Mobile\_Price\_Prediction

Lourin Ejiuwa (CST/19/COM/00284)

2025-03-17

## About Dataset

This dataset contains 2,000 mobile phone records with 21 features detailing their specifications: 1. Performance: RAM ranges from 256 MB to 4 GB, processor speeds from 0.5 GHz to 3.0 GHz, and storage from 2 GB to 64 GB. 2. Battery & Display: Battery capacity varies between 500 and 1998 mAh. Phones weigh around 140g, with screen resolutions from 874×500 to 1633×1960 pixels. 3. Connectivity: 50% of phones have Bluetooth, WiFi, and touchscreens. 76% support 3G, and 52% support 4G. 4. Pricing: Phones are categorized into four price ranges (0 = Low, 1 = Medium, 2 = High, 3 = Very High), influenced by these features.

## Import Libraries

# Load necessary libraries  
library(dplyr)  
library(tidyr)  
library(ggplot2)

## Reading and Understanding the Dataset

**Import Dataset**

# Read the CSV file into a data frame  
data <- read.csv('train.csv')  
  
# View the first few rows to confirm it loaded correctly  
head(data)

## battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory m\_dep mobile\_wt  
## 1 842 0 2.2 0 1 0 7 0.6 188  
## 2 1021 1 0.5 1 0 1 53 0.7 136  
## 3 563 1 0.5 1 2 1 41 0.9 145  
## 4 615 1 2.5 0 0 0 10 0.8 131  
## 5 1821 1 1.2 0 13 1 44 0.6 141  
## 6 1859 0 0.5 1 3 0 22 0.7 164  
## n\_cores pc px\_height px\_width ram sc\_h sc\_w talk\_time three\_g touch\_screen  
## 1 2 2 20 756 2549 9 7 19 0 0  
## 2 3 6 905 1988 2631 17 3 7 1 1  
## 3 5 6 1263 1716 2603 11 2 9 1 1  
## 4 6 9 1216 1786 2769 16 8 11 1 0  
## 5 2 14 1208 1212 1411 8 2 15 1 1  
## 6 1 7 1004 1654 1067 17 1 10 1 0  
## wifi price\_range  
## 1 1 1  
## 2 0 2  
## 3 0 2  
## 4 0 2  
## 5 0 1  
## 6 0 1

**Discover Data**

# Get dimensions of the dataset  
dims <- dim(data) # dims[1] = rows, dims[2] = columns  
  
# Print dimensions  
cat("Dimensions of dataset:", dims, "\n")

## Dimensions of dataset: 2000 21

cat("Rows:", dims[1], "\nColumns:", dims[2], "\n")

## Rows: 2000   
## Columns: 21

**Statistical details**

summary(data)

## battery\_power blue clock\_speed dual\_sim   
## Min. : 501.0 Min. :0.000 Min. :0.500 Min. :0.0000   
## 1st Qu.: 851.8 1st Qu.:0.000 1st Qu.:0.700 1st Qu.:0.0000   
## Median :1226.0 Median :0.000 Median :1.500 Median :1.0000   
## Mean :1238.5 Mean :0.495 Mean :1.522 Mean :0.5095   
## 3rd Qu.:1615.2 3rd Qu.:1.000 3rd Qu.:2.200 3rd Qu.:1.0000   
## Max. :1998.0 Max. :1.000 Max. :3.000 Max. :1.0000   
## fc four\_g int\_memory m\_dep   
## Min. : 0.000 Min. :0.0000 Min. : 2.00 Min. :0.1000   
## 1st Qu.: 1.000 1st Qu.:0.0000 1st Qu.:16.00 1st Qu.:0.2000   
## Median : 3.000 Median :1.0000 Median :32.00 Median :0.5000   
## Mean : 4.309 Mean :0.5215 Mean :32.05 Mean :0.5018   
## 3rd Qu.: 7.000 3rd Qu.:1.0000 3rd Qu.:48.00 3rd Qu.:0.8000   
## Max. :19.000 Max. :1.0000 Max. :64.00 Max. :1.0000   
## mobile\_wt n\_cores pc px\_height   
## Min. : 80.0 Min. :1.000 Min. : 0.000 Min. : 0.0   
## 1st Qu.:109.0 1st Qu.:3.000 1st Qu.: 5.000 1st Qu.: 282.8   
## Median :141.0 Median :4.000 Median :10.000 Median : 564.0   
## Mean :140.2 Mean :4.521 Mean : 9.916 Mean : 645.1   
## 3rd Qu.:170.0 3rd Qu.:7.000 3rd Qu.:15.000 3rd Qu.: 947.2   
## Max. :200.0 Max. :8.000 Max. :20.000 Max. :1960.0   
## px\_width ram sc\_h sc\_w   
## Min. : 500.0 Min. : 256 Min. : 5.00 Min. : 0.000   
## 1st Qu.: 874.8 1st Qu.:1208 1st Qu.: 9.00 1st Qu.: 2.000   
## Median :1247.0 Median :2146 Median :12.00 Median : 5.000   
## Mean :1251.5 Mean :2124 Mean :12.31 Mean : 5.767   
## 3rd Qu.:1633.0 3rd Qu.:3064 3rd Qu.:16.00 3rd Qu.: 9.000   
## Max. :1998.0 Max. :3998 Max. :19.00 Max. :18.000   
## talk\_time three\_g touch\_screen wifi   
## Min. : 2.00 Min. :0.0000 Min. :0.000 Min. :0.000   
## 1st Qu.: 6.00 1st Qu.:1.0000 1st Qu.:0.000 1st Qu.:0.000   
## Median :11.00 Median :1.0000 Median :1.000 Median :1.000   
## Mean :11.01 Mean :0.7615 Mean :0.503 Mean :0.507   
## 3rd Qu.:16.00 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:1.000   
## Max. :20.00 Max. :1.0000 Max. :1.000 Max. :1.000   
## price\_range   
## Min. :0.00   
## 1st Qu.:0.75   
## Median :1.50   
## Mean :1.50   
## 3rd Qu.:2.25   
## Max. :3.00

The overhead table (train) displays: Each feature contains 2000 data recorded. There are some numerical features in the dataset including m\_dep, px height, and sc\_w that their min values don’t make sense.

**Number of uniqe elements in each columns**

# Get the number of unique elements in each column  
unique\_counts <- sapply(data, function(x) length(unique(x)))  
  
# Convert to a data frame and transpose it  
unique\_counts\_df <- as.data.frame(t(unique\_counts))  
  
# View the result  
print(unique\_counts\_df)

## battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory m\_dep mobile\_wt  
## 1 1094 2 26 2 20 2 63 10 121  
## n\_cores pc px\_height px\_width ram sc\_h sc\_w talk\_time three\_g touch\_screen  
## 1 8 21 1137 1109 1562 15 19 19 2 2  
## wifi price\_range  
## 1 2 4

**Information about the dataframe**

str(data)

## 'data.frame': 2000 obs. of 21 variables:  
## $ battery\_power: int 842 1021 563 615 1821 1859 1821 1954 1445 509 ...  
## $ blue : int 0 1 1 1 1 0 0 0 1 1 ...  
## $ clock\_speed : num 2.2 0.5 0.5 2.5 1.2 0.5 1.7 0.5 0.5 0.6 ...  
## $ dual\_sim : int 0 1 1 0 0 1 0 1 0 1 ...  
## $ fc : int 1 0 2 0 13 3 4 0 0 2 ...  
## $ four\_g : int 0 1 1 0 1 0 1 0 0 1 ...  
## $ int\_memory : int 7 53 41 10 44 22 10 24 53 9 ...  
## $ m\_dep : num 0.6 0.7 0.9 0.8 0.6 0.7 0.8 0.8 0.7 0.1 ...  
## $ mobile\_wt : int 188 136 145 131 141 164 139 187 174 93 ...  
## $ n\_cores : int 2 3 5 6 2 1 8 4 7 5 ...  
## $ pc : int 2 6 6 9 14 7 10 0 14 15 ...  
## $ px\_height : int 20 905 1263 1216 1208 1004 381 512 386 1137 ...  
## $ px\_width : int 756 1988 1716 1786 1212 1654 1018 1149 836 1224 ...  
## $ ram : int 2549 2631 2603 2769 1411 1067 3220 700 1099 513 ...  
## $ sc\_h : int 9 17 11 16 8 17 13 16 17 19 ...  
## $ sc\_w : int 7 3 2 8 2 1 8 3 1 10 ...  
## $ talk\_time : int 19 7 9 11 15 10 18 5 20 12 ...  
## $ three\_g : int 0 1 1 1 1 1 1 1 1 1 ...  
## $ touch\_screen : int 0 1 1 0 1 0 0 1 0 0 ...  
## $ wifi : int 1 0 0 0 0 0 1 1 0 0 ...  
## $ price\_range : int 1 2 2 2 1 1 3 0 0 0 ...

