Black Friday Data Vosualization

Parv

8 March 2018

## 1. Problem Statement

To understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories so that sales of a retail company can be boosted.

We will be building models to predict the purchase amount of customer against various products which will help us to create personalized offer for customers against different products.

## 2. Setting up the environment

Setting up environment for analysis, loading data, packages, understanding variables.

## 2.1 Loading Data

getwd()

## [1] "/home/parv/black friday project"

data\_set<-read.csv(paste("train.csv",sep = ""))  
  
## 75% of the sample size  
smp\_size <- floor(0.75 \* nrow(data\_set))  
  
## set the seed to make your partition reproductible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(data\_set)), size = smp\_size)  
  
train\_set <- data\_set[train\_ind, ]  
test\_set <- data\_set[-train\_ind, ]  
  
y\_test<-as.data.frame(test\_set[,12], drop=false)  
names(y\_test)<-c("purchase")  
  
test\_set<-as.data.frame(test\_set[,1:11], drop=false)  
dim(test\_set)

## [1] 137517 11

## 2.2 Checking dimensions of the data

dim(train\_set)

## [1] 412551 12

dim(test\_set)

## [1] 137517 11

**Training data** set consists of 4,12,551 entries across 12 variables.  
**Test Data** set consists of 1,37,517 entries across 11 variables.

12th variable in Training Data set is our response Variable which needs to be predicted in test data set.

# 2.3 Understanding the Data set

library(psych)  
describe(train\_set)

## vars n mean sd median  
## User\_ID 1 412551 1003030.35 1728.22 1003080  
## Product\_ID\* 2 412551 1708.53 1011.98 1668  
## Gender\* 3 412551 1.75 0.43 2  
## Age\* 4 412551 3.50 1.35 3  
## Occupation 5 412551 8.08 6.53 7  
## City\_Category\* 6 412551 2.04 0.76 2  
## Stay\_In\_Current\_City\_Years\* 7 412551 2.86 1.29 3  
## Marital\_Status 8 412551 0.41 0.49 0  
## Product\_Category\_1 9 412551 5.41 3.94 5  
## Product\_Category\_2 10 282116 9.85 5.09 9  
## Product\_Category\_3 11 124786 12.67 4.12 14  
## Purchase 12 412551 9261.00 5022.12 8044  
## trimmed mad min max range skew  
## User\_ID 1003028.59 2180.90 1000001 1006040 6039 0.00  
## Product\_ID\* 1688.75 1196.46 1 3631 3630 0.15  
## Gender\* 1.82 0.00 1 2 1 -1.17  
## Age\* 3.36 1.48 1 7 6 0.81  
## Occupation 7.69 8.90 0 20 20 0.40  
## City\_Category\* 2.05 1.48 1 3 2 -0.07  
## Stay\_In\_Current\_City\_Years\* 2.82 1.48 1 5 4 0.32  
## Marital\_Status 0.39 0.00 0 1 1 0.37  
## Product\_Category\_1 4.91 4.45 1 20 19 1.02  
## Product\_Category\_2 10.00 7.41 2 18 16 -0.17  
## Product\_Category\_3 13.07 2.97 3 18 15 -0.77  
## Purchase 8925.40 4250.61 12 23961 23949 0.60  
## kurtosis se  
## User\_ID -1.20 2.69  
## Product\_ID\* -1.09 1.58  
## Gender\* -0.63 0.00  
## Age\* 0.30 0.00  
## Occupation -1.22 0.01  
## City\_Category\* -1.27 0.00  
## Stay\_In\_Current\_City\_Years\* -1.07 0.00  
## Marital\_Status -1.86 0.00  
## Product\_Category\_1 1.22 0.01  
## Product\_Category\_2 -1.43 0.01  
## Product\_Category\_3 -0.81 0.01  
## Purchase -0.34 7.82

describe(test\_set)

