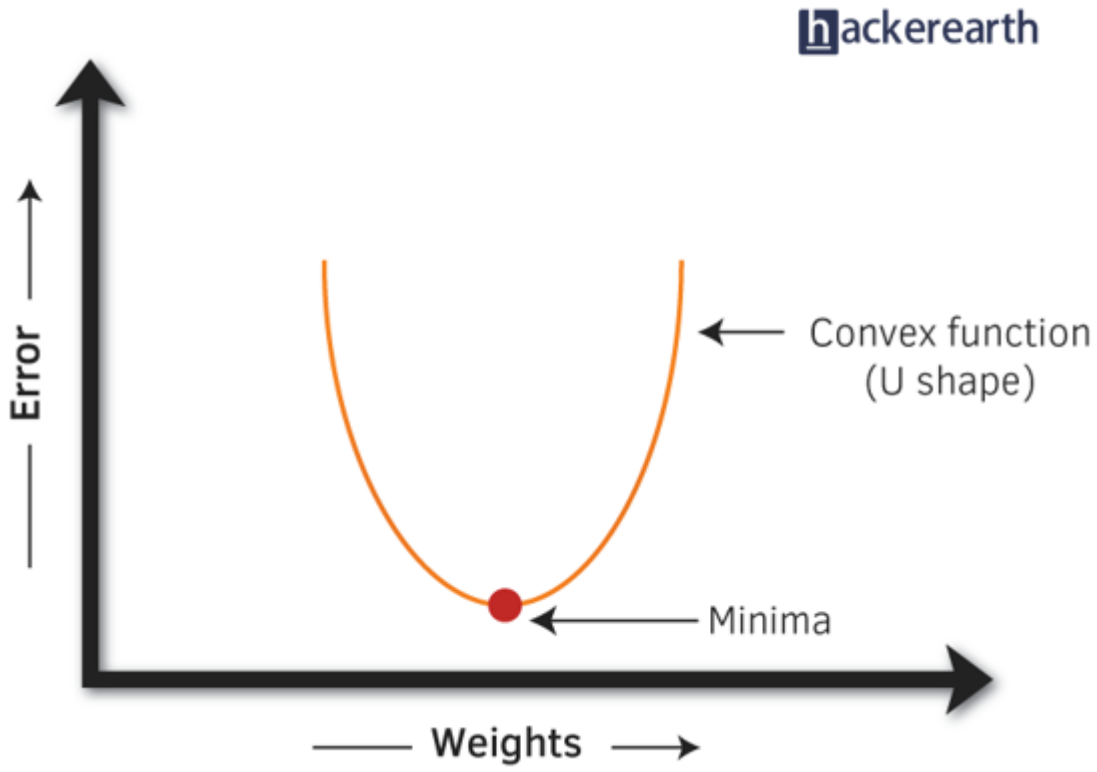
1. Box 1: The first classifier creates a vertical line (split) at D1. It says anything to the left of D1 is + and anything to the right of D1 is -. However, this classifier misclassifies three + points.

Box 2: The next classifier says don't worry I will correct your mistakes. Therefore, it gives more weight to the three + misclassified points (see bigger size of +) and creates a vertical line at D2. Again it says, anything to right of D2 is - and left is +. Still, it makes mistakes by incorrectly classifying three - points.

2. Box 3: The next classifier continues to bestow support. Again, it gives more weight to the three -misclassified points and creates a horizontal line at D3. Still, this classifier fails to classify the points (in circle) correctly.
3. Remember that each of these classifiers has a misclassification error associated with them.
4. Boxes 1,2, and 3 are weak classifiers. These classifiers will now be used to create a strong classifier Box 4.
5. Box 4: It is a weighted combination of the weak classifiers. As you can see, it does good job at classifying all the points correctly.

