Data Science Capstone - ECom_ShippingData

Mehul Mohta

15/05/2021

Executive Summary

The Project is in regards to the HarvardX:PH125:9x Data Science Capstone CYO project. The project I have selected is about predicting accuracy of user ratings for a eCommerce company. Data set from Kaggle has been used for this project. Data description and data cleaning is done along with data visualization so as to get a perspective of the data. Four different models/algorithms are evaluated to decide best accuracy levels. **The goal of the project is to develop a machine learning algorithm using the inputs in one subset to predict review ratings accuracy in the other (validation) set.**

This report contains 5 sections - context & problem definition, data description & loading, exploratory analysis, modeling and data analysis, results and conclusion.During this project report, you will find 4 different models used for estimating the accuracy **The best accuracy derived out of the below modeling exercise is 69.6%**

(1/5). Context & Problem Definition

Context - Year 2020 and 2021 has been worst years in last century due to the raging pandemic. In these difficult times there also has been some silver linings, we all started spending more time with families, nature got a chance to heal itself, etc. Along with this sea of changes, there also has been a change in consumer behaviour. Not only, what is bought but even from where we buy has got prominent importance. Shopping sitting at home or online or as it's known eCommerce has got very important role to play. More and more studies are reflecting that not only developing countries but even developed countries are seeing penetration of eCommerce gaining over General Trade OR Mom & Pop Stores. One of the influencing element while shopping online is whether shipped product will reach on time. In this report we will try and understand which variables affect delivery on time and thereby build a model of accuracy for future deliveries. Being a strategy professional this project has not only personal but even professional implications for myself

Problem Definition - The goal of this project is to train a machine learning algorithm using the inputs in one (training) set to predict accuracy of user ratings in the other (validation) set. I will use four different algorithms - knn, gbm, rpart & random forest to estimate the accuracy. Basis the model which shows highest degree of accuracy on validation data set, shall be the final recommended algorithm. The data is used from Kaggle Data Repository - https://www.kaggle.com/prachi13/customer-analytics. The data set has 10999 entries and 12 columns, details of the same you will find in next section

(2/5). Data Description & Loading

The file can be downloaded from github at: https://github.com/mehulmohta/CYO-Project-E-Commerce-Shipping-Data/blob/main/Train.csv

```
#Install packages we may need
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(matrixStats)) install.packages("matrixStats", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")

#Libraries we are going to work
library(tidyverse)
library(caret)
library(dplyr)
library(rpart)
library(randomForest)
library(matrixStats)
library(gbm)
library(data.table)
library(ggplot2)
library(gridExtra)
library(corrplot)
library(rpart.plot)
```

```
#Download the eCommerce data file

d1 <- tempfile()
download.file("https://raw.githubusercontent.com/mehulmohta/CYO-Project-E-Commerce-Shipping-Data/main/T

#Read the csv file
ecomdata <- read.csv(d1)
```

The dataset used for model building contained 10999 observations of 12 variables. The data contains the following information:

ID: ID Number of Customers. Warehouse block: The Company have big Warehouse which is divided in to block such as A,B,C,D,E. Mode of shipment:The Company Ships the products in multiple way such as Ship, Flight and Road. Customer care calls: The number of calls made by customer for enquiry of the shipment. Customer rating: The rating for every customer. 1 is the lowest (Worst), 5 is the highest (Best). Cost of the product: Cost of the Product in US Dollars. Prior purchases: The Number of Prior Purchases. Product importance: The company has categorized the product in the various parameter such as low, medium, high. Gender: Male and Female. Discount offered: Discount offered on that specific product. This is also in US Dollars Weight in gms: It is the weight in grams. Reached on time: It is the target variable, 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.

(3/5). Exploratory Analysis

We will explore the data to build our understanding and also perform any data modification/cleaning that would be necessary for our modeling

```
#Examine the structure of the dataset
str(ecomdata)
```

```
## 'data.frame':    10999 obs. of  12 variables:
##  $ ï..ID              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Warehouse_block    : chr  "D" "F" "A" "B" ...
##  $ Mode_of_Shipment   : chr  "Flight" "Flight" "Flight" "Flight" ...
##  $ Customer_care_calls: int  4 4 2 3 2 3 3 4 3 3 ...
##  $ Customer_rating    : int  2 5 2 3 2 1 4 1 4 2 ...
##  $ Cost_of_the_Product: int  177 216 183 176 184 162 250 233 150 164 ...
##  $ Prior_purchases    : int  3 2 4 4 3 3 3 2 3 3 ...
##  $ Product_importance : chr  "low" "low" "low" "medium" ...
##  $ Gender             : chr  "F" "M" "M" "M" ...
##  $ Discount_offered   : int  44 59 48 10 46 12 3 48 11 29 ...
##  $ Weight_in_gms      : int  1233 3088 3374 1177 2484 1417 2371 2804 1861 1187 ...
##  $ Reached.on.Time_Y.N: int  1 1 1 1 1 1 1 1 1 1 ...
```

Here we can see that the data has 10999 observations of 12 variables.

```
#Missing value analysis
colSums(is.na(ecomdata))
```

```
##               ï..ID     Warehouse_block    Mode_of_Shipment Customer_care_calls
##                   0                   0                   0                   0
##     Customer_rating Cost_of_the_Product     Prior_purchases  Product_importance
##                   0                   0                   0                   0
##              Gender    Discount_offered       Weight_in_gms Reached.on.Time_Y.N
##                   0                   0                   0                   0
```

As seen above none of the columns have blank values

```
#Summary view of data
summary(ecomdata)
```

```
##      ï..ID       Warehouse_block    Mode_of_Shipment   Customer_care_calls
##  Min.   :    1   Length:10999       Length:10999       Min.   :2.000
##  1st Qu.: 2750   Class :character   Class :character   1st Qu.:3.000
##  Median : 5500   Mode  :character   Mode  :character   Median :4.000
##  Mean   : 5500                                         Mean   :4.054
##  3rd Qu.: 8250                                         3rd Qu.:5.000
##  Max.   :10999                                         Max.   :7.000
##  Customer_rating Cost_of_the_Product Prior_purchases  Product_importance
##  Min.   :1.000   Min.   : 96.0       Min.   : 2.000   Length:10999
##  1st Qu.:2.000   1st Qu.:169.0       1st Qu.: 3.000   Class :character
##  Median :3.000   Median :214.0       Median : 3.000   Mode  :character
##  Mean   :2.991   Mean   :210.2       Mean   : 3.568
##  3rd Qu.:4.000   3rd Qu.:251.0       3rd Qu.: 4.000
```

```
##  Max.   :5.000   Max.   :310.0       Max.   :10.000
##     Gender          Discount_offered Weight_in_gms  Reached.on.Time_Y.N
##  Length:10999       Min.   : 1.00    Min.   :1001   Min.   :0.0000
##  Class :character   1st Qu.: 4.00    1st Qu.:1840   1st Qu.:0.0000
##  Mode  :character   Median : 7.00    Median :4149   Median :1.0000
##                     Mean   :13.37    Mean   :3634   Mean   :0.5967
##                     3rd Qu.:10.00    3rd Qu.:5050   3rd Qu.:1.0000
##                     Max.   :65.00    Max.   :7846   Max.   :1.0000
```

Some quick observations:

1. Customers have made anywhere between 2 to 7 enquiry calls
2. Cost of the products is the range of $96 to $310
3. Customers have made prior purchases between 2 to 10
4. Discount offered is in the range of $1 to $65
5. Weight of the product is in range of 1.001KG to 7.846KG

Now our first step is to identify the categorical and check/convert them into factors. *When the outcome is categorical, we refer to the machine-learning task as classification. Our predictions will be categorical just like our outcomes, and they will be either correct or incorrect*