## vars n mean sd median  
## User\_ID 1 137517 1003024.33 1725.69 1003070  
## Product\_ID\* 2 137517 1708.31 1012.87 1665  
## Gender\* 3 137517 1.75 0.43 2  
## Age\* 4 137517 3.49 1.36 3  
## Occupation 5 137517 8.07 6.51 7  
## City\_Category\* 6 137517 2.04 0.76 2  
## Stay\_In\_Current\_City\_Years\* 7 137517 2.86 1.29 3  
## Marital\_Status 8 137517 0.41 0.49 0  
## Product\_Category\_1 9 137517 5.38 3.93 5  
## Product\_Category\_2 10 94314 9.82 5.08 9  
## Product\_Category\_3 11 42035 12.67 4.13 14  
## trimmed mad min max range skew  
## User\_ID 1003021.88 2172.01 1000001 1006040 6039 0.00  
## Product\_ID\* 1688.48 1200.91 1 3631 3630 0.15  
## Gender\* 1.82 0.00 1 2 1 -1.18  
## Age\* 3.35 1.48 1 7 6 0.81  
## Occupation 7.69 8.90 0 20 20 0.40  
## City\_Category\* 2.05 1.48 1 3 2 -0.07  
## Stay\_In\_Current\_City\_Years\* 2.82 1.48 1 5 4 0.32  
## Marital\_Status 0.39 0.00 0 1 1 0.37  
## Product\_Category\_1 4.88 4.45 1 20 19 1.04  
## Product\_Category\_2 9.97 7.41 2 18 16 -0.15  
## Product\_Category\_3 13.07 2.97 3 18 15 -0.76  
## kurtosis se  
## User\_ID -1.19 4.65  
## Product\_ID\* -1.09 2.73  
## Gender\* -0.60 0.00  
## Age\* 0.30 0.00  
## Occupation -1.22 0.02  
## City\_Category\* -1.27 0.00  
## Stay\_In\_Current\_City\_Years\* -1.07 0.00  
## Marital\_Status -1.87 0.00  
## Product\_Category\_1 1.27 0.01  
## Product\_Category\_2 -1.43 0.02  
## Product\_Category\_3 -0.81 0.02

# 3. Univariate Analysis

Let’s First Find categorical and continuos variables in our training data set.

str(train\_set)

## 'data.frame': 412551 obs. of 12 variables:  
## $ User\_ID : int 1000442 1000778 1004647 1002875 1001675 1003848 1002801 1003626 1004682 1002774 ...  
## $ Product\_ID : Factor w/ 3631 levels "P00000142","P00000242",..: 356 1673 2771 1017 2244 971 3510 1866 2134 1183 ...  
## $ Gender : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 1 2 2 2 ...  
## $ Age : Factor w/ 7 levels "0-17","18-25",..: 3 2 4 3 3 3 3 3 3 3 ...  
## $ Occupation : int 1 17 20 2 6 15 3 17 7 1 ...  
## $ City\_Category : Factor w/ 3 levels "A","B","C": 1 3 2 2 2 3 1 2 2 1 ...  
## $ Stay\_In\_Current\_City\_Years: Factor w/ 5 levels "0","1","2","3",..: 3 5 2 4 2 2 2 4 4 2 ...  
## $ Marital\_Status : int 0 0 1 1 1 0 1 0 1 0 ...  
## $ Product\_Category\_1 : int 5 8 8 1 11 5 5 5 8 1 ...  
## $ Product\_Category\_2 : int NA NA NA 2 NA NA 8 11 NA 2 ...  
## $ Product\_Category\_3 : int NA NA NA 8 NA NA NA NA NA 15 ...  
## $ Purchase : int 5472 9996 5864 19483 4592 8724 5183 5141 8050 15199 ...

## Data Prepartion & Exploratory Data Analysis

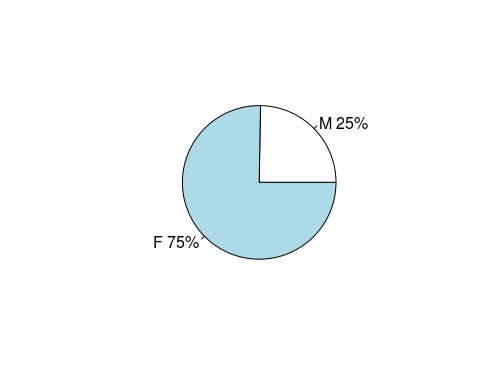
Firstly doing the univariate exploration and modifying the data if deemed necessary.

