Predictive Modelling Project

Sanju Hyacinth C

10/08/2019

Table of Contents

# Problem Statement:

Customer Churn is a burning problem for Telecom companies. In this project, we simulate one such case of customer churn where we work on a data of postpaid customers with a contract. The data has information about the customer usage behavior, contract details and the payment details. The data also indicates which were the customers who canceled their service. Based on this past data, we need to build a model which can predict whether a customer will cancel their service in the future or not.

# Problem Objective:

We need to build a working model which can predict if the customer will cancel or retain their service in the future, based on the past data available to us.

# 1. Exploratory Data Analysis:

## 1.1 Installing the required packages:

#install.packages("DataExplorer")  
#install.packages("car")  
#install.packages("dplyr")  
#install.packages("caTools")  
#install.packages("ROCR")  
#install.packages("e1071")  
#install.packages("mlbench")  
#install.packages("lars")  
#install.packages("elasticnet")  
#install.packages("LiblineaR")  
#install.packages("kknn")  
#install.packages("klaR")

## 1.2 Data loading

setwd("D:/R Progms")  
  
celldt = readxl::read\_xlsx("Cellphone.xlsx", sheet = 2)  
  
head(celldt)

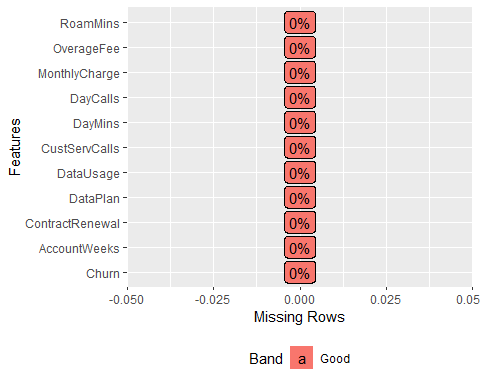
## # A tibble: 6 x 11  
## Churn AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0 128 1 1 2.7 1  
## 2 0 107 1 1 3.7 1  
## 3 0 137 1 0 0 0  
## 4 0 84 0 0 0 2  
## 5 0 75 0 0 0 3  
## 6 0 118 0 0 0 0  
## # ... with 5 more variables: DayMins <dbl>, DayCalls <dbl>,  
## # MonthlyCharge <dbl>, OverageFee <dbl>, RoamMins <dbl>

## 1.3 Checking for Missing values:

library(DataExplorer)

## Warning: package 'DataExplorer' was built under R version 3.6.1

plot\_missing(celldt)



# We find no NA values, hence no removing or imputation of data is required

## 1.4 Data structure:

str(celldt)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 3333 obs. of 11 variables:  
## $ Churn : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ AccountWeeks : num 128 107 137 84 75 118 121 147 117 141 ...  
## $ ContractRenewal: num 1 1 1 0 0 0 1 0 1 0 ...  
## $ DataPlan : num 1 1 0 0 0 0 1 0 0 1 ...  
## $ DataUsage : num 2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...  
## $ CustServCalls : num 1 1 0 2 3 0 3 0 1 0 ...  
## $ DayMins : num 265 162 243 299 167 ...  
## $ DayCalls : num 110 123 114 71 113 98 88 79 97 84 ...  
## $ MonthlyCharge : num 89 82 52 57 41 57 87.3 36 63.9 93.2 ...  
## $ OverageFee : num 9.87 9.78 6.06 3.1 7.42 ...  
## $ RoamMins : num 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

# All of the data types are numeric. We shall later convert the churn, contractrenewal and dataplan to factor variables  
  
summary(celldt)

## Churn AccountWeeks ContractRenewal DataPlan   
## Min. :0.0000 Min. : 1.0 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.: 74.0 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :0.0000 Median :101.0 Median :1.0000 Median :0.0000   
## Mean :0.1449 Mean :101.1 Mean :0.9031 Mean :0.2766   
## 3rd Qu.:0.0000 3rd Qu.:127.0 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :243.0 Max. :1.0000 Max. :1.0000   
## DataUsage CustServCalls DayMins DayCalls   
## Min. :0.0000 Min. :0.000 Min. : 0.0 Min. : 0.0   
## 1st Qu.:0.0000 1st Qu.:1.000 1st Qu.:143.7 1st Qu.: 87.0   
## Median :0.0000 Median :1.000 Median :179.4 Median :101.0   
## Mean :0.8165 Mean :1.563 Mean :179.8 Mean :100.4   
## 3rd Qu.:1.7800 3rd Qu.:2.000 3rd Qu.:216.4 3rd Qu.:114.0   
## Max. :5.4000 Max. :9.000 Max. :350.8 Max. :165.0   
## MonthlyCharge OverageFee RoamMins   
## Min. : 14.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 45.00 1st Qu.: 8.33 1st Qu.: 8.50   
## Median : 53.50 Median :10.07 Median :10.30   
## Mean : 56.31 Mean :10.05 Mean :10.24   
## 3rd Qu.: 66.20 3rd Qu.:11.77 3rd Qu.:12.10   
## Max. :111.30 Max. :18.19 Max. :20.00

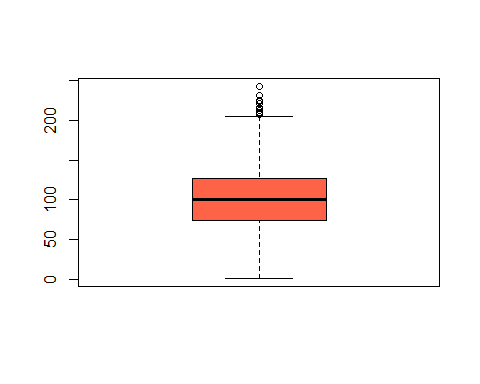
# from the summary,we find that there are no negative or NA values.

## 1.5 Univariate Analysis - Identifying Outliers:

library(ggplot2)  
  
# Excluding Churn, ContractRenewal and DataPlan as their range is between 0 and 1  
  
# 1. Account Weeks:  
  
summary(celldt$AccountWeeks)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.0 74.0 101.0 101.1 127.0 243.0

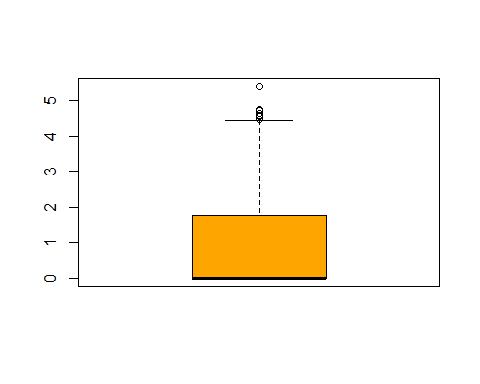
boxplot(celldt$AccountWeeks, data = celldt, col = "tomato")



# Contains outliers  
  
# 2. Data Usage  
  
summary(celldt$DataUsage)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.0000 0.0000 0.8165 1.7800 5.4000

