Classification of Mobile Prices Using Machine Learning

Aung Nyein

May 18, 2024 k

Table of Content

1. INTRODUCTION

In today's rapidly evolving smartphone market, it is progressively getting more difficult to decide on what phones for consumers to buy. With all these new budget smartphones offering features that were once considered premium and all these huge flagship companies offering the same features for a way higher price, it is really hard to navigate exactly what you are paying for. Accounting for the wide range of specifications in these phones, finding the real features that drive the final price listing of phones will help buyers make informed decisions. Consumers can personally assess whether a phone's price aligns with its capabilities, and help them avoid paying a ton of money for unnecessary features. It has been found that "some manufactures inflate prices based on brand reputation rather than hardware performance" . And finding the Machine learning classification models that accurately classify phones into their appropriate price will also help manufacturers optimize their product listings. The point of these models are not for predicting the exact price but to discern the crucial features that influence a phone's price and sort into phones the appropriate price range. I will employ three popular machine learning algorithms ; K-Nearest Neighbors(KNN), Naive Bayes, and Decision Tree to build classification models. These models analyze key features like camera specs, battery capacity, screen, internal hardware, internal storage to sort different phones. And by comparing the performance of these algorithms, we can determine the most effective approach for price classification in the Mobile phone industry.
Objectives:

- Exploring and Preprocessing Data
- Build different classification models to predict price range for phones
- Price range prediction for unseen data and Model Comparison.

2. DATA

2.1 Detailed description of the data

This dataset is from Kaggle, and it has 2000 different phones with a collection of features that makes them distinct and the following chat contains all of the different things that we are gonna be looking into when fitting a classification model;

```
+----------------+------------------------------+  +----------------+-----------------------------------+
| Feature        | Description                  |  | Feature        | Description                       |
+----------------+------------------------------+  +----------------+-----------------------------------+
| battery_power  | battery Capacity (mAh)       |  | ram            | RAM(MB)                           |
| blue           | Has Bluetooth                |  | sc_h           | screen height(cm)                 |
| clock_speed    | Processor speed (GHz)        |  | sc_w           | screen width(cm)                  |
| dual_sim       | Has dual SIM support         |  | talk_time      | Longest battery life per charge   |
| fc             | Front Camera megapixels      |  | three_g        | Has 3G                            |
| four_g         | Has 4G                       |  | touch_screen   | Has touch screen or not           |
| int_memory     | Internal Memory (GB)         |  | wifi           | Has wifi or not                   |
| m_dep          | Mobile thickness (cm)        |  +----------------+-----------------------------------+
| mobile_wt      | Weight of phone (grams)      |  | price_range    | Response: 0(low),1(medium),       |
| n_cores        | Number of processor cores    |  |                |           2(high),3(very high)    |
| pc             | Primary Camera megapixels    |  +----------------+-----------------------------------+
| px_width       | Pixel Resolution Width       |
+----------------+------------------------------+
```