# Additional info: dimensions, column and names  
cat("Dimensions:", dim(data), "\n")

## Dimensions: 2000 21

cat("Column Names:", colnames(data), "\n")

## Column Names: battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory m\_dep mobile\_wt n\_cores pc px\_height px\_width ram sc\_h sc\_w talk\_time three\_g touch\_screen wifi price\_range

## Preprocessing

Because some numeric features in the dataset, including m\_dep, px height, and sc\_w, whose minimum values did not make sense, should be investigated more deeply.

**Mobile Depth (Cm):**

# Summary statistics for the 'm\_dep' column  
summary(data$m\_dep)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1000 0.2000 0.5000 0.5018 0.8000 1.0000

The minimum range of mobile phone depth varies depending on the manufacturer, model, and specific phone design. However, most smartphones have a thickness (depth) in the range of 7mm to 10mm. Some high-end models may be thinner than this range, with depths as low as 6mm or even less.

**Mobile Height (Cm):**

# Summary statistics for the 'px\_height' column  
summary(data$px\_height)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 282.8 564.0 645.1 947.2 1960.0

I considered the dimensions of the Nokia 1100 (96 x 65 pixels, 3:2 ratio) as the minimum of Pixel Resolution.➡️ 65 pixels

**Screen Width (Cm):**

# Summary statistics for the 'sc\_w' column  
summary(data$sc\_w)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 2.000 5.000 5.767 9.000 18.000

I will consider the minimum Screen Width to be 1 inch = 2.54 centimeters.

## Data Cleaning

**Let’s first address what we discussed in the data preprocessing section.**

-Mobile Depth

# Find values below 0.5 cm in 'm\_dep'  
below\_threshold <- data$m\_dep[data$m\_dep < 0.5]  
  
# Count the number of values below 0.5 cm  
num\_below\_threshold <- length(below\_threshold)  
  
# Print the result  
cat("Number of values below 0.5 cm in 'm\_dep' feature:", num\_below\_threshold, "\n")

## Number of values below 0.5 cm in 'm\_dep' feature: 900

# Replace values below 0.5 cm with 0.5 cm in 'm\_dep'  
data$m\_dep[data$m\_dep < 0.5] <- 0.5  
  
# Check summary statistics for 'm\_dep'  
summary(data$m\_dep)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.500 0.500 0.500 0.626 0.800 1.000

-Pixel Resolution

# Find values below 65 pixels in 'px\_height'  
below\_threshold1 <- data$px\_height[data$px\_height < 65]  
  
# Count the number of values below 65 pixels  
num\_below\_threshold1 <- length(below\_threshold1)  
  
# Print the result  
cat("Number of values below 65 pixels in 'px\_height' feature:", num\_below\_threshold1, "\n")

## Number of values below 65 pixels in 'px\_height' feature: 90

# Replace values below 65 pixels with 65 pixels in 'px\_height'  
data$px\_height[data$px\_height < 65] <- 65  
  
# Check summary statistics for 'px\_height'  
summary(data$px\_height)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 65.0 282.8 564.0 646.5 947.2 1960.0

-Screen Width

# Find values below 2.54 cm in 'sc\_w'  
below\_threshold2 <- data$sc\_w[data$sc\_w < 2.54]  
  
# Count the number of values below 2.54 cm  
num\_below\_threshold2 <- length(below\_threshold2)  
  
# Print the result  
cat("Number of values below 2.54 cm in 'sc\_w' feature:", num\_below\_threshold2, "\n")

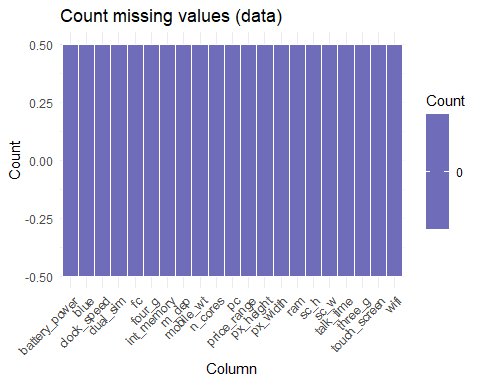
## Number of values below 2.54 cm in 'sc\_w' feature: 546

# Replace values below 2.54 cm with 2.54 cm in 'sc\_w'  
data$sc\_w[data$sc\_w < 2.54] <- 2.54  
  
# Check summary statistics for 'sc\_w'  
summary(data$sc\_w)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.540 2.540 5.000 6.199 9.000 18.000

**Checking for missing values**

# Count missing values for each column  
missing\_values <- colSums(is.na(data))  
  
# Convert to data frame for plotting  
missing\_values\_df <- data.frame(Column = names(missing\_values), Count = missing\_values)  
  
# Plot the missing values heatmap  
ggplot(missing\_values\_df, aes(x = Column, y = Count, fill = Count)) +  
 geom\_tile(color = "white") +  
 scale\_fill\_gradient(low = "lightblue", high = "darkblue") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 ggtitle("Count missing values (data)")



There aren’t any missing values in the dataset.

**Duplicated Data**

# Count duplicated rows in the dataset  
num\_duplicates <- sum(duplicated(data))  
  
# Print the result  
cat("Number of duplicated rows:", num\_duplicates, "\n")

## Number of duplicated rows: 0

**Outliers**

# Separate numerical and categorical features  
num\_cols <- data %>% select(battery\_power, clock\_speed, fc, int\_memory, m\_dep, mobile\_wt,  
 pc, px\_height, px\_width, ram, sc\_h, sc\_w, talk\_time)  
  
cat\_cols <- data %>% select(blue, dual\_sim, four\_g, n\_cores, three\_g, touch\_screen, wifi)  
  
# Separate numerical and categorical column names into different lists  
numerical\_columns <- c('battery\_power', 'clock\_speed', 'fc', 'int\_memory', 'm\_dep',   
 'mobile\_wt', 'pc', 'px\_height', 'px\_width', 'ram', 'sc\_h',   
 'sc\_w', 'talk\_time')  
  
categorical\_columns <- c('blue', 'dual\_sim', 'four\_g', 'n\_cores', 'three\_g',   
 'touch\_screen', 'wifi')  
  
# Print the lists  
print(numerical\_columns)

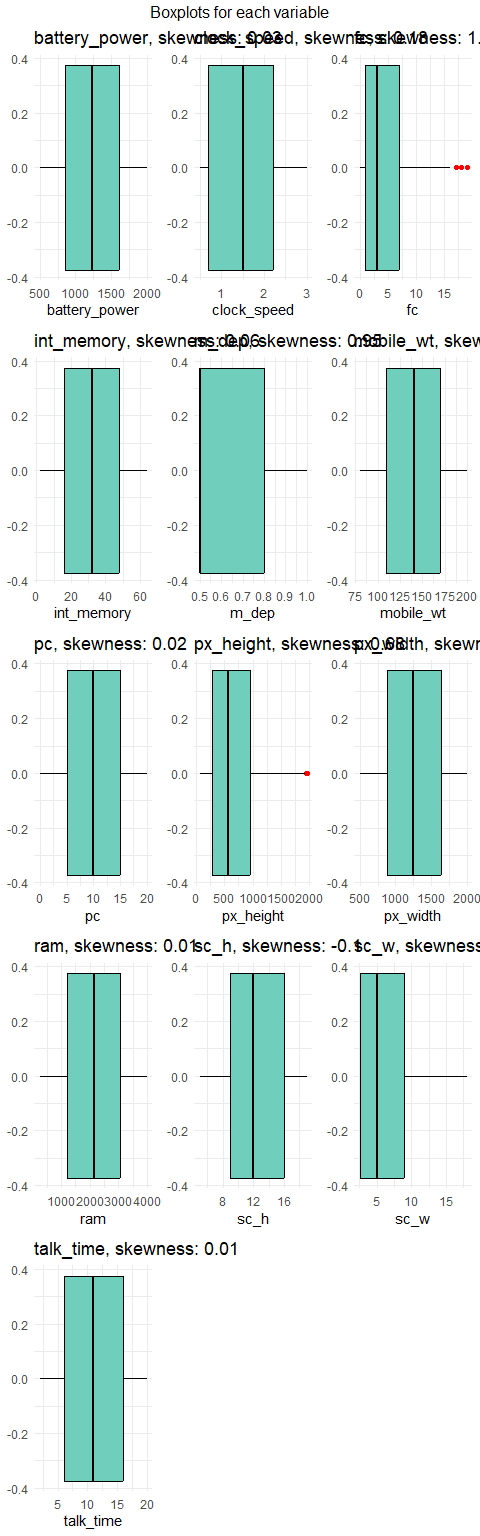
## [1] "battery\_power" "clock\_speed" "fc" "int\_memory"   
## [5] "m\_dep" "mobile\_wt" "pc" "px\_height"   
## [9] "px\_width" "ram" "sc\_h" "sc\_w"   
## [13] "talk\_time"

print(categorical\_columns)

## [1] "blue" "dual\_sim" "four\_g" "n\_cores" "three\_g"   
## [6] "touch\_screen" "wifi"

**Visual features**

library(e1071)  
library(ggplot2)  
library(gridExtra)  
  
