USE CASE STUDY REPORT

STUDY ON BLACK FRIDAY PURCHASES IN A RETAIL COMPANY

GROUP 24: HIMANSHU SAXENA, ANKITA PRADEEP

EXECUTIVE SUMMARY

The aim of the study is to build a model for a retail company where the model will predict the purchase amount of customer against various products. This will help the company create a personalized offer for customers against different products. The data was obtained from Analytics Vidya, a popular site that contains several datasets for analysis. It contained 12 variables. Feature engineering was conducted to convert the variables into suitable numerical values to facilitate smooth analysis. The analysis was done using R programming. Various exploratory data analysis plots were constructed to examine the relationship and distribution of different variables and how they contribute to the target variable. Data mining models such as classification and regression tress, multiple linear regression and extreme gradient boosting were implemented to predict the purchase prices of every customer. The evaluation metric was the RMSE value. This was computed and compared for different models and it was found that XG boost had the minimum and most optimum RMSE value. The predictions obtained can be used to decide the kind of personalized offers to be created for different users.

I. BACKGROUND AND INTRODUCTION

Black Friday is one of the biggest days in the United States for all retail stores and people looking to shop for various consumer products. The year's greatest sales and discounts are available on this day and the various retail stores need to prepare themselves for the sudden surge in the demand for products. Preparation for this event includes an estimation of the number of customers that are expected to shop, the kind of products they are likely to buy and the purchase amount by each customer across different product categories. This enables the sellers to plan their orders, stock their stores with an adequate number of products, devise their profit margins, increase their sales by using recommendation tools to the customers based on their historical buying patterns and be prepared to give the customers a good experience.

This use case aims at helping the sellers do each of these tasks by predicting the purchase amount and thereby assisting them with the knowledge of what products should they recommend to a particular user so that they can increase the chances of converting that recommendation into a sale.

The solution will give an idea about the kind of products every user is likely to shop for and the total amount for which he might make the purchases and how this amount is spread across different product categories.

II. DATA EXPLORATION AND VISUALIZATION

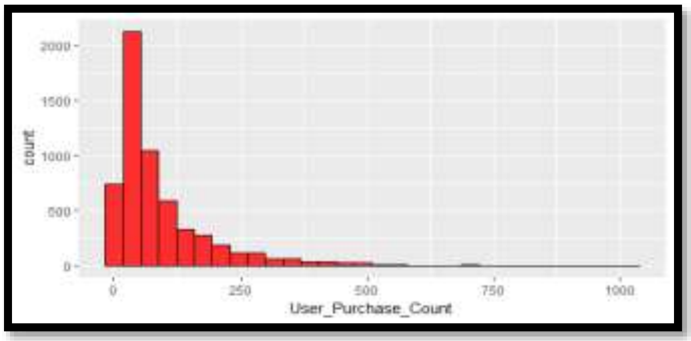
The data was collected from a website called Analytics Vidya which hosts several data hackathons. The dataset has about 550069 rows and 12 columns or variables. The data was extremely clean. It contains the purchase summary of various customers for selected high volume products from a particular month. Various demographics of the customer such as age, gender, marital status, city-type, number of years of stay in the current city, product details and the total purchase amount from the last month are available.

The data was split into training set was split into train and validation set in the ratio 3:2. The test was separately available. The dataset was then observed for missing values. It was found that most parts of the data was clean except that there were missing values in Product category 2 and 3. This essentially means that that particular customer did not shop for products in these categories and the value of these blank rows is considered as zero.

All the variables are then encoded into factors. For example, the age group of different bins was converted into 7 levels of factors, occupation into 21 levels, city category into 3 levels, marital status and gender into 2 levels and number of years of stay in current years as 5 levels.

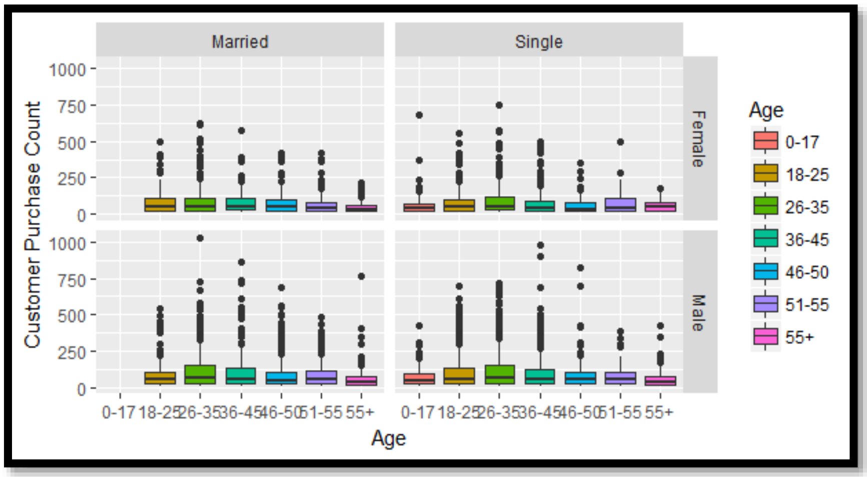
Following this, the count of products purchased by each distinct user ID was computed both in the training and test datasets. Also, the average spending of each user in the last one month was computed. Univariate analysis is applied to study the pattern of distribution of the target variable which is the purchase. It was found that the minimum spending is 46680 and mean is 865000, the maximum being 10540000. This was plotted on the graph and visualized using geom() function in ggplot.

The summary of the distinct user vs the purchase count id found out and plotted. The count of the purchases made is plotted against the variables age, age, occupation, stay in the current city and marital status. The results obtained are depicted below:



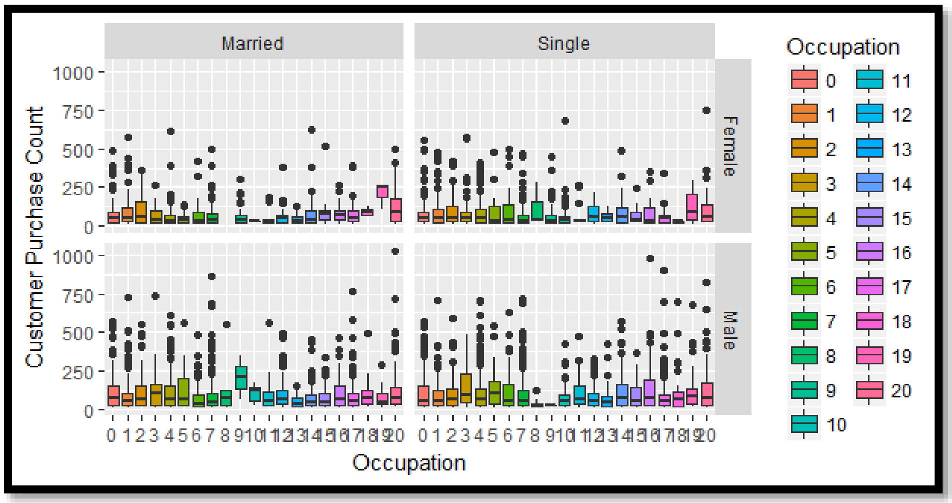
It is observed that the distinct user purchase count goes up to a maximum of a little over 2000 and on average lies between 500-1000.

We further explored the distribution of the customer purchase based on different parameters like marital status, age, and gender.

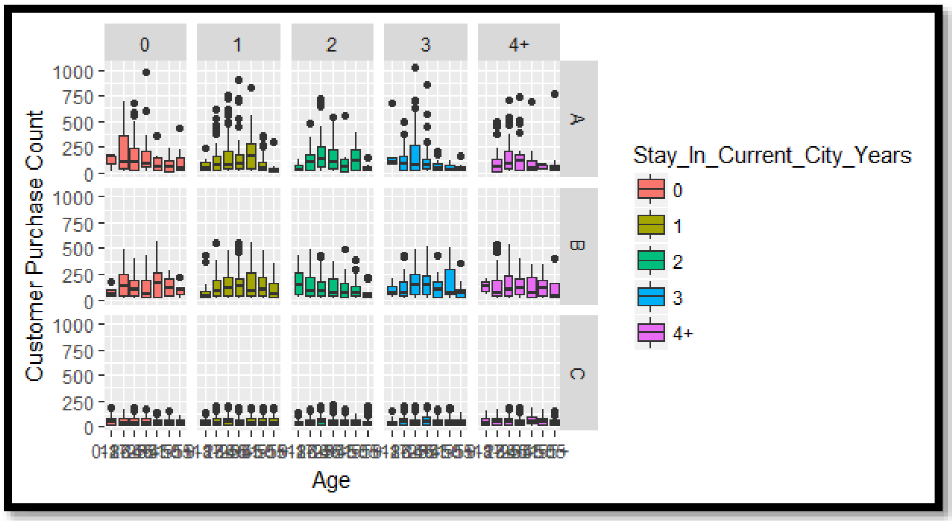


From the grid, it can be concluded that a maximum number of purchases are made by users that fall in the age group of 26-35 and it is high among married females. This group is followed by single males in the age group of 26-35 and 36-45.

Furthermore, another grid considering the four variables of gender, marital status, purchase count and occupation was drawn. The plot is as depicted below.