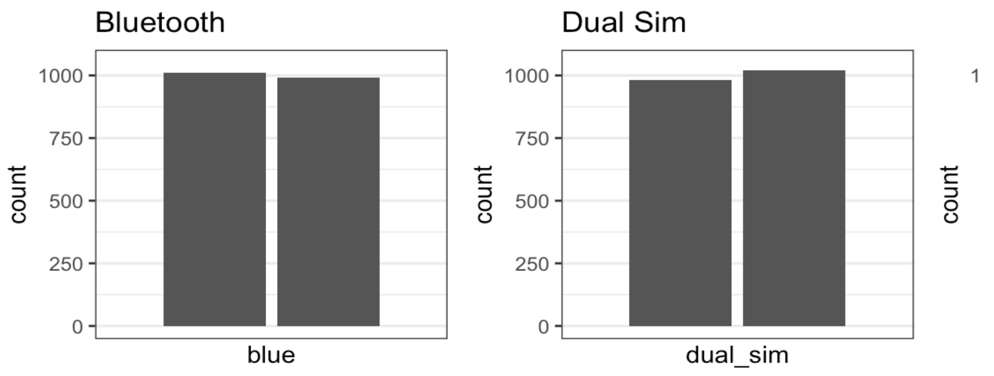
2.2 Univariate Data Analysis

As I mentioned we are going to be looking at internal hardware specs like how many processors the phone has, the processor clocking speed, how much ram and has and things within that scope. Next are add the non-essential features like Screen and Camera specs ; like the screen width and height and the camera megapixels. And for the response variable we are going to be classifying phones into 4 different price ranges ;

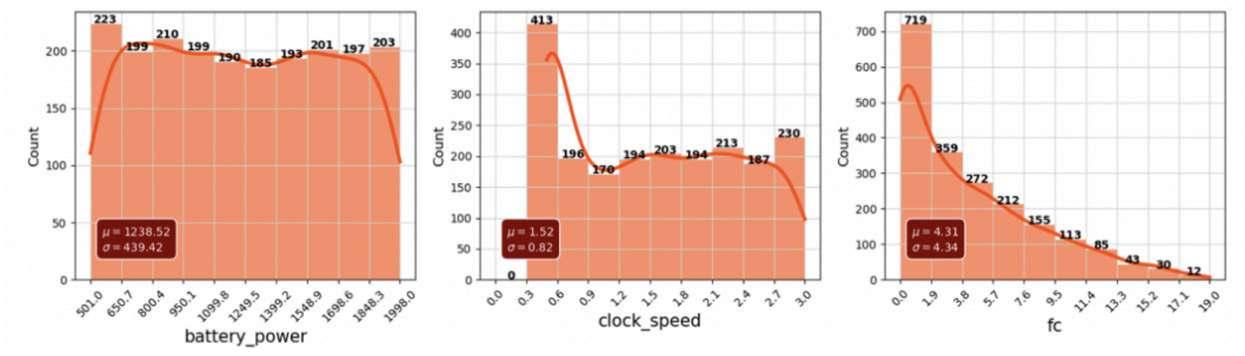
Class	0 (Low price)	1 (Medium price)	2 (High price)	3 (Very high price)
Phone Price	$300 dollars or less	$300 - $600	$600 - $900	$900 or higher

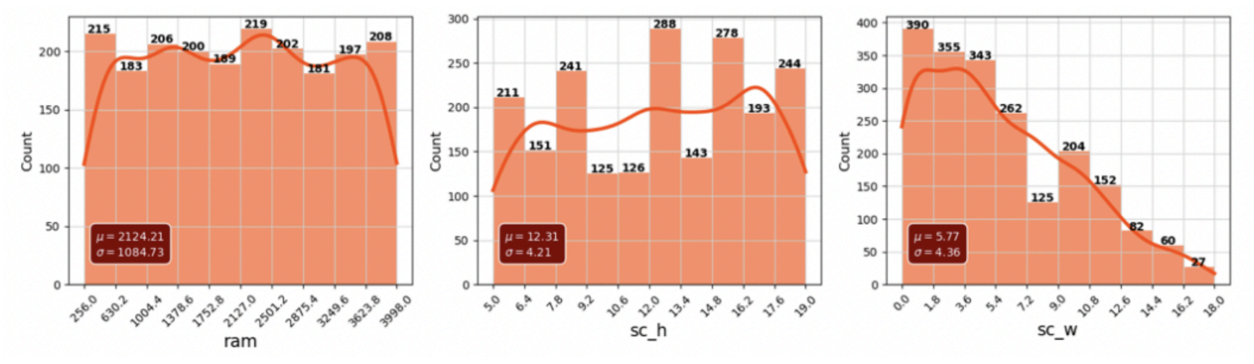
To better understand the data, we perform univariate analysis for continuous and categorical features separately. And first and foremost, looking at the distribution of the response variable I found that the dataset is completely balanced. That there are exactly 500 phones in each of the different price range classes and so we don't have to go out of our way to over or under sampling for the training data. Having a balanced data set is great because when we fit the classification models, there is no bias towards a specific dominant class and there is no overfitting in the model. There are no missing values in any rows of the dataset.