# Custom function to plot boxplots  
boxplots\_custom <- function(dataset, columns\_list, suptitle) {  
 plots <- lapply(columns\_list, function(col) {  
 ggplot(dataset, aes(x = .data[[col]])) +  
 geom\_boxplot(fill = "#6fcfbc", color = "black", outlier.color = "red") +  
 theme\_minimal() +  
 labs(title = paste0(col, ", skewness: ", round(skewness(dataset[[col]], na.rm = TRUE), 2)))  
 })  
   
 # Arrange plots in a grid  
 grid.arrange(grobs = plots, ncol = 3, top = suptitle)  
}  
  
# Call the function  
boxplots\_custom(data, numerical\_columns, "Boxplots for each variable")



**Detect Outliers**

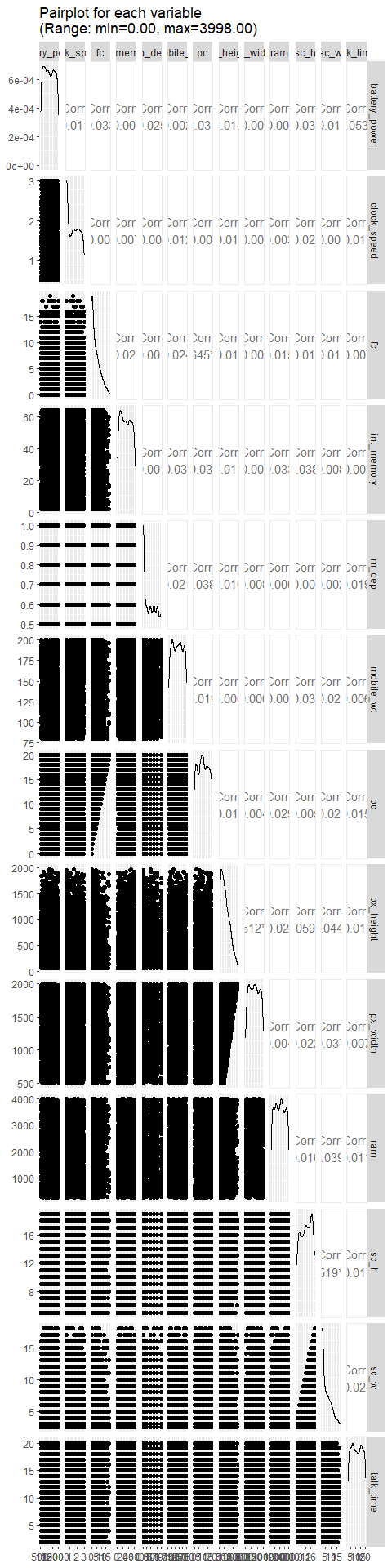
# Calculate Q1, Q3, and IQR for each numerical column  
Q1 <- apply(data[numerical\_columns], 2, quantile, 0.25)  
Q3 <- apply(data[numerical\_columns], 2, quantile, 0.75)  
IQR <- Q3 - Q1  
  
# Identify outliers using the IQR method  
outliers <- sapply(numerical\_columns, function(col) {  
 data[[col]] < (Q1[col] - 1.5 \* IQR[col]) | data[[col]] > (Q3[col] + 1.5 \* IQR[col])  
})  
  
# Count the number of outliers for each variable  
num\_outliers <- colSums(outliers)  
  
# Display the number of outliers for each variable  
t(data.frame(num\_outliers))

## battery\_power clock\_speed fc int\_memory m\_dep mobile\_wt pc  
## num\_outliers 0 0 18 0 0 0 0  
## px\_height px\_width ram sc\_h sc\_w talk\_time  
## num\_outliers 2 0 0 0 0 0

While the boxplots in the table above indicate the presence of outliers in the fc and px\_height features, we cannot justify removing them from the dataset without a strong rationale to do so. Therefore, we have decided to retain these outliers in our analysis.

**Check for Noise**

# Load required package  
library(GGally)  
library(ggplot2)  
  
# Create the pairplot  
dnp <- ggpairs(data[, numerical\_columns],   
 title = sprintf("Pairplot for each variable\n(Range: min=%.2f, max=%.2f)",  
 min(data[, numerical\_columns], na.rm = TRUE),   
 max(data[, numerical\_columns], na.rm = TRUE)))  
  
# Show the plot  
print(dnp)

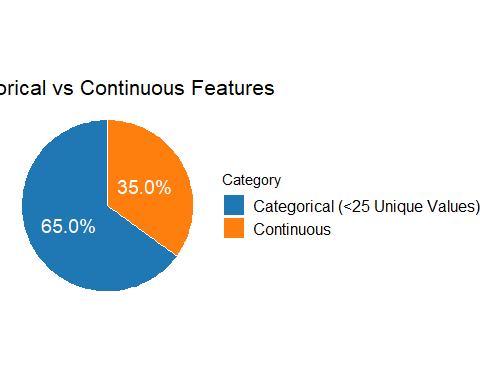


It seems that there isn’t any Noisy data in the train dataset.

##Exploratory Data Analysis (EDA)

**Continuos and Categorical Data Distribution**

# Load libraries  
library(ggplot2)  
  
# Set target and features  
TARGET <- 'price\_range'  
FEATURES <- setdiff(names(data), c('data', TARGET))  
  
# Separate categorical and continuous features  
cat\_features <- FEATURES[sapply(data[FEATURES], function(col) length(unique(col)) < 25)]  
cont\_features <- FEATURES[sapply(data[FEATURES], function(col) length(unique(col)) >= 25)]  
  
num\_cat\_features <- length(cat\_features)  
num\_cont\_features <- length(cont\_features)  
total\_features <- num\_cat\_features + num\_cont\_features  
  
# Create a pie chart dataframe with percentages  
df\_pie <- data.frame(  
 Category = c("Categorical (<25 Unique Values)", "Continuous"),  
 Count = c(num\_cat\_features, num\_cont\_features),  
 Percentage = c((num\_cat\_features / total\_features) \* 100, (num\_cont\_features / total\_features) \* 100)  
)  
  
# Plot with improved colors and percentage labels  
ggplot(df\_pie, aes(x = "", y = Percentage, fill = Category)) +  
 geom\_bar(stat = "identity", width = 1, color = "white") +  
 coord\_polar("y", start = 0) +  
 scale\_fill\_manual(values = c("#1F77B4", "#FF7F0E")) +  
 theme\_void() +  
 geom\_text(aes(label = sprintf("%.1f%%", Percentage)), position = position\_stack(vjust = 0.5), color = "white", size = 5) +  
 ggtitle("Categorical vs Continuous Features") +  
 theme(legend.text = element\_text(size = 12),  
 plot.title = element\_text(size = 16, hjust = 0.5))



# Print the summary  
cat("Total number of features except for the target:", total\_features, "\n")

## Total number of features except for the target: 20

cat("Number of categorical (<25 Unique Values) features:", num\_cat\_features, "\n")

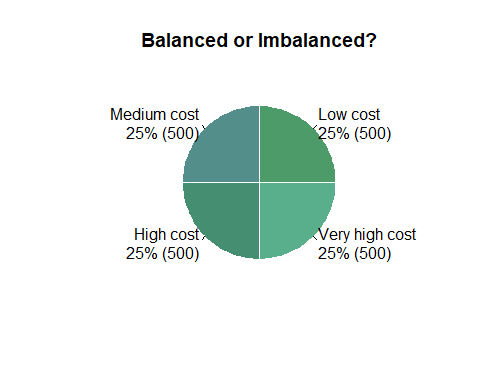
## Number of categorical (<25 Unique Values) features: 13

cat("Number of continuous features:", num\_cont\_features, "\n")

## Number of continuous features: 7

**Data Imbalance**

# Count the number of occurrences of each value in the 'price\_range' column  
value\_counts <- table(data$price\_range)  
  
# Define the label strings correctly  
labels <- c('Low cost', 'Medium cost', 'High cost', 'Very high cost')  
  
# Define the colors for each pie slice  
colors <- c('#4d9b68', '#538e8a', '#468e71', '#59ae8c')  
  
# Create the pie chart with percentages and counts  
percentages <- round(value\_counts / sum(value\_counts) \* 100, 1)  
labels\_with\_values <- paste0(labels, "\n", percentages, "% (", value\_counts, ")")  
  
# Plot the pie chart  
pie(value\_counts, labels = labels\_with\_values, col = colors, main = "Balanced or Imbalanced?", border = "white")