That's the basic idea behind boosting algorithms. The very next model capitalizes on the misclassification/error of previous model and tries to reduce it. Now, let's come to XGBoost.
As we know, XGBoost can used to solve both regression and classification problems. It is enabled with separate methods to solve respective problems. Let's see:
Classification Problems: To solve such problems, it uses booster = gbtree parameter; i.e., a tree is grown one after other and attempts to reduce misclassification rate in subsequent iterations. In this, the next tree is built by giving a higher weight to misclassified points by the previous tree (as explained above).
Regression Problems: To solve such problems, we have two methods: booster = gbtree and booster = gblinear. You already know gbtree. In gblinear, it builds generalized linear model and optimizes it using regularization (L1,L2) and gradient descent. In this, the subsequent models are built on residuals (actual - predicted) generated by previous iterations. Are you wondering what is gradient descent? Understanding gradient descent requires math, however, let me try and explain it in simple words:

- Gradient Descent: It is a method which comprises a vector of weights (or coefficients) where we calculate their partial derivative with respective to zero. The motive behind calculating their partial derivative is to find the local minima of the loss function (RSS), which is convex in nature. In simple words, gradient descent tries to optimize the loss function by tuning different values of coefficients to minimize the error.



Gradient Descent

In R, xgboost package uses a matrix of input data instead of a data frame.Here we select minima to avoid overfitting of data.

Understanding XGBoost Tuning Parameters

Every parameter has a significant role to play in the model's performance. Before hypertuning, let's first understand about these parameters and their importance.
In this article, I've only explained the most frequently used and tunable parameters. XGBoost parameters can be divided into three categories
General Parameters: Controls the booster type in the model which eventually drives overall functioning

- Booster Parameters: Controls the performance of the selected booster
- Learning Task Parameters: Sets and evaluates the learning process of the booster from the given data

1. General Parameters

1. Booster[default=gbtree]
    - Sets the booster type (gbtree, gblinear or dart) to use. For classification problems, you can use gbtree, dart. For regression, you can use any.
2. nthread[default=maximum cores available]
    - Activates parallel computation. Generally, people don't change it as using maximum cores leads to the fastest computation.
3. silent[default=0]
    - If you set it to 1, your R console will get flooded with running messages. Better not to change it.

2. Booster Parameters

As mentioned above, parameters for tree and linear boosters are different. Let's understand each one of them:

Parameters for Tree Booster

1. nrounds[default=100]
    - It controls the maximum number of iterations. For classification, it is similar to the number of trees to grow.
    - Should be tuned using CV
2. eta[default=0.3][range: (0,1)]
    - It controls the learning rate, i.e., the rate at which our model learns patterns in data. After every round, it shrinks the feature weights to reach the best optimum.
    - Lower eta leads to slower computation. It must be supported by increase in nrounds.
    - Typically, it lies between 0.01 - 0.3
3. gamma[default=0][range: (0,Inf)]
    - It controls regularization (or prevents overfitting). The optimal value of gamma depends on the data set and other parameter values.
    - Higher the value, higher the regularization. Regularization means penalizing large coefficients which don't improve the model's performance. default = 0 means no regularization.
    - *Tune trick:* Start with 0 and check CV error rate. If you see train error >>> test error, bring gamma into action. Higher the gamma, lower the difference in train and test CV. If you have no clue what value to use, use gamma=5 and see the performance. Remember that gamma brings improvement when you want to use shallow (low max_depth) trees.
4. max_depth[default=6][range: (0,Inf)]
    - It controls the depth of the tree.
    - Larger the depth, more complex the model; higher chances of overfitting. There is no standard value for max_depth. Larger data sets require deep trees to learn the rules from data.
    - Should be tuned using CV
5. min_child_weight[default=1][range:(0,Inf)]
    - In regression, it refers to the minimum number of instances required in a child node. In classification, if the leaf node has a minimum sum of instance weight (calculated by second order partial derivative) lower than min_child_weight, the tree splitting stops.
    - In simple words, it blocks the potential feature interactions to prevent overfitting. Should be tuned using CV.
6. subsample[default=1][range: (0,1)]
    - It controls the number of samples (observations) supplied to a tree.
    - Typically, its values lie between (0.5-0.8)
7. colsample_bytree[default=1][range: (0,1)]
    - It control the number of features (variables) supplied to a tree
    - Typically, its values lie between (0.5,0.9)