Now starting with the categorical variable analysis, the mobile phones in the dataset have almost the same frequency in most of the categorical terms;



For example, the mobile phones in the dataset have almost the same frequency in terms of having or not having bluetooth as well as supporting and not supporting dual sim cards. Since all the categorical features in the dataset are represented in 0 and 1, we do not have to dummy encode the data set columns. For all the continuous numerical features, everything seems to be ordered as well for the most part. However I noticed that for the variables px_height (Pixel Resolution Height) and sc_W(Screen Width of Phone), there was a phone that had observations that were close to 0 which would be considered outliers in comparison phones. This is potential noise that might skew the classification model fitting so I removed around 2 phones to make the distribution of these features less skewed. There were two entries in “pixel Height” that had values of 0 which were unacceptable



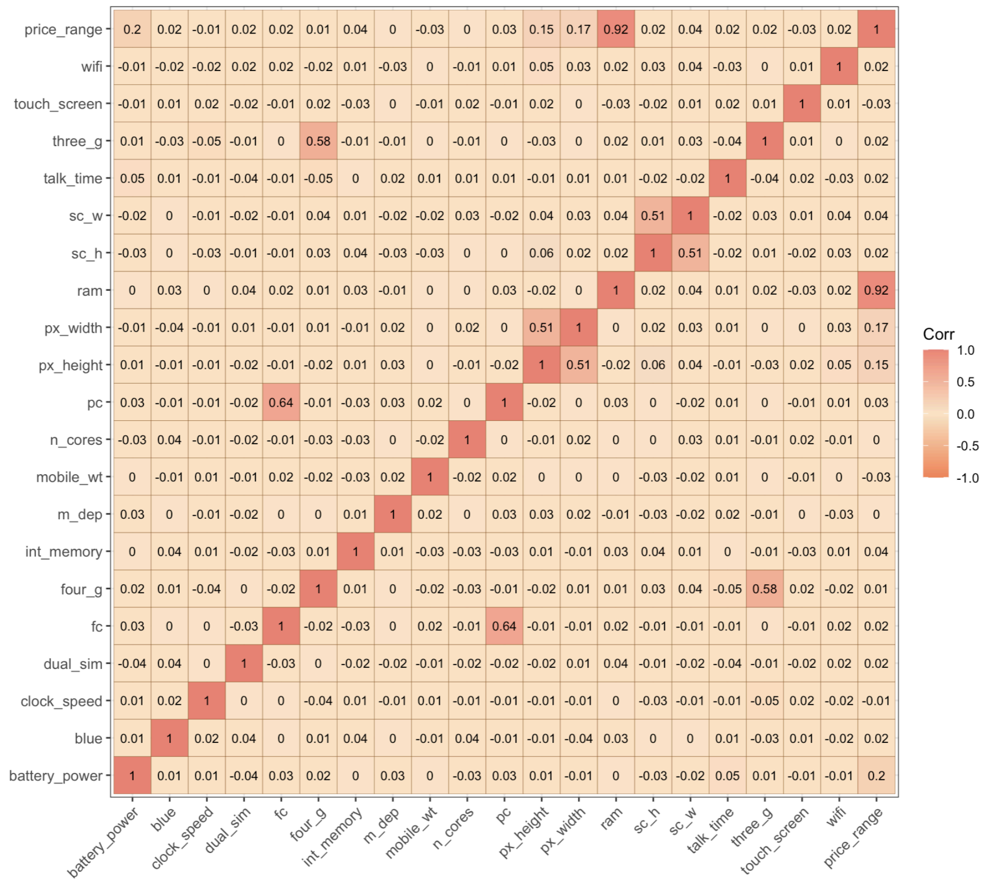


3. Dimensionality Reduction

Now we will look into the real features that work to drive the final price through two different methods to see if we reached the same conclusion. We are working on trying to reduce the number of features in the data while preserving the key essential information that helps us craft a great classification model for predicting phone price ranges. Overall by getting rid of o the irrelevant variables, the models become more efficient, interpretable, and robust while maintaining an efficient predictive performance.