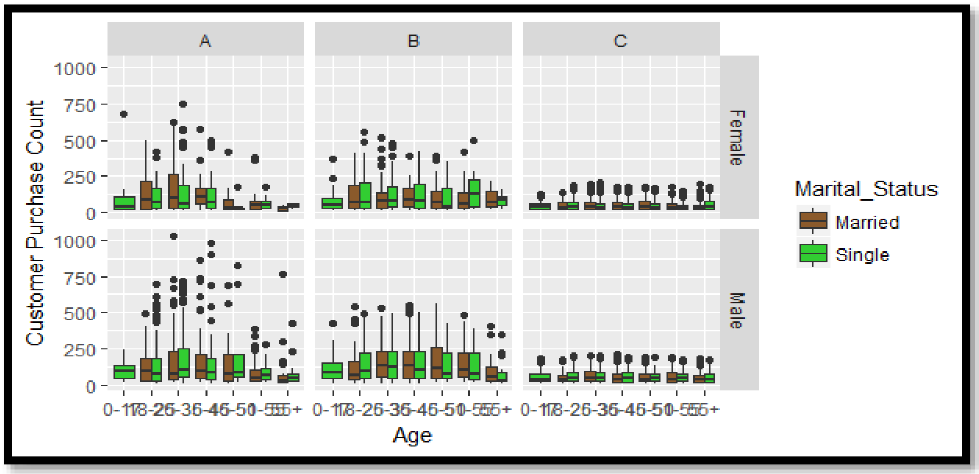


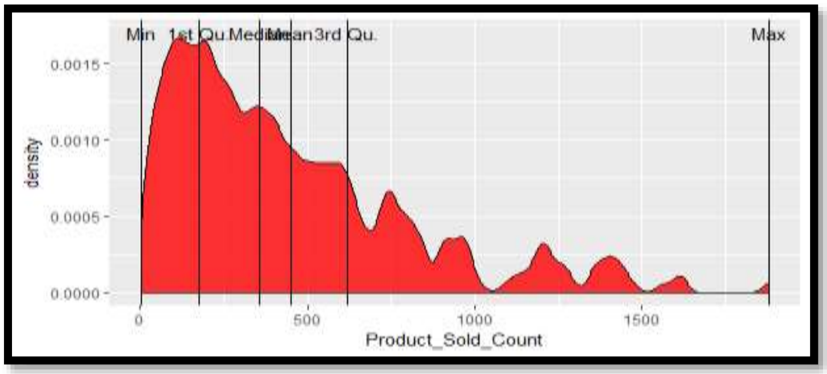
From the plot, it can be interpreted that the distribution of customer purchase count varies significantly among all the occupations. The section of users belonging to married females and single men falling under the occupation 20 and 3 display the maximum range of purchases of different amounts. The least amount of purchases is observed by occupation category 8 and 9 in single males. Among single females, there is a very low purchase trend observed in user falling under the occupation categories of 11 and 18. All married men show significant interest in purchasing as seen from the graph. There have been no purchases made by occupation 8 category of married women.

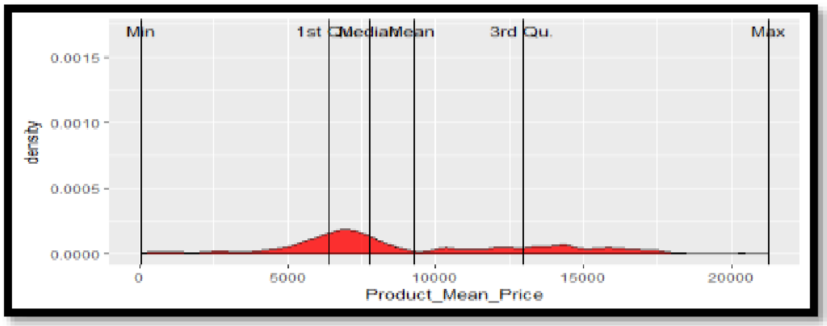


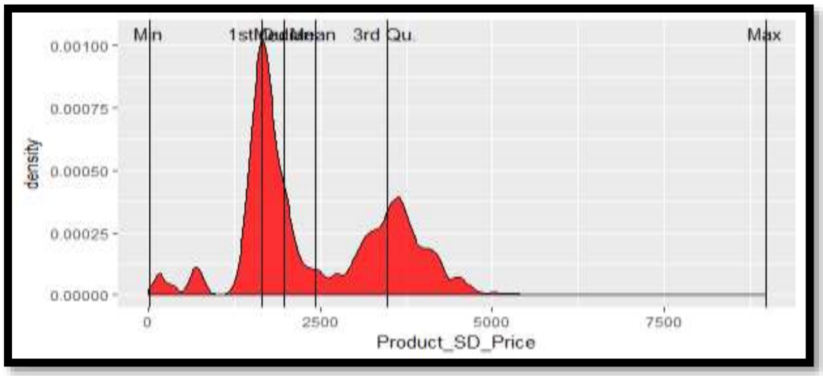
The above graph displays the customer purchase count based on the three variables – the current city category, age and the number of years of stay in the current city. It can be seen that the customer purchase count is least in the city category C and there are a significant number of users (between 0-350) in both city A and city B mostly belonging to the section of people who have just moved into the city. The randomness observed is highest in people residing in city A from 1 year as there are several high counts of purchasers.

Also, the customer purchase count was explored against four more variables namely, age, city category, gender and marital status. It is seen that in city category A, married women have a higher purchase count in the age group 18-25 and 26-35. In the city B, the customer purchase count is higher amoung single females. However, in city C, the customer purchase count is extremely low across both single and married females.





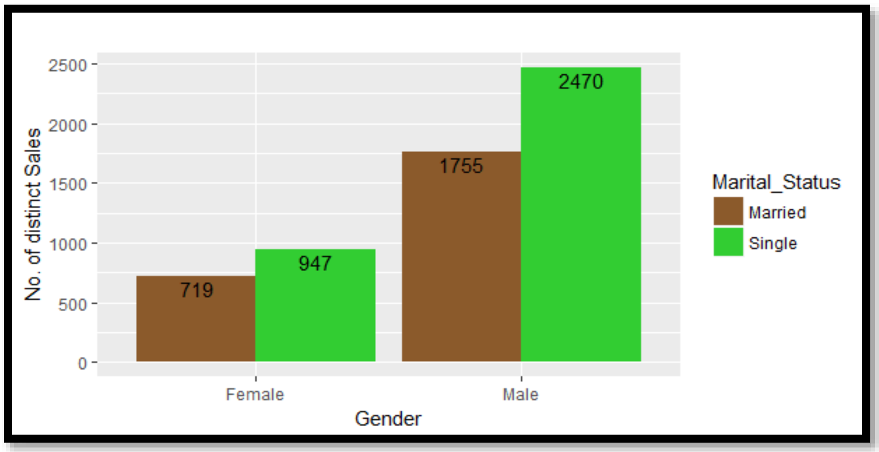




In the above graphs, it can be seen that the density of the sold products count, the product mean price and the product SD price is plotted. It can be understood that the most of the products sold lies between the count of 0-500 and dwindles from there on. The product mean the price is also the concentrated in the first two quartile ranges after which the curve becomes flat. The product SD price is the highest between the first quartile and the median at an approximate value of 1300.

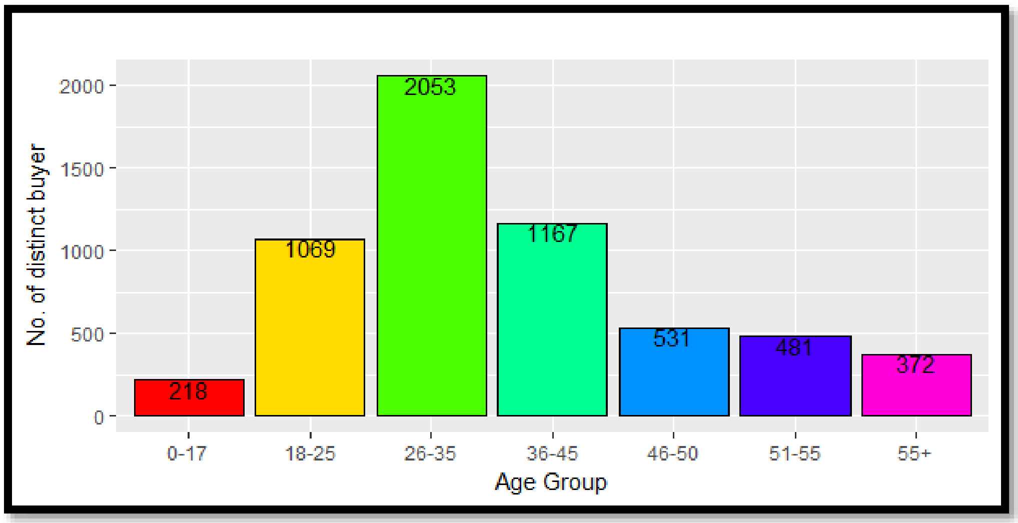
After further exploration of the data about the product that sold most frequently, we find out that there is 1880 number of products carrying the ID P00265242 and this had the maximum demand among all the other products. On the other hand, there are 6 products in the dataset which have the least rate of selling. Just one product of each of these 6 products was bought by the customers indicating a very low demand.

On exploration of the distribution of the demographics of the customers based on marital status and gender, the following results were obtained



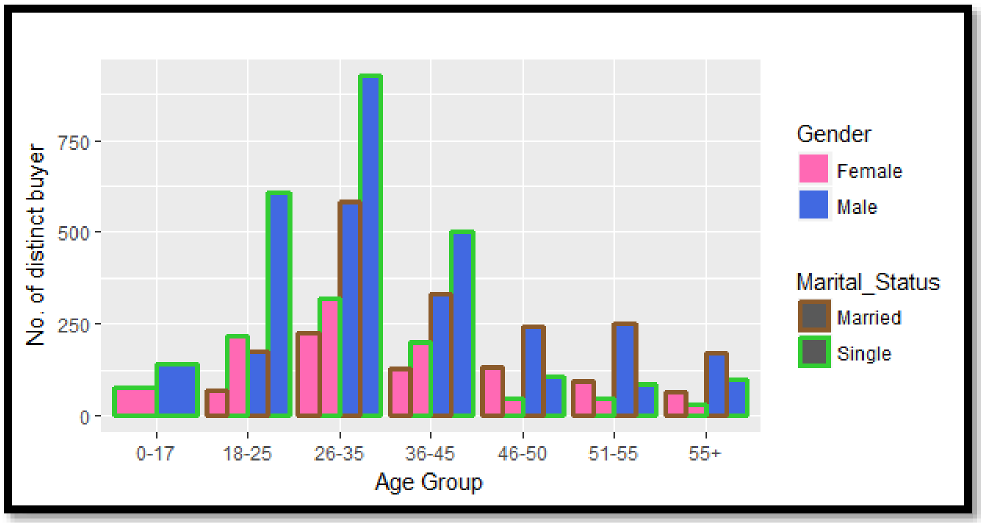
From the above graph, it can be seen that there were 719 female married customers and 1175 male married customers. The distribution in the unmarried/single category is 947 females and 2470 males. It can be concluded that single men formed the maximum percentage of customers giving sales to the retail company.

