

# Mobile Phones Selling Price Report

Final report (MDSA Winter 2023)

Yongpeng Fu (10182778), Rudy Brown (10057171), Jose Palacios (30190988),  
Stuart Finley(30191070), Andrii Voitkiv(30199373)

2023-02-17

## Contents

<b>Introduction</b>	<b>2</b>
<b>Dataset and Cleanup</b>	<b>2</b>
<b>Scope of Analysis</b>	<b>5</b>
<b>Chapter 4: Exploratory Data Analysis</b>	<b>6</b>
4.1: A summary of the dataset . . . . .	6
4.2: Exploratory Data Analysis . . . . .	6
<b>Chapter 5: Statistical Data Analysis</b>	<b>14</b>
5.1: Prediction of Mobile Phone Price Level . . . . .	14
5.2: Cross-Validation . . . . .	23
5.3: Summary for Prediction of Mobile Phone Price Level . . . . .	24
<b>References</b>	<b>25</b>

## Introduction

Mobile phones are everywhere, so are the prices. Despite still having the word “phone” in the name, a typical modern smartphone has much more features than just to make and receive calls. They are boasting a staggering range of applications like brand, memory, storage, camera, resolution, just to name a few. All these cutting edge technology and features packed in one little device does not come without a cost. A [2020 review](#) of premium mobile phones shows a staggering 490% rise in the last two decades.

With so many mobile phones on the market, it can be difficult to decide which one you want to buy. As a customer, we are particularly interested in finding some relation between all these features and its selling price. To this purpose, we collected the [MobilePhone's dataset](#) from Kaggle and apply a set of statistical analysis hoping to answer some guiding questions:

1. Can we estimate the average price for mobile phones?
2. What is the impact of each mobile phone's feature on the selling price?
3. Can a classification model to distinguish the selling price range?
4. Can we build a decent model to predict the selling price for a mobile phone?

## Dataset and Cleanup

The initial dataset consists of 8 columns and 28,036 rows and no missing values. These 8 columns are:

- **Model**: categorical variables with sub-classes. These names include the color of the unit and its storage capacity. The latter being also listed as a separate column. - **Independent Variable**
- **Company**: categorical variable. Name of the phone's manufacturer. - **Independent Variable**
- **Price**: continuous variable. Units in Indian Rupees. - **Dependent Variable**
- **Rating**: continuous variable. - **Independent Variable**
- **Number of ratings**: discrete variable: a simple count. - **Independent Variable**
- **Total reviews**: discrete variable: a simple count. - **Independent Variable**
- **RAM size**: categorical variable. RAM specification of the phone. - **Independent Variable**
- **ROM size**: categorical variable. Storage (non-volatile memory) capacity of the phone. - **Independent Variable**

Some initial steps can be completed to clean the dataset and create new variables which can be used in our analysis. The initial steps for cleaning the dataset are as follows:

1. Remove any duplicates in the dataset;
2. Because **Model** column contains sub-class of a mobile phone, we decide to further break it down to *Model* and *Color*;
3. Convert all units from **RAM size** and **ROM size** measure to GB and then remove unit suffix;
4. Add additional column to segment the **Price** into 4 different levels;
5. Add additional column to determine if a phone has %G feature or not based on **Model** information.

```
mobile_dataset <- read.csv("./Updated_Mobile_Dataset.csv")
#Step 0: convert Model and Company letter to uppercase.
mobile_dataset$Model <- toupper(mobile_dataset$Model)
mobile_dataset$Company <- toupper(mobile_dataset$Company)
#Break down Model column into Model only and Color columns
#Step 1: remove the Company name from Model
Model_no_Company <- stringr::str_remove(mobile_dataset$Model,mobile_dataset$Company)
↪ %>% trimws(., which = c("both"))
#Step 2: remove anything after parentheses to get Model only info
mobile_dataset$Model_Only <- stringr::str_replace(Model_no_Company, "
↪ \\s*\\([^\\)]+\\)", "")
```

```

#Step 3: Get the color information from inside the last ()
#Step 3.1: Get the parenthesis and what is inside from the last ()
Model_no_Company_parenthesis <- stringr::str_extract(Model_no_Company,
  ↳ "(?<=\\()(\\[\\^()]*?)(?=\\)[\\^()]*$)")
#step 3.2: Get the color by just retaining the info before ,
mobile_dataset$Color <- gsub(",.*$", "", Model_no_Company_parenthesis)
#Step 4: Remove duplicated rows in the dataset
mobile_dataset <- mobile_dataset[!duplicated(mobile_dataset),]
#Step 5: cut the price based on the percentile into 4 different levels
mobile_dataset <- mobile_dataset %>% mutate(Price_Level = ntile(Price, n = 4))
#Step 5.1: map each number level to the character
from <- c(1,2,3,4)
to <- c("Low", "Medium", "High", "Very High")
mobile_dataset$Price_Level <- mapvalues(mobile_dataset$Price_Level, from = from, to
  ↳ = to)

#Step 6: Get the numeric part of RomSize (remove GB and MB, but convert MB to GB),
  ↳ discard any record that no numeric in RomSize
#Step 6.1: there are some data input errors for RamSize and RomSize. In the records
  ↳ where RomSize is "Not Known" are swapped with RamSize, so we need to correct
  ↳ that.
RamSize_temp <- ifelse(mobile_dataset$RomSize == "Not Known", "0 GB",
  ↳ mobile_dataset$RamSize)
mobile_dataset$RomSize <- ifelse(mobile_dataset$RomSize == "Not Known",
  ↳ mobile_dataset$RamSize, mobile_dataset$RomSize)
mobile_dataset$RamSize <- RamSize_temp
#Step 6.2: split RomSize into two columns with size number and unit, and convert MB
  ↳ to 1/1000GB, KB to 1/1000000GB
mobile_dataset$RamSize_Ori <- mobile_dataset$RamSize
mobile_dataset$RomSize_Ori <- mobile_dataset$RomSize
mobile_dataset <- mobile_dataset %>% separate(RomSize, c("RomSize_num",
  ↳ "RomSize_Unit")) %>% mutate(RomSize_Unit= mapvalues(.$RomSize_Unit, from =
  ↳ c("GB", "MB", "KB"), to = c(1, 1/1000, 1/1000000)))

## Warning: Expected 2 pieces. Additional pieces discarded in 1 rows [573].

#step 6.3: remove any rows that are not numeric value for RomSize
mobile_dataset <- mobile_dataset[!is.na(as.numeric(mobile_dataset$RomSize_num)),]

## Warning in `[.data.frame`(mobile_dataset, !
## is.na(as.numeric(mobile_dataset$RomSize_num)), : NAs introduced by coercion

mobile_dataset$RomSize_num <- as.numeric(mobile_dataset$RomSize_num)
mobile_dataset$RomSize_Unit <-
  ↳ ifelse(is.na(as.numeric(mobile_dataset$RomSize_Unit)), 0,
  ↳ as.numeric(mobile_dataset$RomSize_Unit))
#Step 6.4: generate the final column RomSize_inGB
mobile_dataset$RomSize_inGB <- mobile_dataset$RomSize_num *
  ↳ mobile_dataset$RomSize_Unit

```

```

#Step 7: Get the numeric part of RamSize (remove GB and MB, but convert MB to GB),
↳ discard any record that no numeric in RamSize
#Step 7.1: split RamSize into two columns with size number and unit, and convert MB
↳ to 1/1000GB
mobile_dataset <- mobile_dataset %>% separate(RamSize, c("RamSize_num",
↳ "RamSize_Unit")) %>% mutate(RamSize_Unit= mapvalues(.$RamSize_Unit, from =
↳ c("GB", "MB"), to = c(1, 1/1000)))
#step 7.2: remove any rows that are not numeric value for RamSize
mobile_dataset <- mobile_dataset[!is.na(as.numeric(mobile_dataset$RamSize_num)),]
mobile_dataset$RamSize_num<- as.numeric(mobile_dataset$RamSize_num)
mobile_dataset$RamSize_Unit <- as.numeric(mobile_dataset$RamSize_Unit)
#Step 7.3: generate the final column RamSize_inGB
mobile_dataset$RamSize_inGB <- mobile_dataset$RamSize_num *
↳ mobile_dataset$RamSize_Unit

#Step 8: Create a new column to determine if the phone is 5G or not
mobile_dataset$Is_5G <- ifelse(str_detect(mobile_dataset$Model_Only, "5G"), "Yes",
↳ "No")
#Step 9: only keep the columns we need
column_names <- c("Model", "Company", "Price", "Rating", "No_of_ratings",
↳ "TotalReviews", "Model_Only", "Color", "Price_Level", "RamSize_inGB",
↳ "RomSize_inGB", "RamSize_Ori", "RomSize_Ori", "Is_5G" )
mobile_dataset <- mobile_dataset[column_names]

# #Step 9: final check up. Convert all company name to uppercase and then do a final
↳ duplicated removal
# mobile_dataset$Company <- toupper(mobile_dataset$Company)
# mobile_dataset <- mobile_dataset[!duplicated(mobile_dataset),]
write.csv(mobile_dataset, file = './Cleaned_Mobile_Dataset.csv', row.names = F)

```

After cleaning and breaking down columns, the dataset now consists of 14 columns and 725 rows and no missing values. These 14 columns are:

- **Model:** categorical variables with sub-classes. These names include the color of the unit and its storage capacity. The latter being also listed as a separate column. - **Independent Variable**
- **Company:** categorical variable. Name of the phone's manufacturer. - **Independent Variable**
- **Price:** continuous variable. Units in Indian Rupees. - **Dependent Variable**
- **Rating:** continuous variable. - **Independent Variable**
- **Number of ratings:** discrete variable: a simple count. - **Independent Variable**
- **Total reviews:** discrete variable: a simple count. - **Independent Variable**
- **Model\_Only:** categorical variable: only contains the model information of a mobile phone. - **Independent Variable**
- **Color:** categorical variable: color of a mobile phone. - **Independent Variable**
- **Price\_Level:** The price level of a mobile phone, with levels of "Low", "Medium", "High", "Very High". - **Independent Variable**
- **RamSize\_inGB:** continuous variable. RAM specification of the phone in GB. - **Independent Variable**
- **RomSize\_inGB:** continuous variable. Storage (non-volatile memory) capacity of the phone in GB. - **Independent Variable**
- **RamSize\_Ori:** categorical variable. RAM specification of the phone, original information. - **Independent Variable**