Converting variables to factors

```
names <- c(2,3,8,9,12)
ecomdata[,names] <- lapply(ecomdata[,names], factor)
str(ecomdata)
```

```
## 'data.frame':    10999 obs. of  12 variables:
##  $ ï..ID              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Warehouse_block    : Factor w/ 5 levels "A","B","C","D",..: 4 5 1 2 3 5 4 5 1 2 ...
##  $ Mode_of_Shipment   : Factor w/ 3 levels "Flight","Road",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Customer_care_calls: int  4 4 2 3 2 3 3 4 3 3 ...
##  $ Customer_rating    : int  2 5 2 3 2 1 4 1 4 2 ...
##  $ Cost_of_the_Product: int  177 216 183 176 184 162 250 233 150 164 ...
##  $ Prior_purchases    : int  3 2 4 4 3 3 3 2 3 3 ...
##  $ Product_importance : Factor w/ 3 levels "high","low","medium": 2 2 2 3 3 3 2 2 2 3 ...
##  $ Gender             : Factor w/ 2 levels "F","M": 1 2 2 2 1 1 1 1 1 1 ...
##  $ Discount_offered   : int  44 59 48 10 46 12 3 48 11 29 ...
##  $ Weight_in_gms      : int  1233 3088 3374 1177 2484 1417 2371 2804 1861 1187 ...
##  $ Reached.on.Time_Y.N: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

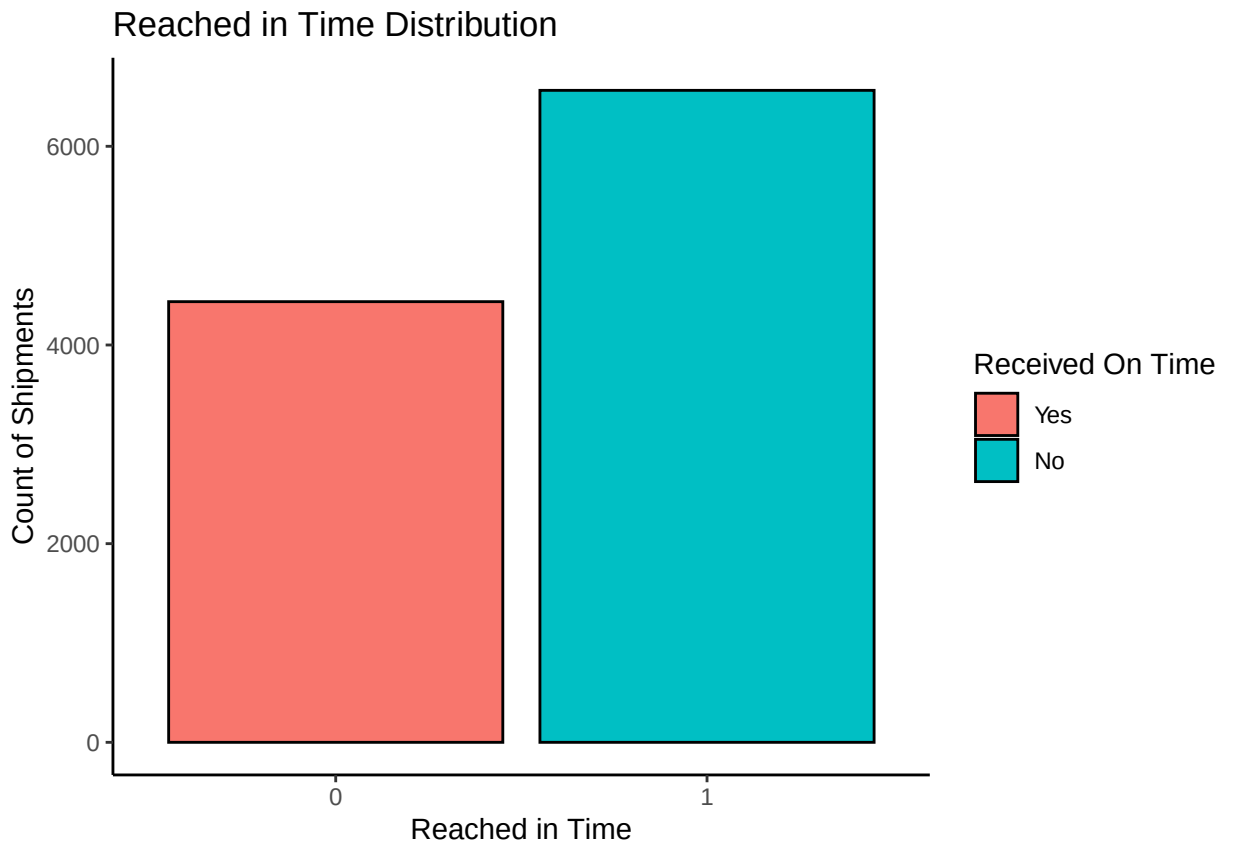
We will now try and understand the variables for two objectives -

1. Their distribution, if there are any outliers
2. Co-relation among themselves and with target variable

But before we do not we would simplify the column headers

```
#We will first see the distribution of the target variable Reached in Time
ecomdata %>% ggplot(aes(Rch_Time)) +
             geom_bar(aes(fill = Rch_Time),color = 'black') +
             theme_classic() +
```

```
            xlab("Reached in Time") +
            ylab("Count of Shipments") +
            scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
            labs(title = "Reached in Time Distribution")
```

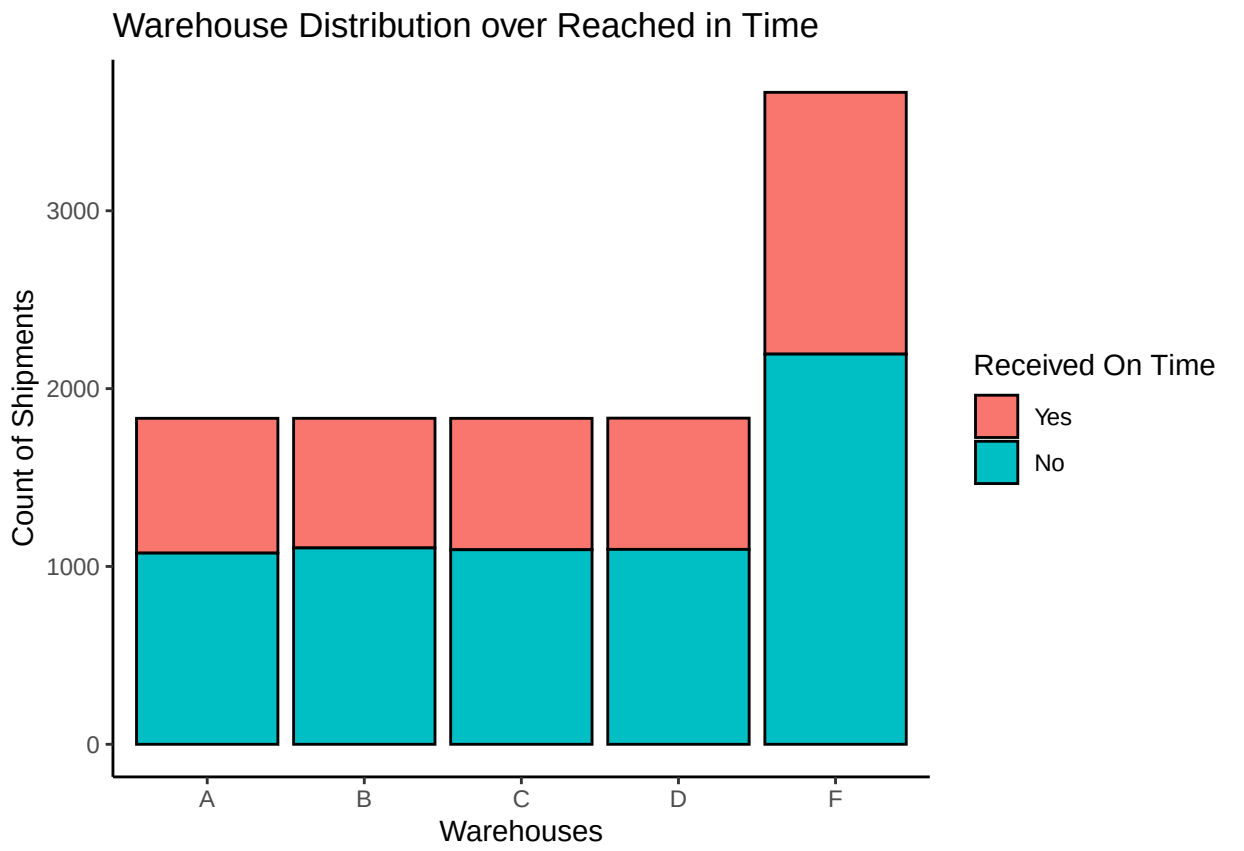