The distribution of the number of customers according to their age groups can be classified as below:

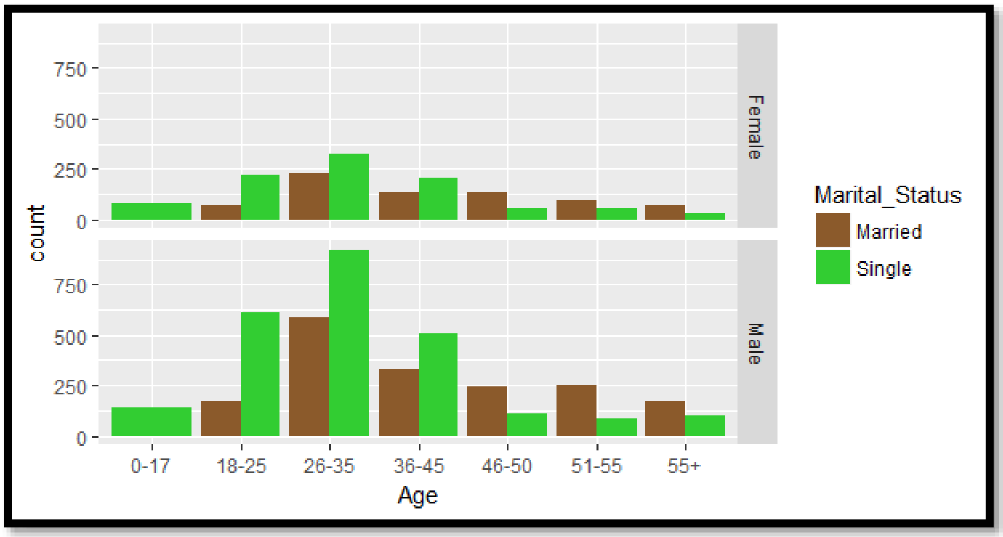


From the graph, it can be concluded the highest percentage of users who purchased the products fell under the 26-35 age group category and 0-17 age group forms the least percentage of users.

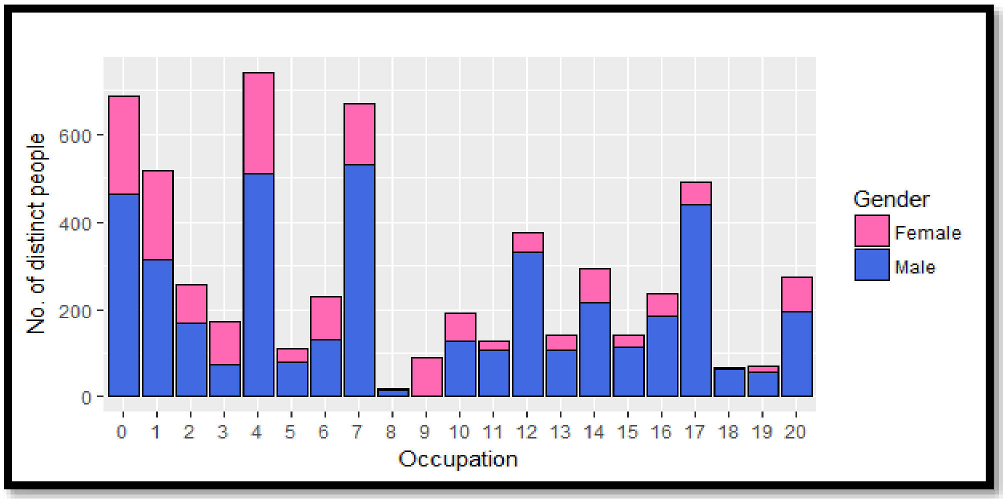
We also studied the distribution of buyers according to the variables gender, age, and marital status. It is observed that single men under the age group of 26-35 are the highest number of buyers followed by single men in the age group of 18-25. Surprisingly the number of female buyers is consistently low across all the age groups with the exception of 26-35.



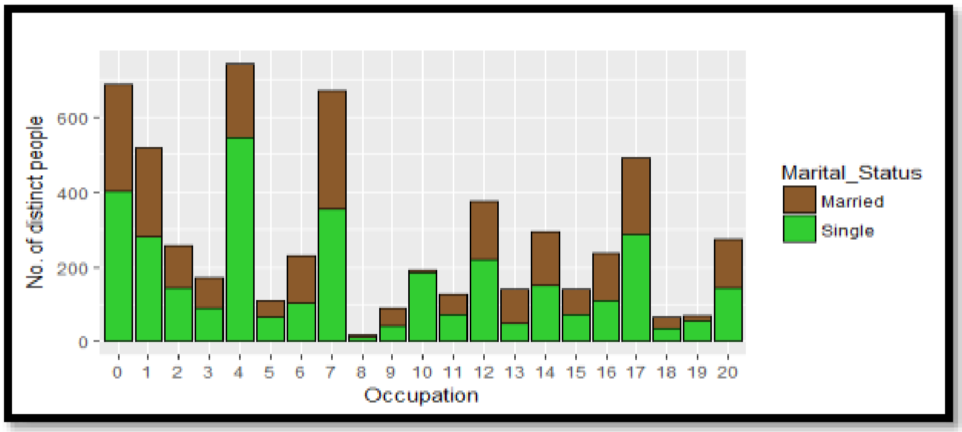
It can be seen that the number of distinct buyers has more males in the age group 0-17 as compared to females. In the age group of 18-25, single men give the highest revenue whereas married females contribute to the least purchases. In the age group 26-35 again it is the single men who dominate, followed by married men, single women, and married women. Single men of this age group form over 1000 distinct buyers thereby being the highest across all demographics. In the age group of 36-45 too the same trend is observed except that the count of distinct buyers is relatively very low. In the age group of 46-50, married men contribute to the highest buyers section follows by single men, married women, and single women. In the age group 51-55, married men again form the highest section of buyers. The same distribution is observed across the age group of 55+ as well.



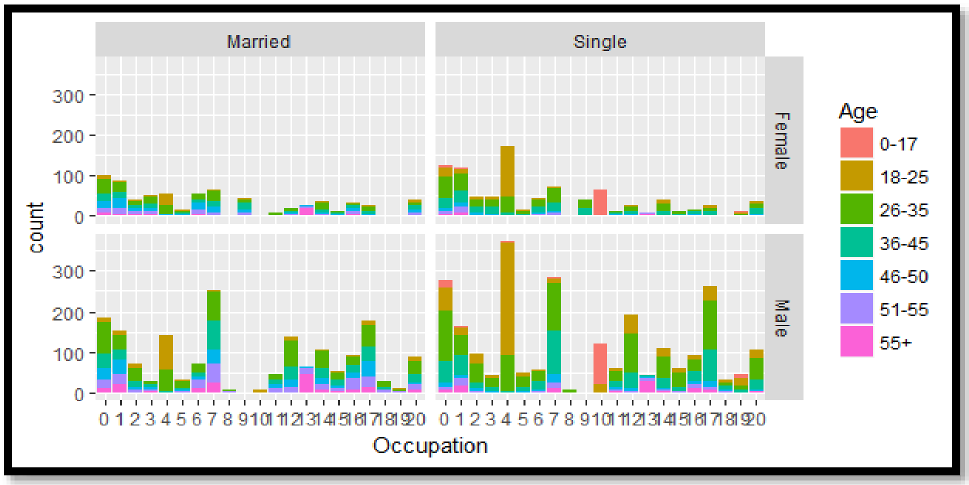
The above distribution again depicts the count of buyers based on gender, marital status and age.



The above graph shows the count of distinct buyers across the 20 categories of occupation with respect to the gender. It can be concluded that males in the occupation categories 7,4 and 0 form the highest number of buyers in decreasing order. Females in the occupation category 4 and 0 also significantly contribute to the count of a number of buyers.

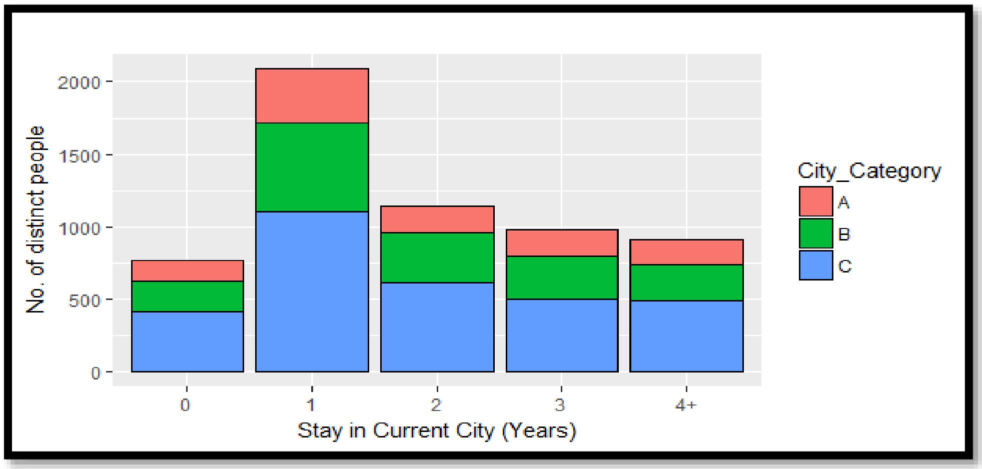


The same data is further explored with respect to occupation and marital status. It can be seen that the highest number of buyers are spread across occupation categories 4,7,0, 17 in the decreasing rank order. The lowest number of buyers are observed in occupation category 8 followed by occupation categories 18 and 19.

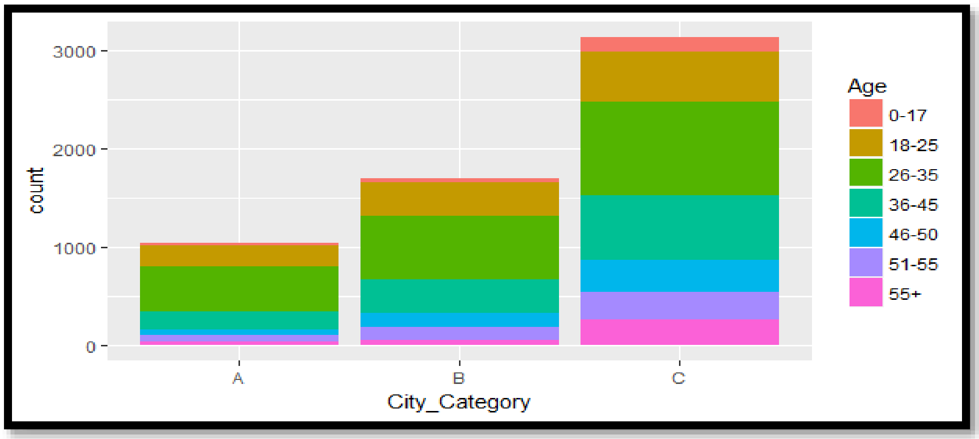


The two variables of stay in the current city and the city category were explored to find out their impact on the count of buyers. It was observed that the highest level of buyers fell in the age group of 18-25 and specifically composed of single males. The single females in the same occupation group of 4 and the age group also constituted a high section of buyers. Occupation category 8,9 and 10 had the least count of buyers across all demographics. In general, the count of a married female was the least among all other groups. It is also observed that single men contribute to the highest section of buyers across all occupational groups.