- **RomSize\_Ori**: categorical variable. Storage (non-volatile memory) capacity of the phone, original information. - **Independent Variable**
- **Is\_5G**: categorical variable. If the phone has 5G service or not. - **Independent Variable**

```
mobile_dataset <- as_tibble(read.csv("./Cleaned_Mobile_Dataset.csv"))
#Specify the level for Price_Level column
mobile_dataset$Price_Level <- factor(mobile_dataset$Price_Level, levels = c("Low",
  ↪ "Medium", "High", "Very High"))
mobile_dataset %>% head(4)
```

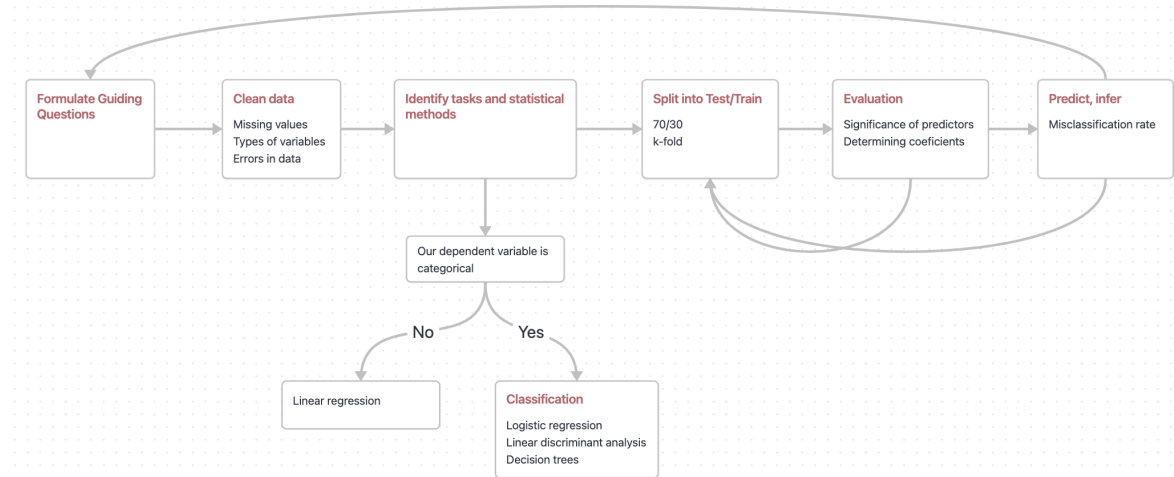
```
## # A tibble: 4 x 14
##   Model      Company Price Rating No_of_ratings TotalReviews Model_Only Color
##   <chr>      <chr>   <int> <dbl>      <int>      <int> <chr>   <chr>
## 1 INFINIX HOT~ INFINIX  8199   4.3         505         52 HOT 20 PL~ LUNA ~
## 2 MOTOROLA E4~ MOTORO~  7999   4.1        56085        5600 E40   CARBO~
## 3 MOTOROLA E4~ MOTORO~  7999   4.1        56085        5600 E40   PINK ~
## 4 POCO C31 (S~ POCO    7499   4.3       183688       11185 C31   SHADO~
## # ... with 6 more variables: Price_Level <fct>, RamSize_inGB <dbl>,
## #   RomSize_inGB <dbl>, RamSize_Ori <chr>, RomSize_Ori <chr>, Is_5G <chr>
```

**Table 1:** The cleaned-up dataset for Mobile Phone

The dataset and detailed analysis can be found at this [repository](#).

## Scope of Analysis

Our team is finalizing what the full analysis of the dataset will look like, but a preliminary template and breakdown of work by team member has been included below. The different colors represent which components of the project different team members would take on. It is anticipated that all members will assist in the finalization of the report.



## Chapter 4: Exploratory Data Analysis

### 4.1: A summary of the dataset

The dimension of the cleaned dataset is 726 rows and 14 columns. A summary of the data is as follows:

```
mobile_dataset <- as_tibble(read.csv("./Cleaned_Mobile_Dataset.csv"))
#Specify the level for Price_Level column
mobile_dataset$Price_Level <- factor(mobile_dataset$Price_Level, levels = c("Low",
  ↪ "Medium", "High", "Very High"))
mobile_dataset %>% summary()
```

```
##      Model          Company      Price      Rating
## Length:725      Length:725      Min.   :   698      Min.   :2.700
## Class :character Class :character 1st Qu.:  7014      1st Qu.:4.100
## Mode  :character Mode  :character Median  : 12889      Median :4.200
##                                     Mean   : 15903      Mean   :4.227
##                                     3rd Qu.: 16999      3rd Qu.:4.300
##                                     Max.   :149900     Max.   :4.800
## No_of_ratings    TotalReviews    Model_Only      Color
## Min.   :      3      Min.   :      0      Length:725      Length:725
## 1st Qu.:   874      1st Qu.:    80      Class :character Class :character
## Median :  7325      Median :   670      Mode  :character Mode  :character
## Mean   : 34522      Mean   : 2496
## 3rd Qu.: 37211      3rd Qu.: 2972
## Max.   :575907      Max.   :33942
## Price_Level      RamSize_inGB      RomSize_inGB      RamSize_Ori
## Low       :171      Min.   : 0.000      Min.   :  0.00      Length:725
## Medium    :185      1st Qu.: 0.064      1st Qu.: 32.00      Class :character
## High      :185      Median : 4.000      Median : 64.00      Mode  :character
## Very High:184      Mean   : 3.882      Mean   : 81.78
##                                     3rd Qu.: 6.000      3rd Qu.:128.00
##                                     Max.   :12.000      Max.   :512.00
## RomSize_Ori      Is_5G
## Length:725      Length:725
## Class :character Class :character
## Mode  :character Mode  :character
##
##
##
```

**Table 2:** A summary for Mobile Phone dataset

### 4.2: Exploratory Data Analysis

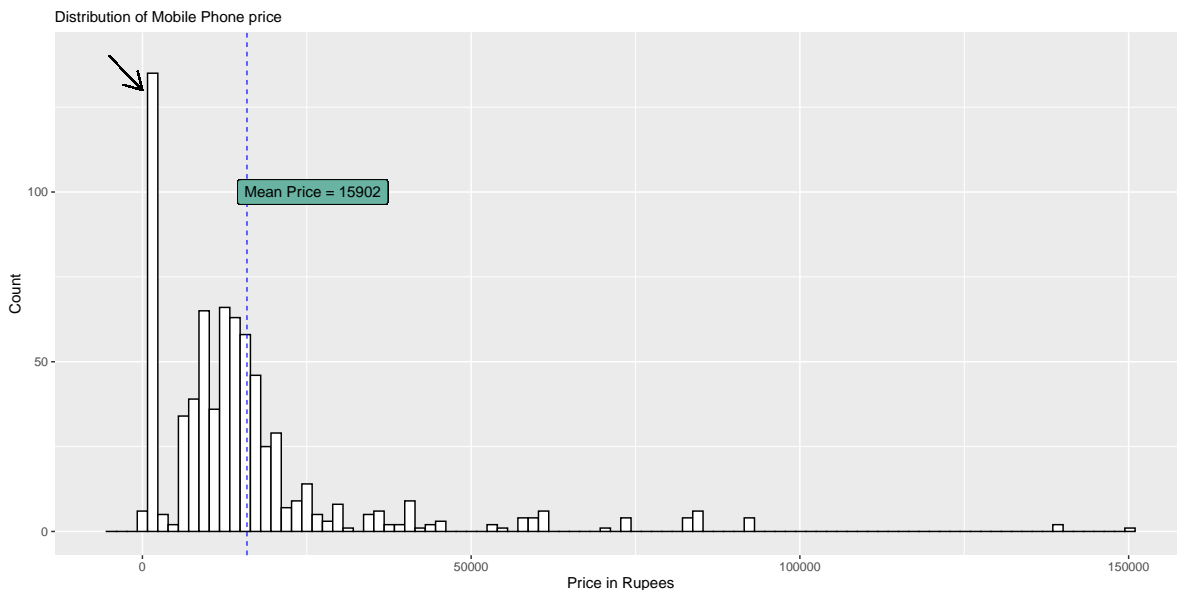
This section is focused on the exploration of relation between variables using various visualization techniques.

The main target is Price in the dataset. In the following visualization it shows the distribution of Mobile Phone price with long tail towards to the high end. The average price is 15902 Rupees. Most price falls in the range of 50000 Rupees. One really stands out price range (arrow indicated) is 1000-2000 Rupees with the highest frequency.

```

ggplot(data = mobile_dataset, mapping = aes(x=Price)) +
  geom_histogram(color="black", fill="white", bins = 100)+
  geom_vline(aes(xintercept=mean(Price)),
             color="blue", linetype="dashed", size=0.4) + geom_label(
    label="Mean Price = 15902",
    x= mean(mobile_dataset$Price) + 10000,
    y=100,
    label.padding = unit(0.35, "lines"),
    label.size = 0.25,
    color = "black",
    fill="#69b3a2"
  ) + geom_segment(aes(x = -5000, y = 140, xend = 0, yend = 130),lineend =
  "round",linejoin = "round",
    arrow = arrow(length = unit(0.5, "cm"))) + labs(y="Count",
    x="Price in Rupees",
    subtitle="Distribution of Mobile Phone price")