```
prop.table(table(ecomdata$Rch_Time))*100
```

```
##
##        0        1
## 40.33094 59.66906
```

0 reflects eCommerce delivery happening in defined time. In the overall data set only 40% deliveries happen on time

```
#We will see the distribution of Warehouse against the target variable Reached in Time
ecomdata %>% ggplot(aes(Wr_bl)) +
            geom_bar(aes(fill = Rch_Time),color = 'black') +
            theme_classic() +
            xlab("Warehouses") +
            ylab("Count of Shipments") +
            scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
            labs(title = "Warehouse Distribution over Reached in Time")
```




```
prop.table(table(ecomdata$Wr_bl,ecomdata$Rch_Time))*100
```

```
##
##             0         1
##   A  6.891536  9.773616
##   B  6.627875 10.037276
##   C  6.718793  9.946359
##   D  6.709701  9.964542
##   F 13.383035 19.947268
```

```
prop.table(table(ecomdata$Wr_bl,ecomdata$Rch_Time),1)*100
```

```
##
##            0        1
##   A 41.35297 58.64703
##   B 39.77087 60.22913
##   C 40.31642 59.68358
##   D 40.23991 59.76009
##   F 40.15276 59.84724
```

First table shows, Warehouse labeled F has the maximum (~3750) deliveries or (~33% of deliveries) and second table shows '60% did not reach on time

```
#We will now see the distribution of Mode of Shipping against the target variable Reached in Time
ecomdata %>% ggplot(aes(M_Ship)) +
             geom_bar(aes(fill = Rch_Time),color = 'black') +
             theme_classic() +
             xlab("Mode of Shipping") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title = "Mode of Shipping Distribution over Reached in Time")
```


```
prop.table(table(ecomdata$M_Ship,ecomdata$Rch_Time))*100
```

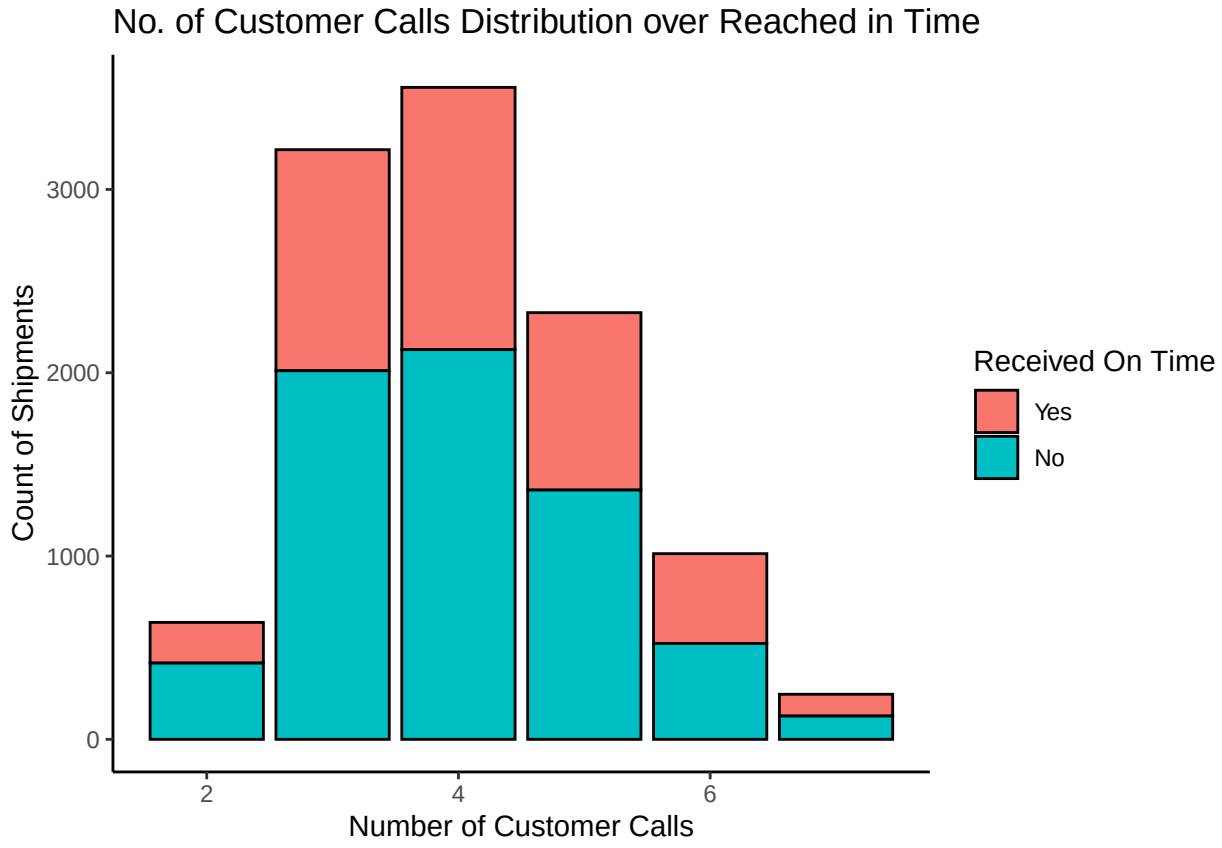
```
##
##                  0         1
##   Flight  6.436949  9.719065
##   Road    6.591508  9.409946
##   Ship   27.302482 40.540049
```

```
prop.table(table(ecomdata$M_Ship,ecomdata$Rch_Time),1)*100
```

```
##
##                  0        1
##   Flight 39.84243 60.15757
##   Road   41.19318 58.80682
##   Ship   40.24390 59.75610
```

First table shows, shipping is most commonly used mode (~67% of shipments) and second table shows again ~60% did not reach on time

```
#We will see the distribution of Customer Calls against the target variable Reached in Time
ecomdata %>% ggplot(aes(Cust_Call)) +
             geom_bar(aes(fill = Rch_Time),color = 'black') +
             theme_classic() +
             xlab("Number of Customer Calls") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title = "No. of Customer Calls Distribution over Reached in Time")
```




```
prop.table(table(ecomdata$Cust_Call,ecomdata$Rch_Time))*100
```

```
##
##            0        1
##   2  2.018365  3.782162
##   3 10.964633 18.283480
##   4 13.010274 19.329030
##   5  8.800800 12.364760
##   6  4.454950  4.754978
##   7  1.081917  1.154650
```

```
prop.table(table(ecomdata$Cust_Call,ecomdata$Rch_Time),1)*100
```

```
##
##            0        1
##   2 34.79624 65.20376
##   3 37.48834 62.51166
##   4 40.23053 59.76947
##   5 41.58076 58.41924
##   6 48.37117 51.62883
##   7 48.37398 51.62602
```

First table shows, most customers ends up calling between 3 to 5 times (~77%) and second table shows again ~60% did not reach on time, however for customers calling 6 to 7 times more deliveries reached in time

```
#We will see the distribution of Customer Rating against the target variable Reached in Time
ecomdata %>% ggplot(aes(Cust_Rat)) +
             geom_bar(aes(fill = Rch_Time),color = 'black') +
             theme_classic() +
             xlab("Customer Rating") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title = "Customer Rating Distribution over Reached in Time")
```