8. lambda[default=0]
    - It controls L2 regularization (equivalent to Ridge regression) on weights. It is used to avoid overfitting.
9. alpha[default=1]
    - It controls L1 regularization (equivalent to Lasso regression) on weights. In addition to shrinkage, enabling alpha also results in feature selection. Hence, it's more useful on high dimensional data sets.

Parameters for Linear Booster

Using linear booster has relatively lesser parameters to tune, hence it computes much faster than gbtree booster.

1. nrounds[default=100]
    - It controls the maximum number of iterations (steps) required for gradient descent to converge.
    - Should be tuned using CV
2. lambda[default=0]
    - It enables Ridge Regression. Same as above
3. alpha[default=1]
    - It enables Lasso Regression. Same as above

3. Learning Task Parameters

These parameters specify methods for the loss function and model evaluation. In addition to the parameters listed below, you are free to use a customized objective / evaluation function.

1. Objective[default=reg:linear]
    - reg:linear - for linear regression
    - binary:logistic - logistic regression for binary classification. It returns class probabilities
    - multi:softmax - multi classification using softmax objective. It returns predicted class labels. It requires setting num_class parameter denoting number of unique prediction classes.
    - multi:softprob - multi classification using softmax objective. It returns predicted class probabilities.
2. eval_metric [no default, depends on objective selected]
    - These metrics are used to evaluate a model's accuracy on validation data. For regression, default metric is RMSE. For classification, default metric is error.

- Available error functions are as follows
  - mae - Mean Absolute Error (used in regression)
  - Log Loss - Negative log likelihood (used in classification)
  - AUC - Area under curve (used in classification)
  - RMSE - Root mean square error (used in regression)
  - error - Binary classification error rate [#wrong cases/#all cases]
  - mlogloss - multiclass logloss (used in classification)

We've looked at how xgboost works, the significance of each of its tuning parameter, and how it affects the model's performance. Let's bolster our newly acquired knowledge by solving a practical problem in R.

Practical - Tuning XGBoost in R

In this practical section, we'll learn to tune xgboost in two ways: using the xgboost package and MLR package. I don't see the xgboost R package having any inbuilt feature for doing grid/random search. To overcome this bottleneck, we'll use MLR to perform the extensive parametric search and try to obtain optimal accuracy.
This data set poses a classification problem where our job is to predict if the given user will have a salary <=50K or >50K.
Using random forest, we achieved an accuracy of 85.8%. Theoretically, xgboost should be able to surpass random forest's accuracy. Let's see if we can do it. I'll follow the most common but effective steps in parameter tuning:

1. First, you build the xgboost model using default parameters. You might be surprised to see that default parameters sometimes give impressive accuracy.
2. If you get a depressing model accuracy, do this: fix eta = 0.1, leave the rest of the parameters at default value, using xgb.cv function get best n_rounds. Now, build a model with these parameters and check the accuracy.
3. Otherwise, you can perform a grid search on rest of the parameters (max_depth, gamma, subsample, colsample_bytree etc) by fixing eta and nrounds. Note: If using gbtree, don't introduce gamma until you see a significant difference in your train and test error.
4. Using the best parameters from grid search, tune the regularization parameters(alpha,lambda) if required.
5. At last, increase/decrease eta and follow the procedure. But remember, excessively lower eta values would allow the model to learn deep interactions in the data and in this process, it might capture noise. So be careful!

Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine.
This makes xgboost at least 10 times faster than existing gradient boosting implementations. It supports various objective functions, including regression, classification and ranking.
Since it is very high in predictive power but relatively slow with implementation, “xgboost” becomes an ideal fit for this data set. It also has additional features for doing cross validation and finding important variables. There are many parameters which needs to be controlled to optimize the model.