```

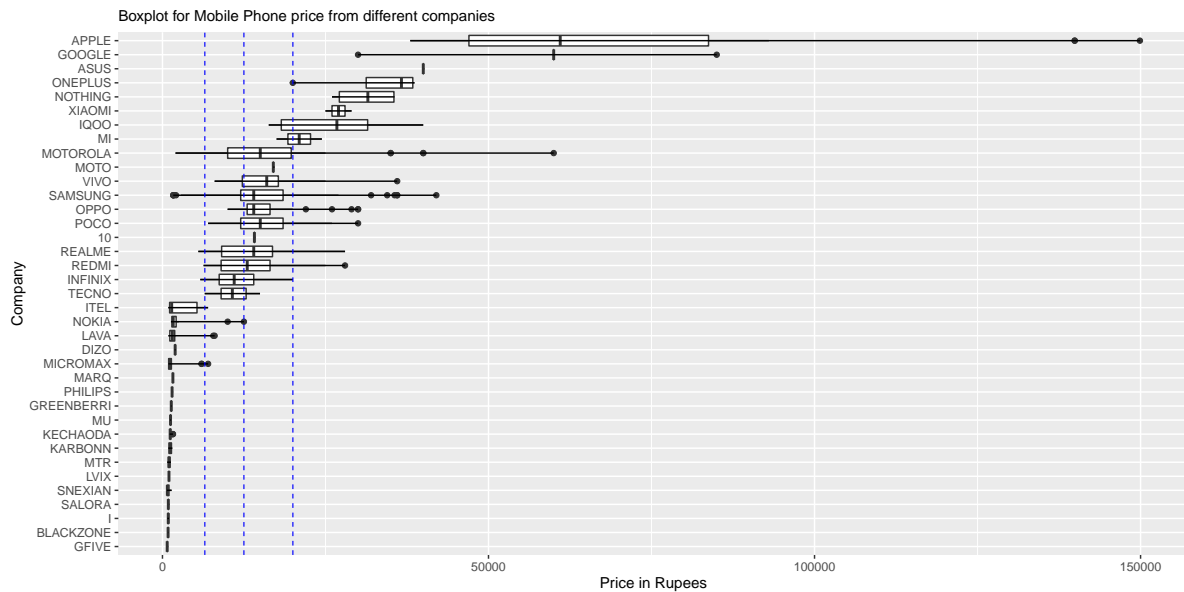


The following figure is to break down the Price by different phone-making companies. There are a total of 37 companies in the dataset. In consistent with previous figure about Price distribution, we see a big portion of Price falls between 9000 (left dash line) and 40000 (right dash line) Rupees. *Apple* alone contributes the most of high priced Mobile Phones, while companies like from *NOKIA* to *GFIVE* contribute to the low-priced ones.

```

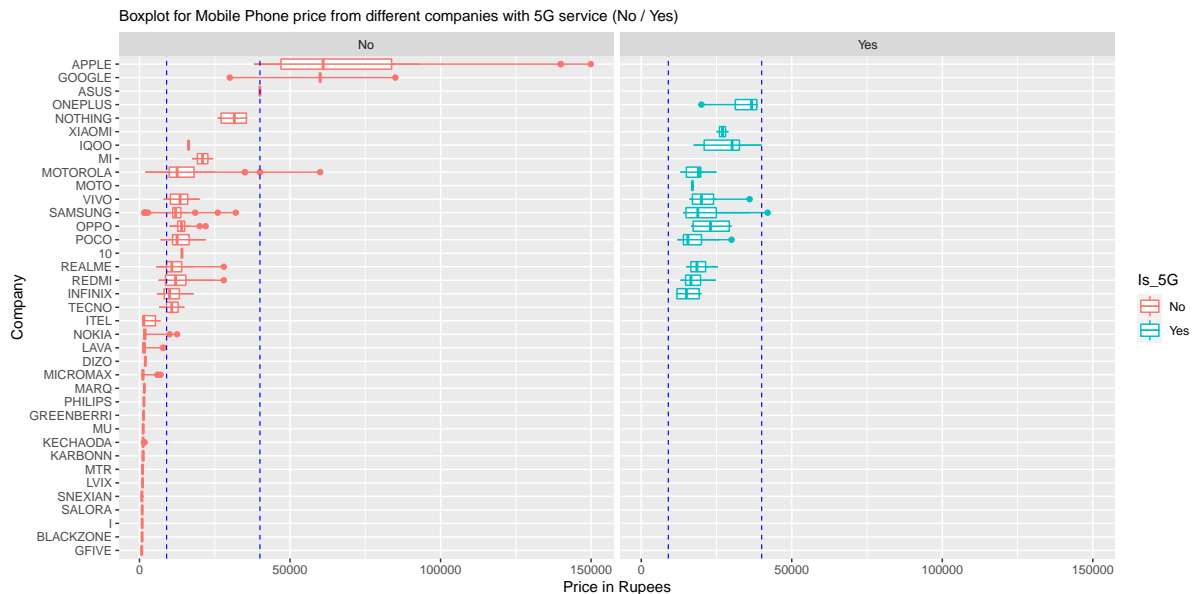
ggplot(data = mobile_dataset, mapping = aes(y = reorder(Company, Price), x = Price))
  +geom_boxplot()+ geom_line() +
  sapply(c(6499, 12490, 20000), function(xint) geom_vline(aes(xintercept = xint),
  color="blue", linetype="dashed", size=0.4)) + labs(y="Company", x="Price in
  Rupees",
  subtitle="Boxplot for Mobile Phone price from different companies")

```



The Price is further broken down by their 5G services. Interestingly, neither high-priced nor low-priced Mobile Phones have equipped 5G service. In contrast, almost all middle-priced Mobile Phones have 5G service. Price for those with 5G service are slightly more expensive than those without. NOTE: the dashed line indicates the price between 9000 (left dash line) and 40000 (right dash line) Rupees.

```
ggplot(data = mobile_dataset, mapping = aes(y = reorder(Company, Price), x = Price,
  ↪ color = Is_5G)) + geom_boxplot(position=position_dodge(width = 0.8)) + geom_line()
  ↪ +
  ↪ sapply(c(9000, 40000), function(xint) geom_vline(aes(xintercept = xint),
  ↪ color="blue", linetype="dashed", size=0.4)) + labs(y="Company", x="Price in
  ↪ Rupees",
  ↪ subtitle="Boxplot for Mobile Phone price from different companies with 5G
  ↪ service (No / Yes)") + facet_wrap(~Is_5G)
```

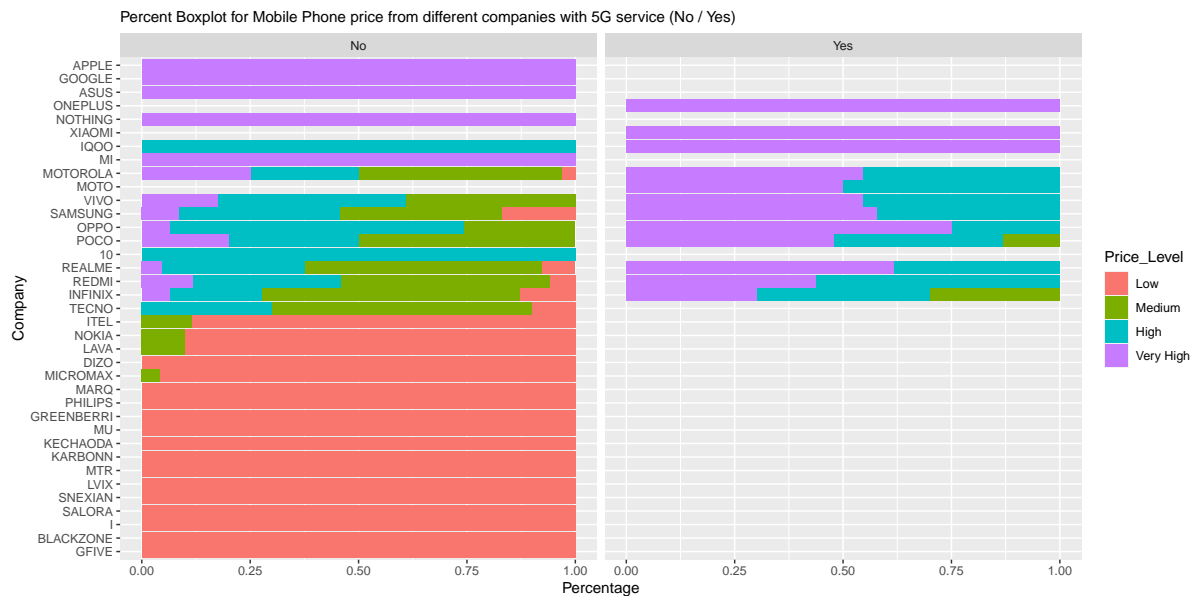


The pattern is more obvious when we provide color for 4 different price level (“Low”, “Medium”,



“High”, “Very High”). Although mobile phones from companies like APPLE, GOOGLE, ASUS and NOTHING have no 5G service, their price are all very high. It is worth investigating if some other attributes like brand effect, Storage capacity (Rom Size), and calculation power (Ram Size) are playing roles. Most Mobile Phones (except MI) with 5G service, again, have a higher proportion falling in a Very High price level, compared to the counterparts without 5G service. Almost all the rest mobile phones without 5G service fall in a Low price level.

```
ggplot(data = mobile_dataset, mapping = aes(y = reorder(Company, Price), fill =
  ↳ Price_Level)) + geom_bar(stat = "count", position="fill") + labs(y="Company",
  ↳ x="Percentage",
  subtitle="Percent Boxplot for Mobile Phone price from different companies
  ↳ with 5G service (No / Yes)") + facet_wrap(~Is_5G)
```



Next, we explored some quantitative features with respect to the Mobile Phone price. The first scatter plot shows an obvious positive relation between Rating and Price as the smooth line indicates. The higher the Rating, the higher the Price. And this holds when we remove those bottom left 2 outlier data points.