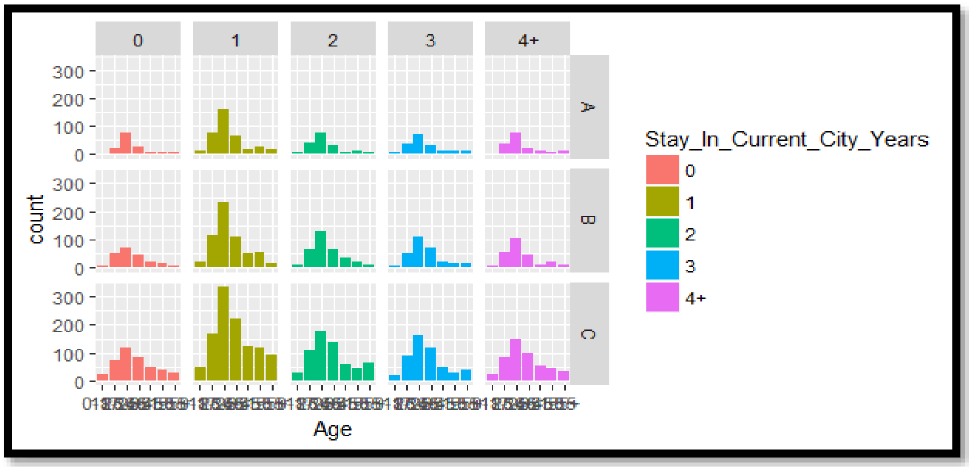
The below graph shows the distribution of distinct buyers amongst different city categories based on the number of years they have lived in the city. In the group where buyers have lived in a city for less than a year, city C forms the highest number. In the group where buyers have been living in the city for a year, city C and B both give a good number of buyers with city C giving more than 1000 buyers. This section of people contributes to the highest number of buyers. The least number of buyers are from the last category where people have been residing in a city for over 4 years. The number of buyers from the city A is consistently low among all the categories. The buyers from city B form an average level of the count and the buyers are mostly from city C.



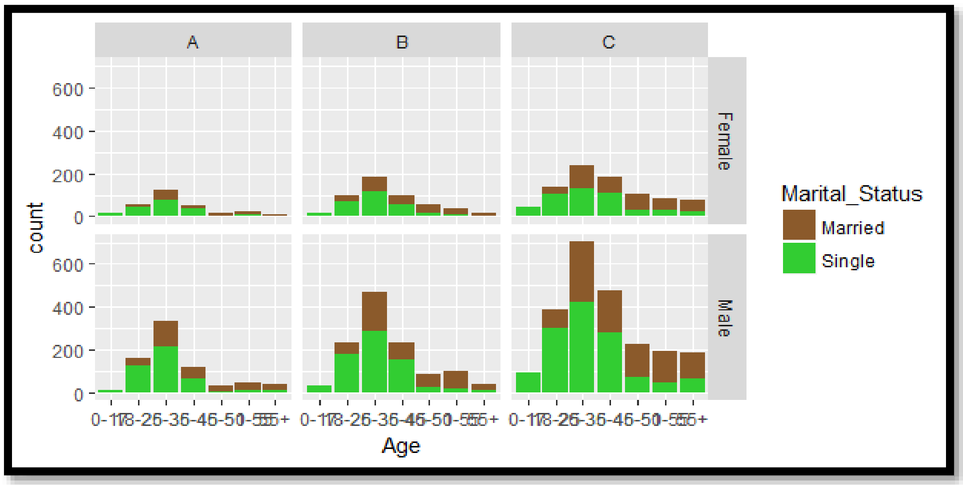
This graph below explores the count of buyers across various age groups based on the city category. It can be observed that the age group 26-35 forms the maximum section of buyers in all the city categories while people falling in the age group of 55+ form the least section of buyers in all the cities. The second highest group of buyers is constituted of people falling in the age group of 18-25. Also, the number of buyers in the age group 0-17 is almost negligible except in the city category C.



We also explored the count of buyers according to the number of years of stay in the current city, the city category and the age to create a matrix or grid of this visualization. The highest number of buyers are from city C and comprise of people who have been residing here for over a year. This is closely followed by the same category of one year residents in both city B and A. The least number of buyers are people who have just moved into the city. Also, the residents in city A who have stayed over 3 years have not been active buyers. Likewise with residents of city A staying for over 4 years in the city. This indicates that the company has a very good chance of sales in city C and they should target people who have been residing in city C for over a year. They can expect maximum sales from them.



A grid was created to explore the count of buyers in different cities according to gender and marital status. It is observed from the first grid that in city A, single men and women in the age group 25-36 are the most significant section of buyers. In the second grid of city B, again the buyers belong to single men but the count of married women buyers is higher in city B as compared to A. The section of female buyers both married and single is highest in C as compared to the other cites. Same is true for the single and married men of the city C who form the maximum number of buyers across all cities and demographics.



III. DATA PREPARATION AND PREPROCESSING

The data obtained was already clean with no missing values in most columns. The variables Product category 2 and product category 3 had missing values of 72344 and 162562 respectively indicating that these categories of products were not purchased by the user. These values were substituted by 0 thereby treating all the missing values in the dataset. This was applied to the training set as well.

Every variable was encoded into factors. The product ID was converted into factors of 3631 levels. The user ID factor count was 5891. This is done to signify the unique product ID and user IDs. The gender strings male and female were encoded as 1 and 2 respectively. Age groups were defined as seven factors ranging from 1-7 depicting the 7 different bins of age groups in the data. The occupation variable was encoded into 21 levels of factors. The city category of A, B, and C was written as 1, 2 and 3.The stay in the current city has five levels from 1 to 5 signifying the number of years of stay in the current city. The marital status variable was encoded as 1 and 2 for single and married respectively.

Further, there are two new variables that were created. The user purchase count which indicates the number of purchases made by every user across different product categories. The second variable created was the Product sold count which indicates the count of each product ID that was sold.

IV. DATA MINING TECHNIQUES AND IMPLEMENTATION

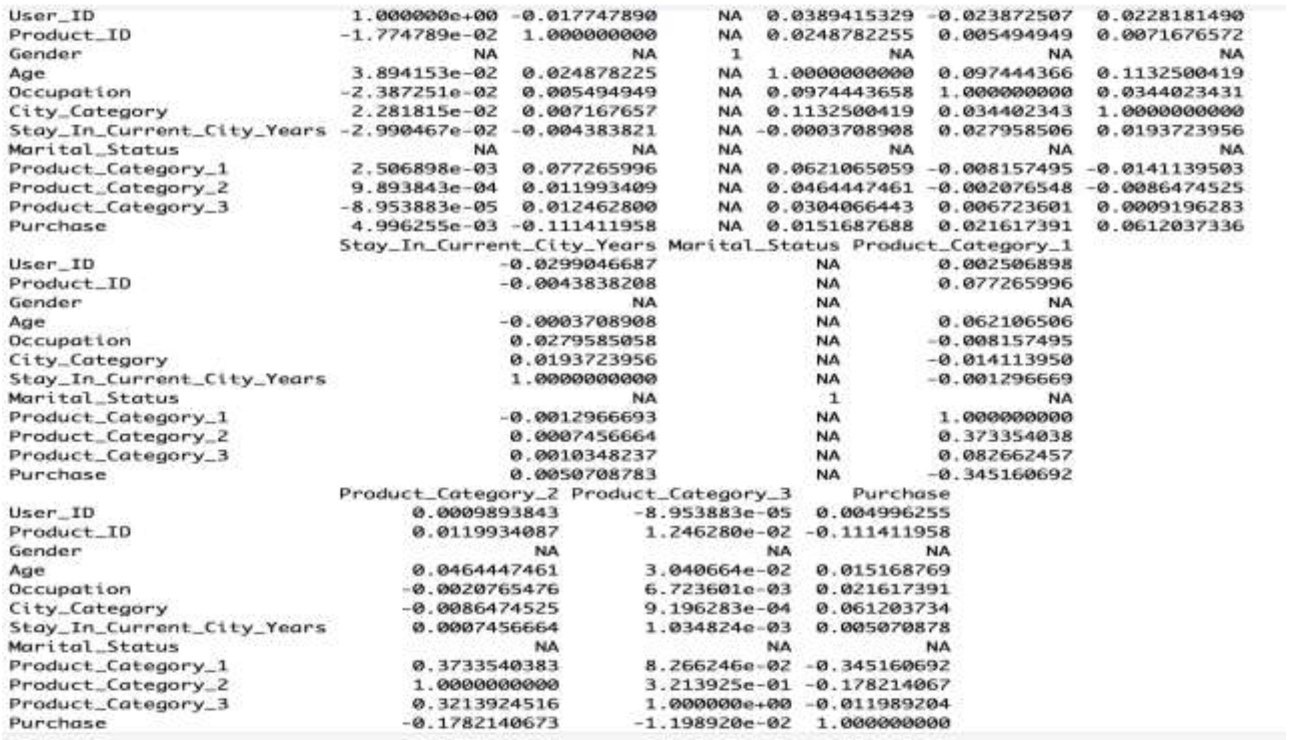
The following approaches were considered for the implementation of data mining techniques

Multiple Linear Regression
As the objective of the problem was to predict the purchase amounts for the test set, Multiple Linear Regression was considered as the first choice for predicting the results. This was also supported by the fact that the predictor variables consisted of categorical and numeric data types.

The dataset (train.csv and test.csv) was imported into the Rstudio. Then, data was processed by replacing the missing values in the product category 2 and product category 3 with the mean of the dataset. This was done to avoid any loss of data and avoid any change in the mean of the data. The structure of the dataset was analyzed and necessary transformations were made. The User_ID, Occupation, and Marital_Status were factorized. Product Category and Purchase were converted to numeric data type.

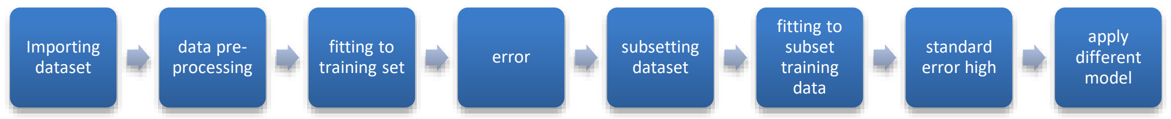
As the test set had no purchase data, it could not be used to evaluate the model performance. Therefore, train data was split into the training data and validation data in the ratio of 3:2. Necessary packages were loaded and a linear model was fitted onto the training set. On running the code, the system showed error –" could not allocate memory of 23.6 GB".