3.1 Correlation Map

First I made a correlation map with all the variables of interest because it helps us identify the relationships between variables in the dataset. It quickly reveals the potential linear associations between pairs of variables and to deal with the redundancy in order to avoid multicollinearity in models all in a visually intuitive way. It also works to forecast if one variable is using data from another. And within the scope of mobile phones, there often exists strong correlation between features and observing how they correlate with the response variable can also help us narrow down the real features that are correlated with the final price. Looking at the correlation map below, I found that Ram , battery power, the pixel width of the screen , pixel height , screen width, and internal memory has the highest correlation with price range(the variable of interest).



And this shows us higher pixel counts and how much ram the phone has contributed to a higher pricing in the final phone. We also noticed; pc and fc, which represent the primary and front camera in pixels respectively. 3G and 4G, which represent the generation of the mobile phone respectively. Px width and px height, which represent the pixel width and height respectively.

3.2 LASSO shrinkage

Lasso regression is a method used in regression analysis, variable selection and regularization. It helps to remove unnecessary features to prevent overfitting. In Lasso a penalty term is added to the OLS equation; it is a tuning parameter that controls the strength of the penalty. This aligns with our goal of feature selection because lasso shrinks the coefficients of less important features to zero.

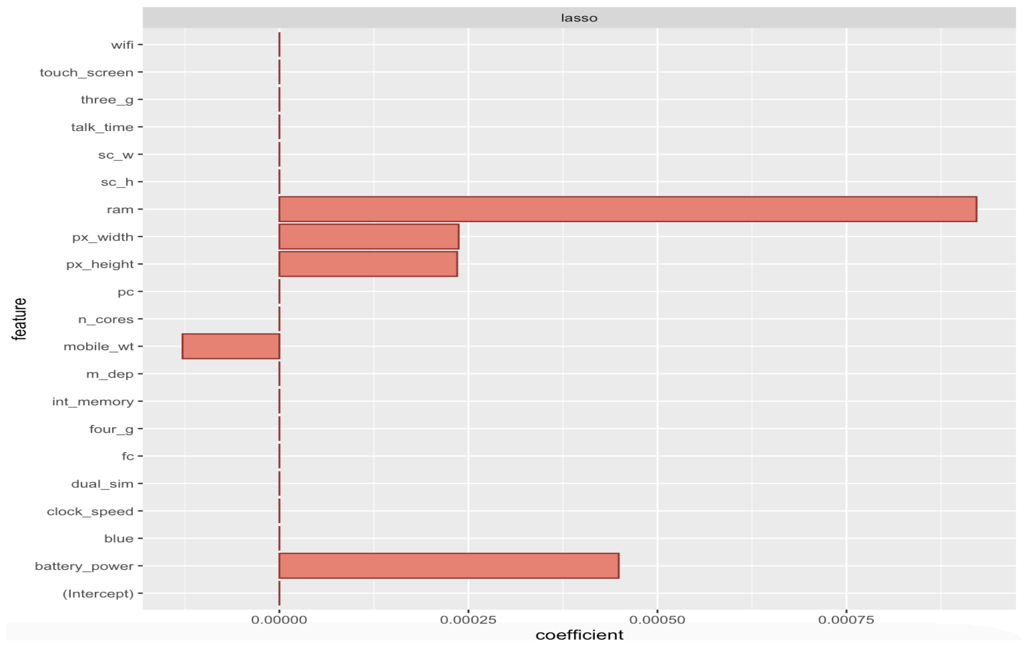


Figure 3. Lasso Shrinkage output.