**one way contigency table**

mytable <- with(train\_set,table(Gender))  
mytable

## Gender  
## F M   
## 102021 310530

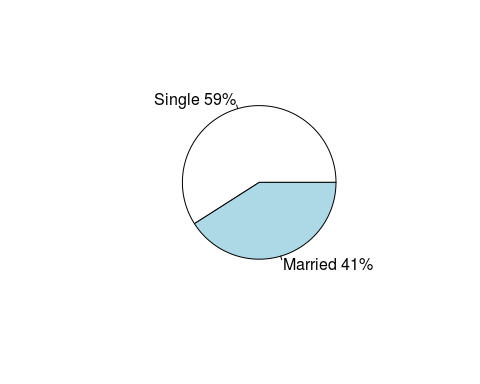
lbls <- c("M","F")  
pct <- round(mytable/sum(mytable)\*100)  
lbls <- paste(lbls, pct)   
lbls <- paste(lbls,"%",sep="")  
pie(mytable,labels = lbls)



mytable1 <- with(train\_set,table(Marital\_Status))  
mytable1

## Marital\_Status  
## 0 1   
## 243555 168996

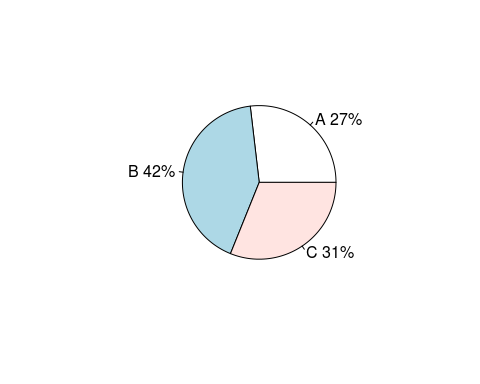
lbls1 <- c("Single","Married")  
pct1 <- round(mytable1/sum(mytable1)\*100)  
lbls1 <- paste(lbls1, pct1)   
lbls1<- paste(lbls1,"%",sep="")  
pie(mytable1,labels = lbls1)



mytable2 <- with(train\_set,table(City\_Category))  
mytable2

## City\_Category  
## A B C   
## 110815 173429 128307

lbls2 <- c("A","B","C")  
pct2 <- round(mytable2/sum(mytable2)\*100)  
lbls2 <- paste(lbls2, pct2)   
lbls2<- paste(lbls2,"%",sep="")  
pie(mytable2,labels = lbls2)



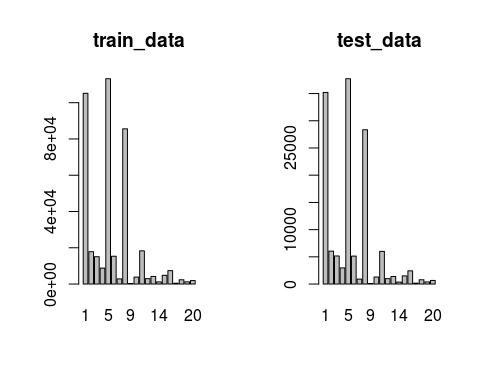
par(mfrow = c(1, 2))  
#train data  
with(train\_set,table(Stay\_In\_Current\_City\_Years))