Alternatively, a subset of 10000 rows was selected at random and then linear regression was fitted again. This time, the model fitted successfully but gave an RMSE of 2029. We decided to move ahead with a better model for lower error.



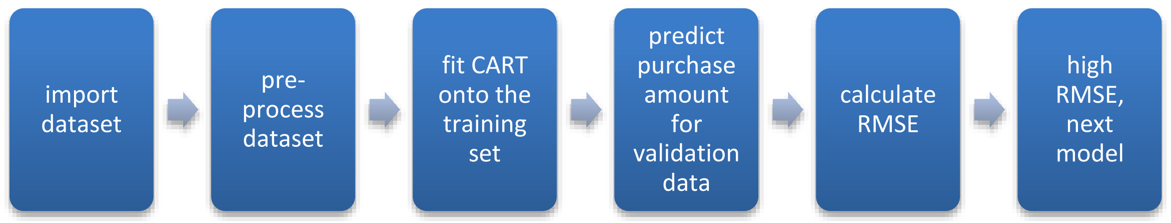
```
User_ID                      1.000000e+00 -0.017747890     NA  0.0389415329 -0.023872507  0.0228181490
Product_ID                  -1.774789e-02  1.000000000     NA  0.0248782255  0.005494949  0.0071676572
Gender                                 NA           NA      1            NA           NA            NA
Age                          3.894153e-02  0.024878225     NA  1.0000000000  0.097444366  0.1132500419
Occupation                  -2.387251e-02  0.005494949     NA  0.0974443658  1.000000000  0.0344023431
City_Category                2.281815e-02  0.007167657     NA  0.1132500419  0.034402343  1.0000000000
Stay_In_Current_City_Years  -2.990467e-02 -0.004383821     NA -0.0003708908  0.027958506  0.0193723956
Marital_Status                         NA           NA     NA            NA           NA            NA
Product_Category_1           2.506898e-03  0.077265996     NA  0.0621065059 -0.008157495 -0.0141139503
Product_Category_2           9.893843e-04  0.011993409     NA  0.0464447461 -0.002076548 -0.0086474525
Product_Category_3          -8.953883e-05  0.012462800     NA  0.0304066443  0.006723601  0.0009196283
Purchase                     4.996255e-03 -0.111411958     NA  0.0151687688  0.021617391  0.0612037336
                           Stay_In_Current_City_Years Marital_Status Product_Category_1
User_ID                                 -0.0299046687             NA        0.002506898
Product_ID                              -0.0043838208             NA        0.077265996
Gender                                             NA             NA                 NA
Age                                     -0.0003708908             NA        0.062106506
Occupation                               0.0279585058             NA       -0.008157495
City_Category                            0.0193723956             NA       -0.014113950
Stay_In_Current_City_Years               1.0000000000             NA       -0.001296669
Marital_Status                                     NA              1                 NA
Product_Category_1                      -0.0012966693             NA        1.000000000
Product_Category_2                       0.0007456664             NA        0.373354038
Product_Category_3                       0.0010348237             NA        0.082662457
Purchase                                 0.0050708783             NA       -0.345160692
                           Product_Category_2 Product_Category_3     Purchase
User_ID                          0.0009893843      -8.953883e-05  0.004996255
Product_ID                       0.0119934087       1.246280e-02 -0.111411958
Gender                                     NA                 NA           NA
Age                              0.0464447461       3.040664e-02  0.015168769
Occupation                      -0.0020765476       6.723601e-03  0.021617391
City_Category                   -0.0086474525       9.196283e-04  0.061203734
Stay_In_Current_City_Years       0.0007456664       1.034824e-03  0.005070878
Marital_Status                             NA                 NA           NA
Product_Category_1               0.3733540383       8.266246e-02 -0.345160692
Product_Category_2               1.0000000000       3.213925e-01 -0.178214067
Product_Category_3               0.3213924516       1.000000e+00 -0.011989204
Purchase                        -0.1782140673      -1.198920e-02  1.000000000
```

Correlation between predictor and response variables for train dataset



Process flow diagram for Multiple Linear Regression

Classification and Regression Trees (CART)
The next approach involved using CART because it works well with all types of predictor variables and thus does not warrant any transformations of variables. Also, variable subset selection is automatic as it is a part of the split selection. Moreover, trees are robust to outliers.
The process was similar to the previous model. The dataset was imported and preprocessed. Again, the train.csv was broken into a training set and validation set.
The model was fitted onto the training set and predicted purchase amounts for the validation set. The RMSE was computed as the performance measure which came out to be 2824. The RMSE was high and a decision of moving forward with the better model was made. However, the additional advantage of this model over Multiple Linear Regression was that CART did not have a problem working with the full dataset involving half a lakh rows.


import dataset
pre-process dataset
fit CART onto the training set
predict purchase amount for validation data
calculate RMSE
high RMSE, next model

Process flow diagram for Classification and Regression Technique

XG Boost or Extreme Gradient Boosting
The next choice of model selection was gradient boosted trees. The selection was made because of the advantages this model has over all other models. In this, trees are built one at a time where each new tree helps to correct errors made by the previously trained trees. By varying the parameters like depth of the tree, number of trees and learning rate, more favorable outcome can be achieved. The data is imported and all the categorical predictors are converted into numeric because XGBoost works best with the numeric predictors. The dataset was divided into training and test set and then the model was fitted onto the training set. The parameter grounds was varied from 500 to 700 for comparison of RMSE value.
The RMSE value for training data for 500 rounds was 2225. For 700 rounds, the RMSE decreased to 2084. The model was fitted onto the validation dataset, the RMSE for 500 rounds was 2148 and for 700 rounds it was 1991. This RMSE is 1.8% improvement over the Multiple Linear Regression and 29.45% improvement over the CART methodology.

Thus, we can see as the rounds increased the RMSE decreased further.

V. PERFORMANCE EVALUATION

The output of a machine learning model is a numeric value for predicting the target variable which in this case is purchase amount. Measuring the model accuracy is done using Root Mean Square Error (RMSE) metric which is an industrial standard. It is a measure of distance between the predicted value and the actual numeric value. The rule of thumb is, the smaller the RMSE, the better if the predictive accuracy of the model. In the ideal case, the model with a perfectly correct prediction would have an RMSE of 0.

The formulae for RMSE :

$$RMSE = \sqrt{1/N) \sum_{i=1}^{N} (actual\ target - predicted\ target)^2}$$

The RMSE values for the various models that have been used are:

Model	**RMSE**
Multiple Linear Regression	2405
CART	2824
XGBoost (validation data, nrounds= 700)	1991

The XGBoost RMSE is 1.8% improvement over the Multiple Linear Regression and 29.45% improvement over the CART methodology. Even though Multiple Linear Regression produced a comparable RMSE, there was compatibility issue of fitting the full dataset into the model. Thus, XGBoost was the most preferred model for predicting the Purchase amount for the test set.

The predicted purchase amounts for test set are saved in “Submit XGBoost.csv”

VI. DISCUSSION AND RECOMMENDATIONS

Through the course of the project, we could understand the practical implementation of everything that we learned in the coursework. By exploratory data analysis, the different factors could be studied well for their relevance to the target variable which is purchase amount. Feature engineering and encoding of variables, converting categorical variables into numerical variables, converting strings into binary and binning of value were explored. Following this model were implemented. It was observed that simpler models like multiple linear regression were not yielding results. Due to the high volume of the dataset and the large numbers in the two columns of User ID and product ID simpler models were unable to capture the non-linearity in the dataset. Hence, models like XG boost, classification, and regression trees were used which resulted in a reasonable RMSE value.

The advantages of using XG boost are that it results in the best RMSE value possible as it creates and an ensemble of decision trees. It improves the predictive force of the model. The shortcomings of this dataset were that it did not fit very well with most regression models. The multiple linear regression could be run only over a random sample of 100 records and resulted in an RMSE value of 2405

Based on the predicted purchase amount and the visualizations, it can be recommended that the company focusses more on sending catalogs/ promotional discount offers to customers in the age group of 26-35 and 18-25. They should primarily target their focus on single men followed by married women. Also, they should expect to make greater sales in city C and could work on increasing their sales in city B.