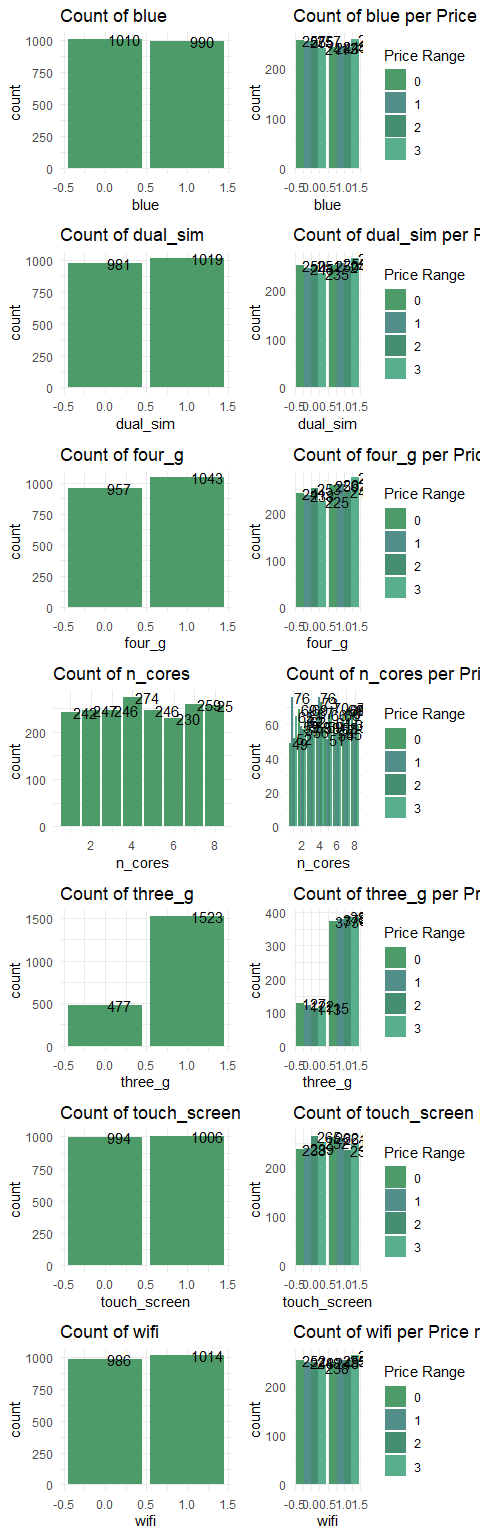


The above charts show that all classes of the target variable have the same count. So, the target data is completely balanced.

**Univariate Analysis**

-Exploring Categorical Features

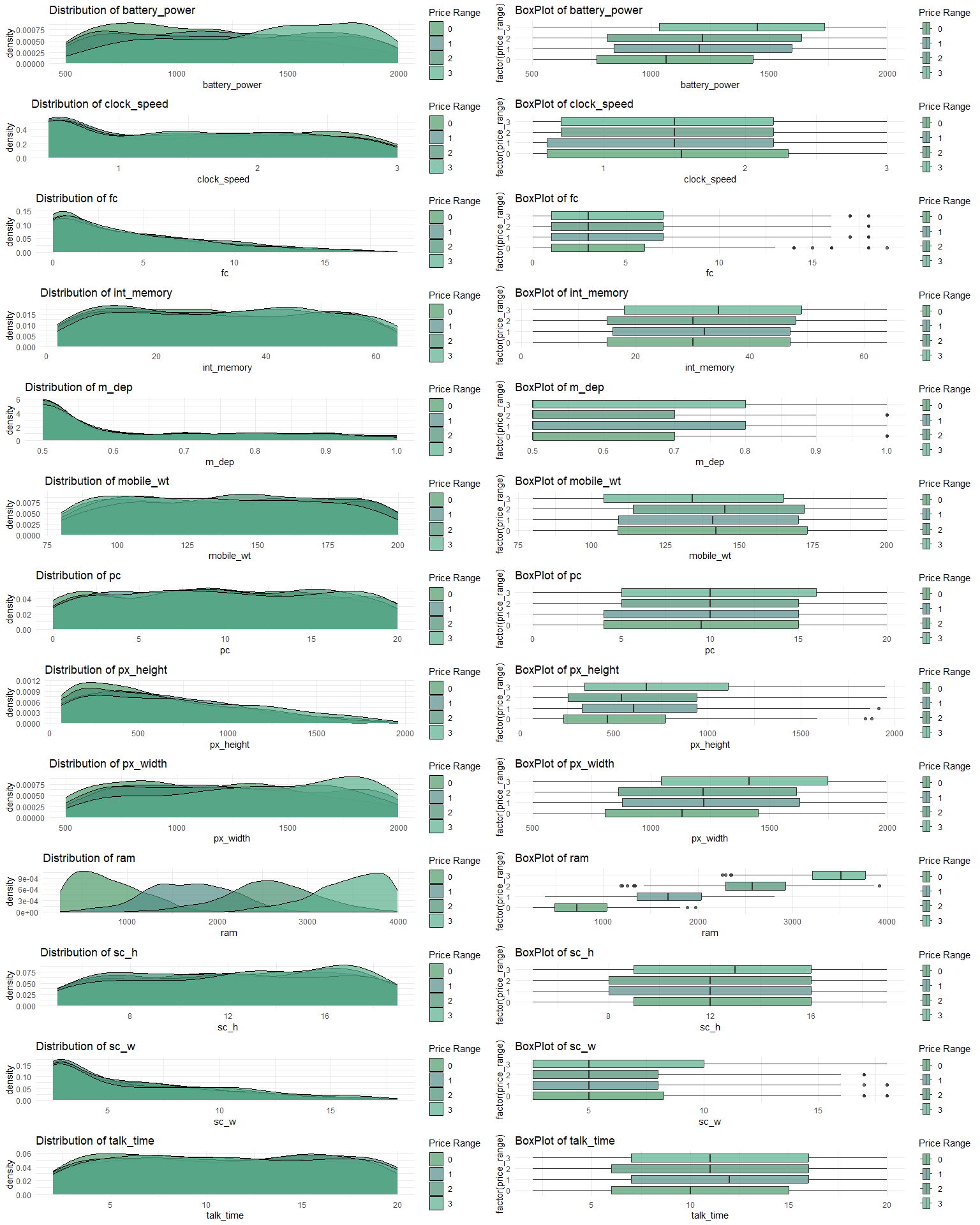
library(ggplot2)  
library(gridExtra)  
  
# Loop through categorical columns to plot counts  
plot\_list <- list()  
  
for (col in categorical\_columns) {  
 # Count plot  
 p1 <- ggplot(data, aes\_string(y = col)) +  
 geom\_bar(fill = '#4d9b68') +  
 ggtitle(paste('Count of', col)) +  
 theme\_minimal() +  
 geom\_text(stat = 'count', aes(label = ..count..), hjust = -0.1) +  
 coord\_flip()  
   
 # Count plot per price range  
 p2 <- ggplot(data, aes\_string(y = col, fill = 'factor(price\_range)')) +  
 geom\_bar(position = 'dodge') +  
 ggtitle(paste('Count of', col, 'per Price range')) +  
 theme\_minimal() +  
 geom\_text(stat = 'count', aes(label = ..count..), position = position\_dodge(width = 0.9), hjust = -0.1) +  
 scale\_fill\_manual(values = c('#4d9b68', '#538e8a', '#468e71', '#59ae8c'), name = "Price Range") +  
 coord\_flip()  
   
 # Store the plots  
 plot\_list <- append(plot\_list, list(p1, p2))  
}  
  
# Display plots in a grid layout  
grid.arrange(grobs = plot\_list, ncol = 2)



Observations: -Bluetooth: The count of the blue chart shows that mobile phones without Bluetooth have the highest frequency than the ones with Bluetooth. Moreover, the count of blue per price range shows that in the group of mobile phones without Bluetooth, the Low-cost and High-cost phones have the highest frequency, and on the other hand, in the group of mobile phones with Bluetooth, the Very high-cost phones have the highest frequency. -Dual Sim: The count of the dual\_sim chart shows that mobile phones which have Dual Sim have the highest frequency. Moreover, the count of dual\_sim per price range shows that in the group of mobile phones without Dual Sim, the High-cost and Low-cost phones have the highest frequency, and in the group of mobile phones with Dual Sim, the Very high-cost phones have the highest frequency. -4G: The count of the four\_g chart shows that mobile phones which have 4G have the highest frequency than the ones without 4G. Moreover, the count of four\_g per price range shows that in the group of mobile phones with 4G, the Medium-cost phones have the highest frequency. Number of Cores: The count of the n-cores chart shows that mobile phones containing 4 cores have the highest frequency. Moreover, the count of n-cores per price range shows that in the group of mobile phones with 4 cores, the High-cost phones have the highest frequency, and in the group of mobile phones containing 4 cores, the Medium-cost phones have the highest frequency. -3G: The count of the three\_g chart shows that mobile phones with 3G have the highest frequency than the ones without 3G. Moreover, the count of three\_g per price range shows that in the group of mobile phones with 3G, the High-cost phones have the highest frequency, and in the group of mobile phones without 3G, the Low-cost phones have the highest frequency. -Touch Screen: The count of the touch\_screen chart shows that mobile phones which have Touch Screen have the highest frequency than the ones without Touch Screen. Moreover, the count of touch\_screen per price range shows that in the group of mobile phones with Touch Screen, the Low-cost phones have the highest frequency, and in the group of mobile phones without Touch Screen, the High-cost phones have the highest frequency. -Wifi: The count of the wifi chart shows that mobile phones which have Wifi have the highest frequency than the ones without Wifi. Moreover, the count of wifi per price range shows that in the group of mobile phones with Wifi, the Very high-cost phones have the highest frequency, and in the group of mobile phones without Wifi, the Low-cost phones have the highest frequency.

