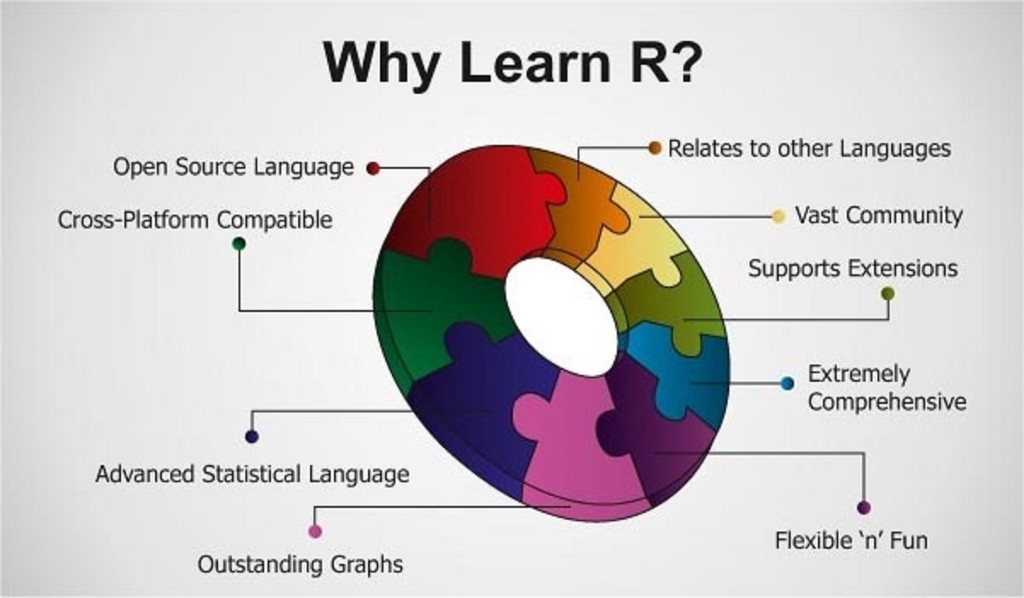
# Getting Started with R Programming

Fonte: <https://www.kdnuggets.com/2020/02/getting-started-r-programming.html>

By Pier Paolo Ippolito, The University of Southampton

An end to end Data Analysis using R, the second most requested programming language in Data Science.



(Source: <https://hackernoon.com/5-free-r-programming-courses-for-data-scientists-and-ml-programmers-5732cb9e10>)

**Introduction**

R is a programming language focused on statistical and graphical analysis. It is therefore commonly used in statistical inference, data analysis and Machine Learning. R is currently one of the most requested programming language in the Data Science job market (Figure 1).,

R is available to be installed from [r-project.org](http://www.r-project.org/) and one of R most commonly used integrated development environment (IDE) is certainly [RStudio](http://www.rstudio.com/ide/).

There are two main types of packages (libraries) which can be used to add functionalities to R: base packages and distributed packages. Base packages come with the installation of R, distributed packages can instead be downloaded for free using [CRAN](https://cran.r-project.org/web/packages/available_packages_by_date.html).

Once installed R, we can then get started doing some data analysis!

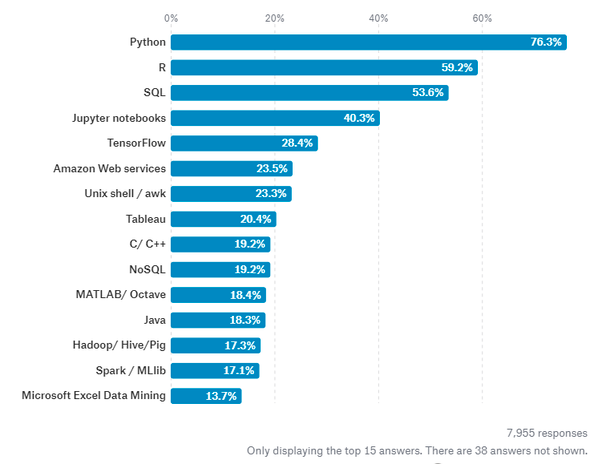


Figure 1: Most Requested programming languages for Data Science in 2019 [1]