```
prop.table(table(ecomdata$Cust_Rat,ecomdata$Rch_Time))*100
```

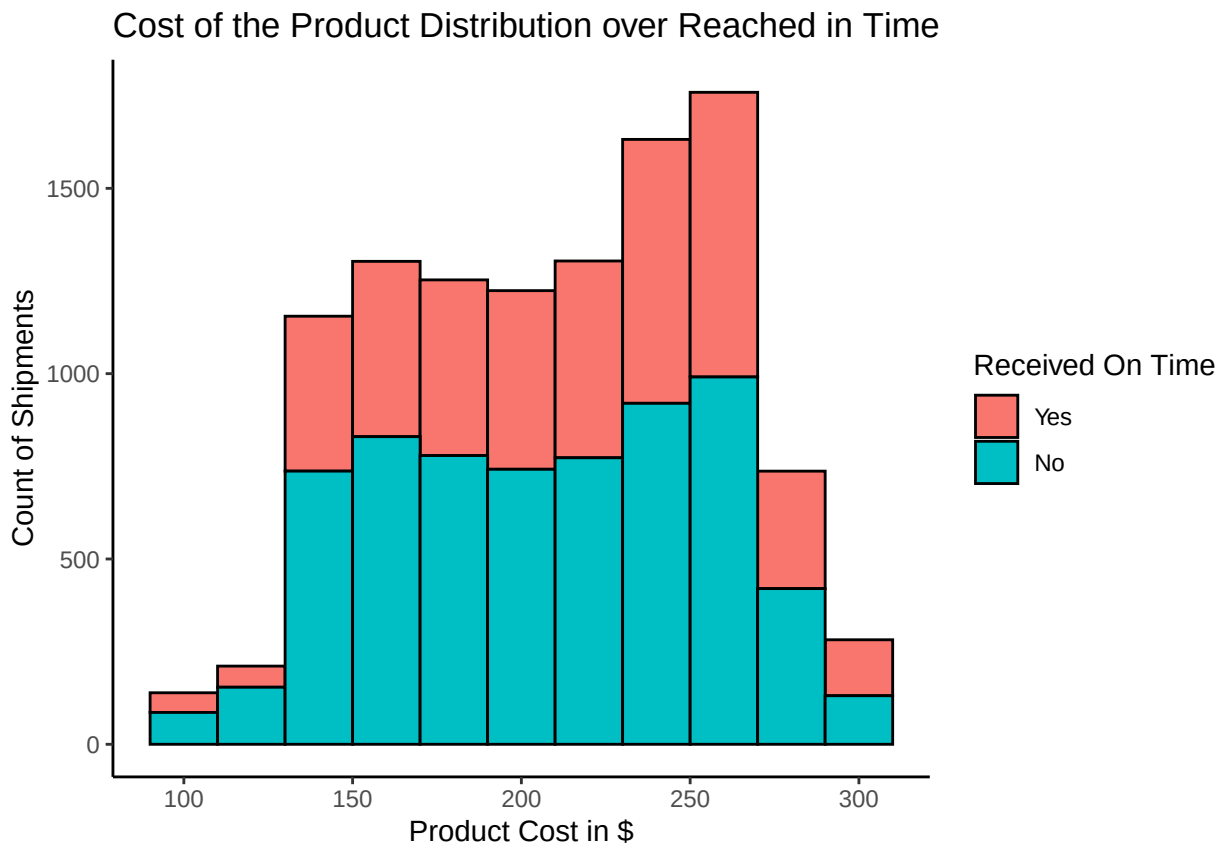
```
##
##             0         1
##   1  8.382580 11.937449
##   2  8.109828 11.573779
##   3  8.018911 12.337485
##   4  8.055278 11.846532
##   5  7.764342 11.973816
```

```
prop.table(table(ecomdata$Cust_Rat,ecomdata$Rch_Time),1)*100
```

```
##
##            0        1
##   1 41.25280 58.74720
##   2 41.20092 58.79908
##   3 39.39259 60.60741
##   4 40.47510 59.52490
##   5 39.33671 60.66329
```

The first table shows there is almost equal number of customers between each rating and second table shows again~60% deliveries did not reach on time

```
#We will see the distribution of Cost of Product against the target variable Reached in Time
ecomdata %>% ggplot(aes(Cost_Prd)) +
             geom_histogram(aes(fill = Rch_Time),color= 'black' ,binwidth=20) +
             theme_classic() +
             xlab("Product Cost in $") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title= "Cost of the Product Distribution over Reached in Time")
```

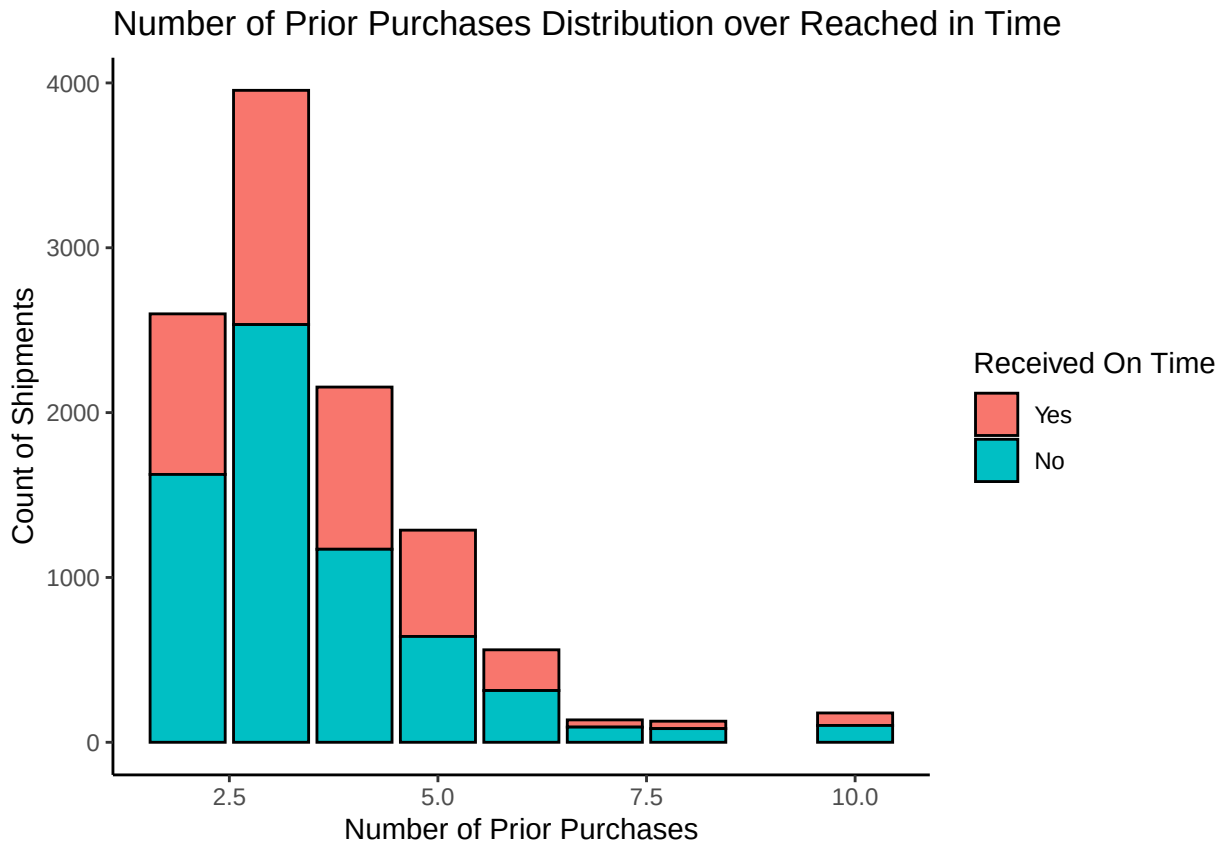



```
v1 <- ecomdata %>% filter(Cost_Prd >224, Cost_Prd<276) %>% nrow()
v1/nrow(ecomdata)
```

```
## [1] 0.3811256
```

We can see highest number of products costed between $225-$275 (~38% of products sold & delivered)

```
#We will see the distribution of Prior Purchases against the target variable Reached in Time
ecomdata %>% ggplot(aes(Pri_Pur)) +
             geom_bar(aes(fill = Rch_Time),color = 'black') +
             theme_classic()  +
             xlab("Number of Prior Purchases") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title= "Number of Prior Purchases Distribution over Reached in Time")
```