**Exploring Numerical Features**

library(ggplot2)  
library(gridExtra)  
  
# Create an empty list to store plots  
plot\_list <- list()  
  
# Loop through numerical columns to plot distributions and boxplots  
for (col in numerical\_columns) {  
   
 # KDE Plot (Density Plot)  
 p1 <- ggplot(data, aes\_string(x = col, fill = "factor(price\_range)")) +  
 geom\_density(alpha = 0.7) +  
 ggtitle(paste('Distribution of', col)) +  
 scale\_fill\_manual(values = c('#4d9b68', '#538e8a', '#468e71', '#59ae8c'), name = "Price Range") +  
 theme\_minimal()  
   
 # Box Plot  
 p2 <- ggplot(data, aes\_string(y = "factor(price\_range)", x = col, fill = "factor(price\_range)")) +  
 geom\_boxplot(alpha = 0.7) +  
 ggtitle(paste('BoxPlot of', col)) +  
 scale\_fill\_manual(values = c('#4d9b68', '#538e8a', '#468e71', '#59ae8c'), name = "Price Range") +  
 theme\_minimal()  
   
 # Add plots to the list  
 plot\_list <- append(plot\_list, list(p1, p2))  
}  
  
# Arrange the plots in a grid (13 rows, 2 columns)  
grid.arrange(grobs = plot\_list, ncol = 2)



Observations: A normal distribution (with no skewness) is observed in the features of int\_memory, mobile\_wt, pc and talk time for all price ranges.

Battery Power:

Mobile phones with price ranges of 0-3 mostly have battery\_power at the range of 600-800, 700-1350, 600-900, and 1500-1900 mAh, respectively. Clock Speed:

Mobile phones with all price ranges mostly have clock\_speed at the range of 0.4-0.8. A positive skewness is observed in the distribution of clock\_speed for all price ranges. The box plot of clock\_speed indicates that all price ranges have the same median equal 1.5. Front Camera:

Mobile phones with all price ranges mostly have fc in the range of 0-2.5 megapixels. A positive skewness is observed in the distribution of fc for all price ranges. The box plot of fc indicates that all price ranges have the same median equal 3 megapixels. Internal Memory:

Mobile phones with price ranges of 0-3 mostly have int\_memory at the range of 10-30, 35-50, 13-20, and 40-50 gigabytes, respectively. Mobile Depth:

A positive skewness is observed in the distribution of m\_dep for all price ranges. Mobile phones with all price ranges mostly have m\_dep at the range of 0.5-0.55 cm. Pixel Resolution Height:

A positive skewness is observed in the distribution of px\_height for all price ranges. Mobile phones with all price ranges mostly have px\_height in the range of 65-500 pixels. Pixel Resolution Width:

Mobile phones with price ranges of 0-3 mostly have px\_width in the range of 750-900, 700-1350, 750-1200, and 1500-1700 pixels, respectively. Ram:

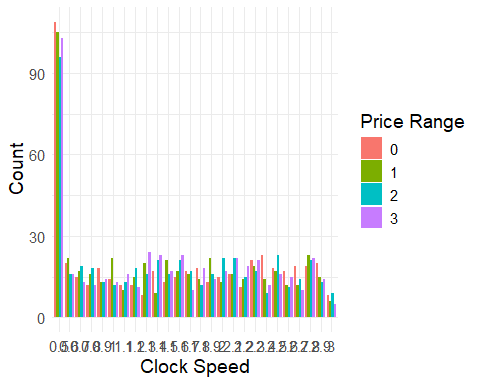
Mobile phones with price ranges of 0-3 mostly have ram at the range of 450-750, 1350-1900, 2500-2800, and 3500-3950 megabytes, respectively. Screen Width:

A positive skewness is observed in the distribution of sc\_w for all price ranges. Mobile phones with all price ranges mostly have sc\_w in the range of 2.5-4 cm. The box plot of sc\_w indicates that all price ranges have the same median equal 1.5 cm.

**Bivariate Analysis**

-Clock Speed

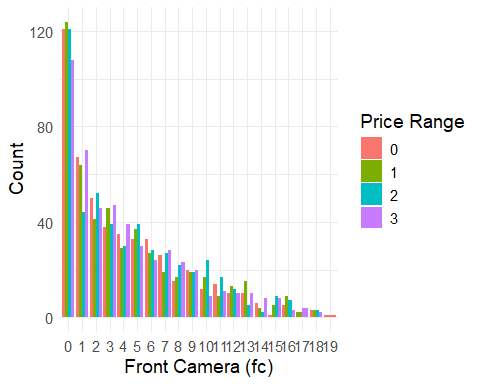
library(ggplot2)  
  
# Clock speed based on price range  
ggplot(data, aes(x = factor(clock\_speed), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "Clock Speed", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: The above chart illustrates that the phones with a clock\_speed of 0.5 contains the highest count among all mobile phones.

**Front Camera (Megapixels):**

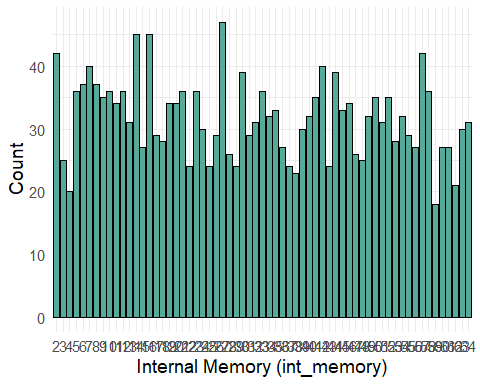
library(ggplot2)  
  
# fc based on price range  
ggplot(data, aes(x = factor(fc), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "Front Camera (fc)", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: The above chart illustrates that with the increase in the value of fc from 0-19, their count will decrease. This indicates that as the front camera becomes more powerful, the number of phones decreases.

**Internal Memory (Gigabytes):**

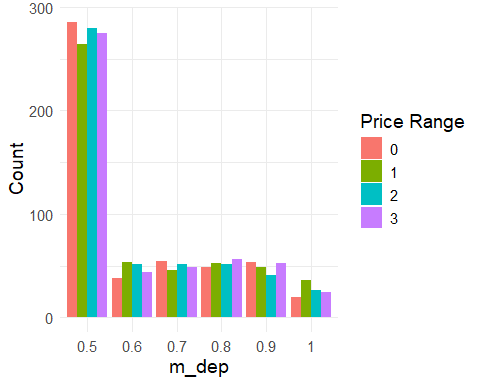
library(ggplot2)  
  
# Count of int\_memory  
ggplot(data, aes(x = factor(int\_memory))) +  
 geom\_bar(fill = "#5A9", color = "black") +  
 labs(x = "Internal Memory (int\_memory)", y = "Count") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: Mobile phones with 27 gigabytes of int\_memory with the value of 64 have the highest count among all phones.

**Mobile Depth (Cm):**

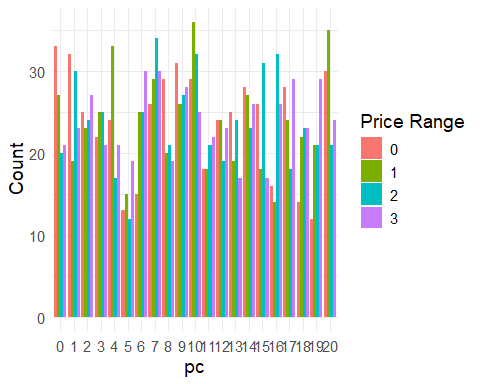
library(ggplot2)  
  
# m\_dep based on price\_range  
ggplot(data, aes(x = factor(m\_dep), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "m\_dep", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: Mobile phones with 0.5 cm of m\_dep have the highest count among all phones.

**Primary Camera (Megapixels):**

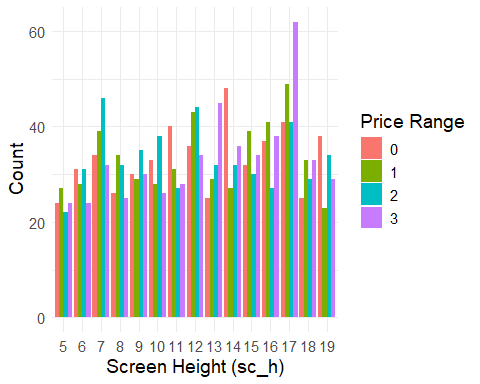
library(ggplot2)  
  
# pc based on price\_range  
ggplot(data, aes(x = factor(pc), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "pc", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: In general, mobile phones with 5 Megapixels of pc have the lowest count among all phones.