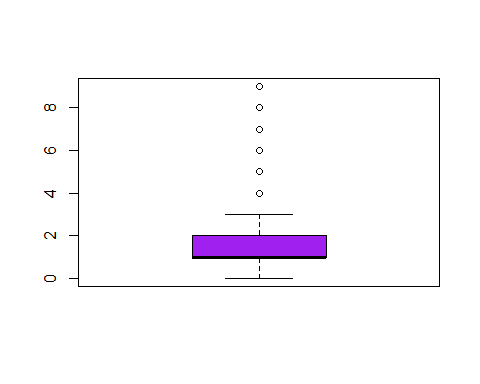
boxplot(celldt$DataUsage, data = celldt, col = "orange")



# both min value and median is 0  
# the variable contains outliers  
  
# 3. Customer Service Calls:  
  
summary(celldt$CustServCalls)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.000 1.000 1.563 2.000 9.000

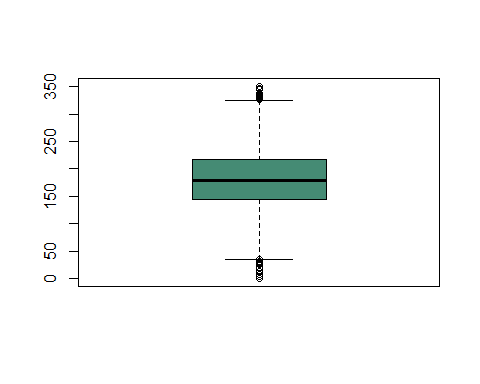
boxplot(celldt$CustServCalls, data = celldt, col = "purple")



# we find that there are outliers as most of them make customer service calls between 0-2  
  
# 4. Day Minutes:  
  
summary(celldt$DayMins)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 143.7 179.4 179.8 216.4 350.8

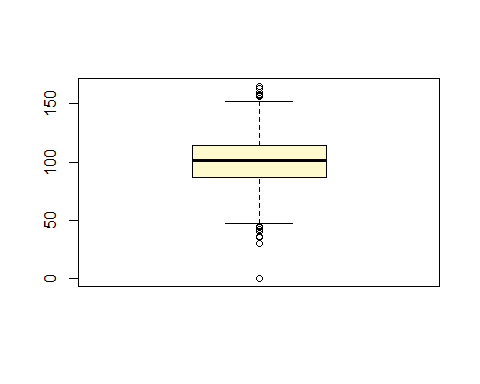
boxplot(celldt$DayMins, data = celldt, col = "aquamarine4")



# the majority range of data is between 140 - 210, and the median is 179.  
# outliers present beyond 300 and below 50  
  
# 5. Day Calls:  
  
summary(celldt$DayCalls)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 87.0 101.0 100.4 114.0 165.0

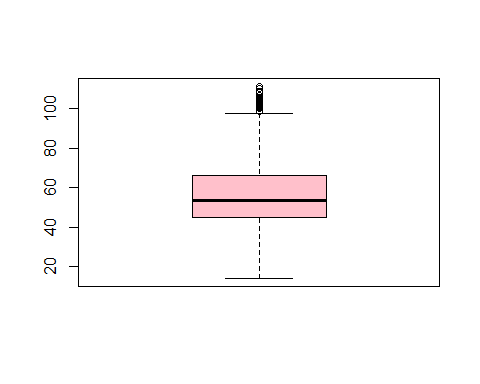
boxplot(celldt$DayCalls, data = celldt, col = "lemonchiffon")



# outliers present  
  
# 6. Monthly Charge:  
  
summary(celldt$MonthlyCharge)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 14.00 45.00 53.50 56.31 66.20 111.30

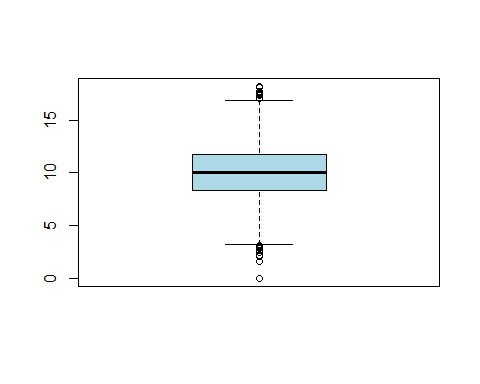
boxplot(celldt$MonthlyCharge, data = celldt, col = "pink")



# outliers are present in the data  
  
# 7. Overage Fee:  
  
summary(celldt$OverageFee)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 8.33 10.07 10.05 11.77 18.19

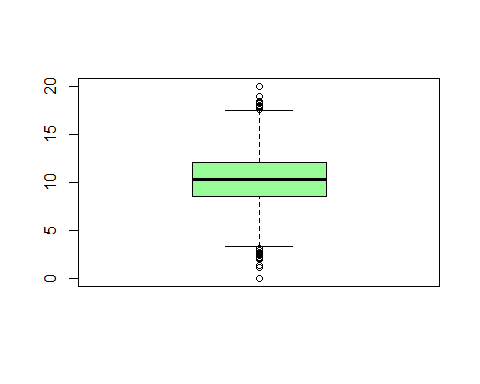
boxplot(celldt$OverageFee, data = celldt, col = "light blue")



# outliers present in the data  
  
# 8. Roam Minutes:  
  
summary(celldt$RoamMins)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 8.50 10.30 10.24 12.10 20.00

boxplot(celldt$RoamMins, data = celldt, col = "pale green")



# outliers present in the data

From the boxplots, we find that all the numeric variables contain outliers (extreme values).

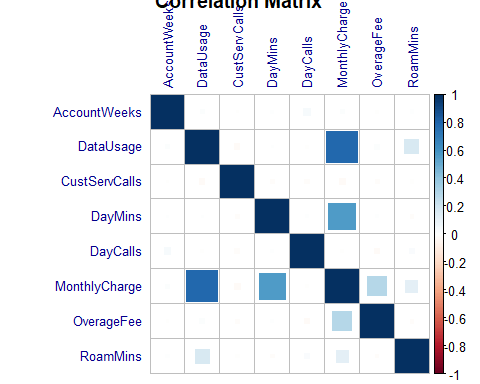
## 1.6 Correlation matrix:

library(corrplot)

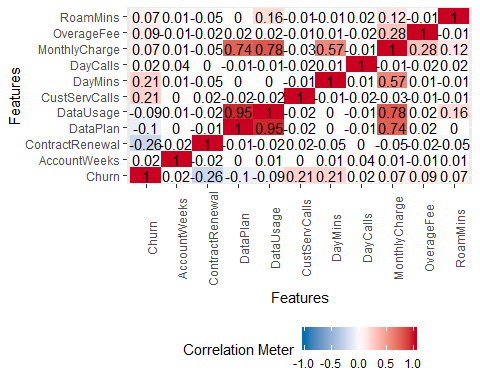
## Warning: package 'corrplot' was built under R version 3.6.1

## corrplot 0.84 loaded

cellcorr = corrplot(cor(celldt[, c(2, 5:11)]), method = "square", title = "Correlation Matrix", tl.cex = 0.8, tl.col = "dark blue")



library(DataExplorer)  
  
plot\_correlation(celldt, type = "c")



### 1.6.1 Inference from Correlation Plot:

From the correlation plot, we find that there is a:

1. Very high correlation between Data Usage and Data Plan
2. High correlation between Data Plan and Monthly Charge, as well as between Data Usage and Monthly Charge
3. A good amount of correlation between Monthly Charge and DayMins

Hence, we can fairly assume that there is a presence of collinearity between a few independent variables.