```
prop.table(table(ecomdata$Pri_Pur,ecomdata$Rch_Time))*100
```

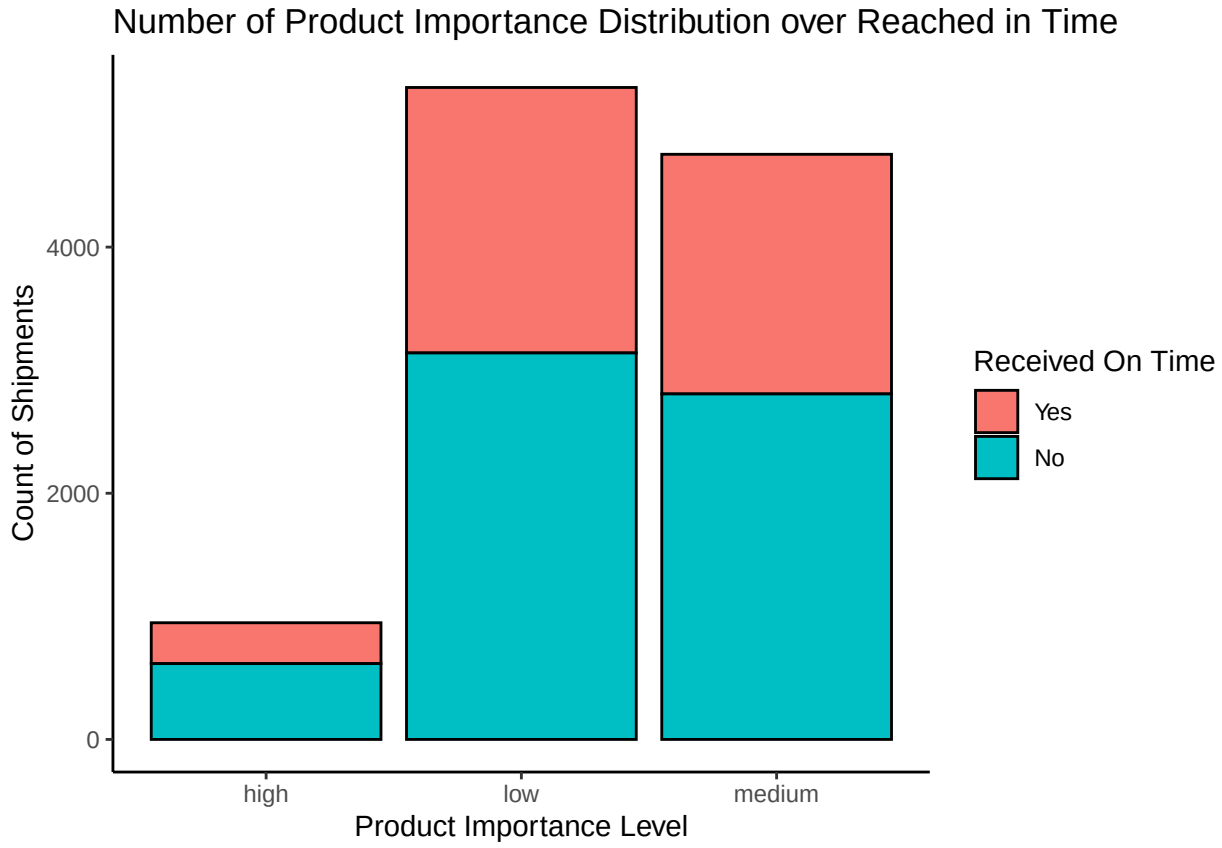
```
##
##               0         1
##   2   8.8553505 14.7740704
##   3  12.9193563 23.0384580
##   4   8.9462678 10.6464224
##   5   5.8641695  5.8368943
##   6   2.2456587  2.8548050
##   7   0.4000364  0.8364397
##   8   0.4091281  0.7546141
##   10  0.6909719  0.9273570
```

```
prop.table(table(ecomdata$Pri_Pur,ecomdata$Rch_Time),1)*100
```

```
##
##             0        1
##   2  37.47595 62.52405
##   3  35.92920 64.07080
##   4  45.66125 54.33875
##   5  50.11655 49.88345
##   6  44.02852 55.97148
##   7  32.35294 67.64706
##   8  35.15625 64.84375
##   10 42.69663 57.30337
```

The first table shows ~35% of customers have made 3 prior purchases and second table shows again ~64% deliveries did not reach on time

```
#We will  see the distribution of Product Importance against the target variable Reached in Time
ecomdata %>% ggplot(aes(Prd_Imp)) +
             geom_bar(aes(fill = Rch_Time),color = 'black') +
             theme_classic()  +
             xlab("Product Importance Level") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title= "Number of Product Importance Distribution over Reached in Time")
```




```
prop.table(table(ecomdata$Prd_Imp,ecomdata$Rch_Time))*100
```

```
##
##                  0         1
##   high    3.018456  5.600509
##   low    19.610874 28.548050
##   medium 17.701609 25.520502
```

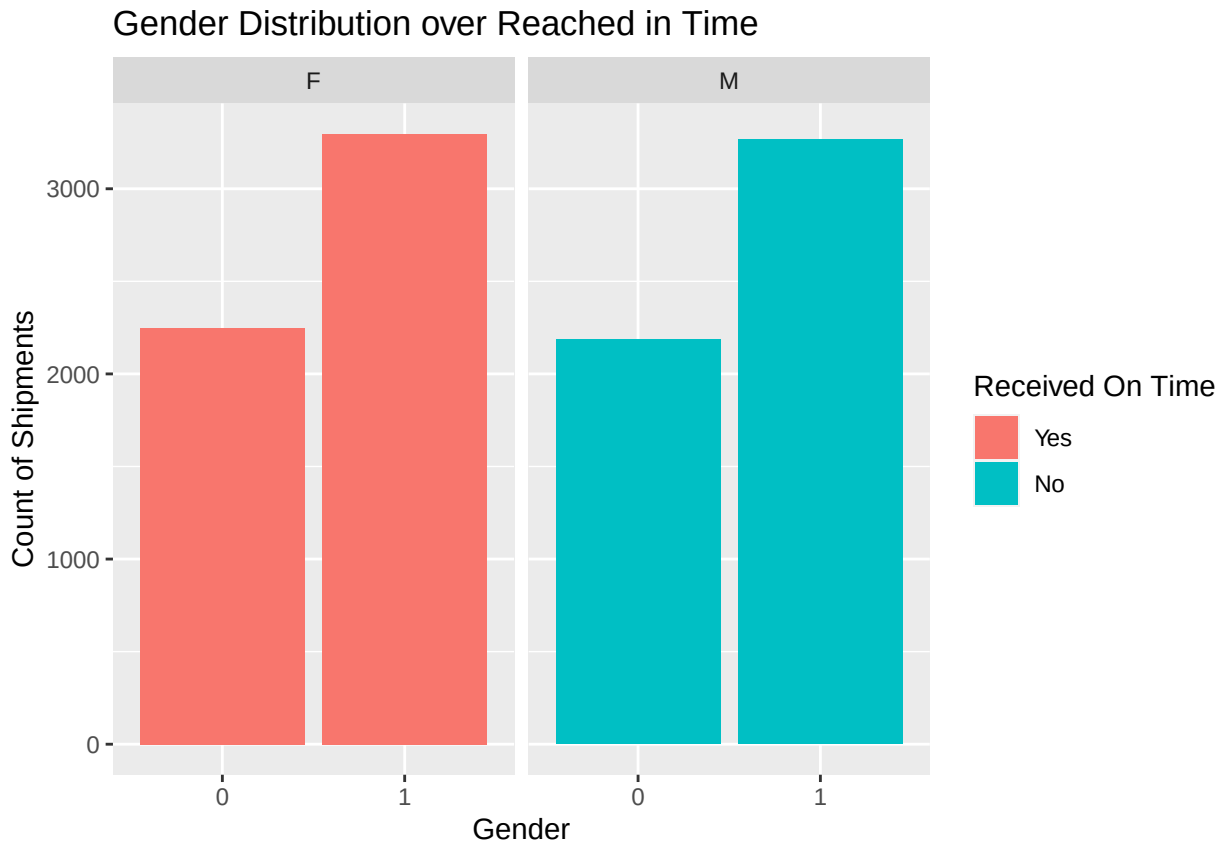
```
prop.table(table(ecomdata$Prd_Imp,ecomdata$Rch_Time),1)*100
```

```
##
##                 0        1
```

```
##   high   35.02110 64.97890
##   low    40.72116 59.27884
##   medium 40.95499 59.04501
```

The first table shows ~48% of product purchased were in low importance category and 42% in medium importance, and second table shows again ~60% deliveries did not reach on time for these category purchases