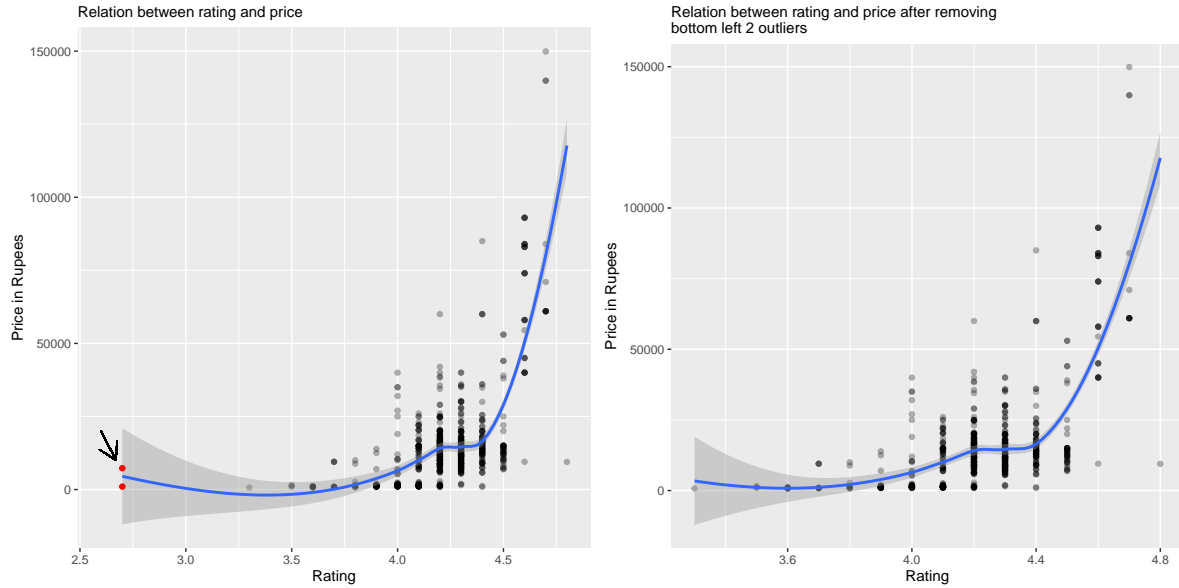
```
#The first plot is one with outlier
plot1 = ggplot(data = mobile_dataset, mapping = aes(x = Rating, y = Price)) +
  ↳ geom_point(alpha=0.3) + geom_smooth(se=T) +
  geom_segment(aes(x = 2.6, y = 20000, xend = 2.67, yend = 10000), lineend =
  ↳ "round", linejoin = "round", arrow = arrow(length = unit(0.5, "cm"))) +
  geom_point(data=mobile_dataset[mobile_dataset$Rating<3.0,], aes(x = Rating, y =
  ↳ Price), color='red') +
  labs(y="Price in Rupees", x="Rating", subtitle="Relation between rating and
  ↳ price")

#The first plot is a one with outliers removed
plot2 = ggplot(data = mobile_dataset[mobile_dataset$Rating>3.0, ], mapping = aes(x =
  ↳ Rating, y = Price)) + geom_point(alpha=0.3) + geom_smooth(se=T) +
  labs(y="Price in Rupees", x="Rating", subtitle="Relation between rating and price
  ↳ after removing \nbottom left 2 outliers")
```

```
grid.arrange(plot1, plot2, ncol=2)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

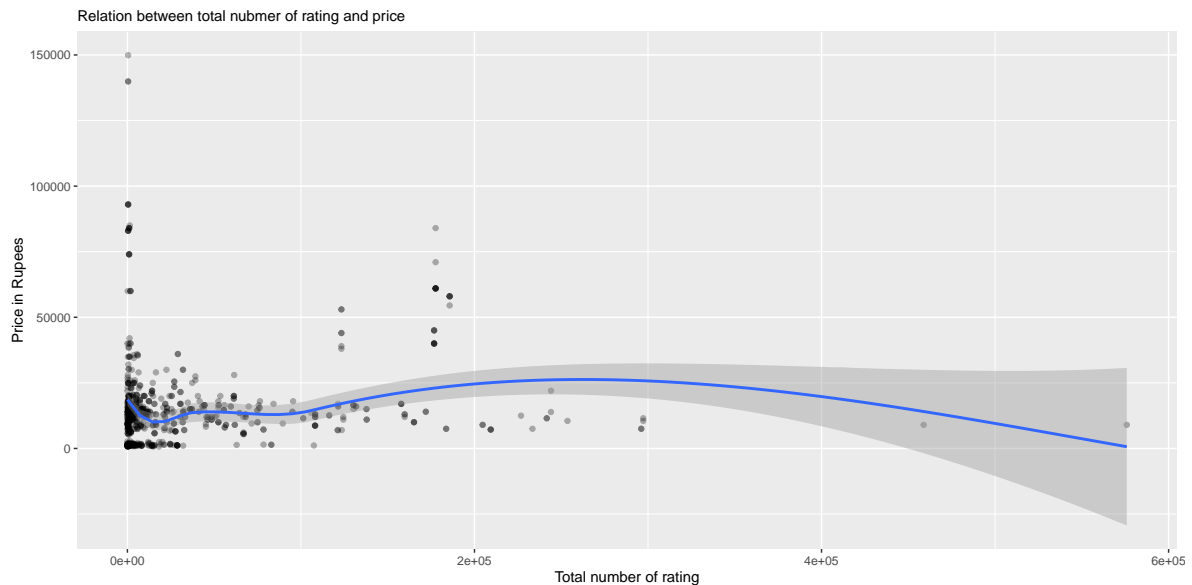
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The second scatter plot as follows shows the relation between Total number of rating and Price. As the smooth line indicates, there is no obvious correlation between them. The result holds even when we remove some seemingly outliers on the far right end.

```
ggplot(data = mobile_dataset, mapping = aes(x = No_of_ratings, y = Price)) +
  ↳ geom_point(alpha=0.3) + geom_smooth() + labs(y="Price in Rupees", x="Total number
  ↳ of rating", subtitle="Relation between total nubmer of rating and price")
```

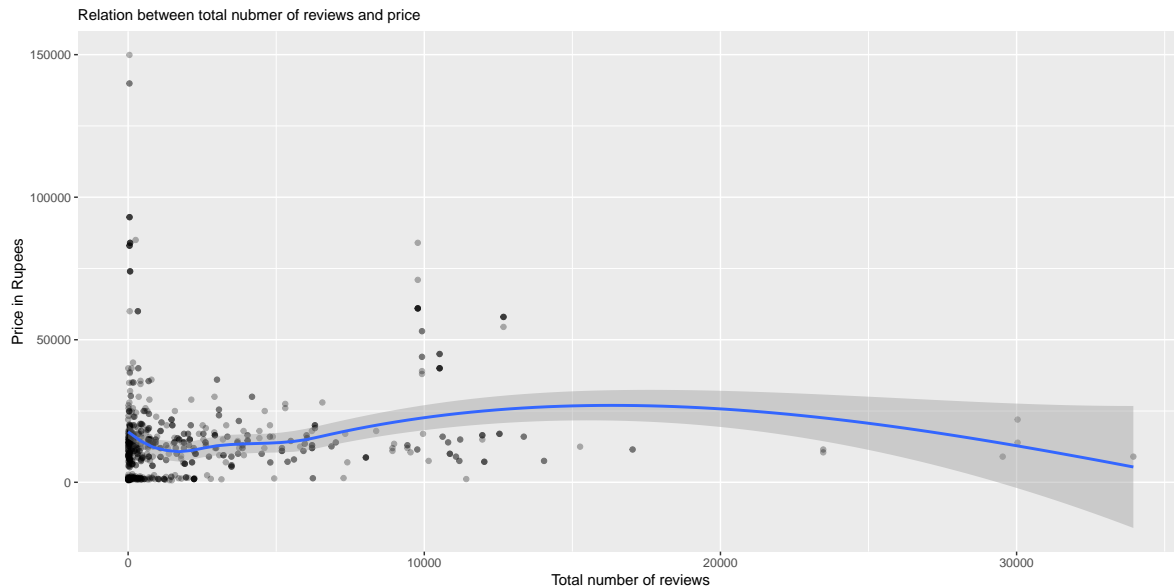
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The same pattern is also observed in the third scatter plot where relation between Total number of reviews and Price is plotted. As the smooth line indicates, there is no obvious correlation between them. The result holds even when we remove some seemingly outliers on the far right end.

```
ggplot(data = mobile_dataset, mapping = aes(x = TotalReviews, y = Price)) +
  ↳ geom_point(alpha=0.3) + geom_smooth() + labs(y="Price in Rupees", x="Total number
  ↳ of reviews", subtitle="Relation between total nubmer of reviews and price")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



In the fourth scatter plot, below, there seems a slight positive relation between Ram size (in GB) and the Price. Ram is normally associated with speed and performance of an operating system. The higher ram is, the better it is in speed and performance. It makes sense that a mobile phone with a larger Ram will charge more. However, because nowadays most mobile phones are very fast and stable, people wont tell too much difference, making the added on value from ram is only weekly associated with price.

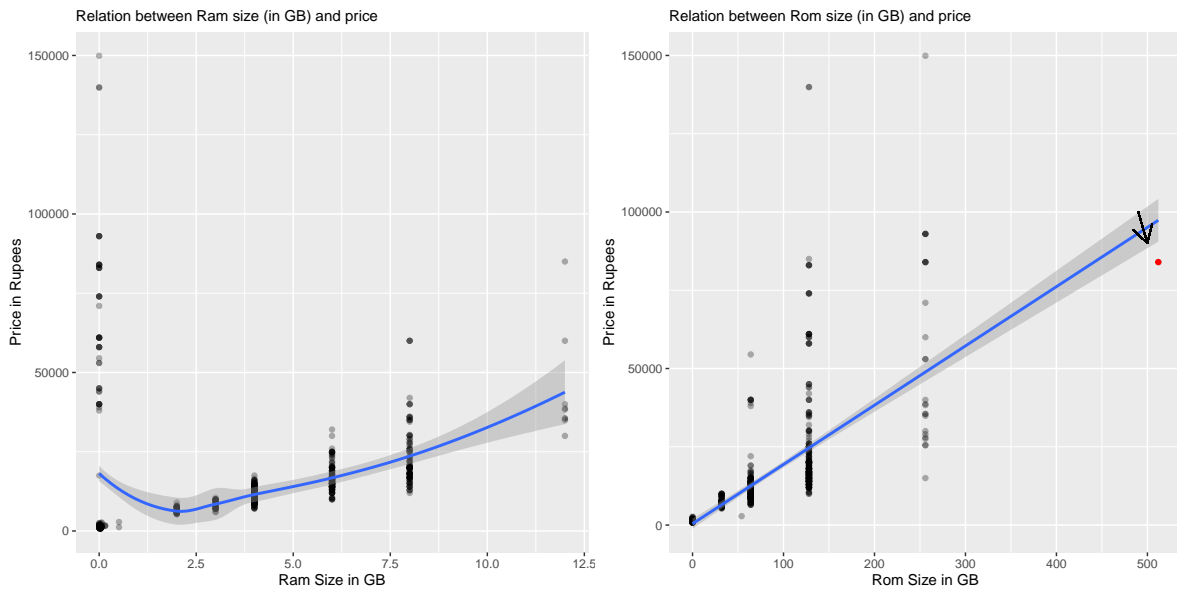
```
plot1 = ggplot(data = mobile_dataset, mapping = aes(x = RamSize_inGB, y = Price)) +
  ↳ geom_point(alpha=0.3) + geom_smooth(se=T) + labs(y="Price in Rupees", x="Ram Size
  ↳ in GB", subtitle="Relation between Ram size (in GB) and price")
```

```
plot2 = ggplot(data = mobile_dataset, mapping = aes(x = RomSize_inGB, y = Price)) +
  ↳ geom_point(alpha=0.3) + geom_smooth(se=T, method = lm) +
  geom_segment(aes(x = 490, y = 100000, xend = 500, yend = 90000), lineend =
  ↳ "round", linejoin = "round", arrow = arrow(length = unit(0.5, "cm"))) +
  geom_point(data=mobile_dataset[mobile_dataset$RomSize_inGB>400,], aes(x =
  ↳ RomSize_inGB, y = Price), color='red') +
  labs(y="Price in Rupees", x="Rom Size in GB", subtitle="Relation between Rom size
  ↳ (in GB) and price")
```

```
grid.arrange(plot1, plot2, ncol=2)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## `geom_smooth()` using formula 'y ~ x'
```