VII. APPENDIX

R Code for Use Case Study

Data Exploration

```
#{echo=FALSE, cache=FALSE, results=FALSE, warning=FALSE,
comment=FALSE, Message=FALSE}
install.packages('data.table')
library(data.table)
library(ggplot2)
install.packages('gridExtra')
library(gridExtra)
install.packages('corrplot')
library(corrplot)
install.packages('rpart')
library(rpart)
install.packages('randomForest')
library(randomForest)
install.packages('stepPlr')
library(stepPlr)
install.packages('C50')
library(C50)
install.packages('plyr')
library(plyr)
library(MASS)
install.packages('caret')
library(caret)
install.packages('caretEnsemble')
library(caretEnsemble)
library(dplyr)

train_set = read.csv("train.csv", stringsAsFactors = FALSE, header =
TRUE, na.strings = c('NA',''))
test_set = read.csv("test.csv",  stringsAsFactors = FALSE, header =
TRUE, na.strings = c('NA',''))

str(train_set)

sapply(train_set, function(x) sum(is.na(x)))

str(test_set)

sapply(test_set, function(x) sum(is.na(x)))

train_set$User_ID = as.factor(train_set$User_ID)
train_set$Product_ID = as.factor(train_set$Product_ID)
train_set$Marital_Status = as.factor(ifelse(train_set$Marital_Status
== 1, 'Married', 'Single'))
```

```
train_set$Age = as.factor(train_set$Age)
train_set$Gender = as.factor(ifelse(train_set$Gender=='M', 'Male',
'Female'))
train_set$Occupation = as.factor(train_set$Occupation)
train_set$City_Category = as.factor(train_set$City_Category)
train_set$Stay_In_Current_City_Years =
as.factor(train_set$Stay_In_Current_City_Years)

test_set$User_ID = as.factor(test_set$User_ID)
test_set$Product_ID = as.factor(test_set$Product_ID)
test_set$Marital_Status = as.factor(ifelse(test_set$Marital_Status ==
1, 'Married', 'Single'))
test_set$Age = as.factor(test_set$Age)
test_set$Gender = as.factor(ifelse(test_set$Gender=='M', 'Male',
'Female'))
test_set$Occupation = as.factor(test_set$Occupation)
test_set$City_Category = as.factor(test_set$City_Category)
test_set$Stay_In_Current_City_Years =
as.factor(test_set$Stay_In_Current_City_Years)

str(train_set)
str(test_set)

EDA_Distinct = distinct(train_set, User_ID, Age, Gender,
Marital_Status, Occupation, City_Category, Stay_In_Current_City_Years)
str(EDA_Distinct)
head(EDA_Distinct)

head(train_set$User_ID,40)

head(test_set$User_ID,40)

#creating a new data frame to stor the number of purchase made by each
user
userIDCount = as.data.frame(table(train_set$User_ID))
names(userIDCount) = c("User_ID","User_Purchase_Count")
head(userIDCount)

# joining i.e. storing the user purchase count in original data frame
#By using the merge function and its optional parameters:
#Inner join: merge(df1, df2) will work for these examples because R
automatically joins the frames by common variable names, but you would
most likely want to specify merge(df1, df2, by = "CustomerId") to make
sure that you were matching on only the fields you desired. You can
also use the by.x and by.y parameters if the matching variables have
different names in the different data frames.
#Outer join: merge(x = df1, y = df2, by = "CustomerId", all = TRUE)
#Left outer: merge(x = df1, y = df2, by = "CustomerId", all.x = TRUE)
#Right outer: merge(x = df1, y = df2, by = "CustomerId", all.y = TRUE)
#Cross join: merge(x = df1, y = df2, by = NULL)

# User_Purchase_Count i.e. no. of item purchased by a customer
```

```
train_set = merge(x = train_set, y = userIDCount, by = "User_ID",
all.x = TRUE)
str(train_set)

# writing code such that if a new user comes for the first time his
count is set to one in test dataset
test_set = merge(x = test_set, y = userIDCount, by = "User_ID", all.x
= TRUE)
#Now we can remove the UserIDCount dataframe
rm(userIDCount)
'%!in%' <- function(x,y)!('%in%'(x,y))
#if(test_set$User_ID %!in%
train_set$User_ID){ assign(test_set$userIDCount, 0)}
test_set[is.na(test_set$User_Purchase_Count), "User_Purchase_Count"]
<- 1
class(test_set$User_Purchase_Count)

str(test_set)

test_set$User_Purchase_Count =
as.integer(test_set$User_Purchase_Count)

# avg_spending i.e. average spending by a user
# creating a data frame to store the total spending by a user
totspend = aggregate(train_set$Purchase,
by=list(Category=train_set$User_ID), FUN=sum)
# or this code >tapply(train_set$Purchase, train_set$User_ID, FUN=sum)
names(totspend) = c("User_ID","Total_Spending")
head(totspend)

# univariate analysis of total spending by users
#We ain't adding this possible variable to training set as this is a
function of y and only using it for explanatory data analysis.
summary(totspend$Total_Spending)


ggplot(totspend, aes(x=Total_Spending)) + geom_density(fill="red",
col="black", alpha=0.80)

rm(totspend)

#Updating EDA_Distinct dataframe
EDA_Distinct = distinct(train_set, User_ID, Age, Gender,
Marital_Status, Occupation, City_Category, Stay_In_Current_City_Years,
User_Purchase_Count)

d1 = summary(EDA_Distinct$User_Purchase_Count)

p1 = ggplot(EDA_Distinct, aes(x=User_Purchase_Count)) +
geom_density(fill="red", col="black", alpha=0.80) + annotate(geom =
"text", x = 6, y = 0.0125, label = "Min")  + annotate(geom = "text", x
= 24, y = 0.013, label = "1st Qu.") + annotate(geom = "text", x = 50,
y = 0.0125, label = "Median") + annotate(geom = "text", x = 90, y =
0.013, label = "Mean") + annotate(geom = "text", x = 112, y = 0.0125,
label = "3rd Qu.") + annotate(geom = "text", x = 1015, y = 0.0125,
label = "Max") + geom_vline(xintercept = c(6, 26, 54, 93.37, 117,
1026), size = 0.2, col = 'black') #+ lims(x = )

p2 = ggplot(EDA_Distinct, aes(x=User_Purchase_Count))
+geom_histogram(fill="red", col="black", alpha=0.80)

p3 = ggplot(EDA_Distinct,aes(x= Age,y=User_Purchase_Count, fill=Age))
+ geom_boxplot() + facet_grid(Gender~Marital_Status) +
labs(x="Age",y="Customer Purchase Count")

p4 = ggplot(EDA_Distinct,aes(x= Occupation,y=User_Purchase_Count,
fill=Occupation)) + geom_boxplot() + facet_grid(Gender~Marital_Status)
+ labs(x="Occupation",y="Customer Purchase Count")

p5 =
ggplot(EDA_Distinct,aes(x=Age,y=User_Purchase_Count,fill=Stay_In_Curre
nt_City_Years))+geom_boxplot()+facet_grid(City_Category~
Stay_In_Current_City_Years) + labs(x="Age",y="Customer Purchase
Count")

p5i =
ggplot(EDA_Distinct,aes(x=Age,y=User_Purchase_Count,fill=Stay_In_Curre
nt_City_Years))+geom_boxplot()+facet_grid( Stay_In_Current_City_Years
~ City_Category) + labs(x="Age",y="Customer Purchase Count")

p6 =
ggplot(EDA_Distinct,aes(x=Age,y=User_Purchase_Count,fill=Marital_Statu
s))+geom_boxplot()+facet_grid(Gender~City_Category) +
scale_fill_manual(values=c("tan4","limegreen"))  +
labs(x="Age",y="Customer Purchase Count")

#grid.arrange(p1, p2, p3, p4 ,p5i ,p6, ncol = 1, nrow = 6);
d1;
p1;
p2;
p3;
p4;
p5;
p5i;
p6

head(train_set$Product_ID,15)

#head(test_set$Product_ID,15)

# SoldProdCount : creating a new data frame to store the number of
each product sold
# Product_Sold_Count --> variable which stores the number of each
product sold
```

```
SoldProdCount = as.data.frame(table(train_set$Product_ID))
names(SoldProdCount) <- c("Product_ID","Product_Sold_Count")

# SoldProdPriceMean : creating a new data frame to store the
# Product_Mean_Price --> variable which stores the mean of the product
price
# or this code >tapply(train_set$Purchase, train_set$Product_ID,
FUN=mean)
SoldProdPriceMean = aggregate(train_set$Purchase,
by=list(Category=train_set$Product_ID), FUN=mean)
names(SoldProdPriceMean) = c("Product_ID","Product_Mean_Price")

# SD_Product Price i.e. SD in product price
# creating a data frame to store the total Price of product
SoldProdPriceSD = aggregate(train_set$Purchase,
by=list(Category=train_set$Product_ID), FUN=sd)
names(SoldProdPriceSD) = c("Product_ID","Product_SD_Price")

str(SoldProdCount); str(SoldProdPriceMean); str(SoldProdPriceSD)

# merging these three data in single dataframe
#ProductData = data.frame(Product_ID)
ProductData = as.data.frame(train_set$Product_ID)
colnames(ProductData) = c("Product_ID")
# joining i.e. storing the Product sold count in original data frame
ProductData = merge(x = ProductData, y = SoldProdCount, by =
"Product_ID", all.x = TRUE)
ProductData = merge(x = ProductData, y = SoldProdPriceMean, by =
"Product_ID", all.x = TRUE)
ProductData = merge(x = ProductData, y = SoldProdPriceSD, by =
"Product_ID", all.x = TRUE)

ProductData$Product_Sold_Count =
as.integer(ProductData$Product_Sold_Count)
ProductData$Product_Mean_Price =
as.integer(ProductData$Product_Mean_Price)
ProductData$Product_SD_Price =
as.integer(ProductData$Product_SD_Price)

str(ProductData)

# other variable rather than Product_Sold_Count (from SoldProdCount
dataframe ) aren't added as they are a function of Response
train_set = merge(x = train_set, y = SoldProdCount, by = "Product_ID",
all.x = TRUE)

## plot using Product Data since EDA set wont deal with product data
d2 = summary(ProductData$Product_Sold_Count)

p7 = ggplot(ProductData, aes(x=Product_Sold_Count))
+geom_density(fill="red", col="black", alpha=0.80) + annotate(geom =
"text", x = 1, y = 0.0017, label = "Min")  + annotate(geom = "text", x
= 174, y = 0.0017, label = "1st Qu.") + annotate(geom = "text", x =
357, y = 0.0017, label = "Median") + annotate(geom = "text", x = 450,
y = 0.0017, label = "Mean") + annotate(geom = "text", x = 620, y =
0.0017, label = "3rd Qu.") + annotate(geom = "text", x = 1880, y =
0.0017, label = "Max") + geom_vline(xintercept =
c(1,174,357,450.5,620,1880), size = 0.2, col = 'black')

d2e = summary(ProductData$Product_Mean_Price)

p7e = ggplot(ProductData, aes(x=Product_Mean_Price))
+geom_density(fill="red", col="black", alpha=0.80) + annotate(geom =
"text", x = 30, y = 0.0017, label = "Min")  + annotate(geom = "text",
x = 6340, y = 0.0017, label = "1st Qu.") + annotate(geom = "text", x =
7750, y = 0.0017, label = "Median") + annotate(geom = "text", x =
9245, y = 0.0017, label = "Mean") + annotate(geom = "text", x = 12950,
y = 0.0017, label = "3rd Qu.") + annotate(geom = "text", x = 21240, y
= 0.0017, label = "Max") + geom_vline(xintercept = c(36, 6372, 7785,
9263, 12970, 21260), size = 0.2, col = 'black')

d2ee = summary(ProductData$Product_SD_Price)

p7ee = ggplot(ProductData, aes(x=Product_SD_Price))
+geom_density(fill="red", col="black", alpha=0.80) + annotate(geom =
"text", x = 5, y = 0.00105, label = "Min")  + annotate(geom = "text",
x = 1620, y = 0.00105, label = "1st Qu.") + annotate(geom = "text", x
= 1968, y = 0.00105, label = "Median") + annotate(geom = "text", x =
2410, y = 0.00105, label = "Mean") + annotate(geom = "text", x = 3450,
y = 0.00105, label = "3rd Qu.") + annotate(geom = "text", x = 8950, y
= 0.00105, label = "Max") + geom_vline(xintercept = c(11, 1642, 1978,
2433, 3470, 8970), size = 0.2, col = 'black')

d2;
p7;
d2e;
p7e;
d2ee;
p7ee

# Product which are sold most frequent and least frequent
head(ProductData[order(-ProductData$Product_Sold_Count),])

tail(ProductData[order(-ProductData$Product_Sold_Count),])

# writing code such that if a new user comes for the first time his
count is set to one in test dataset
test_set = merge(x = test_set, y = SoldProdCount, by = "Product_ID",
all.x = TRUE)
#Now we can remove the UserIDCount dataframe
rm(SoldProdCount, SoldProdPriceMean, SoldProdPriceSD, ProductData)
test_set[is.na(test_set$Product_Sold_Count), "Product_Sold_Count"] <-
0
#str(test_set)
```

```
test_set$Product_Sold_Count <- as.integer(test_set$Product_Sold_Count)

#Gender and Marital Status
head(train_set$Gender); head(train_set$Marital_Status)
d3 = table(EDA_Distinct$Gender, EDA_Distinct$Marital_Status)

p8 = ggplot(EDA_Distinct, aes(x=Gender, fill= Marital_Status)) +
geom_bar(position = "dodge") + ggtitle("") +  labs(x="Gender",y="No.
of distinct Sales") + annotate(geom = "text", x = 0.775, y = 619,
label = "719")   + annotate(geom = "text", x = 1.225, y = 847, label =
"947") + annotate(geom = "text", x = 1.775, y = 1655, label = "1755")
+ annotate(geom = "text", x = 2.225, y = 2370, label = "2470") +
scale_fill_manual(values=c("tan4","limegreen"))
d3;
p8

head(train_set, 10)

d4 = table(EDA_Distinct$Age)

p9 = ggplot(EDA_Distinct, aes(x=Age)) + geom_bar(fill=rainbow(7),
col="black") + ggtitle("") +  labs(x="Age Group",y="No. of distinct
buyer") + annotate(geom = "text", x = 1, y = 168, label = "218") +
annotate(geom = "text", x = 2, y = 1019, label = "1069") +
annotate(geom = "text", x = 3, y = 2000, label = "2053") +
annotate(geom = "text", x = 4, y = 1117, label = "1167") +
annotate(geom = "text", x = 5, y = 481, label = "531") + annotate(geom
= "text", x = 6, y = 431, label = "481") + annotate(geom = "text", x =
7, y = 322, label = "372")

d5 = table(EDA_Distinct$Marital_Status, EDA_Distinct$Gender,
EDA_Distinct$Age)

p10 = ggplot(EDA_Distinct, aes(x= Age,fill= Gender, col=
Marital_Status)) + geom_bar(position = "dodge", size=1.25) +
labs(x="Age Group",y="No. of distinct buyer") +
scale_fill_manual(values=c("hotpink", "royalblue")) +
scale_color_manual(values=c("tan4","limegreen")) + ggtitle("")

p11 =
ggplot(EDA_Distinct,aes(x=Age,fill=Marital_Status))+geom_bar(position
= "dodge")+facet_grid(Gender~.) +
scale_fill_manual(values=c("tan4","limegreen"))

#grid.arrange(p9, p10, p11, ncol = 5, nrow = 1); wastrying to create
thumbnail
d4;
p9;
d5;
p10;
p11
```

```
#Occupation
head(train_set$Occupation, 10)

d6 = table(EDA_Distinct$Occupation)

d7 = table(EDA_Distinct$Gender, EDA_Distinct$Occupation)

p12 = ggplot(EDA_Distinct, aes(x=Occupation, fill=Gender)) +
geom_bar( col="black") + ggtitle("") +  labs(x="Occupation",y="No. of
distinct people") + scale_fill_manual(values=c("hotpink",
"royalblue"))

d8 = table(EDA_Distinct$Marital_Status, EDA_Distinct$Occupation)

p13 = ggplot(EDA_Distinct, aes(x=Occupation, fill=Marital_Status)) +
geom_bar( col="black") + ggtitle("") +  labs(x="Occupation",y="No. of
distinct people") + scale_fill_manual(values=c("tan4","limegreen"))

p14 = ggplot(EDA_Distinct,aes(x=Occupation,
fill=Age))+geom_bar()+facet_grid(Gender~Marital_Status)

#grid.arrange(p12, p13, p14, ncol = 5, nrow = 1);
d6;
d7;
p12;
d8;
p13;
p14

#City Category and Stay in Current City

head(train_set$Stay_In_Current_City_Years, 10);
head(train_set$City_Category, 10)

d9 = table(EDA_Distinct$City_Category,
EDA_Distinct$Stay_In_Current_City_Years)

p15 = ggplot(EDA_Distinct, aes(x=Stay_In_Current_City_Years,
fill=City_Category)) + geom_bar( col="black") + ggtitle("") +
labs(x="Stay in Current City (Years)",y="No. of distinct people")

p16 = ggplot(EDA_Distinct,aes(City_Category,fill=Age))+geom_bar()

p17 =
ggplot(EDA_Distinct,aes(x=Age,fill=Stay_In_Current_City_Years))+geom_b
ar()+facet_grid(City_Category~ Stay_In_Current_City_Years)

p18 =
ggplot(EDA_Distinct,aes(x=Age,fill=Marital_Status))+geom_bar()+facet_g
rid(Gender~City_Category) +
scale_fill_manual(values=c("tan4","limegreen"))
```

```
#grid.arrange(p15, p16, p17, p18, ncol = 5, nrow = 1);
d9;
p15;
p16;
p17;
p18

#Product Category
head(as.factor(train_set$Product_Category_1))
head(as.factor(train_set$Product_Category_2))
head(as.factor(train_set$Product_Category_3))

#Feature Manipulation and Creation - Training Set

# Converting Variable to Factor type
train_set$Product_Category_1 = as.factor(train_set$Product_Category_1)
train_set$Product_Category_2 = as.factor(train_set$Product_Category_2)
train_set$Product_Category_3 = as.factor(train_set$Product_Category_3)

#Adding another factor level for missing value
train_set$Product_Category_2 = factor(train_set$Product_Category_2,
levels=c(levels(train_set$Product_Category_2), "0"))
train_set[is.na(train_set$Product_Category_2), "Product_Category_2"]
="0"
#head(train_set$Product_Category_2, 10)
train_set$Product_Category_3 = factor(train_set$Product_Category_3,
levels=c(levels(train_set$Product_Category_3), "0"))
train_set[is.na(train_set$Product_Category_3), "Product_Category_3"]
="0"
#head(train_set$Product_Category_3, 10)

# Creating new binary variable based on product category
train_set$Cat_1 = as.factor(ifelse((train_set$Product_Category_1=='1'
| train_set$Product_Category_2=='1' |
train_set$Product_Category_3=='1'), 1,0))
for(i in 2:20)
{
  assign(paste("Cat_", as.character(i),
sep=""),as.factor(ifelse((train_set$Product_Category_1==i |
train_set$Product_Category_2==i | train_set$Product_Category_3==i),
1,0)))
}
train_set = cbind(train_set, Cat_2, Cat_3, Cat_4, Cat_5, Cat_6, Cat_7,
Cat_8, Cat_9, Cat_10, Cat_11, Cat_12, Cat_13, Cat_14, Cat_15, Cat_16,
Cat_17, Cat_18, Cat_19, Cat_20)

# Dropping the unnecessary variables
to_drop = c("Product_Category_1", "Product_Category_2",
"Product_Category_3")
train_set = train_set[,!names(train_set)%in% to_drop]
```

```
rm(Cat_2, Cat_3, Cat_4, Cat_5, Cat_6, Cat_7, Cat_8, Cat_9, Cat_10,
Cat_11, Cat_12, Cat_13, Cat_14, Cat_15, Cat_16, Cat_17, Cat_18,
Cat_19, Cat_20)

#Checking the final structure of the dataset
dim(train_set)


as.matrix(sapply(train_set, function(x) class(x)))

#Feature Manipulation and Creation - Test Result

# Converting Variable to Factor type
test_set$Product_Category_1 = as.factor(test_set$Product_Category_1)
test_set$Product_Category_2 = as.factor(test_set$Product_Category_2)
test_set$Product_Category_3 = as.factor(test_set$Product_Category_3)

#Adding another factor level for missing value
test_set$Product_Category_2 = factor(test_set$Product_Category_2,
levels=c(levels(test_set$Product_Category_2), "0"))
test_set[is.na(test_set$Product_Category_2), "Product_Category_2"]
="0"
#head(test_set$Product_Category_2, 10)
test_set$Product_Category_3 = factor(test_set$Product_Category_3,
levels=c(levels(test_set$Product_Category_3), "0"))
test_set[is.na(test_set$Product_Category_3), "Product_Category_3"]
="0"
#head(test_set$Product_Category_3, 10)

# Creating new binary variable based on product category
for(i in 1:20)
{
  assign(paste("Cat_", as.character(i),
sep=""),as.factor(ifelse((test_set$Product_Category_1==i |
test_set$Product_Category_2==i | test_set$Product_Category_3==i),
1,0)))
}
test_set <- cbind(test_set, Cat_1, Cat_2, Cat_3, Cat_4, Cat_5, Cat_6,
Cat_7, Cat_8, Cat_9, Cat_10, Cat_11, Cat_12, Cat_13, Cat_14, Cat_15,
Cat_16, Cat_17, Cat_18, Cat_19, Cat_20)

# Dropping the unnecessary variables
to_drop = c("Product_Category_1", "Product_Category_2",
"Product_Category_3")
test_set = test_set[,!names(test_set)%in% to_drop]
rm(Cat_1, Cat_2, Cat_3, Cat_4, Cat_5, Cat_6, Cat_7, Cat_8, Cat_9,
Cat_10, Cat_11, Cat_12, Cat_13, Cat_14, Cat_15, Cat_16, Cat_17,
Cat_18, Cat_19, Cat_20)

#Checking the final structure of the dataset
dim(test_set)
```

```
as.matrix(sapply(test_set, function(x) class(x)))

#Final

#str(train_set, list.len = 6);str(test_set, list.len = 6)
str(train_set);
str(test_set)

sapply(train_set, function(x) sum(is.na(x)));
sapply(test_set, function(x) sum(is.na(x)))

head(train_set[order(train_set$Product_ID),])
```