**Screen Height (Cm):**

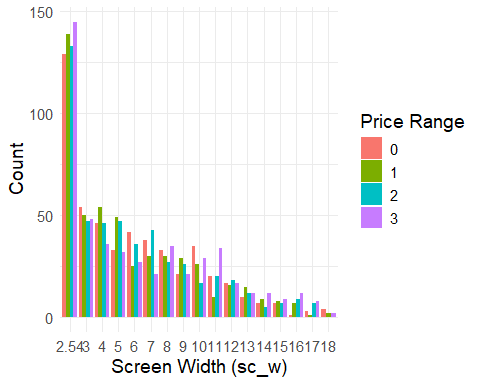
library(ggplot2)  
  
# sc\_h based on price\_range  
ggplot(data, aes(x = factor(sc\_h), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "Screen Height (sc\_h)", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: In general, mobile phones with 17 Cm of sc\_h have the highest count among all phones.

**Screen Width (Cm):**

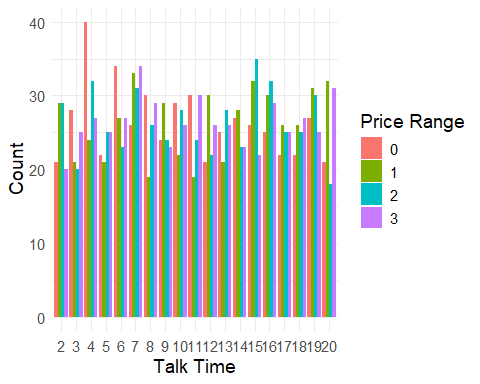
library(ggplot2)  
  
# sc\_w based on price\_range  
ggplot(data, aes(x = factor(sc\_w), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "Screen Width (sc\_w)", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: The above chart demonstrates that with the increase in the value of sc\_w from 2.54-18, their count will decrease. This indicates that as the Screen Width becomes bigger, the number of phones decreases. In general, mobile phones with 2.54 cm or 1-inch of sc\_w have the highest count among all phones.

**Talk Time:**

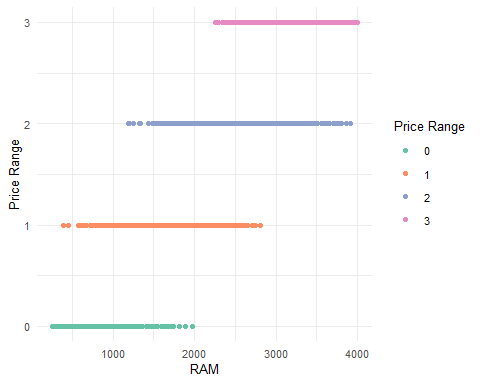
library(ggplot2)  
  
# talk\_time based on price\_range  
ggplot(data, aes(x = factor(talk\_time), fill = factor(price\_range))) +  
 geom\_bar(position = "dodge") +  
 labs(x = "Talk Time", y = "Count", fill = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 14))



Observations: The range of talk\_time varies from 2 to 20. Mobile phones with talk\_time 4 with a Low-cost price range have the highest count among all phones.

**Ram (Megabytes):**

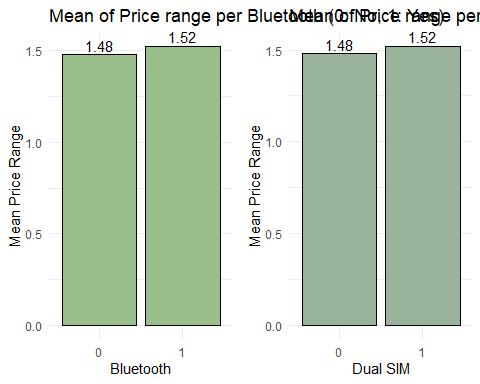
library(ggplot2)  
  
# Create scatter plot  
ggplot(data, aes(x = ram, y = price\_range, color = factor(price\_range))) +  
 geom\_point() +  
 scale\_color\_brewer(palette = "Set2") + # Adjust the palette as needed  
 labs(x = "RAM", y = "Price Range", color = "Price Range") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 10))



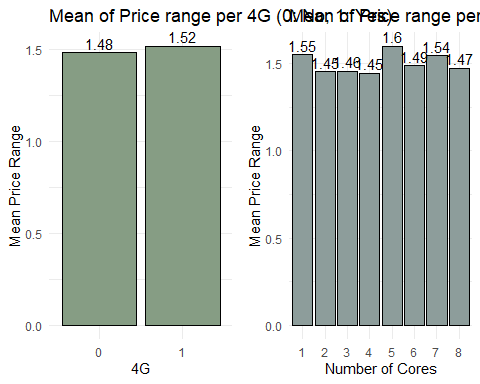
Observations: By increasing the value of ram from 256-4000 megabytes, the price range will increase.

**Mean of ‘Price range’ per each unique value of different categorical features:**

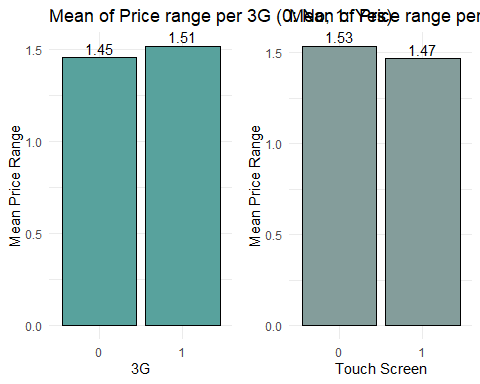
library(ggplot2)  
library(dplyr)  
  
# Plot for Bluetooth  
bluetooth\_plot <- data %>%  
 group\_by(blue) %>%  
 summarise(mean\_price = mean(price\_range)) %>%  
 ggplot(aes(x = factor(blue), y = mean\_price, fill = factor(blue))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#9bbf8a") +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per Bluetooth (0: No, 1: Yes)", x = "Bluetooth", y = "Mean Price Range") +  
 theme\_minimal() +  
 theme(legend.position = "none")  
  
# Plot for Dual SIM  
dual\_sim\_plot <- data %>%  
 group\_by(dual\_sim) %>%  
 summarise(mean\_price = mean(price\_range)) %>%  
 ggplot(aes(x = factor(dual\_sim), y = mean\_price, fill = factor(dual\_sim))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#99b49a") +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per Dual SIM (0: No, 1: Yes)", x = "Dual SIM", y = "Mean Price Range") +  
 theme\_minimal() +  
 theme(legend.position = "none")  
  
# Arrange side by side  
library(gridExtra)  
grid.arrange(bluetooth\_plot, dual\_sim\_plot, ncol = 2)



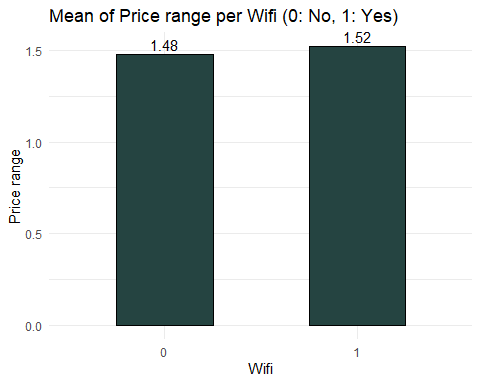
library(ggplot2)  
library(dplyr)  
library(gridExtra)  
  
# Plot for 4G  
four\_g\_plot <- data %>%  
 group\_by(four\_g) %>%  
 summarise(mean\_price = mean(price\_range)) %>%  
 ggplot(aes(x = factor(four\_g), y = mean\_price, fill = factor(four\_g))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#869d84") +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per 4G (0: No, 1: Yes)", x = "4G", y = "Mean Price Range") +  
 theme\_minimal() +  
 theme(legend.position = "none")  
  
# Plot for Number of Cores  
n\_cores\_plot <- data %>%  
 group\_by(n\_cores) %>%  
 summarise(mean\_price = mean(price\_range)) %>%  
 ggplot(aes(x = factor(n\_cores), y = mean\_price, fill = factor(n\_cores))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#8d9d9b") +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per Number of cores", x = "Number of Cores", y = "Mean Price Range") +  
 theme\_minimal() +  
 theme(legend.position = "none")  
  
# Arrange side by side  
grid.arrange(four\_g\_plot, n\_cores\_plot, ncol = 2)



library(ggplot2)  
library(dplyr)  
library(gridExtra)  
  
# Plot for 3G  
three\_g\_plot <- data %>%  
 group\_by(three\_g) %>%  
 summarise(mean\_price = mean(price\_range)) %>%  
 ggplot(aes(x = factor(three\_g), y = mean\_price, fill = factor(three\_g))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#58a29d") +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per 3G (0: No, 1: Yes)", x = "3G", y = "Mean Price Range") +  
 theme\_minimal() +  
 theme(legend.position = "none")  
  
# Plot for Touch Screen  
touch\_screen\_plot <- data %>%  
 group\_by(touch\_screen) %>%  
 summarise(mean\_price = mean(price\_range)) %>%  
 ggplot(aes(x = factor(touch\_screen), y = mean\_price, fill = factor(touch\_screen))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#849d9b") +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per Touch Screen (0: No, 1: Yes)", x = "Touch Screen", y = "Mean Price Range") +  
 theme\_minimal() +  
 theme(legend.position = "none")  
  