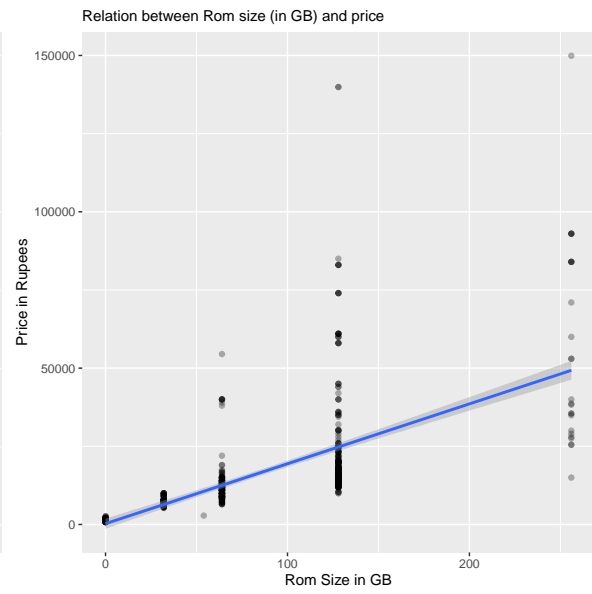
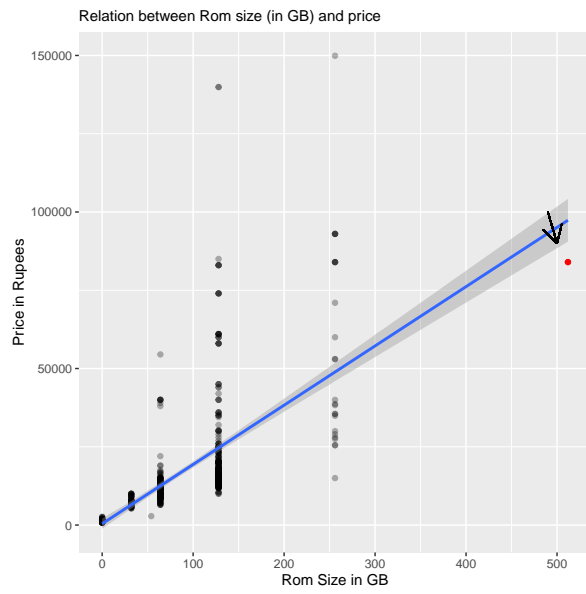
In the final scatter plot, again, we have two charts, one the original dataset, the other with outlier removed. Both cases tells the same story. In contrast to the previous scatter plot (Ram size (in GB) and the Price), there is a very obvious positive linear relation between Rom size (in GB) and the Price. This makes sense because often times a mobile phone is more limited by its non-volatile memory space than its speed & performance.

```
#The first plot is a one with outliers
plot1 = ggplot(data = mobile_dataset, mapping = aes(x = RomSize_inGB, y = Price)) +
  ↪ geom_point(alpha=0.3) + geom_smooth(se=T, method = lm) +
  ↪ geom_segment(aes(x = 490, y = 100000, xend = 500, yend = 90000), lineend =
  ↪ "round", linejoin = "round", arrow = arrow(length = unit(0.5, "cm"))) +
  ↪ geom_point(data=mobile_dataset[mobile_dataset$RomSize_inGB>400,], aes(x =
  ↪ RomSize_inGB, y = Price), color='red') +
  ↪ labs(y="Price in Rupees", x="Rom Size in GB", subtitle="Relation between Rom size
  ↪ (in GB) and price")

#The second plot is a one with outliers removed
plot2 = ggplot(data = mobile_dataset[mobile_dataset$RomSize_inGB<400, ], mapping =
  ↪ aes(x = RomSize_inGB, y = Price)) + geom_point(alpha=0.3) +
  ↪ geom_smooth(se=T, method = lm) +
  ↪ labs(y="Price in Rupees", x="Rom Size in GB", subtitle="Relation between Rom size
  ↪ (in GB) and price")

grid.arrange(plot1, plot2, ncol=2)
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



## Chapter 5: Statistical Data Analysis

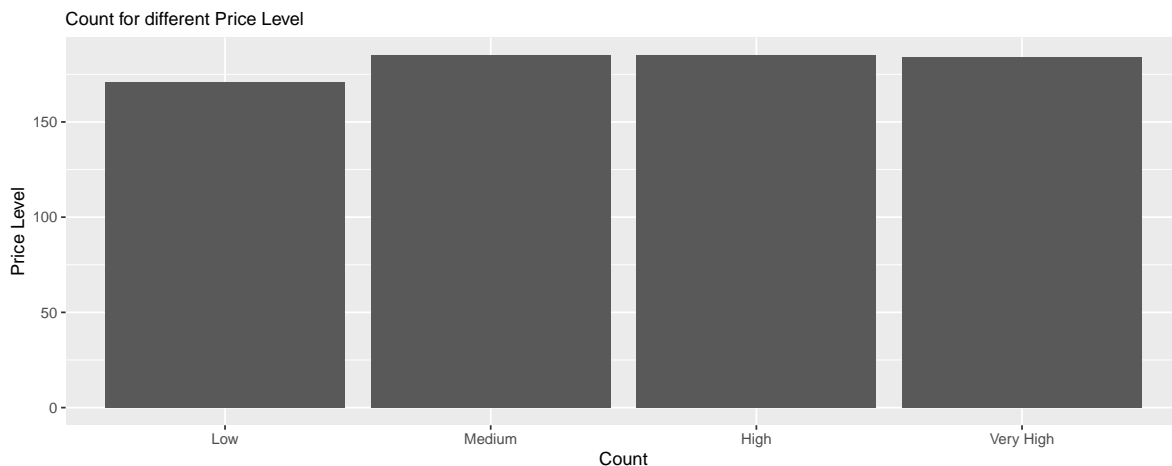
### 5.1: Prediction of Mobile Phone Price Level

In this section, we will use a range of statistical models to predict mobile phone price level: “Low”, “Medium”, “High”, “Very High” based on some selected features.

- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Decision Tree (Classification)
- Multinomial Regression

First, let us explore the distribution for different price level. From the following bar plot, you can see it is very close to uniformly distributed across the dataset. Balanced dataset is important for us when we do the classification.

```
price_level_count <- table(mobile_dataset$Price_Level) %>% as.data.frame()
ggplot(data = price_level_count, mapping = aes(x = Var1, y = Freq)) + geom_bar(stat
↪ = "identity") +
  labs(y="Price Level", x="Count", subtitle="Count for different Price Level")
```



### 5.1.1 Quadratic Discriminant Analysis (LDA) We want to use LDA to classify our observations into mobile phone price level: “Low”, “Medium”, “High”, “Very High”. However, we know LDA has 2 assumptions:

- The distribution of the predictors is normally distributed within each group.
- The homogeneity of variance-covariance matrices

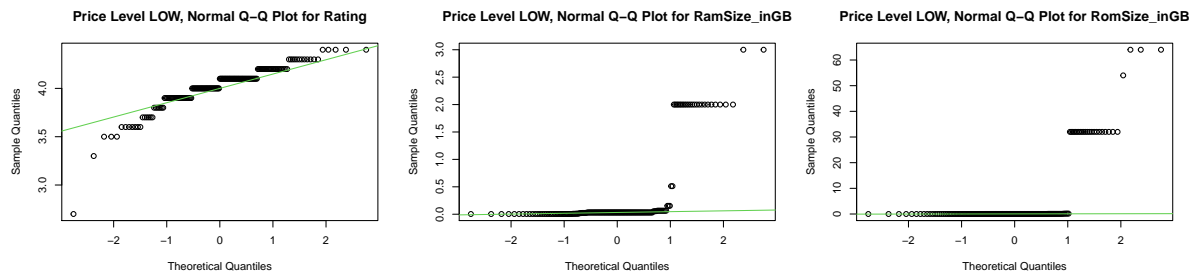
Based on EDA in section 4.2, we know three **numeric features** have a great impact on price level a mobile phone belongs to, which are *Rating*, *RamSize\_inGB* and *RomSize\_inGB*. As such, we will do a test for normal distribution for each predictor in each price level. Because we have 4 different price levels and 3 variables, the resulting normality check would be  $4 \times 3 = 12$  comparison. We use Q-Q plot and Shapiro test for normality check.

```
#split the dataset based on the price level
mobile_dataset_low <- mobile_dataset[mobile_dataset$Price_Level == "Low", ]
mobile_dataset_medium <- mobile_dataset[mobile_dataset$Price_Level == "Medium", ]
mobile_dataset_high <- mobile_dataset[mobile_dataset$Price_Level == "High", ]
mobile_dataset_very_high <- mobile_dataset[mobile_dataset$Price_Level == "Very
↪ High", ]
#generate the QQ plot for each price level
```

```

variables <- c("Rating", "RamSize_inGB", "RomSize_inGB")
par(mfrow = c(1,3))
for (i in variables){
  qqnorm(mobile_dataset_low[[i]], main = paste("Price Level LOW, Normal Q-Q Plot
  ↳ for", i))
  qqline(mobile_dataset_low[[i]], col = 3)
}

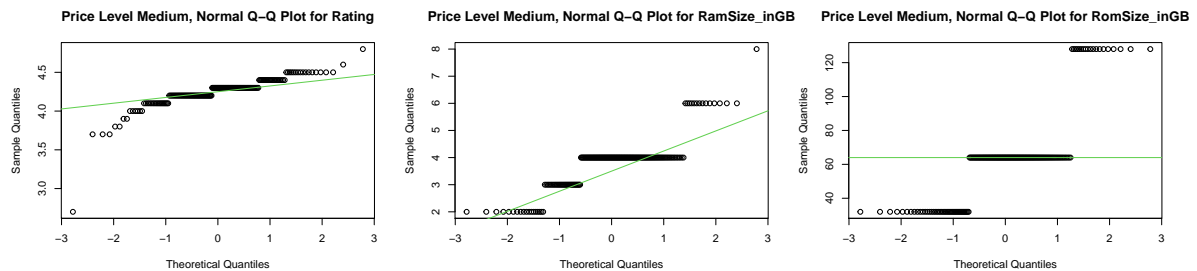
```



```

par(mfrow = c(1,3))
for (i in variables){
  qqnorm(mobile_dataset_medium[[i]], main = paste("Price Level Medium, Normal Q-Q
  ↳ Plot for", i))
  qqline(mobile_dataset_medium[[i]], col = 3)
}