```
#We will see the distribution of Gender against the target variable Reached in Time
qplot(Rch_Time, data = ecomdata, fill = Gender) + facet_grid (. ~ Gender) +
            xlab("Gender") +
            ylab("Count of Shipments") +
            scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
            labs(title= "Gender Distribution over Reached in Time")
```



We can see that the data is equally split between Men and Women and also the share are of deliveries which reached in time

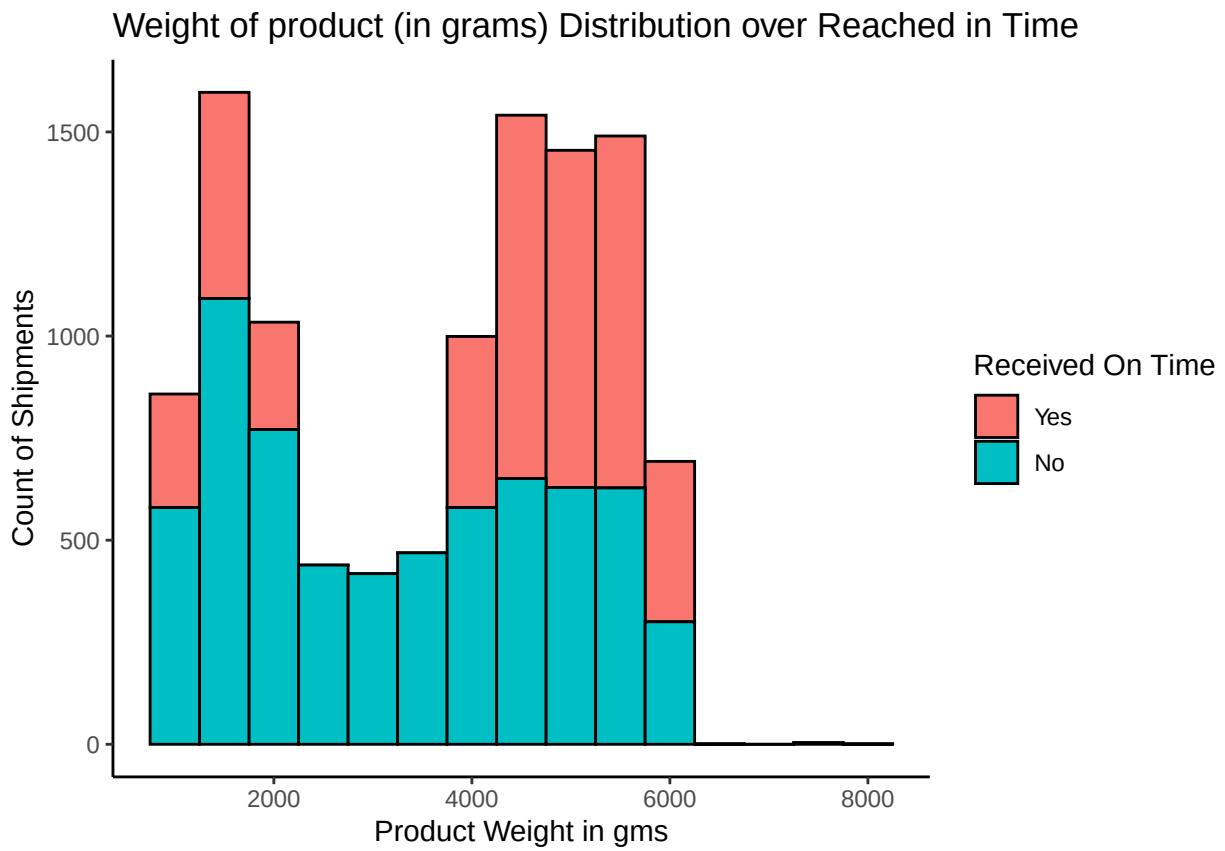
```
#We will see the distribution of Discount Offered against the target variable Reached in Time
ecomdata %>% ggplot(aes(Dis_Off)) +
            geom_histogram(aes(fill = Rch_Time),color= 'black' ,binwidth=5) +
            theme_classic() +
            xlab("Discount Offered in $") +
            ylab("Count of Shipments") +
            scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
            labs(title= "Discounts Offered Distribution over Reached in Time")
```


```
v2 <- ecomdata %>% filter(Dis_Off >1.9, Dis_Off<6.1) %>% nrow()
v2/nrow(ecomdata)
```

```
## [1] 0.3802164
```

We can see majority of discount offered between $2-$6 (~38% of products sold & delivered)and also any product which was offered more than $6 never reached in time

```
#We will see the distribution of Product Weight in grams against the target variable Reached in Time
ecomdata %>% ggplot(aes(Wt_gms)) +
             geom_histogram(aes(fill = Rch_Time),color= 'black' ,binwidth=500) +
             theme_classic() +
             xlab("Product Weight in gms") +
             ylab("Count of Shipments") +
             scale_fill_discrete(name = "Received On Time", labels = c("Yes", "No")) +
             labs(title= "Weight of product (in grams) Distribution over Reached in Time")
```




```
v3 <- ecomdata %>% filter(Wt_gms >3999, Wt_gms<6001) %>% nrow()
v3/nrow(ecomdata)
```

```
## [1] 0.5418674
```

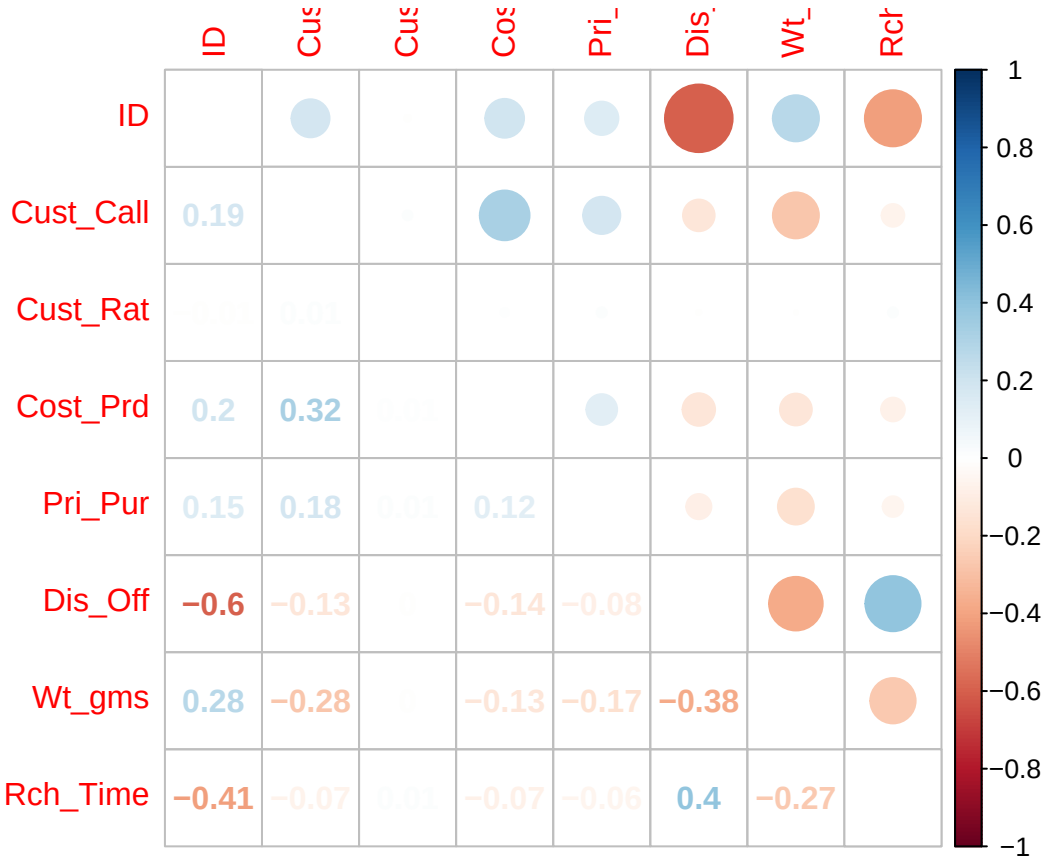
We can see maximum number of products sold, weighted in range of 4Kgs to 6Kgs (~54% of products sold & delivered)

(4/5). Modelling & Data Analysis

Let us perform correlation analysis, which is used to quantify the association between two variables. However since correlations will work on only numbers, we will convert the target variable to integer

```
ecomdata$Rch_Time <- as.integer(as.factor(ecomdata$Rch_Time))

##Correlation analysis
correlation <- cor(ecomdata[sapply(ecomdata, function(x) !is.factor(x))])
diag (correlation) = 0 #Remove self correlations
corrplot.mixed(correlation,tl.pos = "lt")
```




```
# We will turn the target variable back into factor class so that we can use in all our modelling exerc
ecomdata$Rch_Time <- as.factor(ecomdata$Rch_Time)
```

Above matrix shows the variables having positive and negative correlation

We will now split the dataset into train and validation sets.