# Model Fitting

# Multiple Linear Regression

```
#importing the dataset
dataset= read.csv("train.csv")
dataset=dataset[1:10000,]

#replacing missing values in product category 1, 2 and 3 with mean
dataset$Product_Category_2 = ifelse(is.na(dataset$Product_Category_2),
                                    ave(dataset$Product_Category_2,
FUN = function(x) mean(x, na.rm = TRUE)),
                                    dataset$Product_Category_2)
dataset$Product_Category_3 = ifelse(is.na(dataset$Product_Category_3),
                                    ave(dataset$Product_Category_3,
FUN = function(x) mean(x, na.rm = TRUE)),
                                    dataset$Product_Category_3)
str(dataset)

dataset$User_ID=as.factor(dataset$User_ID)
dataset$Occupation= as.factor(dataset$Occupation)
dataset$Marital_Status=as.factor(dataset$Marital_Status)
dataset$Product_Category_1=as.numeric(dataset$Product_Category_1)
dataset$Purchase=as.numeric(dataset$Purchase)

str(dataset)

# subsetting the dataset into train and validation set with a ratio of
60/40

install.packages('caTools')
library(caTools)
split = sample.split(dataset$User_ID, SplitRatio = 3/5)
training_set = subset(dataset, split == TRUE)
valid_set = subset(dataset, split == FALSE)
```

```
# fitting multiple regression on training set
regressor_m= lm(formula= dataset$Purchase ~., data = dataset)
summary(regressor_m)
regressor_predict= predict(regressor_m, valid_set)
regressor_predict=as.data.frame(regressor_predict)
install.packages("Metrics")
library(Metrics)
rmse(valid_set$Purchase, regressor_m)
sse = sum((regressor_predict - valid_set$Purchase)^2)
rmse_tree = sqrt(sse/nrow(valid_set))
```