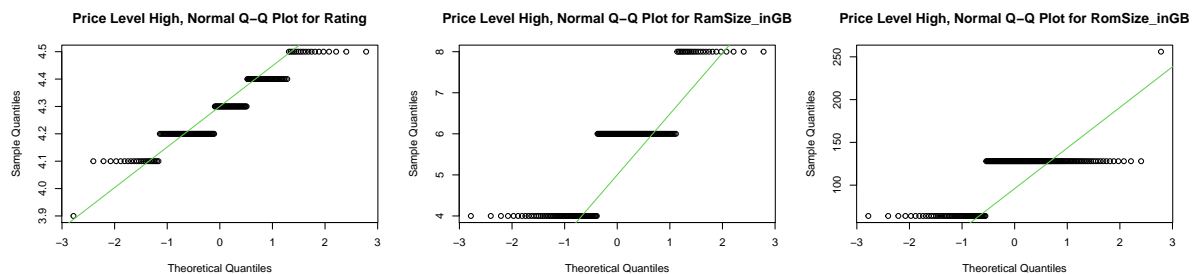
```



```

par(mfrow = c(1,3))
for (i in variables){
  qqnorm(mobile_dataset_high[[i]], main = paste("Price Level High, Normal Q-Q Plot
  ↳ for", i))
  qqline(mobile_dataset_high[[i]], col = 3)
}

```



```

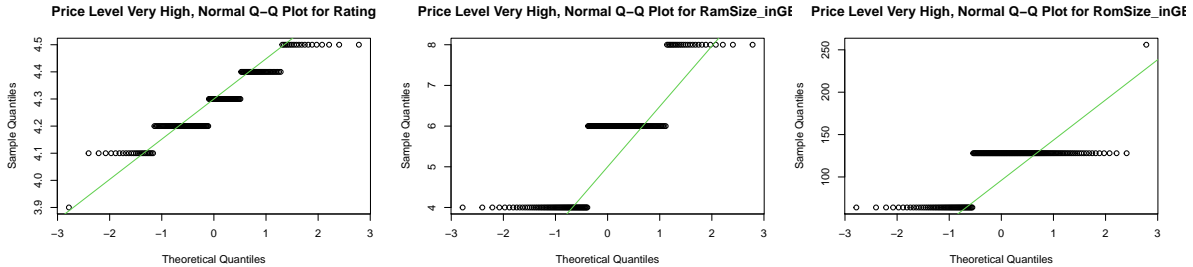
par(mfrow = c(1,3))
for (i in variables){

```

```

qqnorm(mobile_dataset_high[[i]], main = paste("Price Level Very High, Normal Q-Q
↳ Plot for", i))
qqline(mobile_dataset_high[[i]], col = 3)
}

```



```

#We also run the shapiro test to statistically conclude the result
p_value_rating <- c(shapiro.test(mobile_dataset_low$Rating)$p.value,
↳ shapiro.test(mobile_dataset_medium$Rating)$p.value,
                    shapiro.test(mobile_dataset_high$Rating)$p.value,
                    shapiro.test(mobile_dataset_very_high$Rating)$p.value)
p_value_RamSize_inGB <- c(shapiro.test(mobile_dataset_low$RamSize_inGB)$p.value,
↳ shapiro.test(mobile_dataset_medium$RamSize_inGB)$p.value,
                        shapiro.test(mobile_dataset_high$RamSize_inGB)$p.value,
                        shapiro.test(mobile_dataset_very_high$RamSize_inGB)$p.value)
p_value_rating_RomSize_inGB <-
↳ c(shapiro.test(mobile_dataset_low$RomSize_inGB)$p.value,
    shapiro.test(mobile_dataset_medium$RomSize_inGB)$p.value,
    shapiro.test(mobile_dataset_high$RomSize_inGB)$p.value,
    shapiro.test(mobile_dataset_very_high$RomSize_inGB)$p.value)
p_value_df <-
↳ data.frame(p_value_rating, p_value_RamSize_inGB, p_value_rating_RomSize_inGB)
rownames(p_value_df) <- c("Low", "Medium", "High", "Very High")
p_value_df

```

	p_value_rating	p_value_RamSize_inGB	p_value_rating_RomSize_inGB
## Low	1.470349e-10	1.875399e-22	7.757042e-23
## Medium	6.944763e-16	6.976122e-16	2.960373e-18
## High	9.714119e-09	3.072293e-15	1.522047e-20
## Very High	1.078638e-07	5.203225e-15	1.132641e-21

Based on the Q-Q plots (dots not fall on the line) and Shapiro test (p value is < 5%), we can conclude that all variables are not normal. Therefore, we will not proceed with LDA model. Although Quadratic Discriminant Analysis (QDA) also requires normality assumption, we still want to try QDA (in the next section) since it normally has less strict assumptions. NOTE, the above code is inspired by Pascal Schmidt from post [Assumption Checking of LDA vs. QDA – R Tutorial Pima Indians Data Set](#).

### 5.1.2 Quadratic Discriminant Analysis (QDA)

In consistent with LDA, we use the same three numeric features that have a great impact on price level a mobile phone belongs to, which are *Rating*, *RamSize\_inGB* and *RomSize\_inGB*. To that end, we split the dataset into 75% for training and 25% for testing purpose. We use each “Price\_Level” as the stratum and the proportional allocation principle to get a stratified sample. Then we fit the data into Quadratic Discriminant Analysis (QDA) as follows:



```

set.seed(2023)
Ny <- round(table(mobile_dataset$Price_Level)[unique(mobile_dataset$Price_Level)] *
  ↪ 0.75)
#Create the sample from the population
idx=sampling::strata(mobile_dataset, stratanames=c("Price_Level"), size=Ny,
  ↪ method="srswor")
strata_sample <- getdata(mobile_dataset,idx)
train <- mobile_dataset[idx$ID_unit,]
test <- mobile_dataset[-idx$ID_unit,]
#Train the data using Quadratic Discriminant Analysis
Model.fit<-qda(Price_Level~Rating+RamSize_inGB+RomSize_inGB, data = train)
Model.fit
cat(rep("-", 50), "\n")
#This plot is showing that LD1 can do a really good to separate these species rather
  ↪ than LD2
# partimat(Price_Level~Rating+RamSize_inGB+RomSize_inGB, data = train, method="qda")

```

```

## Call:
## qda(Price_Level ~ Rating + RamSize_inGB + RomSize_inGB, data = train)
##
## Prior probabilities of groups:
##      Low      Medium      High Very High
## 0.2352941 0.2555147 0.2555147 0.2536765
##
## Group means:
##           Rating RamSize_inGB RomSize_inGB
## Low           4.012500    0.3342734    5.949688
## Medium        4.237410    3.8201439    62.848921
## High          4.276259    5.6258993    113.266187
## Very High    4.341304    5.5797536    142.376812
## - - - - -

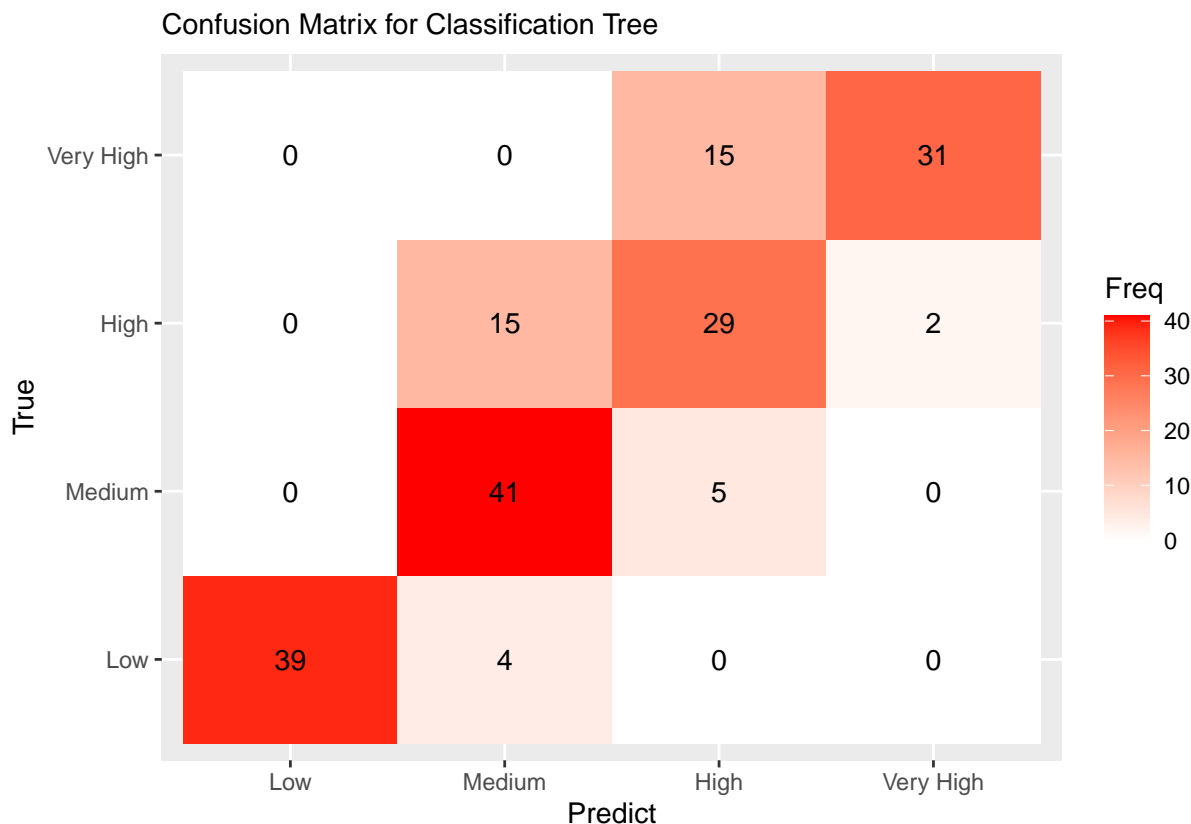
```

At last, we apply this model on the testing dataset, and calculate the misclassification rate.