```
#Splitting the dataset for validation and training.
set.seed(1,sample.kind = "Rounding")   #if using R3.5 or earlier set.seed(1)
test_index <- createDataPartition(ecomdata$Rch_Time, times = 1, p = 0.2, list = FALSE)
ecomdata_validation <- ecomdata[test_index, ]
ecomdata_training <- ecomdata[-test_index, ]
```

We will split the ecomdata_training dataset once more so we can use it to test out various models before choosing a final model.

```
#Splitting the ecomdata_training dataset again for testing
set.seed(10,sample.kind = "Rounding")   #if using R3.5 or earlier set.seed(10)
test_indexsplit <- createDataPartition(ecomdata_training$Rch_Time, times = 1, p = 0.2, list = FALSE)
testing <- ecomdata_training[test_indexsplit, ]
training <- ecomdata_training[-test_indexsplit, ]
```

Machine Learning Models

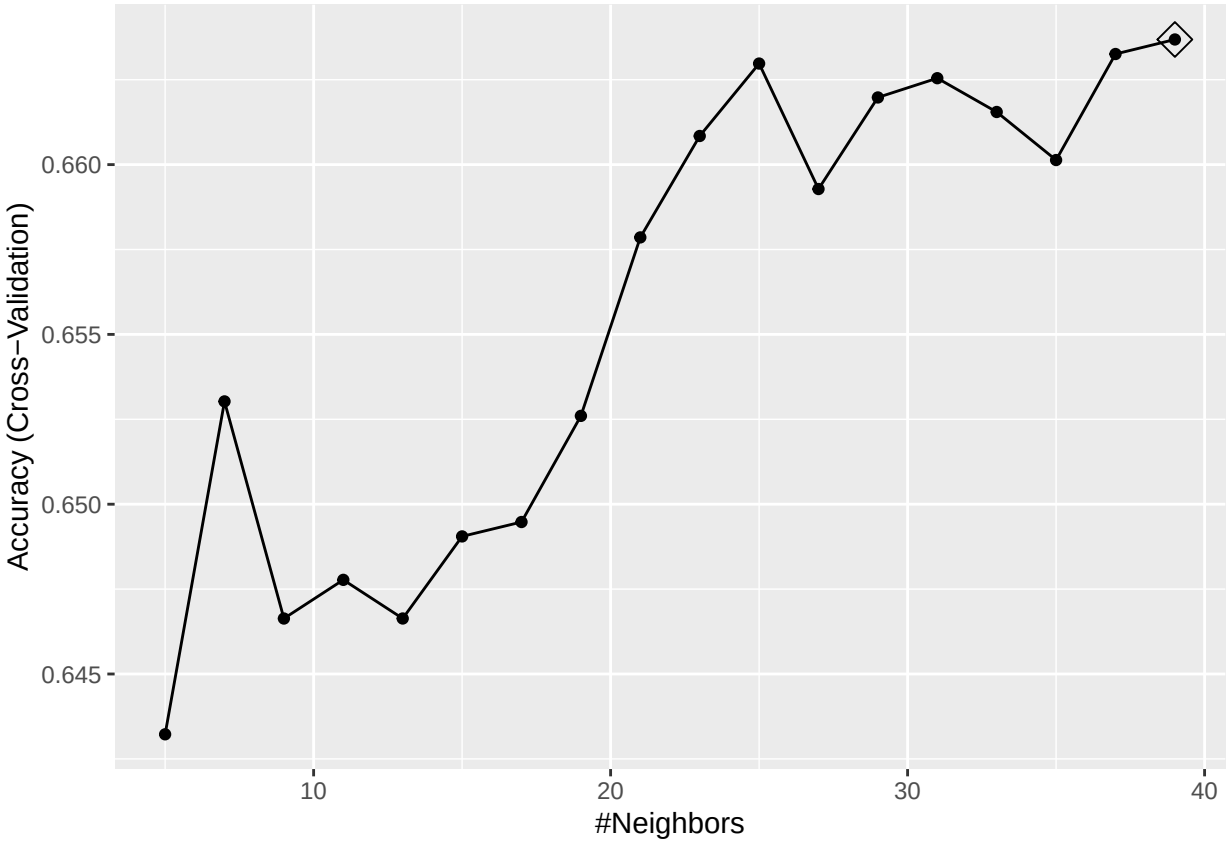
Now we will begin to test various machine learning models to see which has the highest overall accuracy in predicting target variable

knn (K nearest neighbors) Model

We will first try a knn, or k nearest neighbors, model. It is based on a similarity concept and is similar to bin smoothing. Knn is also adaptable to multiple dimensions. It works by calculating the distance between observations based on the attributes. New data points, or observations, are predicted by looking at the k-nearest points and averaging them. Therefore, if the majority of K-neighbors belong to a certain class, the new observation also belongs to that same class.

Here we use k as our tuning parameter, which represents the number of neighbors to be considered. We use a 10-fold cross-validation to make our code run faster and to avoid over-fitting. We will have 10 validation samples that use 10% of the observations in each sample that are used to create separate boosted models. The final model is an ensemble based on all of these models.

```
#Train a knn model on our training dataset optimizing k as the tuning parameter
set.seed(100,sample.kind = "Rounding")  #if using R3.5 or earlier set.seed(100)
#Using a 10 fold cross-validation method to make our code run faster.
control <- trainControl(method = "cv", number = 10)
train_knn <- train(Rch_Time ~ .,
                   method = "knn",
                   data = training,
                   tuneGrid = data.frame(k = seq(5,40,2)),
                   trControl = control)
#Highlighting the optimized k value on this plot
ggplot(train_knn, highlight = TRUE)
```



```
#Use this code to see the best k value
train_knn$bestTune
```

```
##     k
## 18 39
```

```
#Compute the accuracy of the knn model on the testing dataset
knn_accuracy <- confusionMatrix(predict(train_knn, testing, type = "raw"),
                testing$Rch_Time)$overall["Accuracy"]
#Create a table to save our results for each model
accuracy_results <- tibble(method = "knn", Accuracy = knn_accuracy)
#View the knn accuracy results in our table
accuracy_results %>% knitr::kable()
```

method	Accuracy
knn	0.6721591

gbm (Gradient Boosting Machines) Model

For our second model, we will try a gbm, or Gradient Boosting Machines model. This approach creates an ensemble where new models are added sequentially rather than simply averaging the predicted values of all the models. The Gradient Boosting Machine method builds an ensemble of shallow and successive trees where each learns and improves based on previous trees. We will also use a 10 fold cross-validation method here.

```
#Train a gbm model on our training dataset using 10-fold cross-validation
set.seed(200,sample.kind = "Rounding") #if using R3.5 or earlier set.seed(200)
#Using a 10 fold cross-validation method to make our code run faster
 trCtrl <- trainControl (method = "cv", number = 10)
#Train a gbm model
 train_gbm <- train (Rch_Time~ .,
                trControl = trCtrl,
                method = "gbm",
                preProc="zv",
                data = training,
                verbose = FALSE)