# CART

```
#importing the dataset
train_set = read.csv("train.csv")
test_set = read.csv("test.csv")

train_set$User_ID = as.factor(train_set$User_ID)
train_set$Product_ID = as.factor(train_set$Product_ID)
train_set$Marital_Status = as.factor(ifelse(train_set$Marital_Status ==
1, 'Married', 'Single'))
train_set$Age = as.factor(train_set$Age)
train_set$Gender   =   as.factor(ifelse(train_set$Gender=='M',   'Male',
'Female'))
train_set$Occupation = as.factor(train_set$Occupation)
train_set$City_Category = as.factor(train_set$City_Category)
train_set$Stay_In_Current_City_Years                                  =
as.factor(train_set$Stay_In_Current_City_Years)

test_set$User_ID = as.factor(test_set$User_ID)
test_set$Product_ID = as.factor(test_set$Product_ID)
test_set$Marital_Status = as.factor(ifelse(test_set$Marital_Status == 1,
'Married', 'Single'))
test_set$Age = as.factor(test_set$Age)
test_set$Gender   =   as.factor(ifelse(test_set$Gender=='M',   'Male',
'Female'))
test_set$Occupation = as.factor(test_set$Occupation)
test_set$City_Category = as.factor(test_set$City_Category)
test_set$Stay_In_Current_City_Years                                   =
as.factor(test_set$Stay_In_Current_City_Years)


str(train_set)
str(test_set)

sapply(train_set, function(x) sum(is.na(x)))
sapply(test_set, function(x) sum(is.na(x)))

train_set$Product_Category_2                                          =
ifelse(is.na(train_set$Product_Category_2),
                                    ave(train_set$Product_Category_2,
FUN = function(x) mean(x, na.rm = TRUE)),
                                    train_set$Product_Category_2)
train_set$Product_Category_3                                         =
ifelse(is.na(train_set$Product_Category_3),
                                    ave(train_set$Product_Category_3,
FUN = function(x) mean(x, na.rm = TRUE)),train_set$Product_Category_3)

test_set$Product_Category_2 = ifelse(is.na(test_set$Product_Category_2),
                                      ave(test_set$Product_Category_2,
FUN = function(x) mean(x, na.rm = TRUE)),
                                      train_set$Product_Category_2)
test_set$Product_Category_3 = ifelse(is.na(test_set$Product_Category_3),
                                      ave(test_set$Product_Category_3,
FUN = function(x) mean(x, na.rm = TRUE)),test_set$Product_Category_3)
sapply(train_set, function(x) sum(is.na(x)))
sapply(test_set, function(x) sum(is.na(x)))

# breaking train set into training set and validation set

install.packages('caTools')
library(caTools)
split = sample.split(train_set$User_ID, SplitRatio = 3/5)
training_set = subset(train_set, split == TRUE)
valid_set = subset(train_set, split == FALSE)

library(rpart)
library(rpart.plot)

# fitting CART

treeModel = rpart(Purchase ~ ., data = training_set, minbucket = 50 )
treePredictionCv = predict(treeModel, valid_set)

sse_tree = sum((treePredictionCv - valid_set$Purchase)^2)
rmse_tree = sqrt(sse_tree/nrow(valid_set))
```

#XGBoost

```
#importing the dataset
dataset= read.csv("train.csv")



#replacing missing values in product category 1, 2 and 3 with mean
dataset$Product_Category_2                                        =
ifelse(is.na(dataset$Product_Category_2),

ave(dataset$Product_Category_2, FUN = function(x) mean(x, na.rm =
TRUE)),
                                    dataset$Product_Category_2)
dataset$Product_Category_3                                    =
ifelse(is.na(dataset$Product_Category_3),

ave(dataset$Product_Category_3, FUN = function(x) mean(x, na.rm =
TRUE)),
                                    dataset$Product_Category_3)
str(dataset)


# XGBoost only works with numeric data type so we need to convert
all the data to numeric

dataset$User_ID=as.numeric(dataset$User_ID)
dataset$Product_ID=as.numeric(dataset$Product_ID)
dataset$Gender=as.numeric(ifelse(dataset$Gender=="Male", 1, 0))
dataset$Age=as.numeric(ifelse(dataset$Age=='0-17',            17,
ifelse(dataset$Age=='18-25', 25, ifelse(dataset$Age=='26-35', 35,
ifelse(dataset$Age=='36-45', 45, ifelse(dataset$Age=='46-50', 50,
ifelse(dataset$Age=='51-55', 55, 65)))))))
dataset$Marital_Status=as.numeric(ifelse(dataset$Marital_Status=
="Married", '1','0'))
dataset$Occupation=as.numeric(dataset$Occupation)
dataset$City_Category=as.numeric(ifelse(dataset$City_Category=='
A', 1, ifelse(dataset$City_Category=='B', 2, 3)))
dataset$Stay_In_Current_City_Years=as.numeric(ifelse(dataset$Sta
y_In_Current_City_Years=='4+',                                 6,
dataset$Stay_In_Current_City_Years))

dataset$Product_Category_1=as.numeric(dataset$Product_Category_1)
dataset$Purchase=as.numeric(dataset$Purchase)

str(dataset)

# subsetting the dataset into train and validation set with a ratio
of 60/40

install.packages('caTools')
library(caTools)
split = sample.split(dataset$User_ID, SplitRatio = 3/5)
training_set = subset(dataset, split == TRUE)
valid_set = subset(dataset, split == FALSE)

xgb_train_mat = data.matrix(training_set)
xgb_valid_mat = data.matrix(valid_set)
xgb_dataset_mat=data.matrix(dataset)
class(xgb_train_mat)
```

```
# Model Preparation
library(xgboost)
xgb_1 = xgboost(xgb_train_mat, label = training_set$Purchase, cv=5,
objective="reg:linear", nrounds = 500, max.depth=10, eta = 0.1,
colsample_bytree =0.5, seed=235, metric="rmse", importance=1)


xgb_2 = xgboost(xgb_train_mat, label = training_set$Purchase, cv=5,
objective="reg:linear", nrounds = 700, max.depth=10, eta = 0.1,
colsample_bytree =0.5, seed=235, metric="rmse", importance=1)


xgb_3 = xgboost(xgb_dataset_mat, label = dataset$Purchase, cv=5,
objective="reg:linear", nrounds = 500, max.depth=10, eta = 0.1,
colsample_bytree =0.5, seed=235, metric="rmse", importance=1)


xgb_4 = xgboost(xgb_valid_mat, label = valid_set$Purchase, cv=5,
objective="reg:linear", nrounds = 700, max.depth=10, eta = 0.1,
colsample_bytree =0.5, seed=235, metric="rmse", importance=1)

# Prediction XGBoost

# Setting up test dataset
test_set = read.csv("test.csv")

summary(is.na(test_set))

#replacing missing values in product category 1, 2 and 3 with mean
test_set$Product_Category_2                                        =
ifelse(is.na(test_set$Product_Category_2),

ave(test_set$Product_Category_2, FUN = function(x) mean(x, na.rm
= TRUE)),
                                    test_set$Product_Category_2)
test_set$Product_Category_3                                        =
ifelse(is.na(test_set$Product_Category_3),

ave(test_set$Product_Category_3, FUN = function(x) mean(x, na.rm
= TRUE)),
                                    test_set$Product_Category_3)
str(test_set)

# changing to numeric

test_set$User_ID=as.numeric(test_set$User_ID)
```

```
test_set$Product_ID=as.numeric(test_set$Product_ID)
test_set$Gender=as.numeric(ifelse(test_set$Gender=="Male", 1, 0))
test_set$Age=as.numeric(ifelse(test_set$Age=='0-17',         17,
ifelse(test_set$Age=='18-25',  25,  ifelse(test_set$Age=='26-35',
35,  ifelse(test_set$Age=='36-45',  45,  ifelse(test_set$Age=='46-
50', 50, ifelse(test_set$Age=='51-55', 55, 65)))))))
test_set$Marital_Status=as.numeric(ifelse(test_set$Marital_Statu
s=="Married", '1','0'))
test_set$Occupation=as.numeric(test_set$Occupation)
test_set$City_Category=as.numeric(ifelse(test_set$City_Category=
='A', 1, ifelse(test_set$City_Category=='B', 2, 3)))
test_set$Stay_In_Current_City_Years=as.numeric(ifelse(test_set$S
tay_In_Current_City_Years=='4+',                               6,
test_set$Stay_In_Current_City_Years))

test_set$Product_Category_1=as.numeric(test_set$Product_Category
_1)

str(test_set)
test_set_mat=data.matrix(test_set)
class(test_set_mat)
# predicting

pred=predict(xgb_1, test_set_mat, outputmargin = TRUE)

submit=test_set[,c("User_ID","Product_ID")]
class(submit)

submit$Purchase = pred

write.csv(submit, "Submit XGBoost.csv", row.names=FALSE)

install.packages("PerformanceAnalytics")
library(PerformanceAnalytics)
cor(xgb_train_mat)


# correlation matrix with p-values
cor.prob(xgb_train_mat)
# "flatten" that table
flattenSquareMatrix(cor.prob(mydata))
# plot the data
library(PerformanceAnalytics)
chart.Correlation(xgb_dataset_mat)
```

VIII. REFERENCES

- Machine Learning A-Z: Udemy.com
- Data Mining for Business Analytics: Concepts. Techniques, and Applications with XLMiner
- Analytics Vidhya