```

Prob.predict<-predict(Model.fit,test,type="response")
ggplot(as.data.frame(table(Prob.predict$class,test$Price_Level)), aes(x = Var1, y =
  ↪ Var2, fill = Freq)) +geom_tile() + geom_text(aes(label = round(Freq, 1))) +
  ↪ scale_fill_gradient(low = "white", high = "red") + labs(y="True", x="Predict",
  ↪ subtitle="Confusion Matrix for Classification Tree")

```



```
cat(rep("-", 45), "\n")
cat("The misclassification rate for QDA is: ", mean(Prob.predict$class !=
  ↪ test$Price_Level))
```

```
## -----
## The misclassification rate for QDA is: 0.2265193
```

Conclusion: Quadratic Discriminant Analysis has a good performance to predict the mobile phone price level using only 3 variables (*Rating*, *RamSize\_inGB* and *RomSize\_inGB*), with a relatively high accuracy (77%).

### 5.1.3 Decision Tree (Classification)

To be consistent and comparable with QDA, we again use these three variables *Rating*, *RamSize\_inGB* and *RomSize\_inGB* to build the decision tree model. We split the dataset into 75% for training and 25% for testing purpose. We use each “Price\_Level” as the stratum and the proportional allocation principle to get a stratified sample. Then we fit the data into Classification tree model as follows:

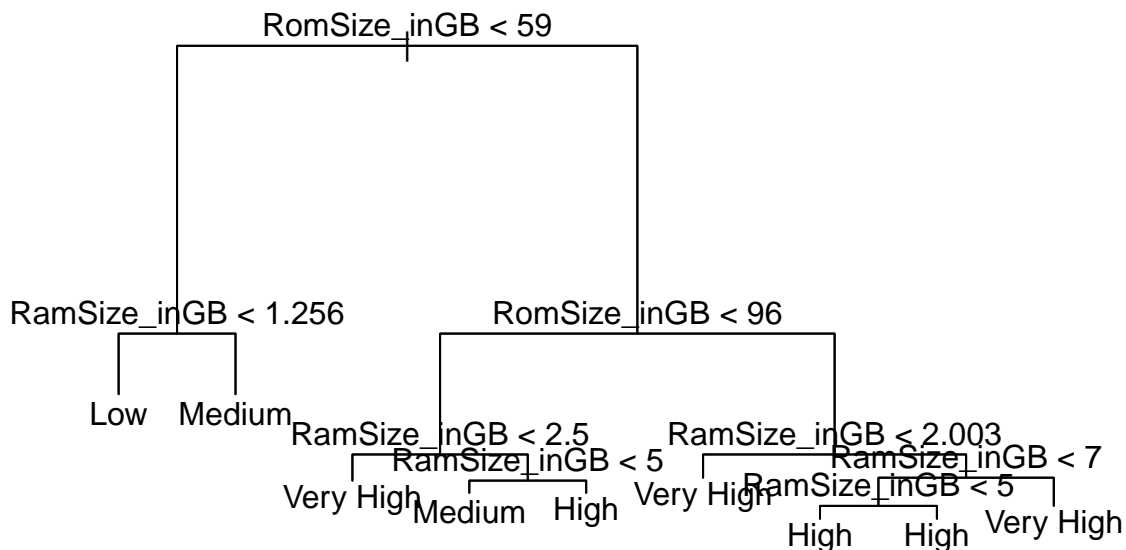
```
set.seed(2023)
Ny <- round(table(mobile_dataset$Price_Level)[unique(mobile_dataset$Price_Level)] *
  ↪ 0.75)
#Create the sample from the population
idx=sampling::strata(mobile_dataset, stratanames=c("Price_Level"), size=Ny,
  ↪ method="srswor")
strata_sample <- getdata(mobile_dataset,idx)
train <- mobile_dataset[idx$ID_unit,]
test <- mobile_dataset[-idx$ID_unit,]
```

```
#Fit the train data into the tree model
tree.class<-tree(factor(Price_Level)~Rating+RamSize_inGB+RomSize_inGB, train)
summary(tree.class)
```

```
##
## Classification tree:
## tree(formula = factor(Price_Level) ~ Rating + RamSize_inGB +
##       RomSize_inGB, data = train)
## Variables actually used in tree construction:
## [1] "RomSize_inGB" "RamSize_inGB"
## Number of terminal nodes: 9
## Residual mean deviance: 1.055 = 564.3 / 535
## Misclassification error rate: 0.2335 = 127 / 544
```

The tree is plotted without pruning as follows. There are a total of 9 terminal nodes in the tree.

```
#Let us plot the tree as well
plot(tree.class)
text(tree.class,pretty=0)
```



```
# rpart_fit <- rpart(factor(Price_Level)~Rating+RamSize_inGB+RomSize_inGB, train,
↪ method = 'class')
# rpart.plot(rpart_fit)
```

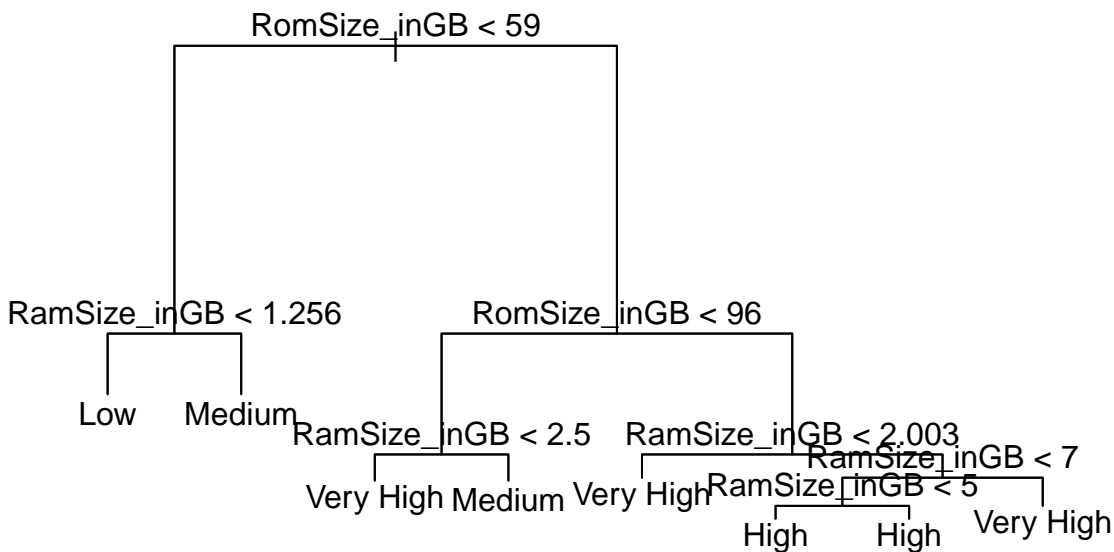
To find the best tree size, we used cross-validation to select the “best” number of tree size (the terminal nodes). Based on the following plot, it seems tree size 9 gives the smallest deviance. Just to demonstrate how pruned tree works and also reduce the complexity of tree, we decide to use 8 as the “best” tree size.

```
cv.class<-cv.tree(tree.class, FUN = prune.misclass)
plot(cv.class$size, cv.class$dev,type="b")
```



A tree after pruning is plotted as follows. There are a total of 8 terminal nodes in the tree.

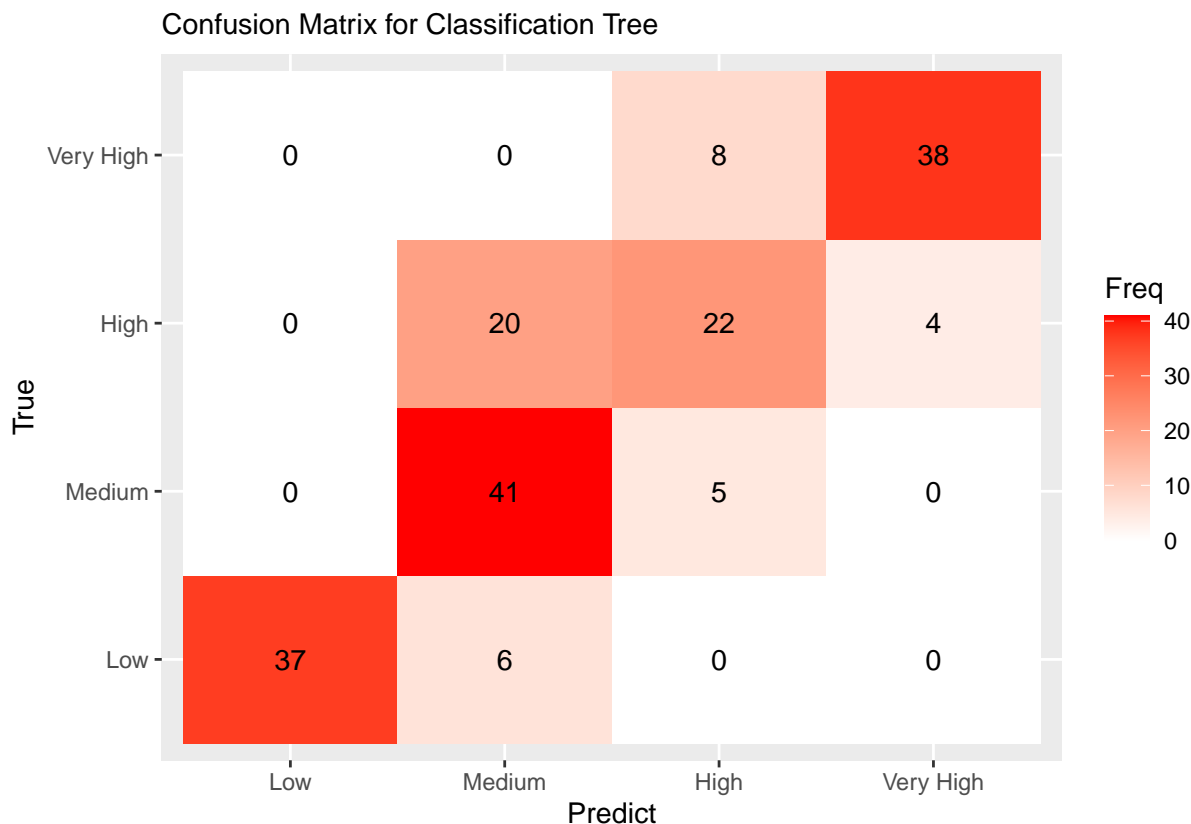
```
prune.class=prune.tree(tree.class,best=8)
plot(prune.class)
text(prune.class,pretty=0)
```



Fi-

nally, we use the pruned tree to do prediction.