Based on figure 3, aside from Ram, Px_width, px_height, mobile_wt, and battery power, all the other features shrink to zero. I also found that the best lambda for the penalty term was 0.0272. And so by observing both the heat map and Lasso Shrinkage, I found that ram, pixel width and height, and battery power were the four recurring important features for the prediction of price ranges. And so these are the only features that I will fit into the classification models.

4. Model Selection

This section provides an overview of the prediction models used in this study to classify the pricing of different mobile phones based on their features. Before we begin we will be splitting our dataset into subsets; one for training/ fitting the data and one for testing the fitted model to see its performance by means of cross validation.

4.1. Naive Bayes

Naive Bayes is a classification algorithm that uses the feature probability to predict which category a data point belongs to. The classifier uses the Bayes' Theorem to classify data based on the probabilities of the different classes given the features of the data. After fitting the classification mode to the data I applied it on to the test data portion to cast predictions and compared it to the actual classes. From the results, we see that the Naive Bayes works to correctly classify the correct price ranges for the test phones 80% of the time. While this works relatively well, it could be more accurate. This reason why it might not have performed well is because the naive bayes assumed that each of the features are independent but based on the correlation map there was a correlation between px_width and px_height and that might have affected but overall it does pretty well with data that we have.

4.2. KNN

The KNN algorithm is a non parametric method because it does not make any assumptions regarding underlying data. It essentially compares pre-existing data points and compares it to new cases to correctly categorize them into their appropriate classes. The classification of data points is based on how their neighbours are classified. In the case of this research, we will look at the neighboring phones in the data sheet that have almost the same spec to classify them in the right price ranges.

# of K	Accuracy	Lowest Precision
3	0.9593	Class 2 : 0.9444
5	0.9756	Class 3 : 0.9630
7	0.9512	Class 3 : 0.8889
9	0.8889	Class 2 : 0.8889
11	0.9106	Class 1 : 0.8333
13	0.9024	Class 1 : 0.8000
15	0.8780	Class 3 : 0.7778

In the case of KNN, I did the same cross-validation method to see how well the model performed. And I found that the model that looked at 5 neighboring data points gave us the best classification accuracy for the test phones.

4.3. Decision Tree and Random Forest

The Decision tree, as its name suggests, is a tree-like model where an input is processed through a series of decisions based on the features, that ultimately leads to a prediction. They are great because out of all the previous models the trees are very easy to understand. It has a bunch of branches that split each of the relevant features at a specific threshold until it reaches a specific class of the dependent variable called the leaf nodes.