#Compute the accuracy of our gbm model on the testing dataset
gbm_accuracy <- confusionMatrix(predict(train_gbm, testing, type = "raw"),
                testing$Rch_Time)$overall["Accuracy"]
#Save the gbm accuracy results to our table
accuracy_results <- bind_rows(accuracy_results, tibble(method="gbm",
                                     Accuracy = gbm_accuracy))
#View the gbm accuracy results in our table
accuracy_results %>% knitr::kable()
```

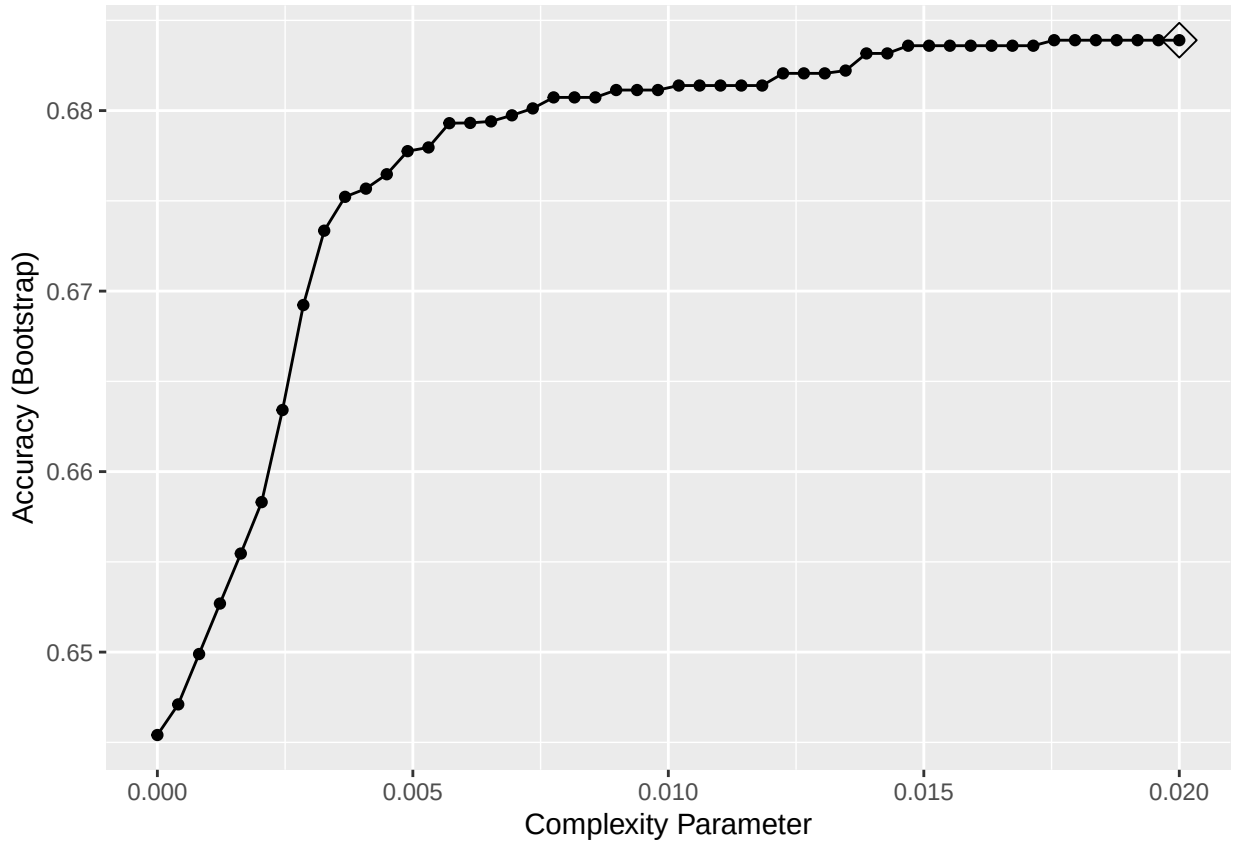
method	Accuracy
knn	0.6721591
gbm	0.6795455

Classification Tree Model

For our third model, we will train a Classification Tree algorithm using the rpart method from the caret package. A tree can be described as a flow chart with yes or no questions and predictions at the ends that are called nodes. Decision trees are a type of supervised learning algorithm that work by partitioning the predictor space in order to predict an outcome, which in our case is received in time The partitions are created recursively.

We will use cross-validation to choose the best cp (complexity parameter).

```
#Train a Classification Tree model using rpart and optimizing for the complexity parameter
set.seed(300,sample.kind = "Rounding")  #if using R3.5 or earlier set.seed(300)
train_rpart <- train(Rch_Time ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0, 0.02, len=50)),
                     data = training)
#Highlight the optimized complexity parameter
ggplot(train_rpart, highlight=TRUE)
```




```
#Use this code to see the best cp value
train_rpart$bestTune
```

```
##      cp
## 50 0.02
```

```
#Compute the accuracy of our Classification Tree model on the testing dataset
rpart_accuracy <- confusionMatrix(predict(train_rpart, testing),
                                  testing$Rch_Time)$overall["Accuracy"]
#Save the Classification Tree model accuracy results to our table
accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="rpart", Accuracy = rpart_accuracy))
#View the rpart accuracy results in our table
accuracy_results %>% knitr::kable()
```

method	Accuracy
knn	0.6721591
gbm	0.6795455
rpart	0.6943182

Random Forest Model

For our last model, we will train a random forest model using the randomForest package in r. Random Forests take the average of multiple decision trees in order to improve predictions. Random Forests use the bootstrap to introduce randomness and ensure that individual trees are unique. They sample N observations with replacement from the training set to create a bootstrap training set. The second way that Random Forests introduce randomness is that each tree is built from its own randomly selected subset of features. This random selection process helps reduce the correlation between the trees. Finally, the Random Forest algorithm creates an ensemble by averaging the predictions of all the trees to form a final prediction.

```
#Train a random forest model on our training dataset
set.seed(400,sample.kind = "Rounding")  #if using R3.5 or earlier set.seed(400)
train_rf <- randomForest(Rch_Time ~ ., data = training)
#Compute the accuracy of our random forest model on the testing dataset
rf_accuracy <- confusionMatrix(predict(train_rf, testing),
                               testing$Rch_Time)$overall["Accuracy"]
#Save the random forest accuracy results to our table
accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="random forest", Accuracy = rf_accuracy))
#View the random forest accuracy results in our table
accuracy_results %>% knitr::kable()
```

method	Accuracy
knn	0.6721591
gbm	0.6795455
rpart	0.6943182
random forest	0.6738636

Testing the Final Model on our Validation Set

From our results table, we can see that our Classification Tree Model model achieved the highest accuracy, so we will now test that model on our validation set.

```
#Train our final random forest model on our census_training dataset
set.seed(300, sample.kind = "Rounding") #if using R3.5 or earlier set.seed(300)
final_train_rpart <- train(Rch_Time ~ .,
                      method = "rpart",
                      tuneGrid = data.frame(cp = seq(0, 0.02, len=50)),
                      data = ecomdata_training)
#Compute the accuracy of our final random forest model on the validation set
final_accuracy <- confusionMatrix(predict(final_train_rpart,
                              ecomdata_validation),
                          ecomdata_validation$Rch_Time)$overall["Accuracy"]
#Save the classification tree accuracy results to our table.
accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="Final rpart",
                                     Accuracy = final_accuracy))
#View the final classification tree model accuracy results in our table
accuracy_results %>% knitr::kable()
```

method	Accuracy
knn	0.6721591
gbm	0.6795455
rpart	0.6943182
random forest	0.6738636
Final rpart	0.6960473

(5/5). Results & Conclusion

The goal of this project (as defined at the beginning) was to train a machine learning algorithm using the inputs in one (training) set to predict delivery reaching in time in the other (validation) set. The data set used was from Kaggle Data Repository - https://www.kaggle.com/prachi13/customer-analytics. We first analyzed each variable independently, and then we analyzed 4 different algorithms - knn, gbm, rpart & random forest. We saw rpart (Classification Tree Model) estimated best accuracy.

We can see the same in table below, the classification tree model has the highest overall accuracy of 69.4% on the data we set aside for training and testing purposes. After choosing classification tree as our final model, we tested our predictions on the validation dataset and achieved an overall accuracy of 69.6% for predicting whether the delivery via eCommerce will reach in time or not.

```
#Results
accuracy_results %>% knitr::kable()
```

method	Accuracy
knn	0.6721591
gbm	0.6795455
rpart	0.6943182
random forest	0.6738636
Final rpart	0.6960473

Learnings & Limitations

As I studied this data closely, some of the thoughts I would like to share before closing the report. Data analysis and modeling exercise made it very clear that there is a relationship between variables and if studied effectively can give very valuable insights into consumer shopping behavior. Though at certain points the data looked little distant from reality *(as seen in some of the exploratory analysis - almost near perfect distribution).* Also there could have been a couple of more variables like option for faster delivery and classification of product type as either *'Consumer Durable' or 'Fashion' or 'Home Improvement' or 'Grocery* cause the type of product purchased could have inherent expectation on delivery time.