# Arrange side by side  
grid.arrange(three\_g\_plot, touch\_screen\_plot, ncol = 2)



library(ggplot2)  
library(dplyr)  
  
# Group by wifi and calculate mean of price\_range  
wifi\_mean <- data %>%  
 group\_by(wifi) %>%  
 summarise(mean\_price = mean(price\_range))  
  
# Plot  
ggplot(wifi\_mean, aes(x = factor(wifi), y = mean\_price, fill = factor(wifi))) +  
 geom\_bar(stat = "identity", color = "black", fill = "#254441", width = 0.5) +  
 geom\_text(aes(label = round(mean\_price, 2)), vjust = -0.3) +  
 labs(title = "Mean of Price range per Wifi (0: No, 1: Yes)",   
 x = "Wifi", y = "Price range") +  
 theme\_minimal() +  
 theme(legend.position = "none", axis.text.x = element\_text(angle = 0, hjust = 0.5))

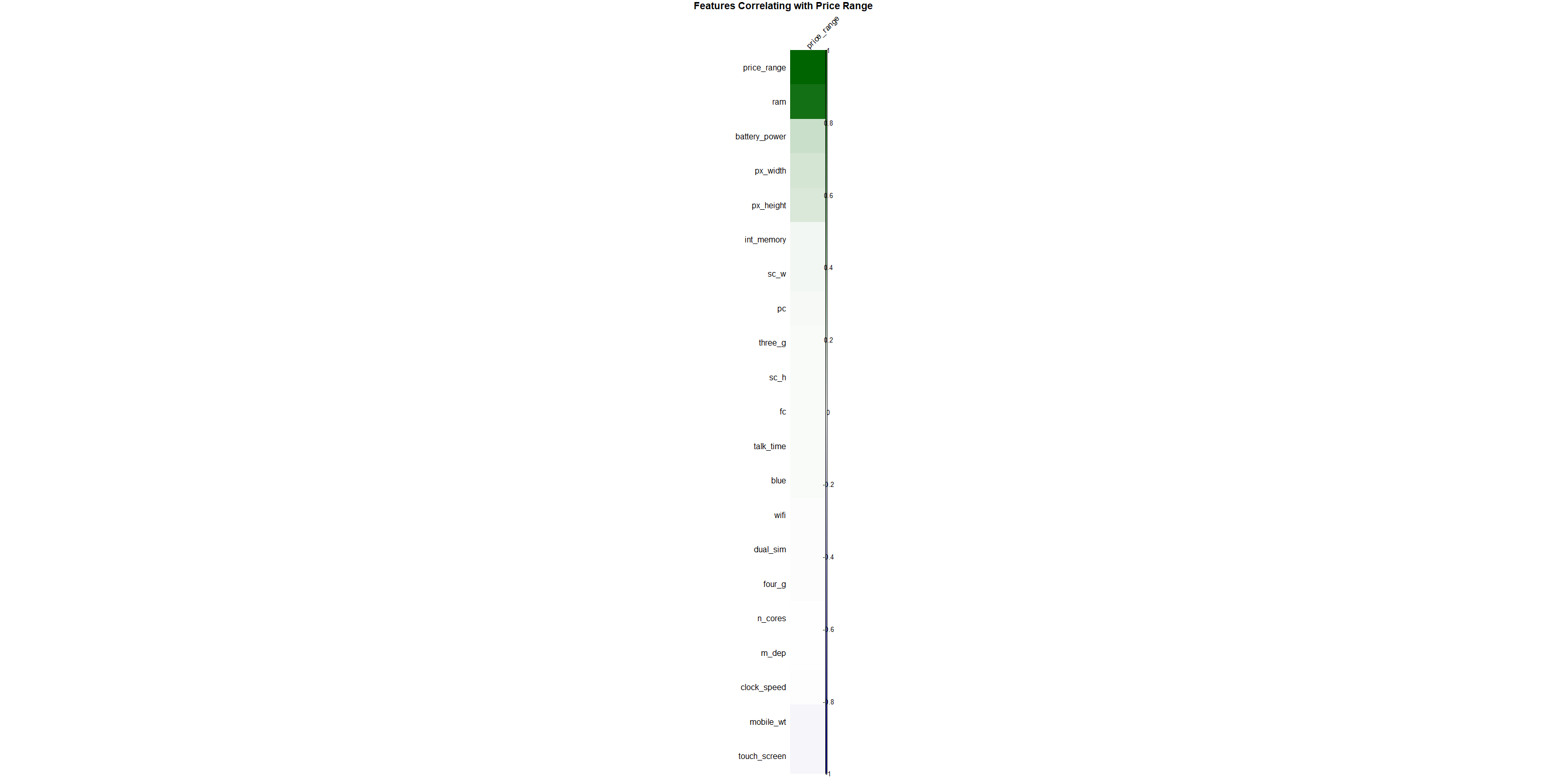


Observations: The Mean of Price range per Number of cores shows that mobile phones with 5 cores have the highest mean of the price range.

**Correlation**

Our target is Price Range. So, we should check how each attribute correlates with the Price Range variable. We can do it as follows:

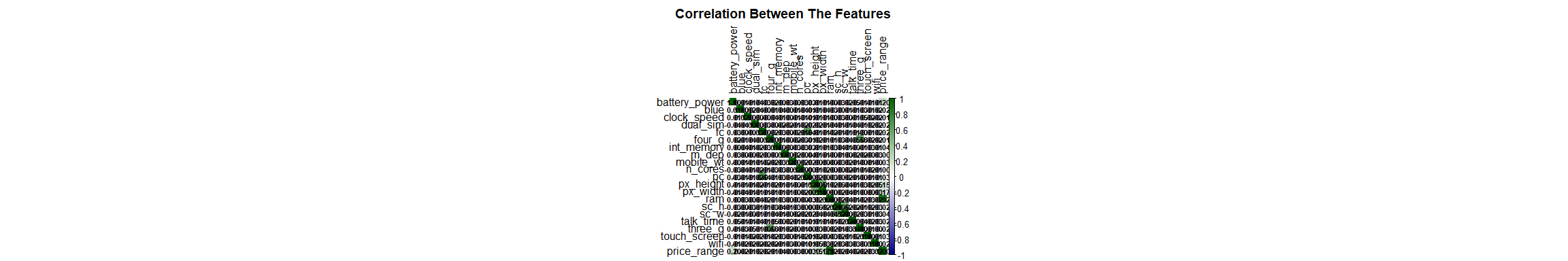
library(ggplot2)  
library(corrplot)  
  
# Calculate correlation matrix  
cor\_matrix <- cor(data)  
  
# Extract correlations with price\_range and sort  
price\_corr <- cor\_matrix[, "price\_range", drop = FALSE]  
price\_corr <- price\_corr[order(-price\_corr[, "price\_range"]), , drop = FALSE]  
  
# Plot heatmap  
corrplot(price\_corr, method = "color",   
 col = colorRampPalette(c("darkblue", "white", "darkgreen"))(200),  
 tl.col = "black", tl.srt = 45,   
 title = "Features Correlating with Price Range",  
 mar = c(0, 0, 1, 0))



Interpretation: The correlation coefficient ranges from -1 to +1. When it is close to +1, this signifies that there is a strong positive correlation. So, we can see that there is a positive correlation between Price Range and ram, Price Range and battery\_power, and Price Range and px\_width. When it is close to -1, it means that there is a strong negative correlation. When it is close to 0, it means that there is no correlation. We can see that most of the variables except clock\_speed, mobile\_wt and touch\_screen are positively correlated with the target.

**Correlation Between The Features**

library(ggplot2)  
library(corrplot)  
  
# Calculate correlation matrix  
cor\_matrix <- cor(data)  
  
# Plot heatmap  
corrplot(cor\_matrix, method = "color", col = colorRampPalette(c("darkblue", "white", "darkgreen"))(200),  
 tl.col = "black", tl.srt = 90, addCoef.col = "black", number.cex = 0.7,  
 title = "Correlation Between The Features", mar = c(0, 0, 2, 0))



Interpretation: There is a strong correlation between ram and price\_range. In addition, the heatmap above indicates a moderate correlation between 4G and 3G, fc and pc, px\_height and px\_width, and sc\_h and sc\_w.