Parameters used in Xgboost

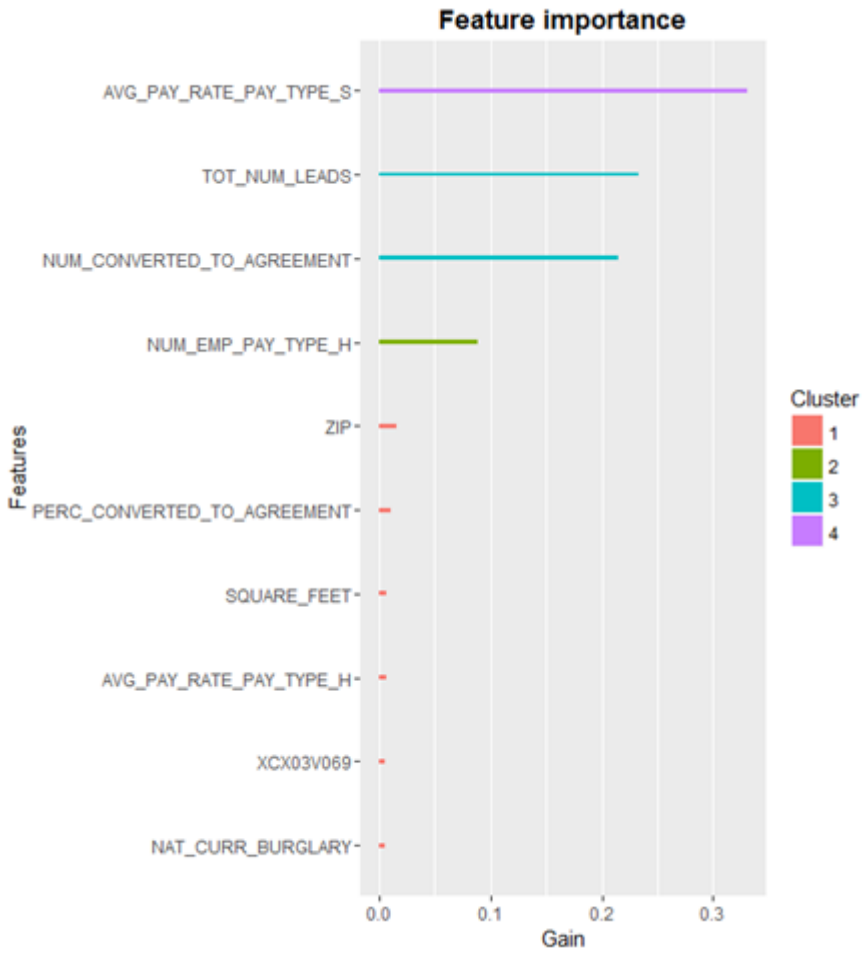
There are three types of parameters: General Parameters, Booster Parameters and Task Parameters.