```
prune.pred=predict(prune.class,test,type="class")
ggplot(as.data.frame(table(prune.pred,test$Price_Level)), aes(x = prune.pred, y =
  ↳ Var2, fill = Freq)) +geom_tile() + geom_text(aes(label = round(Freq, 1))) +
  ↳ scale_fill_gradient(low = "white", high = "red") + labs(y="True", x="Predict",
  ↳ subtitle="Confusion Matrix for Classification Tree")
```



```
cat(rep("-", 45), "\n")
cat("The misclassification rate for regression tree is: ", mean(prune.pred !=
  ↳ test$Price_Level))
```

```
## -----
## The misclassification rate for regression tree is: 0.2375691
```

Conclusion: Decision Tree (Classification) has also a good performance to predict the mobile phone price level using only 3 variables ( *Rating*, *RamSize\_inGB* and *RomSize\_inGB*), with a relatively high accuracy (76.2%).

#### 5.1.4 Multinomial Regression

This is an experimental analysis. Based on EDA in section 4.2, we know that Company name (branding) also plays a big role in the price level. We also know that Rom size of a mobile phone matters a lot as well. We are curious to see if a combination of *Company* and *RomSize* can also be used to distinguish the price level.

Normally, we would split the dataset into 75% for training and 25% for testing purpose. However, because our sample size is small, not every Company has multiple records in our dataset. Therefore, in the following test, we use whole sample as the training and test purpose.

Then we fit the data into Multinomial Regression as follows. Note, because our response is ordinal, we use Cumulative logit model; also parallel = TRUE: means beta is the same, this is a simple model.

We also use Chi square test to test for the goodness-of-fit for this model.

$H_0$  : The Cumulative logit model fits the observations

$H_1$  : The Cumulative logit model does not fit the observations

```
comany_rom_train <- mobile_dataset[c("Company", "RomSize_Ori", "Price_Level")]
multi_train_fit <- vglm(Price_Level~Company+RomSize_Ori,family=cumulative(parallel =
  ↪ TRUE),data=comany_rom_train)
cat("The p value for the chi test is: ",
  ↪ 1-pchisq(deviance(multi_train_fit),df.residual(multi_train_fit)))
cat("\n\nThe p-value here is large = 1, so we cannot reject null hypothesis. Our model
  ↪ seems to be provide a good fit.")
```

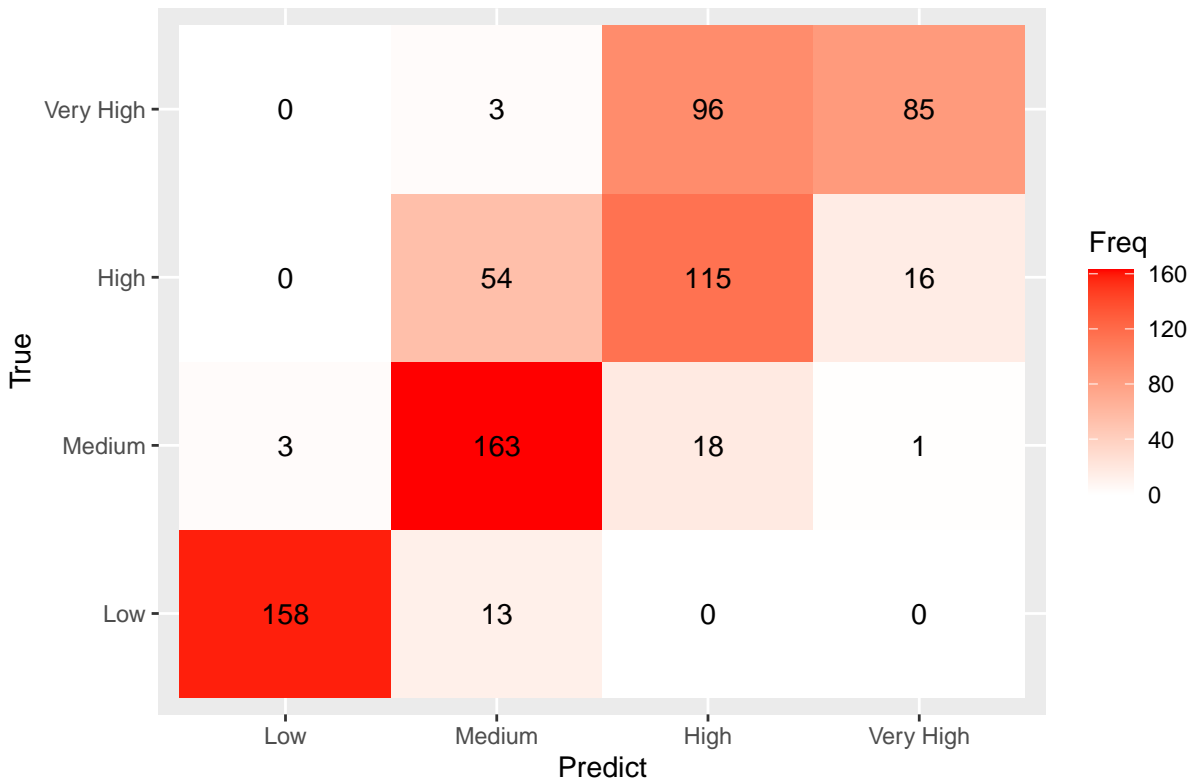
```
## The p value for the chi test is: 1
```

```
## The p-value here is large = 1, so we cannot reject null hypothesis. Our model seems to be provide
```

Finally, we use this model to predict use the same training dataset, as a demonstration purpose.

```
pred_fit <- predict(multi_train_fit, type="response")
#Use the record that has the highest probability as the final predication for each
  ↪ row.
pred_fit_class <-factor(dimnames(pred_fit)[[2]][max.col(pred_fit, 'first')], levels
  ↪ = c("Low", "Medium", "High", "Very High"))
ggplot(as.data.frame(table(pred_fit_class, mobile_dataset$Price_Level)), aes(x =
  ↪ pred_fit_class, y = Var2, fill = Freq)) +geom_tile() + geom_text(aes(label =
  ↪ round(Freq, 1))) + scale_fill_gradient(low = "white", high = "red") +
  ↪ labs(y="True", x="Predict", subtitle="Confusion Matrix for Classification Tree")
```

Confusion Matrix for Classification Tree



```
cat(rep("-", 45), "\n")
cat("The misclassification rate for Multinomial Regression is: ",
  ↪ mean(pred_fit_class != mobile_dataset$Price_Level))
```

```
## -----
## The misclassification rate for Multinomial Regression is: 0.2813793
```

Conclusion: Multinomial Regression has also a good performance to predict the mobile phone price level using only 2 variables ( *Company* and *RomSize*), with a relatively high accuracy (71.9%).

## 5.2: Cross-Validation

In section 5.1, we have compared three different models based on their misclassification rate: **Quadratic Discriminant Analysis (QDA)** = 0.2265193, **Decision Tree (Classification)** = 0.2375691, and **Multinomial Regression** = 0.2813793. In order to compare model performance, we decide to also use stratified 10-fold cross-validation to calculate the cross-validation errors of these 3 methods.

```
cat("\nFirst I am using the 10-fold cross validation for Quadratic Discriminant
↪ Analysis")
set.seed(2023)
#Create 10 folds
folds<-createFolds(mobile_dataset$Price_Level, k=10)
#Define a function to calculate the misclassification rate for each fold
misrate_fold<-function(idx){
  Train<-mobile_dataset[-idx,]
  Test<-mobile_dataset[idx,]
  fit <- qda(Price_Level~Rating+RamSize_inGB+RomSize_inGB, data = Train)
  pred<-predict(fit,Test)
  return(1- mean(pred$class == Test$Price_Level))
}
#Get the misclassification rate from each fold
type_misrate_fold=lapply(folds,misrate_fold)
#The average of misclassification rate for 10 folds is:
cat("\nThe average of misclassification rate for 10 folds
cross validation of Quadratic Discriminant Analysis is :",
mean(as.numeric(type_misrate_fold)))
cat("\n")
cat(rep("-", 45), "\n")

cat("\nSecond I am using the 10-fold cross validation for Decision Tree
↪ (Classification)")
set.seed(2023)
#Create 10 folds
folds<-createFolds(mobile_dataset$Price_Level, k=10)
#Define a function to calculate the misclassification rate for each fold
misrate_fold<-function(idx){
  Train<-mobile_dataset[-idx,]
  Test<-mobile_dataset[idx,]
  fit <- tree(factor(Price_Level)~Rating+RamSize_inGB+RomSize_inGB, data = Train)
  pred<-predict(fit,Test, type = 'class')
  return(1- mean(pred == Test$Price_Level))
}

#Get the misclassification rate from each fold
```

```

type_misrate_fold=lapply(folds,misrate_fold)
#The average of misclassification rate for 10 folds is:
cat("\nThe average of misclassification rate for 10 folds cross
    validation of Decision Tree (Classification) is :",
    ↪ mean(as.numeric(type_misrate_fold)))
cat("\n")
cat(rep("-", 45), "\n")

cat("\nBecause our sample size is small, not every Company has multiple records in
    ↪ our dataset. \nTherefore, we are not able to perform cross-validation for
    ↪ Multinomial regression.")

```

```

##
## First I am using the 10-fold cross validation for Quadratic Discriminant Analysis
## The average of misclassification rate for 10 folds
##     cross validation of Quadratic Discriminant Analysis is : 0.2690109
## - - - - -
##
## Second I am using the 10-fold cross validation for Decision Tree (Classification)
## The average of misclassification rate for 10 folds cross
##     validation of Decision Tree (Classification) is : 0.2314057
## - - - - -
##
## Because our sample size is small, not every Company has multiple records in our dataset.
## Therefore, we are not able to perform cross-validation for Multinomial regression.

```

### 5.3: Summary for Prediction of Mobile Phone Price Level

To summarize this part of analysis, we find some contradictory results. We used the balanced dataset trying to classify the Mobile Phone Price Level. In Section 5.1, we have the following misclassification rate from each model:

- **Quadratic Discriminant Analysis (QDA):** 0.2265193
- **Decision Tree (Classification):** 0.2375691
- **Multinomial Regression:** 0.2813793

We can see Quadratic Discriminant Analysis (QDA) gives the lowest misclassification rate, making this model the best one. However, when we do the 10-folds cross validation, we have the following misclassification rate from each model:

- **Quadratic Discriminant Analysis (QDA):** 0.2690109
- **Decision Tree (Classification):** 0.2314057
- **Multinomial Regression:** Not Available (see section 5.2 explanation)

From this result, Decision Tree (Classification) gives the lowest misclassification rate. Because cross-validation is averaging multiple misclassification rate, we think this result is more reliable.

So our final conclusion is that Decision Tree (Classification) can best predict the Mobile Phone Price Level using only 3 variables ( *Rating*, *RamSize\_inGB* and *RomSize\_inGB*), with a relatively high accuracy (76.9%).



## References

2020 review. “High-End Mobile Phones Price Have Soared 490% in 20 Years | This Is Money.” This Is Money, This Is Money, 23 July 2020, <https://www.thisismoney.co.uk/money/bills/article-8548235/High-end-mobile-phones-price-soared-490-20-years.html>.

MobilePhone’s dataset. “MobilePhone’s Dataset | Kaggle.” Kaggle: Your Machine Learning and Data Science Community, Kaggle, 20 Dec. 2022, <https://www.kaggle.com/datasets/sudhanshuy17/mobilephone>.

Pascal Schmidt. “Assumption Checking of LDA vs. QDA – R Tutorial Pima Indians Data Set”, retrieved Feb 17 2023, <https://thatdatatho.com/assumption-checking-lda-vs-qda-r-tutorial-2/>.