### 1.6.2 Multi Collinearity Remedy:

A data with multicollinearity can be dealt with using a method called **Variance Inflation Method** by removing the highly correlated variables. The variables having VIF value more than 5 are assumed to be highly correlated. Hence, we will have to do a regression model on the dependent variable with all the independent variables. If all the variables are found to be having a VIF value less than 5, we can go about the problem without the need to remove any variables. But if we find variables having a higher VIF than 5, we will need to remove the variables one by one from the highest VIF. Upon performing regression with the other variables, we will eventually repeat the same steps to arrive at a consistent model with independent variables having VIF less than 5.

# Regression model  
  
library(car)

## Warning: package 'car' was built under R version 3.6.1

## Loading required package: carData

# Running the model on all independent variables:  
mod1 = lm(Churn~. , data = celldt)  
summary(mod1)

##   
## Call:  
## lm(formula = Churn ~ ., data = celldt)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.66572 -0.16629 -0.08236 0.02060 1.08844   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.433e-01 5.363e-02 -2.672 0.007580 \*\*   
## AccountWeeks 8.888e-05 1.396e-04 0.637 0.524402   
## ContractRenewal -2.993e-01 1.882e-02 -15.904 < 2e-16 \*\*\*  
## DataPlan -4.175e-02 4.381e-02 -0.953 0.340650   
## DataUsage -2.835e-02 1.933e-01 -0.147 0.883401   
## CustServCalls 5.829e-02 4.222e-03 13.804 < 2e-16 \*\*\*  
## DayMins 1.021e-03 3.272e-03 0.312 0.754936   
## DayCalls 3.409e-04 2.769e-04 1.231 0.218433   
## MonthlyCharge 1.428e-03 1.924e-02 0.074 0.940838   
## OverageFee 1.046e-02 3.280e-02 0.319 0.749780   
## RoamMins 8.765e-03 2.307e-03 3.800 0.000147 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3203 on 3322 degrees of freedom  
## Multiple R-squared: 0.1747, Adjusted R-squared: 0.1722   
## F-statistic: 70.31 on 10 and 3322 DF, p-value: < 2.2e-16

vif(mod1)

## AccountWeeks ContractRenewal DataPlan DataUsage   
## 1.003791 1.007216 12.473470 1964.800207   
## CustServCalls DayMins DayCalls MonthlyCharge   
## 1.001945 1031.490608 1.002935 3243.300555   
## OverageFee RoamMins   
## 224.639750 1.346583

mod2 = lm(Churn~AccountWeeks+ContractRenewal+DataPlan+DataUsage+CustServCalls+DayMins+DayCalls+OverageFee+RoamMins, data = celldt)  
summary(mod2)

##   
## Call:  
## lm(formula = Churn ~ AccountWeeks + ContractRenewal + DataPlan +   
## DataUsage + CustServCalls + DayMins + DayCalls + OverageFee +   
## RoamMins, data = celldt)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.66565 -0.16649 -0.08222 0.02050 1.08913   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.426e-01 5.280e-02 -2.700 0.006959 \*\*   
## AccountWeeks 8.871e-05 1.396e-04 0.636 0.525107   
## ContractRenewal -2.993e-01 1.881e-02 -15.910 < 2e-16 \*\*\*  
## DataPlan -4.181e-02 4.379e-02 -0.955 0.339837   
## DataUsage -1.405e-02 1.561e-02 -0.900 0.368048   
## CustServCalls 5.828e-02 4.222e-03 13.806 < 2e-16 \*\*\*  
## DayMins 1.264e-03 1.020e-04 12.390 < 2e-16 \*\*\*  
## DayCalls 3.409e-04 2.769e-04 1.231 0.218296   
## OverageFee 1.289e-02 2.190e-03 5.886 4.34e-09 \*\*\*  
## RoamMins 8.763e-03 2.306e-03 3.800 0.000147 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3203 on 3323 degrees of freedom  
## Multiple R-squared: 0.1747, Adjusted R-squared: 0.1725   
## F-statistic: 78.15 on 9 and 3323 DF, p-value: < 2.2e-16

vif(mod2)

## AccountWeeks ContractRenewal DataPlan DataUsage   
## 1.003497 1.006516 12.469560 12.813803   
## CustServCalls DayMins DayCalls OverageFee   
## 1.001778 1.003334 1.002929 1.001657   
## RoamMins   
## 1.346471

mod3 = lm(Churn~AccountWeeks+ContractRenewal+DataPlan+CustServCalls+DayMins+DayCalls+OverageFee+RoamMins, data = celldt)  
summary(mod3)

##   
## Call:  
## lm(formula = Churn ~ AccountWeeks + ContractRenewal + DataPlan +   
## CustServCalls + DayMins + DayCalls + OverageFee + RoamMins,   
## data = celldt)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.66253 -0.16688 -0.08283 0.02049 1.08558   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.325e-01 5.159e-02 -2.568 0.010276 \*   
## AccountWeeks 8.424e-05 1.395e-04 0.604 0.545924   
## ContractRenewal -2.990e-01 1.881e-02 -15.896 < 2e-16 \*\*\*  
## DataPlan -7.962e-02 1.241e-02 -6.417 1.59e-10 \*\*\*  
## CustServCalls 5.832e-02 4.221e-03 13.817 < 2e-16 \*\*\*  
## DayMins 1.262e-03 1.020e-04 12.373 < 2e-16 \*\*\*  
## DayCalls 3.422e-04 2.769e-04 1.236 0.216494   
## OverageFee 1.288e-02 2.190e-03 5.884 4.41e-09 \*\*\*  
## RoamMins 7.714e-03 1.990e-03 3.876 0.000108 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3203 on 3324 degrees of freedom  
## Multiple R-squared: 0.1745, Adjusted R-squared: 0.1725   
## F-statistic: 87.82 on 8 and 3324 DF, p-value: < 2.2e-16

vif(mod3)

## AccountWeeks ContractRenewal DataPlan CustServCalls   
## 1.002227 1.006143 1.000937 1.001659   
## DayMins DayCalls OverageFee RoamMins   
## 1.002862 1.002901 1.001646 1.002987

# AIC (Akaike information criterion) to assess model quality:  
  
AIC(mod1)

## [1] 1882.875

AIC(mod2)

## [1] 1880.881

AIC(mod3)

## [1] 1879.694

### 1.6.3 Model Analysis

Let us analyse the models.