## Stay\_In\_Current\_City\_Years  
## 0 1 2 3 4+   
## 55676 145504 76472 71518 63381

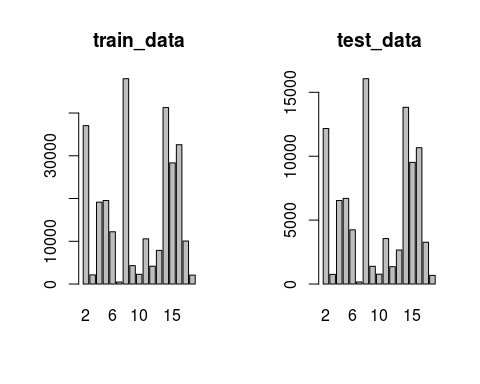
#test data  
with(test\_set,table(Stay\_In\_Current\_City\_Years))

## Stay\_In\_Current\_City\_Years  
## 0 1 2 3 4+   
## 18722 48317 25366 23767 21345

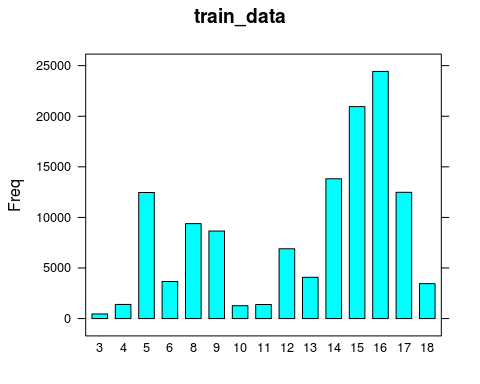
par(mfrow = c(1, 2))  
a<-with(train\_set,table(Product\_Category\_1))  
#a  
barplot(a,main ="train\_data")  
b<-with(test\_set,table(Product\_Category\_1))  
#b  
barplot(b,main ="test\_data")



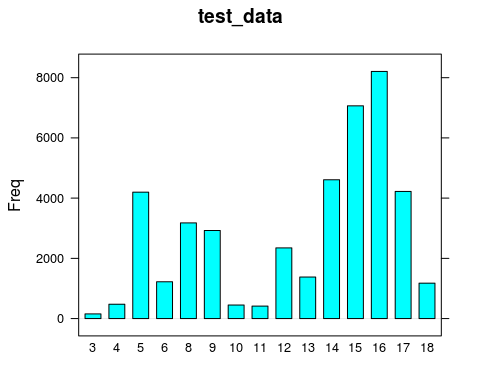
par(mfrow = c(1, 2))  
a<-with(train\_set,table(Product\_Category\_2))  
#a  
barplot(a,main ="train\_data")  
b<-with(test\_set,table(Product\_Category\_2))  
#b  
barplot(b,main ="test\_data")



library(lattice)  
par(mfrow = c(1, 2))  
a<-with(train\_set,table(Product\_Category\_3))  
#a  
barchart(a,main ="train\_data",horizontal = "FALSE")



b<-with(test\_set,table(Product\_Category\_3))  
#b  
barchart(b,main ="test\_data",horizontal = "FALSE")



**2 way table**

xtabs(~Marital\_Status+Gender,data=train\_set)

## Gender  
## Marital\_Status F M  
## 0 59207 184348  
## 1 42814 126182

xtabs(Purchase~Product\_Category\_3,aggregate(Purchase~Product\_Category\_3,train\_set,mean))

## Product\_Category\_3  
## 3 4 5 6 8 9 10   
## 14058.795 9806.369 12128.368 13179.604 13047.715 10388.366 13507.104   
## 11 12 13 14 15 16 17   
## 12113.981 8703.027 13237.030 10036.091 12342.484 11960.804 11737.853   
## 18   
## 11015.255

xtabs(Purchase~Product\_Category\_2,aggregate(Purchase~Product\_Category\_2,train\_set,mean))

## Product\_Category\_2  
## 2 3 4 5 6 7 8   
## 13610.833 11222.781 10216.105 9033.057 11516.350 6894.512 10247.663   
## 9 10 11 12 13 14 15   
## 7329.170 15715.654 8893.459 6996.297 9662.622 7101.230 10345.805   
## 16 17 18   
## 10289.805 9421.296 9328.248

xtabs(Purchase~Product\_Category\_1,aggregate(Purchase~Product\_Category\_1,train\_set,mean))