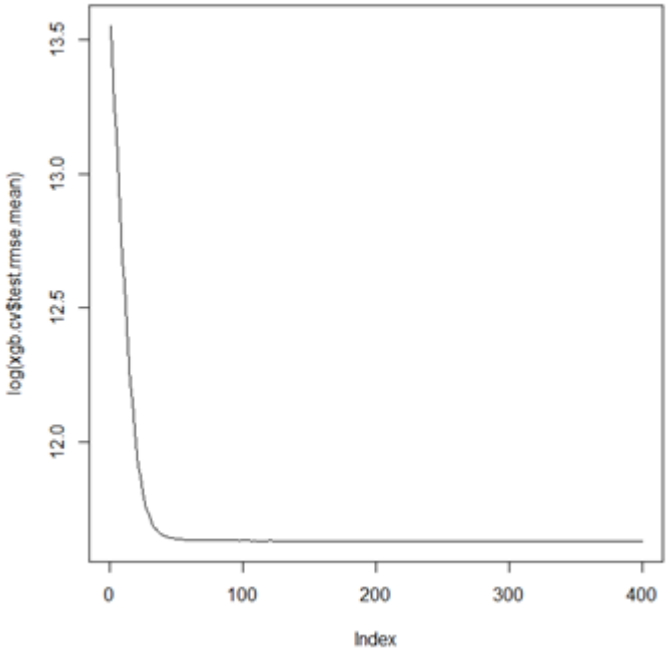
Booster Parameters

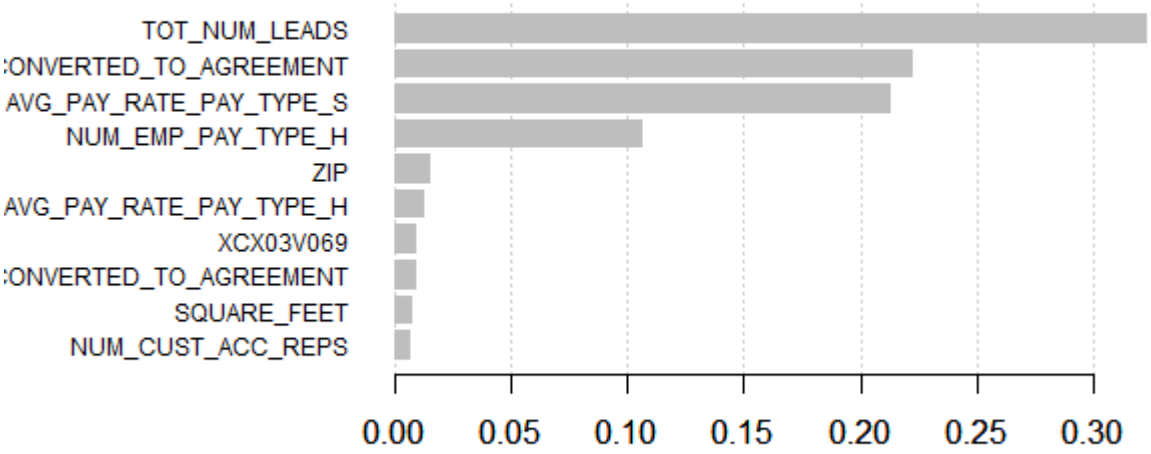
The tree specific parameters –

- eta : The default value is set to 0.3.We changed it to 0.1 to avoid from overfitting..
- max_depth :Set to 11.
- Nthred is set to 2.
- Nround is set to 600