* While performing linear regression mod1, we find that the model explains 17% variability with only 3 variables significant, though there are 10 independent variables. But we find that the overall model is significant with **p value 2.2e-16**. We have then conducted VIF on the model in order to confirm the presence of multicollinearity. We find the independent variable **MonthlyCharge** to have the highest VIF value.
* In performing the second linear regression model (mod2) with the variable MonthlyCharge removed, we arrive at model that now has 5 significant independent variables out of the 9. Upon running the VIF function on this model, we now have values of 1 plus for most variables except for the variables of data plan and usage
* We have thus removed the next variable with the highest vif value (DataUsage) and conduct regression. We noe have a model with 6 significant variables out of the 8 that explain the variability. The Vif of this model provides an image of all the VIFs being about 1.

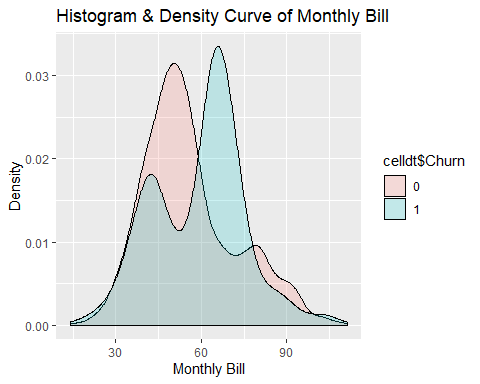
This is a remedy to remove multicollinearity in order to construct stable models

## 1.7 Bivariate Analysis - Plots and Inferences:

# Converting the required fields from numerics to factors   
  
celldt$Churn = factor(celldt$Churn)  
celldt$ContractRenewal = factor(celldt$ContractRenewal)  
celldt$DataPlan = factor(celldt$DataPlan)  
  
str(celldt)

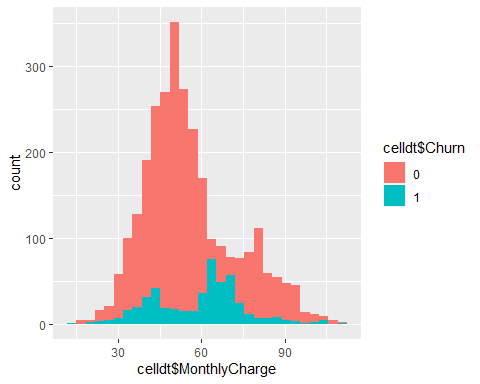
## Classes 'tbl\_df', 'tbl' and 'data.frame': 3333 obs. of 11 variables:  
## $ Churn : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ AccountWeeks : num 128 107 137 84 75 118 121 147 117 141 ...  
## $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 1 ...  
## $ DataPlan : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 1 1 2 ...  
## $ DataUsage : num 2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...  
## $ CustServCalls : num 1 1 0 2 3 0 3 0 1 0 ...  
## $ DayMins : num 265 162 243 299 167 ...  
## $ DayCalls : num 110 123 114 71 113 98 88 79 97 84 ...  
## $ MonthlyCharge : num 89 82 52 57 41 57 87.3 36 63.9 93.2 ...  
## $ OverageFee : num 9.87 9.78 6.06 3.1 7.42 ...  
## $ RoamMins : num 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

library(ggplot2)  
  
den.plot <- ggplot(data=celldt, aes(x=celldt$MonthlyCharge, fill=celldt$Churn))  
den.plot + geom\_density(stat="density", alpha=I(0.2)) + xlab("Monthly Bill") + ylab("Density") + ggtitle("Histogram & Density Curve of Monthly Bill")



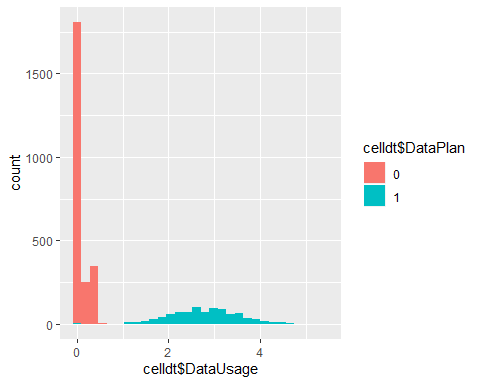
# The above plot shows that people tend to churn out with an increase in Monthly Bill. A higher range of monthly bill (more than 60) shows a higher intensity of churn rate, whilst most people retain to the service in case of lower monthly Charge. There are also instances when churn rate exists in lower monthly charge and retention in the higher monthly charge levels.  
  
p1 = ggplot(data = celldt) + geom\_histogram(mapping = aes(x = celldt$MonthlyCharge, fill = celldt$Churn), position = "identity")  
p1

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

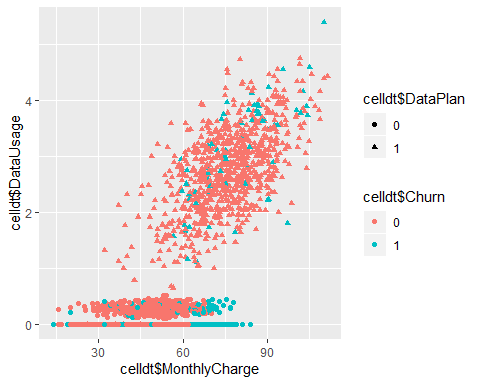


# This plot is another illustration as the above which shows the churn out intensifies a bit as the monthly charge is specially between 60 to 70, after which the churn rate drops down. Although the people who retained the service is the highest upto 60, is comes down a little as the monthly bill increases.  
  
p2 = ggplot(data = celldt) + geom\_histogram(mapping = aes(x = celldt$DataUsage, fill = celldt$DataPlan), position = "identity")  
p2

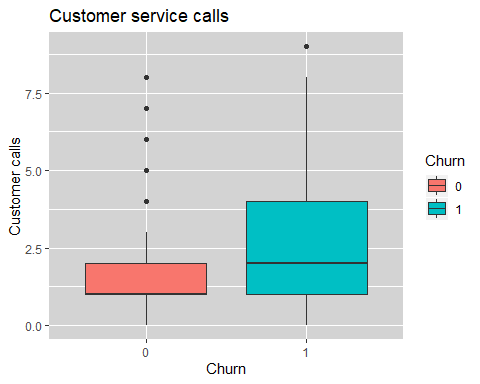
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