## Model Building

**Min-max normalization**

# Formula  
normalize <- function(x) {  
 return((x - min(x)) / (max(x) - min(x)))  
}  
  
# Normalize relevant numerical columns in the mobile features dataset  
data[, c("battery\_power", "clock\_speed", "fc", "int\_memory", "m\_dep",   
 "mobile\_wt", "n\_cores", "pc", "px\_height", "px\_width",   
 "ram", "sc\_h", "sc\_w", "talk\_time")] <-   
 apply(data[, c("battery\_power", "clock\_speed", "fc", "int\_memory", "m\_dep",   
 "mobile\_wt", "n\_cores", "pc", "px\_height", "px\_width",   
 "ram", "sc\_h", "sc\_w", "talk\_time")], 2, normalize)  
  
# Show first rows  
head(data)

## battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory m\_dep  
## 1 0.22778891 0 0.68 0 0.05263158 0 0.08064516 0.2  
## 2 0.34736139 1 0.00 1 0.00000000 1 0.82258065 0.4  
## 3 0.04141617 1 0.00 1 0.10526316 1 0.62903226 0.8  
## 4 0.07615230 1 0.80 0 0.00000000 0 0.12903226 0.6  
## 5 0.88176353 1 0.28 0 0.68421053 1 0.67741935 0.2  
## 6 0.90714763 0 0.00 1 0.15789474 0 0.32258065 0.4  
## mobile\_wt n\_cores pc px\_height px\_width ram sc\_h sc\_w  
## 1 0.9000000 0.1428571 0.10 0.0000000 0.1708945 0.6127739 0.2857143 0.2884864  
## 2 0.4666667 0.2857143 0.30 0.4432718 0.9933244 0.6346873 0.8571429 0.0297542  
## 3 0.5416667 0.5714286 0.30 0.6321900 0.8117490 0.6272047 0.4285714 0.0000000  
## 4 0.4250000 0.7142857 0.45 0.6073879 0.8584780 0.6715660 0.7857143 0.3531695  
## 5 0.5083333 0.1428571 0.70 0.6031662 0.4753004 0.3086585 0.2142857 0.0000000  
## 6 0.7000000 0.0000000 0.35 0.4955145 0.7703605 0.2167290 0.8571429 0.0000000  
## talk\_time three\_g touch\_screen wifi price\_range  
## 1 0.9444444 0 0 1 1  
## 2 0.2777778 1 1 0 2  
## 3 0.3888889 1 1 0 2  
## 4 0.5000000 1 0 0 2  
## 5 0.7222222 1 1 0 1  
## 6 0.4444444 1 0 0 1

All the rows has been normalized

## KNN Modelling

**Split the data into training and testing sets:** **Train KNN**

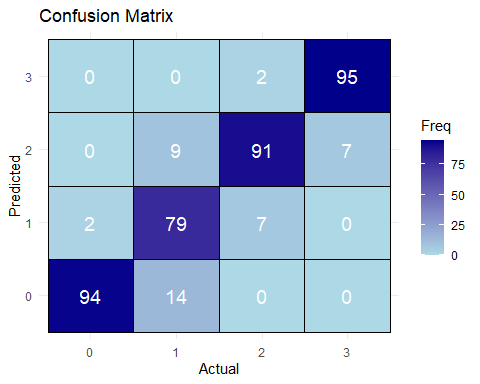
# Polynomial Features (up to degree 2)  
data$ram\_squared <- data$ram^2  
data$battery\_power\_squared <- data$battery\_power^2  
  
# Interaction Terms  
data$ram\_battery\_interaction <- data$ram \* data$battery\_power  
data$screen\_area <- data$px\_height \* data$px\_width  
  
# Scale the new features along with the original ones  
scaled\_data <- scale(data[, -ncol(data)]) # Exclude price\_range if it's the last column  
labels <- data$price\_range  
  
# Train/Test Split  
set.seed(42)  
train\_idx <- sample(1:nrow(data), 0.8 \* nrow(data))  
train\_features <- scaled\_data[train\_idx, ]  
test\_features <- scaled\_data[-train\_idx, ]  
train\_labels <- labels[train\_idx]  
test\_labels <- labels[-train\_idx]  
  
# Train KNN with new features  
library(class)  
k\_value <- 41  
knn\_pred <- knn(train = train\_features,   
 test = test\_features,   
 cl = train\_labels,   
 k = k\_value)  
  
# Evaluate  
library(caret)  
# Ensure labels and predictions are factors with the same levels  
knn\_pred <- factor(knn\_pred, levels = levels(factor(train\_labels)))  
test\_labels <- factor(test\_labels, levels = levels(factor(train\_labels)))  
  
# Evaluate performance  
conf\_matrix <- confusionMatrix(knn\_pred, test\_labels)  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2 3  
## 0 94 14 0 0  
## 1 2 79 7 0  
## 2 0 9 91 7  
## 3 0 0 2 95  
##   
## Overall Statistics  
##   
## Accuracy : 0.8975   
## 95% CI : (0.8635, 0.9254)  
## No Information Rate : 0.255   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8634   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3  
## Sensitivity 0.9792 0.7745 0.9100 0.9314  
## Specificity 0.9539 0.9698 0.9467 0.9933  
## Pos Pred Value 0.8704 0.8977 0.8505 0.9794  
## Neg Pred Value 0.9932 0.9263 0.9693 0.9769  
## Prevalence 0.2400 0.2550 0.2500 0.2550  
## Detection Rate 0.2350 0.1975 0.2275 0.2375  
## Detection Prevalence 0.2700 0.2200 0.2675 0.2425  
## Balanced Accuracy 0.9666 0.8722 0.9283 0.9623

##Evaluation

**COnfusion Matrix**

# Load necessary libraries  
library(caret)  
library(ggplot2)  
  
# Create the confusion matrix  
conf\_matrix <- confusionMatrix(knn\_pred, test\_labels)  
  
# Extract the matrix  
cm\_table <- as.data.frame(conf\_matrix$table)  
colnames(cm\_table) <- c("Prediction", "Reference", "Freq")  
  
# Plot the confusion matrix  
ggplot(cm\_table, aes(x = Reference, y = Prediction, fill = Freq)) +  
 geom\_tile(color = "black") +  
 geom\_text(aes(label = Freq), color = "white", size = 5) +  
 scale\_fill\_gradient(low = "lightblue", high = "darkblue") +  
 labs(title = "Confusion Matrix", x = "Actual", y = "Predicted") +  
 theme\_minimal()



**Model Metrics**

# Load required libraries  
library(caret)  
library(knitr)  
  
# Compute confusion matrix  
conf\_matrix <- confusionMatrix(knn\_pred, test\_labels)  
  
# Extract values from the confusion matrix  
cm\_table <- conf\_matrix$table  
  
# Calculate overall metrics  
accuracy <- conf\_matrix$overall['Accuracy']  
kappa <- conf\_matrix$overall['Kappa']  
  
# Calculate per-class metrics  
class\_metrics <- data.frame(  
 Class = levels(test\_labels),  
 Sensitivity = conf\_matrix$byClass[, "Sensitivity"],  
 Specificity = conf\_matrix$byClass[, "Specificity"],  
 Precision = conf\_matrix$byClass[, "Pos Pred Value"],  
 Recall = conf\_matrix$byClass[, "Sensitivity"],  
 F1\_Score = 2 \* (conf\_matrix$byClass[, "Pos Pred Value"] \* conf\_matrix$byClass[, "Sensitivity"]) /  
 (conf\_matrix$byClass[, "Pos Pred Value"] + conf\_matrix$byClass[, "Sensitivity"])  
)  
  
# Add overall metrics  
overall\_metrics <- data.frame(  
 Metric = c("Accuracy", "Kappa"),  
 Value = c(accuracy, kappa)  
)  
  
# Display metrics  
cat("## Overall Performance Metrics\n")

## ## Overall Performance Metrics

print(kable(overall\_metrics, format = "markdown", digits = 4))

##   
##   
## | |Metric | Value|  
## |:--------|:--------|------:|  
## |Accuracy |Accuracy | 0.8975|  
## |Kappa |Kappa | 0.8634|

cat("\n## Per-Class Performance Metrics\n")

##   
## ## Per-Class Performance Metrics

print(kable(class\_metrics, format = "markdown", digits = 4))

##   
##   
## | |Class | Sensitivity| Specificity| Precision| Recall| F1\_Score|  
## |:--------|:-----|-----------:|-----------:|---------:|------:|--------:|  
## |Class: 0 |0 | 0.9792| 0.9539| 0.8704| 0.9792| 0.9216|  
## |Class: 1 |1 | 0.7745| 0.9698| 0.8977| 0.7745| 0.8316|  
## |Class: 2 |2 | 0.9100| 0.9467| 0.8505| 0.9100| 0.8792|  
## |Class: 3 |3 | 0.9314| 0.9933| 0.9794| 0.9314| 0.9548|

Kappa: 0.8634 — Indicates strong agreement between predicted and actual classes. Class-wise performance: Class 0: Sensitivity is 0.9792, meaning it correctly identifies 97.92% of class 0 cases. Class 1: Sensitivity is 0.7745, which is slightly lower, meaning some class 1 cases are misclassified. Class 2: Improved to 0.9100 sensitivity, showing good performance. Class 3: Sensitivity is 0.9314, meaning it correctly identifies 93.14% of class 3 cases.

## Conclusion

**Conclusion: KNN Model Performance Analysis**

The KNN model achieved **89.75%** accuracy, demonstrating strong multi-class classification capability. Its high balanced accuracy indicates robustness against class imbalances.

Key metrics include: - **Sensitivity**: 77.45%-97.92% (strongest for Classes 0 and 3) - **Specificity**: Consistently high, peaking at 99.33% for Class 3 - **Precision**: 85.05%-97.94% (strongest for Classes 0 and 3) - **Kappa**: 0.8634, showing strong prediction reliability