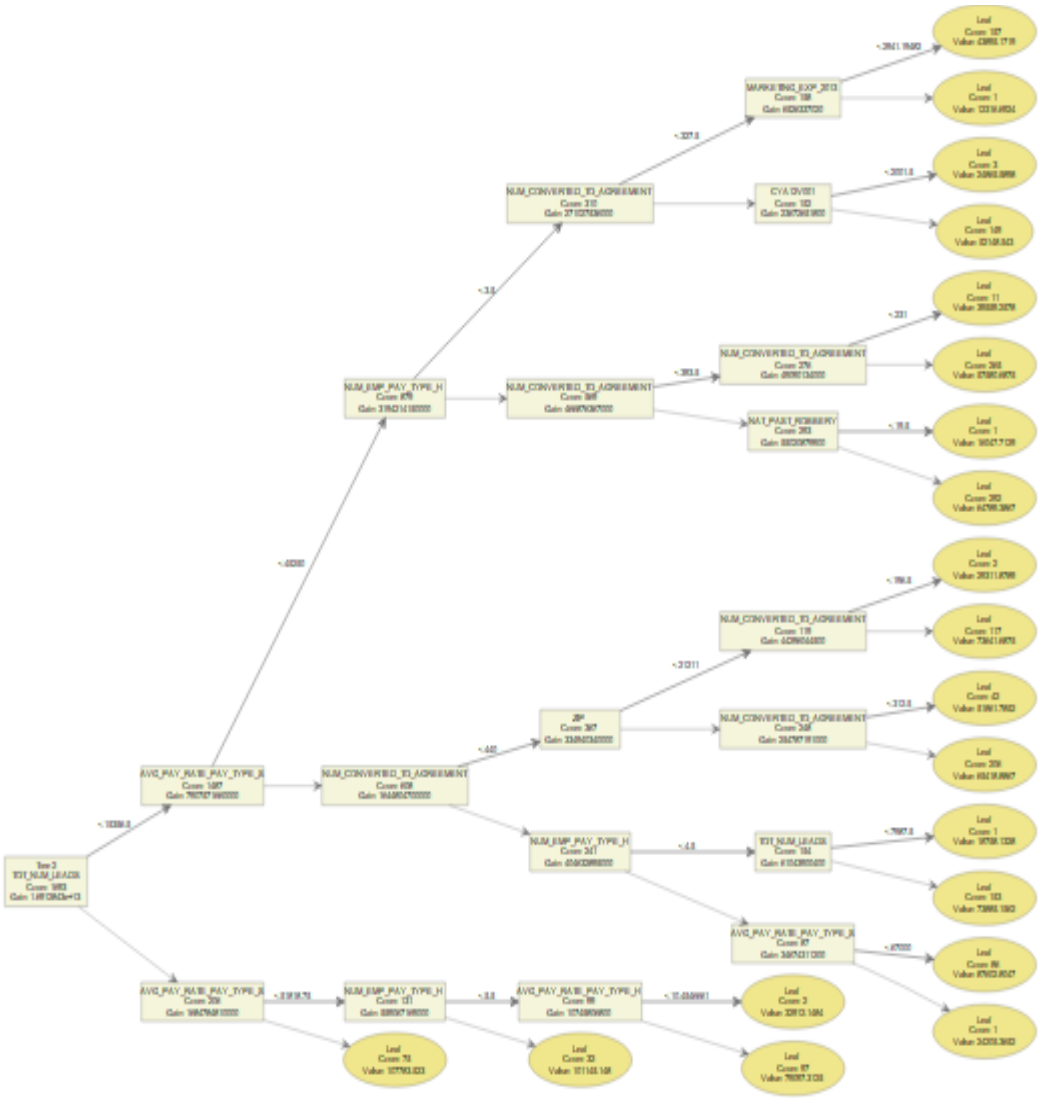
Rest of the values are set to default


Feature importance
AVG_PAY_RATE_PAY_TYPE_S
TOT_NUM_LEADS
NUM_CONVERTED_TO_AGREEMENT
NUM_EMP_PAY_TYPE_H
ZIP
PERC_CONVERTED_TO_AGREEMENT
SQUARE_FEET
AVG_PAY_RATE_PAY_TYPE_H
XCX03V069
NAT_CURR_BURGLARY
Features
0.0
0.1
0.2
0.3
Gain
Cluster
1
2
3
4





PLOT OF IMPORTANCE MATRIX FOR FIRST 10 ELEMENTS



Decision Tree for a weak learner in XGBoost

REFERENCES

[1] Wensen Dai et al. “A Clustering-based Sales Forecasting Scheme Using Support Vector Regression for Computer Server.” Procedia Manufacturing 2 (2015) 82 – 86.

[2] Marco Hulsmann et al. “General Sales Forecast Models for Automobile Markets and their Analysis.” Transactions on Machine Learning and Data Mining Vol. 5, No. 2 (2012) 65-86.

[3] "Analytics for an Online Retailer: Demand Forecasting and Price Optimization” case study by hbs by Kris Johnson Ferreira,Bin Hong Alex Lee and David Simchi-Levi.

[4] A Service-Oriented Solution for Retail Store Network Planning. Xinxin Bai,Wei Shang,Wenjun Yin, Jin Dong.