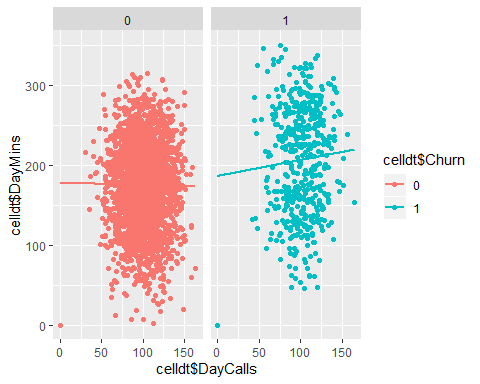
# This plot shows it is mostly people with a data plan highly use the mobile data. We can also see the majority of the people do not have a data plan and use the data, whilst there are just a few who have a dataplan and not use mobile data.  
   
p2.1 = ggplot(data = celldt,aes(y= celldt$DataUsage, x= celldt$MonthlyCharge))+  
 geom\_point(aes(color = celldt$Churn, shape = celldt$DataPlan))   
p2.1



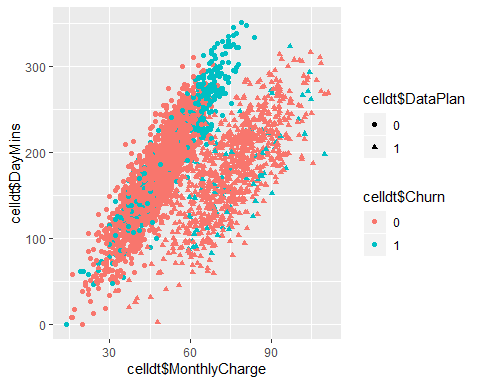
# This plot is interesting as we can observe 4 things in one plot. Most people use the mobile data whilst having a data plan. And a higher percentage of these people do not churn with people only churning out beyond 60 of monthly charge.  
  
# It is interesting to note from the plot that the person with lowest monthly charge without a data plan and the person with almost the highest monthly charge with a data plan have both churned out.  
  
# There are people who have churned out without having a data plan and much data usage, intensifying in the higher range of monthly charge where there are only a few people who retained beyond the range.  
  
p3 = ggplot(data = celldt, aes(x=celldt$Churn, y=celldt$CustServCalls)) + geom\_boxplot(aes(fill = celldt$Churn))  
p31 = p3 + labs(title = "Customer service calls",x="Churn", y="Customer calls",fill = "Churn")  
p32 = p31 + theme(panel.background = element\_rect(fill = "light grey"))  
p32



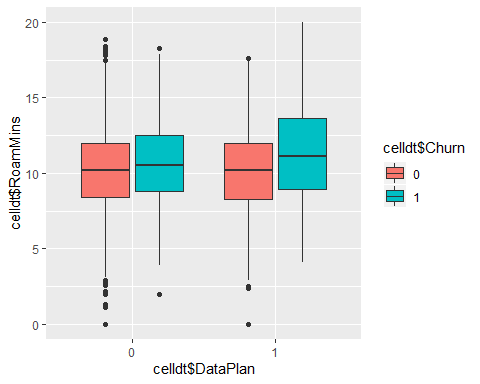
# In this plot we find that the median customer service calls made by the people who retained the service is about 1, while the median customer service calls made by the people who churned is 2. We can also see that there are outliers present for retention as there are people who made more calls than 2 to a maximum of 8 calls have also retained the service. We find that the person who churned out made the most calls of 9.   
  
p4 = ggplot(data = celldt,aes(y= celldt$DayMins, x= celldt$DayCalls, col = celldt$Churn))+ geom\_point() + geom\_smooth(method = "lm",se = F)+ facet\_grid(~celldt$Churn) + facet\_wrap(~celldt$Churn, ncol = 2)  
p4



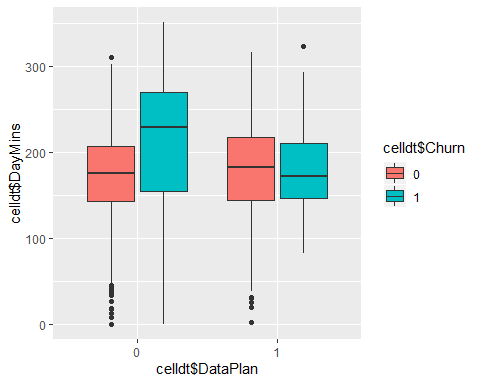
# This scatter plot shows a trend line that shows the increase or decrease of day minutes with respect to day calls, separated by the churn variable. We can see from the plot that the retention rate is more than churn out rate, and that it follows a rather negative trend line (daymins decreases with increase in day calls) But on the other hand, the people who churned out are less in number, and the trend is positive (day mins increases with day calls)  
  
p5 = ggplot(data = celldt,aes(y= celldt$DayMins, x= celldt$MonthlyCharge))+  
 geom\_point(aes(color = celldt$Churn, shape = celldt$DataPlan))  
p5



# The first thing we see from this plot is that, the trend is a positive one (an increase in day minutes increases the monthly charge) It is classified on the basis of data plan and churn rate.Most of them without a data plan and monthly charge less than 60, show retention. Though there are many who show retention with a data plan and charge more than 60. But the intensity of churn out rate of people without data plan is more between 60-80 monthly charge.  
  
# We can safely assume that many people without a data plan have churned out.  
  
p6 = ggplot(data = celldt, aes(x=celldt$DataPlan, y=celldt$RoamMins)) + geom\_boxplot(aes(fill = celldt$Churn))  
p6



# In this boxplot for both the cases where people have a dataplan or not, the roaming minutes is in the same range for the people who have retained the service, although there are extreme values for both on either sides. But the blue areas show a difference, the median has comparatively increased and the churn rate is higher with people having a data plan.  
  
p7 = ggplot(data = celldt, aes(x=celldt$DataPlan, y=celldt$DayMins)) + geom\_boxplot(aes(fill = celldt$Churn))  
p7



# This plot shows in both cases of churning, how has having a data plan or not affected the day call minutes. In both the cases of data plans (0 or 1) here the people have retained the service, the day minutes range seems to be similar with a median of 175-180. But in case of churning out, people without a data plan seem to use more day minutes whereas people with a data plan use lesser day minutes.   
  