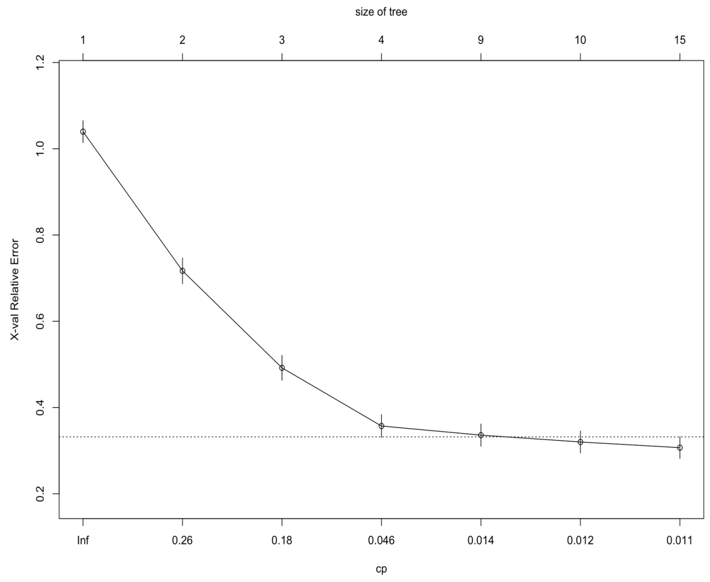


Figure 4. The size of the tree relative to Cross-validation Error

After fitting the data, I had the whole fitted tree but in order to avoid overfitting I looked into pruning the tree by means of cross-validation error and relative error. And I found that anything past the forth split was not necessary, and did not do much to improve the overall prediction accuracy of the model and so I was able to find a subtree that performed well; This confusion matrix shows us that the tree correctly predicted the phones' price range 87% of the time. And so accounting for all the different classification models that we looked at , KNN by far performed the best for the classification of mobile phone prices. Then moving on, random forest is an ensemble learning method that builds multiple decision trees during the training prices and combines all their predictions to improve accuracy. Trains each tree on a random subset of data then averages it out across all the trees for robustness . This model is great because it does very well with overfitting. Then looking at the confusion matrix, we see that, for most classes there were high counts for strong accuracy. And with an over accuracy of 93.7 % the model is reliably precise.

4.4. Multinomial Regression model and

The multinomial logistics regression extends binary logistics regression to handle categorical dependent variables with more than two outcomes. It is well suited for the questions we are trying to answer because it models the probability of each class in the face of both continuous and discrete predictors. Multinomial predicted probabilities for each of the price ranges 0, 1, 2, and 3. In this case it treats class 0 as the reference and works to estimate the different coefficients of the predictors for each of the response classes. Each of these coefficients show how each predictor affects the log-odds of a phone being in a class with respect to the reference class. And as expected, we had difference coefficients for each of the different classes;

$$\log(\frac{\pi_1}{\pi_0}) = 86.09525 - 86.09525 + 0.03313 \cdot ram + 0.01969 \cdot battery\_power + 0.01319 \cdot px\_height + 0.01146 \cdot px\_width$$

The logs-odds equation comparing class 1 to class 0. And we had different ones for each of the classes. Then using those equations we predicted the test phones and compared to the actual results. It gave a really good accuracy of 0.9674. These models can correctly classify phones into their price ranges 96 percent of the time. With this, we have fitted a wide range of classification models and evaluated how well they did for further prediction.

5. CONCLUSION

In this study, we explored the key factors influencing phone pricing and developed a Machine learning classification model to accurately classify phones into their respective price ranges. Through data analysis we have found that hardware capabilities and some aspect of the display on a phone plays a crucial role in the final price of phones in the market. Among the classification models tested—K-Nearest Neighbors (KNN), Naive Bayes, and Decision Tree, Random Forest, and Multinomial logistic regression, the KNN algorithm demonstrated the highest accuracy in predicting price ranges. These insights not only help consumers make informed choices by understanding what truly drives smartphone pricing but also provide manufacturers with valuable feedback on the features that justify different price points. When conducting this research, I noticed that a lot of the features in the data did not drive the final pricing and believe that we need to look at the features that actually reflect the mobile smartphones that we have today. The pool of phones that we had in out dataset was from 2018 and as you know the mobile phone market is highly competitive, and the standard is constantly rising. With features such as bluetooth and touch screen being the bare minimum in order to have a place in the market. And so I want to look into what brand the phone is because nowadays the deciding factor for phones is its brand, with these monopoly companies dominating the field like samsung and apple. Next I also want to look at the refresh rate of the screen, because phone use is not beyond the scope of just everyday use, and for a lot of people its primary use is for playing video games. So a lot of younger kids are looking into how responsive the screen is and how good the screen looks. It is features like these that reflect the constant evolving market of mobile phones. So I would like to work with a list of features that reflects today's phone market and see what actually drive its final price.