[5] A sales forecasting model for consumer products based on the influence of online word-of-Mouth.Ching-Chin Chern,Chih-Ping Wei,Fang-Yi Shen,Yu-Neng Fan.

[6] Simulation Based Sales Forecasting On Retail Small Stores. Hai Rong Lv Xin Xin Bai Wen Jun Yin Jin Dong.

[7] Service-Oriented Enterprise: The Technology Path to Business Transformation.”, http://www.intel.com/ business/bss/technologies/soe/ accessed at Jan. 15th, 2007.

[8] http://en.wikipedia.org/wiki/R(programming_language)

[9]http://www.yellowfinbi.com/YFCommunityNews-DataAnalysis-for-the-Retail-Industry-Part-1-100068

CODE

```
#project code
library(caret) # for dummyVars
library(RCurl) # download https data
library(Metrics) # calculate errors
library(xgboost) # model
library(MASS)
library(Metrics)
library(corrplot)
library(randomForest)
library(lars)
library(ggplot2)
library(xgboost)
library(Matrix)
library(methods)
#load the csv train and test files into train1 and test1 respectively.
train1=read.csv(choose.files(),header = TRUE,stringsAsFactors = T)
test1=read.csv(choose.files(),header = TRUE,stringsAsFactors = T)
#count the null values and delete the rows
for(i in c(1:56))
{
 count=0;
 count<-sum(is.na(train1[,i]))/1693;
 if(count>.8)
 {
  print(i);
 }
}
```

```
#As the count is very less,we will not delete the attributes.
#test data cleaning
#have set all other null values to zero
test1[is.na(test1)]<-0
#change categorical data into numeric data
for(i in 1:48){
  if(is.factor(test1[,i])){
    test1[,i]<-as.integer(test1[,i])
  }
}
#train data
#train data cleaning
train1[is.na(train1)]<-0
#convert categorical data into the numeric data
for(i in 1:49){
  if(is.factor(train1[,i])){
    train1[,i]<-as.integer(train1[,i])
  }
}
#divided the training data into 2 sets Training_Inner and Testing_Inner to check the model
Training_Inner<- train1[1:floor(length(train1[,1])*0.8),]
## We have done crossm validation for the Test_Inner values which is 20 percent of the train
data.
Test_Inner<- train1[(length(Training_Inner[,1])+1):1680,]

set.seed(10)
param<- list("objective" = "reg:linear",
             "eval_metric" = "rmse",
             "eta" = 0.1, "max.depth" = 11)
xgb.cv = xgb.cv(param=param,data = as.matrix(train1[,3:49]),seed=1,
label=train1$REVENUE_2013,nfold = 40, nrounds = 400)
```

```
plot(log(xgb.cv$test.rmse.mean),type = "l")
xgb<- xgboost(data = as.matrix(train1[,3:49]), label =train1$REVENUE_2013, max.depth = 11,
              eta = 0.1, nround = 600,
nthread = 2,seed=1, objective = "reg:linear")

trees = xgb.model.dt.tree(dimnames(train1[,3:49])[[2]],model = xgb)
trees
#to display the most important attributes of the dataset
model <- xgb.dump(bst, with.stats = T)
model[1:10]
names <- dimnames(train1[,3:49])[[2]]
names
importance_matrix<- xgb.importance(names, model = xgb)
importance_matrix
#plot the importance matrix which will display the 10 most important attributes.
xgb.plot.importance(importance_matrix[1:10,])
xgb.plot.tree(feature_names = names, model = xgb, n_first_tree = 2)
#Then following code is used for the final prediction of the REVENUE_2013  from the train
model xgb for the test data.
y_pred<- predict(xgb, data.matrix(test1[,2:48]))
y_pred
#convert the data into csv file and export it
write.csv(y_pred,file="C:/Users/dell pc/Desktop/project/output1.csv")
```