# Also people without a data plan have not used much mobile data thereby only making calls, and the people with a dataplan have reduced a bit on their daycalls and used more data. But on both occasions, they have left the service.

xtabs(~ Churn+DataPlan, data = celldt)

## DataPlan  
## Churn 0 1  
## 0 2008 842  
## 1 403 80

xtabs(~ Churn+ContractRenewal, data = celldt)

## ContractRenewal  
## Churn 0 1  
## 0 186 2664  
## 1 137 346

# 2. Model Building:

## 2.1 Logistic Regression:

# binomial tells the model to produce a logistic regression model  
  
logit = glm(Churn~., data = celldt, family = "binomial")  
summary(logit)

##   
## Call:  
## glm(formula = Churn ~ ., family = "binomial", data = celldt)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0058 -0.5112 -0.3477 -0.2093 2.9981   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -5.9510252 0.5486763 -10.846 < 2e-16 \*\*\*  
## AccountWeeks 0.0006525 0.0013873 0.470 0.638112   
## ContractRenewal1 -1.9855172 0.1436107 -13.826 < 2e-16 \*\*\*  
## DataPlan1 -1.1841611 0.5363668 -2.208 0.027262 \*   
## DataUsage 0.3636565 1.9231751 0.189 0.850021   
## CustServCalls 0.5081349 0.0389682 13.040 < 2e-16 \*\*\*  
## DayMins 0.0174407 0.0324841 0.537 0.591337   
## DayCalls 0.0036523 0.0027497 1.328 0.184097   
## MonthlyCharge -0.0275526 0.1909074 -0.144 0.885244   
## OverageFee 0.1868114 0.3256902 0.574 0.566248   
## RoamMins 0.0789226 0.0220522 3.579 0.000345 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2758.3 on 3332 degrees of freedom  
## Residual deviance: 2188.4 on 3322 degrees of freedom  
## AIC: 2210.4  
##   
## Number of Fisher Scoring iterations: 5

# spliting data to train and test  
  
library(caTools)

## Warning: package 'caTools' was built under R version 3.6.1

set.seed(1234)  
spl.cd = sample.split(celldt,SplitRatio = 0.7)  
ctrain = subset(celldt,spl.cd == TRUE)

## Warning: Length of logical index must be 1 or 3333, not 11

ctest = subset(celldt, spl.cd == FALSE)

## Warning: Length of logical index must be 1 or 3333, not 11

# Logistic Model 1  
  
logit1 = glm(Churn~., data = ctrain, family = "binomial")  
summary(logit1)

##   
## Call:  
## glm(formula = Churn ~ ., family = "binomial", data = ctrain)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9972 -0.5166 -0.3470 -0.2115 3.0243   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -6.101874 0.683762 -8.924 <2e-16 \*\*\*  
## AccountWeeks 0.001052 0.001759 0.598 0.5497   
## ContractRenewal1 -1.956987 0.182713 -10.711 <2e-16 \*\*\*  
## DataPlan1 -0.689064 0.662037 -1.041 0.2980   
## DataUsage 0.473587 2.377604 0.199 0.8421   
## CustServCalls 0.503941 0.049383 10.205 <2e-16 \*\*\*  
## DayMins 0.023286 0.040179 0.580 0.5622   
## DayCalls 0.004521 0.003497 1.293 0.1961   
## MonthlyCharge -0.054531 0.236139 -0.231 0.8174   
## OverageFee 0.228351 0.402776 0.567 0.5708   
## RoamMins 0.062001 0.027374 2.265 0.0235 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1743.3 on 2120 degrees of freedom  
## Residual deviance: 1401.5 on 2110 degrees of freedom  
## AIC: 1423.5  
##   
## Number of Fisher Scoring iterations: 5

# From this logistic regression model on the train data, we find that there are 3 significant variables  
  
# Let us now predict on the test data and obtain the confusio matrix  
ctest.pred = predict(logit1, newdata = ctest, type = "response")  
table(ctest$Churn, ctest.pred>0.5)

##   
## FALSE TRUE  
## 0 999 34  
## 1 140 39

# Accuracy  
(999+39)/nrow(ctest)

## [1] 0.8564356

# Let us view the accuracy of the original data   
table(celldt$Churn)

##   
## 0 1   
## 2850 483

prop.table(table(celldt$Churn))

##   
## 0 1   
## 0.8550855 0.1449145

# We find that the accuracy of the original data is almost the same to the test data.  
# We can assume that accuracy is not very important in this instance  
  
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.6.1

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.6.1

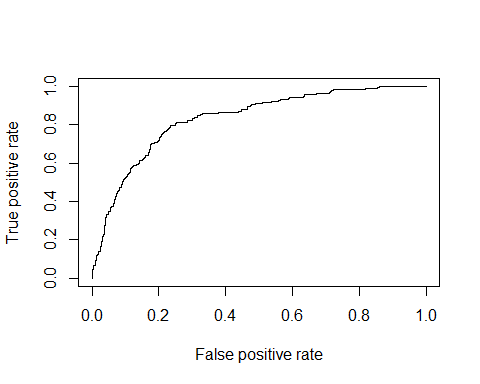
##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

rocr.pred = prediction(ctest.pred, ctest$Churn)  
  
# Area Under the Curve is calculated  
as.numeric(performance(rocr.pred, "auc")@y.values)

## [1] 0.8312503

# 83 percent of the time, the model has given correct predictions  
  
cperf = performance(rocr.pred, "tpr", "fpr")  
plot(cperf)



# false positive rate is very important for this data where we should know that the model has correctly predicted most of the time. Here in this instance we can see the model has wrongly predicted just 3 percent of the time, with correct predictions 97 percent of the time.  
  
34/(999+34)

## [1] 0.03291384

## 2.2 K Nearest Neighbour Algorithm:

KNN, as it is popularly called, is a Supervised Machine Learning algorithm that helps to classify a data point to it’s assigned variable or target class, by analysing the nearest neighbours of the data point. Hence the name, nearest neighbour. And K is the value we assign to the model in order for the algorithm to focus on the desired K number of neighbours for analysis. KNN works only on numerical variables as it involves calculation of distances between each datapoint of the numeric variable. Mostly, the assigned K value is **odd** to avoid any ties between the selected data point.

Let us go about the algorithm

### 2.2.1 Data Exploring:

# Load the dataset, read and understand it's structure:  
  
datac = readxl::read\_xlsx("Cellphone.xlsx", sheet = 2)  
  
str(datac)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 3333 obs. of 11 variables:  
## $ Churn : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ AccountWeeks : num 128 107 137 84 75 118 121 147 117 141 ...  
## $ ContractRenewal: num 1 1 1 0 0 0 1 0 1 0 ...  
## $ DataPlan : num 1 1 0 0 0 0 1 0 0 1 ...  
## $ DataUsage : num 2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...  
## $ CustServCalls : num 1 1 0 2 3 0 3 0 1 0 ...  
## $ DayMins : num 265 162 243 299 167 ...  
## $ DayCalls : num 110 123 114 71 113 98 88 79 97 84 ...  
## $ MonthlyCharge : num 89 82 52 57 41 57 87.3 36 63.9 93.2 ...  
## $ OverageFee : num 9.87 9.78 6.06 3.1 7.42 ...  
## $ RoamMins : num 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

summary(datac)