Code:

```
library(data.table)
library(ggplot2)
library(dplyr)
library(gridExtra)
library(glmnet)
library(ggcorrplot)
library(caTools)
library(rpart)
library(caret)
library(rpart.plot)
library(e1071)
library(caTools)
library(class)
library(nnet)
library(randomForest)

tr_data <- read.csv("~/Downloads/train.csv")


##correlation map
corr <- round(cor(tr_data), 8)
print(ggcorrplot(
  corr,
  method = "square",      # or "circle" for circular markers
  type = "full",          # shows full matrix (use "lower" or "upper" for half)
  colors = c("coral", "bisque", "salmon"),  # Brown -> White -> Orange
  outline.color = "darkorange4",  # border color of tiles
  ggtheme = theme_bw(),    # clean theme
  lab = TRUE,             # display correlation coefficients
  lab_size = 3,           # size of correlation text
  tl.cex = 10            # size of variable labels
))
```

```
## Feature Selection

x = model.matrix(price_range~., tr_data)      # matrix of predictors
y = tr_data$price_range                  # vector y values
set.seed(123)                        # replicate  results
lasso_model <- cv.glmnet(x, y, alpha=1)     # alpha=1 is lasso
best_lambda_lasso <- lasso_model$lambda.1se
best_lambda_lasso# largest lambda in 1 SE
lasso_coef <- lasso_model$glmnet.fit$beta[,lasso_model$glmnet.fit$lambda ==
best_lambda_lasso]
coef_l = data.table(lasso = lasso_coef)     # build table
coef_l[, feature := names(lasso_coef)]      # add feature names
to_plot_r = melt(coef_l                  # label table
             , id.vars='feature'
             , variable.name = 'model'
             , value.name = 'coefficient')
ggplot(data=to_plot_r,                  # plot coefficients
     aes(x=feature, y=coefficient, fill=model)) +
  coord_flip() +
  geom_bar(stat='identity', fill='salmon', color='brown',) +
  facet_wrap(~ model) + guides(fill=FALSE)

##STEPWISE SELECTION




## MAKING GRAPHS
tr_data$blue <- as.factor(tr_data$blue)
tr_data$dual_sim <- as.factor(tr_data$dual_sim)
tr_data$four_g <- as.factor(tr_data$four_g)
tr_data$three_g<- as.factor(tr_data$three_g)
tr_data$touch_screen <- as.factor(tr_data$touch_screen)
tr_data$wifi <- as.factor(tr_data$wifi)
```

```
tr_data$n_cores <- as.factor(tr_data$n_cores)
tr_data$price_range <- as.factor(tr_data$price_range)


p1 <-  ggplot(tr_data, aes(x=blue, fill=blue)) + theme_bw() + geom_bar() + ylim(0, 1050) +
labs(title = "Bluetooth") + scale_x_discrete(labels = c('Not Supported','Supported')) +
  scale_fill_manual(values = c("salmon", "orange"))
p2 <- ggplot(tr_data, aes(x=dual_sim, fill=dual_sim)) + theme_bw() + geom_bar() + ylim(0,
1050) + labs(title = "Dual Sim") + scale_x_discrete(labels = c('Not Supported','Supported'))+
  scale_fill_manual(values = c("salmon", "orange"))
p3 <- ggplot(tr_data, aes(x=four_g, fill=four_g)) + theme_bw() + geom_bar() + ylim(0, 1050)
+ labs(title = "4G") + scale_x_discrete(labels = c('Not Supported','Supported'))+
  scale_fill_manual(values = c("salmon", "orange"))
p4 <- ggplot(tr_data, aes(x=price_range, fill=price_range)) + theme_bw() + geom_bar() +
ylim(0, 600) + labs(title = "Price") + scale_x_discrete(labels = c('0','1','2','3'))+
  scale_fill_manual(values = c("salmon", "orange", "coral1","coral4"))
p5 <- ggplot(tr_data, aes(x=three_g, fill=three_g)) + theme_bw() + geom_bar() + ylim(0,
1600) + labs(title = "3G") + scale_x_discrete(labels = c('Not Supported','Supported'))+
  scale_fill_manual(values = c("salmon", "orange"))
p6 <- ggplot(tr_data, aes(x=touch_screen, fill=touch_screen)) + theme_bw() + geom_bar() +
ylim(0,1050) + labs(title ="Touch Screen") + scale_x_discrete(labels = c('Not
Supported','Supported'))+
  scale_fill_manual(values = c("salmon", "orange"))
p7 <- ggplot(tr_data, aes(x=wifi, fill=wifi)) + theme_bw() + geom_bar() + ylim(0, 1050) +