## Product\_Category\_1  
## 1 2 3 4 5 6   
## 13600.94860 11232.00870 10109.66465 2334.29817 6240.79881 15830.96146   
## 7 8 9 10 11 12   
## 16374.65185 7497.88064 15754.68932 19668.12699 4687.24631 1344.33266   
## 13 14 15 16 17 18   
## 720.56800 13143.73906 14790.16472 14754.58678 10167.59811 2979.82756   
## 19 20   
## 36.78061 369.85307

## 3.1. Unique Data For EDA & Data type modification

Since our data is stored based on product id i.e. if a person buying 10 products then his data will be stored in 10 observation and hence there will cause a repetition of same person data.

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)

The function distinct() in dplyr package can be used to keep only unique/distinct rows from a data frame. If there are duplicate rows, only the first row is preserved. It’s an efficient version of the R base function unique().

library("dplyr")

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

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)

## User\_ID Gender Age Occupation City\_Category Stay\_In\_Current\_City\_Years  
## 1 1000442 Male 26-35 1 A 2  
## 2 1000778 Male 18-25 17 C 4+  
## 3 1004647 Male 36-45 20 B 1  
## 4 1002875 Male 26-35 2 B 3  
## 5 1001675 Female 26-35 6 B 1  
## 6 1003848 Male 26-35 15 C 1  
## Marital\_Status  
## 1 Single  
## 2 Single  
## 3 Married  
## 4 Married  
## 5 Married  
## 6 Single

# 3.2. User\_ID

**Creating new variables User\_Purchase\_Count, Total\_Spending and Avg\_Spending\_on\_each\_Product**

**Total number of product purchased bu customer**

#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)

## User\_ID User\_Purchase\_Count  
## 1 1000001 24  
## 2 1000002 63  
## 3 1000003 25  
## 4 1000004 13  
## 5 1000005 78  
## 6 1000006 37

merging df

train\_set <- merge(x = train\_set, y = userIDCount, by = "User\_ID", all.x = TRUE)

changing the test dataset too

# 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)  
test\_set[is.na(test\_set$User\_Purchase\_Count), "User\_Purchase\_Count"] <- 1  
#Changing the datatype  
test\_set$User\_Purchase\_Count <- as.integer(test\_set$User\_Purchase\_Count)

**Total Spending by a User**

Changing the Training Dataset

# 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)  
names(totspend) <- c("User\_ID","Total\_Spending")  
train\_set <- merge(x = train\_set, y = totspend, by = "User\_ID", all.x = TRUE)  
#head(train\_set)

changing test too

test\_set <- merge(x = test\_set, y = totspend, by = "User\_ID", all.x = TRUE)  
# writing code such that if a new user comes for the first time his count is set to zero for total spending in test dataset  
test\_set[is.na(test\_set$Total\_Spending), "Total\_Spending"] <- 0  
#Changing the datatype  
test\_set$Total\_Spending <- as.integer(test\_set$Total\_Spending)

**Average Spending on each product by a customer**

Now calculating the average spending of user on each product we can divide Total\_Spending by User\_Purchase Count. Avg\_Spending\_on\_each\_Product = Total\_Spending / User\_Purchase\_Count

train\_set$Avg\_Spending\_on\_each\_Product <- train\_set$Total\_Spending / train\_set$User\_Purchase\_Count  
test\_set$Avg\_Spending\_on\_each\_Product <- test\_set$Total\_Spending / test\_set$User\_Purchase\_Count  
#Changing the datatype   
train\_set$Avg\_Spending\_on\_each\_Product <- as.integer(train\_set$Avg\_Spending\_on\_each\_Product)  
test\_set$Avg\_Spending\_on\_each\_Product <- as.integer(test\_set$Avg\_Spending\_on\_each\_Product)

summary(totspend$Total\_Spending)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 20125 178411 388922 648555 842252 8050867

library(ggplot2)

##   
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':  
##   
## %+%, alpha

ggplot(totspend, aes(x=Total\_Spending)) + geom\_density(fill="red", col="black", alpha=0.80)