## Churn AccountWeeks ContractRenewal DataPlan   
## Min. :0.0000 Min. : 1.0 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.: 74.0 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :0.0000 Median :101.0 Median :1.0000 Median :0.0000   
## Mean :0.1449 Mean :101.1 Mean :0.9031 Mean :0.2766   
## 3rd Qu.:0.0000 3rd Qu.:127.0 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :243.0 Max. :1.0000 Max. :1.0000   
## DataUsage CustServCalls DayMins DayCalls   
## Min. :0.0000 Min. :0.000 Min. : 0.0 Min. : 0.0   
## 1st Qu.:0.0000 1st Qu.:1.000 1st Qu.:143.7 1st Qu.: 87.0   
## Median :0.0000 Median :1.000 Median :179.4 Median :101.0   
## Mean :0.8165 Mean :1.563 Mean :179.8 Mean :100.4   
## 3rd Qu.:1.7800 3rd Qu.:2.000 3rd Qu.:216.4 3rd Qu.:114.0   
## Max. :5.4000 Max. :9.000 Max. :350.8 Max. :165.0   
## MonthlyCharge OverageFee RoamMins   
## Min. : 14.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 45.00 1st Qu.: 8.33 1st Qu.: 8.50   
## Median : 53.50 Median :10.07 Median :10.30   
## Mean : 56.31 Mean :10.05 Mean :10.24   
## 3rd Qu.: 66.20 3rd Qu.:11.77 3rd Qu.:12.10   
## Max. :111.30 Max. :18.19 Max. :20.00

# We should make sure to remove any na or missing values from the data.  
# In our case, there are no missing values present.

### 2.2.2 Normalising the Data:

In order to develop a proper algorithm, KNN uses normalised data. In a data, we may have datapoints that are not same in their measurements. In order to bring these datapoints to a standard measurement, we do the **normalisation of datapoints**. Upon doing this, we can find all the datapoints ranging between 0 and 1 thus facilitating the development of a good and unbiased model.

Let us normalise the data with the below fundction

# Normalizing the data  
  
norm = function(x) { (x- min(x))/(max(x) - min(x)) }  
  
# eliminating the dependent variable  
norm.data = as.data.frame(lapply(datac[,-1], norm))  
  
# Bring back the dependent variable to the normalised data:  
usable.data = cbind(datac[,1], norm.data)  
  
str(usable.data)

## 'data.frame': 3333 obs. of 11 variables:  
## $ Churn : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ AccountWeeks : num 0.525 0.438 0.562 0.343 0.306 ...  
## $ ContractRenewal: num 1 1 1 0 0 0 1 0 1 0 ...  
## $ DataPlan : num 1 1 0 0 0 0 1 0 0 1 ...  
## $ DataUsage : num 0.5 0.685 0 0 0 ...  
## $ CustServCalls : num 0.111 0.111 0 0.222 0.333 ...  
## $ DayMins : num 0.756 0.461 0.694 0.853 0.475 ...  
## $ DayCalls : num 0.667 0.745 0.691 0.43 0.685 ...  
## $ MonthlyCharge : num 0.771 0.699 0.391 0.442 0.277 ...  
## $ OverageFee : num 0.543 0.538 0.333 0.17 0.408 ...  
## $ RoamMins : num 0.5 0.685 0.61 0.33 0.505 0.315 0.375 0.355 0.435 0.56 ...

View(usable.data)

### 2.2.3 KNN Classification:

Let us now partition the data into training and testing data, after which we will run the KNN algorithm. Here is where we will select the correct number for K. One thing to keep in mind is an **optimum number for K is what that will produce a good model**. A smaller k will result in a rather overfitting model and a larger k may not capture as much details from the data, thereby missing on important ones.

By default, K can take a value of 19 or the square root of the number of observations in the data. But we have to make sure to select the optimum number so as to avoid over or under fitting of the model.

# Data partitioning  
  
library(caTools)  
  
set.seed(2000)  
spl = sample.split(usable.data$Churn, SplitRatio = 0.7)  
train = subset(usable.data, spl == T)  
test = subset(usable.data, spl == F)  
  
# KNN Classifier:  
table(sqrt(3333), sqrt(2333))

##   
## 48.3011387029333  
## 57.7321400954442 1

# taking k=19 first as the sqrt of the dataset is about 58.  
  
library(class)  
  
pred1 = knn(train[-1], test[-1], train[,1], k = 19)   
table.knn1 = table(test[,1], pred1)  
sum(diag(table.knn1)/sum(table.knn1))

## [1] 0.904

# Accuracy of 89.1 keeping k as 19  
  
# Assigning the sqrt values of 57 and 58  
  
pred2 = knn(train[-1], test[-1], train[,1], k = 58)  
table.knn2 = table(test[,1], pred2)  
sum(diag(table.knn2)/sum(table.knn2))

## [1] 0.878

# Accuracy of 87.8  
  
  
pred = knn(train[-1], test[-1], train[,1], k = 57)  
table.knn = table(test[,1], pred)  
sum(diag(table.knn)/sum(table.knn))

## [1] 0.878

# Accuracy of 88  
  
pred3 = knn(train[-1], test[-1], train[,1], k = 56)  
table.knn3 = table(test[,1], pred3)  
sum(diag(table.knn3)/sum(table.knn3))

## [1] 0.877

# Accuracy of 88.1  
  
pred4 = knn(train[-1], test[-1], train[,1], k = 55)  
table.knn4 = table(test[,1], pred4)  
sum(diag(table.knn4)/sum(table.knn4))

## [1] 0.878

# Accuracy of 88  
  
# The difference in accuracy from 19 to 57 is not a lot. But between 57 and 58, the accuracy has dropped by .2 points. As per understanding, the accuracy from 57 to 56 has increased. But it seemed to have come down at K = 55. The optimum number of k here may be assumed to be 57 or 56

## 2.3 Naïve Bayes:

Naïve Bayes, an extremely powerful tool, is technically used only when the data set has all of its variables in a categorical format. However, it can also be used with continuous features but is more suited to categorical variables. Naïve Bayes is a recommended algorithm if all the independent and dependent variables are categorical. Naïve Bayes is a parametric algorithm. Hence, we may not obtain different results upon re executing the function unless the data remains unchanged.

In our case, although our dependent variable is categorical (as in the case of any data with a problem statement), most of our dependent variables are of numeric format.

# e1071 has the naiveBayes function   
library(e1071)

## Warning: package 'e1071' was built under R version 3.6.1

# training the model on the train dataset  
NB = naiveBayes(Churn~., data = train)  
print(NB)

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.8551222 0.1448778   
##   
## Conditional probabilities:  
## AccountWeeks  
## Y [,1] [,2]  
## 0 0.4171607 0.1662111  
## 1 0.4180400 0.1632241  
##   
## ContractRenewal  
## Y [,1] [,2]  
## 0 0.9303258 0.2546610  
## 1 0.7278107 0.4457467  
##   
## DataPlan  
## Y [,1] [,2]  
## 0 0.2897243 0.4537481  
## 1 0.1656805 0.3723448  
##   
## DataUsage  
## Y [,1] [,2]  
## 0 0.1569489 0.2374961  
## 1 0.0986796 0.2076403  
##   
## CustServCalls  
## Y [,1] [,2]  
## 0 0.1595656 0.1274755  
## 1 0.2416174 0.2002582  
##   
## DayMins  
## Y [,1] [,2]  
## 0 0.4981119 0.1425293  
## 1 0.5895822 0.1987181  
##   
## DayCalls  
## Y [,1] [,2]  
## 0 0.6072606 0.1225323  
## 1 0.6126771 0.1345831  
##   
## MonthlyCharge  
## Y [,1] [,2]  
## 0 0.4275030 0.1678857  
## 1 0.4603921 0.1624915  
##   
## OverageFee  
## Y [,1] [,2]  
## 0 0.5472717 0.1381949  
## 1 0.5766759 0.1366146  
##   
## RoamMins  
## Y [,1] [,2]  
## 0 0.5065890 0.1375518  
## 1 0.5353846 0.1377075