labs(title = "Wifi") + scale_x_discrete(labels = c('Not Supported','Supported'))+
  scale_fill_manual(values = c("salmon", "orange"))
p8 <- ggplot(tr_data, aes(x=n_cores, fill=n_cores))  + geom_bar() + ylim(0, 500) + labs(title =
"Number of Processor Cores ") +
  scale_fill_manual(values = c("salmon", "orange","chocolate", "darksalmon","coral1",
"coral2","coral3", "coral"))

print(grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, nrow = 3))
print(prop.table(table(tr_data$blue))) # cell percentages
print(prop.table(table(tr_data$dual_sim))) # cell percentages
print(prop.table(table(tr_data$four_g))) # cell percentages

##Decision Tree
#Importing Libraries And reading The file ----


set.seed(123)
split<-sample.split(tr_data,SplitRatio=0.8)
train_data<-subset(tr_data,split==TRUE)
ts_data<-subset(tr_data,split==FALSE)

#set.seed(12345)
# Training with classification tree
model <- rpart(price_range ~ram+battery_power+px_height+px_width, data=train_data,
method="class")
#print(model, digits = 3)
rpart.plot(model)
printcp(model)
plotcp(model)
pred <- predict(model, ts_data, type = "class")
#pred
# Accuracy and other metrics
t1<-table(ts_data$price_range, pred)
confusionMatrix(t1, mode="everything", positive="1")


##KNN
df<- train_data[,c(1,5,11,12,13,14,21)]


#Applying knn for k = 1----
```

```
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 1)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 3----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 3)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 5----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 5)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 7----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 7)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 9----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 9)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 11----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 11)
#classifier_knn
#summary(classifier_knn)
```

```
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 13----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 13)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 17----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 17)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")
#Applying knn for k = 15----
classifier_knn <- knn(train = tr_data, test = ts_data, cl = tr_data$price_range, k = 15)
#classifier_knn
#summary(classifier_knn)
cm <- table(ts_data$price_range, classifier_knn)
confusionMatrix(cm, mode="everything", positive="1")




##Naive Bayes
model <- naiveBayes(price_range ~ram+battery_power+px_height+px_width, data = tr_data)


print(summary(model))


predicted <- predict(model, ts_data)
head(predicted)
```

```
accu<-mean(predicted == ts_data$price_range)  #accuracy of the multinomial logistic
regression
t1<-table(ts_data$price_range, predicted)
confusionMatrix(t1, mode="everything", positive="1")



## Multinomial Logistic regression

modelML <- nnet::multinom(price_range ~ram+battery_power+px_height+px_width,
tr_data)

summary(modelML)

predicted <- modelML %>% predict(ts_data)
head(predicted)
ts_data <- cbind(ts_data, predicted)


accu<-mean(predicted == ts_data$price_range)  #accuracy of the multinomial logistic
regression
t1<-table(ts_data$price_range, predicted)
confusionMatrix(t1, mode = "everything", positive="1")


#Building Random Forest classifier
rf1 <- randomForest(price_range ~ ram + battery_power + px_height + px_width,
            data = train_data, importance = TRUE)


# Make predictions (will be factors)
```

```
pred1 <- predict(rf1, ts_data)
cm1 = table(ts_data$price_range, pred1)
confusionMatrix(cm1, mode = "everything", positive = "1")
```