# Changing the numeric fields to factors:  
train$Churn = factor(train$Churn)  
train$ContractRenewal = factor(train$ContractRenewal)  
train$DataPlan = factor(train$DataPlan)  
test$Churn = factor(test$Churn)  
test$ContractRenewal = factor(test$ContractRenewal)  
test$DataPlan = factor(test$DataPlan)  
  
# Creating naiveBayes model only on categorical variables  
NB1 = naiveBayes(Churn~ContractRenewal+DataPlan, data = train)  
print(NB1)

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1   
## 0.8551222 0.1448778   
##   
## Conditional probabilities:  
## ContractRenewal  
## Y 0 1  
## 0 0.06967419 0.93032581  
## 1 0.27218935 0.72781065  
##   
## DataPlan  
## Y 0 1  
## 0 0.7102757 0.2897243  
## 1 0.8343195 0.1656805

# A priori probabilities and Conditional probabilities   
# Conditional probabilities for the categorical variables are given correctly in the summary of the model  
# Even from this model we can justify that many people who have cancelled the service have been without a data plan. Also most of them without a data plan have not churned out

## 2.4 Model Comparison Using Model Performance Metrics - Regression

### 2.4.1 Model Performance Metrics - Regression:

# loading the fresh data:  
celldt = readxl::read\_xlsx("Cellphone.xlsx", sheet = 2)  
  
str(celldt)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 3333 obs. of 11 variables:  
## $ Churn : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ AccountWeeks : num 128 107 137 84 75 118 121 147 117 141 ...  
## $ ContractRenewal: num 1 1 1 0 0 0 1 0 1 0 ...  
## $ DataPlan : num 1 1 0 0 0 0 1 0 0 1 ...  
## $ DataUsage : num 2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...  
## $ CustServCalls : num 1 1 0 2 3 0 3 0 1 0 ...  
## $ DayMins : num 265 162 243 299 167 ...  
## $ DayCalls : num 110 123 114 71 113 98 88 79 97 84 ...  
## $ MonthlyCharge : num 89 82 52 57 41 57 87.3 36 63.9 93.2 ...  
## $ OverageFee : num 9.87 9.78 6.06 3.1 7.42 ...  
## $ RoamMins : num 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

library(mlbench)

## Warning: package 'mlbench' was built under R version 3.6.1

library(caret)

## Warning: package 'caret' was built under R version 3.6.1

## Loading required package: lattice

library(lars)

## Loaded lars 1.2

library(elasticnet)  
  
# CROSS VALIDATION: technique used is K Fold cross validation   
# prepare training scheme  
  
control <- trainControl(method = "repeatedcv", number = 10, repeats = 2)

# train the Linear Regression model  
  
set.seed(2000)  
celldt.reg = celldt[,c(-3,-4)]  
model.ols <- train(Churn ~., data = celldt.reg, method = "lm", trControl = control)

## Warning in train.default(x, y, weights = w, ...): You are trying to do  
## regression and your outcome only has two possible values Are you trying to  
## do classification? If so, use a 2 level factor as your outcome column.

# train the Lasso model  
  
set.seed(2000)  
  
model.Lasso <- train(Churn ~., data = celldt.reg, method = "lasso", trControl = control)

## Warning in train.default(x, y, weights = w, ...): You are trying to do  
## regression and your outcome only has two possible values Are you trying to  
## do classification? If so, use a 2 level factor as your outcome column.

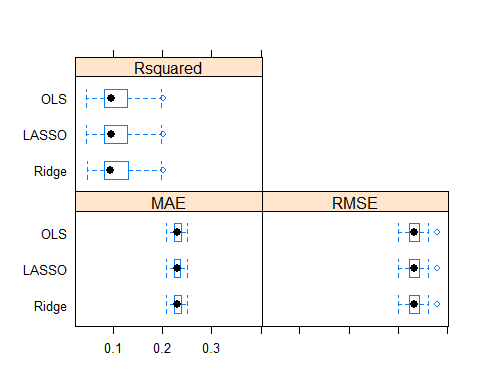
# train the ridge model:  
  
set.seed(2000)  
  
model.Ridge <- train(Churn ~., data = celldt.reg, method = "ridge", trControl = control)

## Warning in train.default(x, y, weights = w, ...): You are trying to do  
## regression and your outcome only has two possible values Are you trying to  
## do classification? If so, use a 2 level factor as your outcome column.

# collect resamples  
  
results <- resamples(list(OLS = model.ols, LASSO = model.Lasso, Ridge = model.Ridge))  
  
# summarizing the distributions  
  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: OLS, LASSO, Ridge   
## Number of resamples: 20   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## OLS 0.2089019 0.2251575 0.2304035 0.2309001 0.2371858 0.2501352 0  
## LASSO 0.2088687 0.2250991 0.2303677 0.2308261 0.2370034 0.2501223 0  
## Ridge 0.2086390 0.2252148 0.2300344 0.2307734 0.2372263 0.2499894 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## OLS 0.2992415 0.3228164 0.3331185 0.3327399 0.3410310 0.3787526 0  
## LASSO 0.2991953 0.3234839 0.3331018 0.3327667 0.3409652 0.3786895 0  
## Ridge 0.2988873 0.3227941 0.3329981 0.3326385 0.3404879 0.3785809 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## OLS 0.04631406 0.08269700 0.09598503 0.1094379 0.1271957 0.2021192 0  
## LASSO 0.04657575 0.08285620 0.09602734 0.1096325 0.1272693 0.2025405 0  
## Ridge 0.04866260 0.08335177 0.09578556 0.1098541 0.1277072 0.2029407 0

# OLS has the least median Root Mean Square Error (RMSE) of 0.329 when compared to the models of LASSO and RIDGE  
# And it also has the highest median and mean R Squared of 0.1037 and 0.1082  
# We find that the other two models are close to each other and quite far apart from OLS.  
  
# Boxplot results:  
  
bwplot(results)



### 2.4.2 Model Performance Metrics - Interpretation:

* OLS has the least median Root Mean Square Error (RMSE) of 0.329 when compared to the models of LASSO and RIDGE
* And it also has the highest median and mean R Squared of 0.1037 and 0.1082
* We find that the other two models are close to each other and quite far apart from OLS