## Demonstration

In this example, I will walk you through an end to end analysis of the [Mobile Price Classification Dataset](https://www.kaggle.com/iabhishekofficial/mobile-price-classification#train.csv) to predict the price range of Mobile Phones. The code I used for this demonstration is available on both my [GitHub](https://github.com/pierpaolo28/R-Programming/blob/master/Smartphone%20Prices/workflow.r) and [Kaggle](https://www.kaggle.com/pierpaolo28/mobile-price-classification?scriptVersionId=20002929) account.

## Importing Libraries

First of all, we need to import all the necessary libraries.

Packages can be installed in R using the ***install.packages()*** command and then loaded using the ***library()*** command. In this case, I decided to install first PACMAN (Package Management Tool) and then use it to install and load all the other packages. PACMAN makes loading library easier because it can install and load all the necessary libraries in just one line of code.

install.packages("pacman")

library(pacman)

pacman::p\_load(pacman, dplyr, ggplot2, rio, gridExtra, scales, ggcorrplot, caret, e1071)

The imported packages are used to add the following functionalities:

* **dplyr:**data processing and analysis.
* **ggplot2:**data visualization.
* **rio:**data import and export.
* **gridExtra:**to make plots graphical objects to which can be freely arranged on a page.
* **scales:**used to scale data in plots.
* **ggcorrplot:**tovisualize correlation matrices using ggplot2 in the backend.
* **caret:**is used totrain and plot classification and regression models.
* **e1071:**contains functions to perform Machine Learning algorithms such as Support Vector Machines, Naive Bayes, etc…

## Data Pre-processing

We can now go on loading our dataset, displaying it’s first 5 columns (Figure 2) and print a summary of the main characteristics of each feature (Figure 3). In R, we can create new objects using the **<-** operator.

# Loading our dataset

df <- import("./mobile\_price.csv")

head(df)

# Getting set of descriptive statistics, depending on the type of variable.

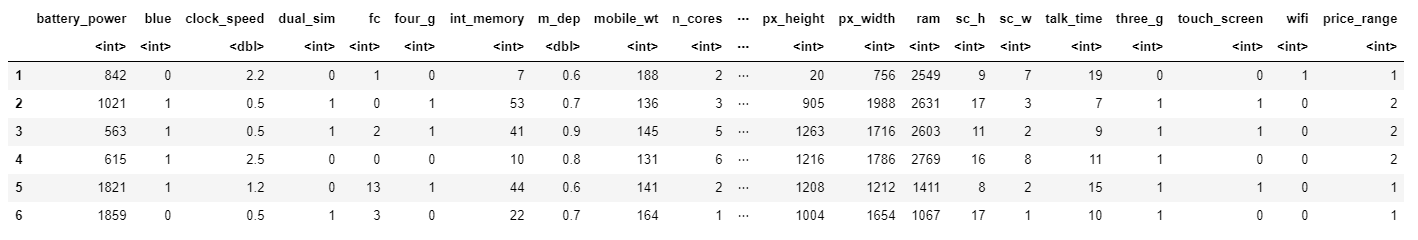
# In case of a Numerical Variable -> Gives Mean, Median, Mode, Range and Quartiles.

# In case of a Factor Variable -> Gives a table with the frequencies.

# In case of Factor + Numerical Variables -> Gives the number of missing values.

# In case of character variables -> Gives the length and the class.

summary(df)



The summary function provides us with a brief statistical description of each feature in our dataset. Depending on the nature of the feature in consideration, different statistics will be provided:

* **Numeric Features:**Mean, Median, Mode, Range and Quartiles.
* **Factor Features:**Frequencies.
* **A mixture of Factor and Numeric Features:**Number of missing values.
* **Character Features:**Lenght of the class.

Factors are a type of data object used in R to categorize and store data (eg. integers or strings) as levels. They can, for example, be used to one hot encode a feature or to create Bar Charts (as we will see later on). Therefore they are especially useful when working with columns with few unique values.

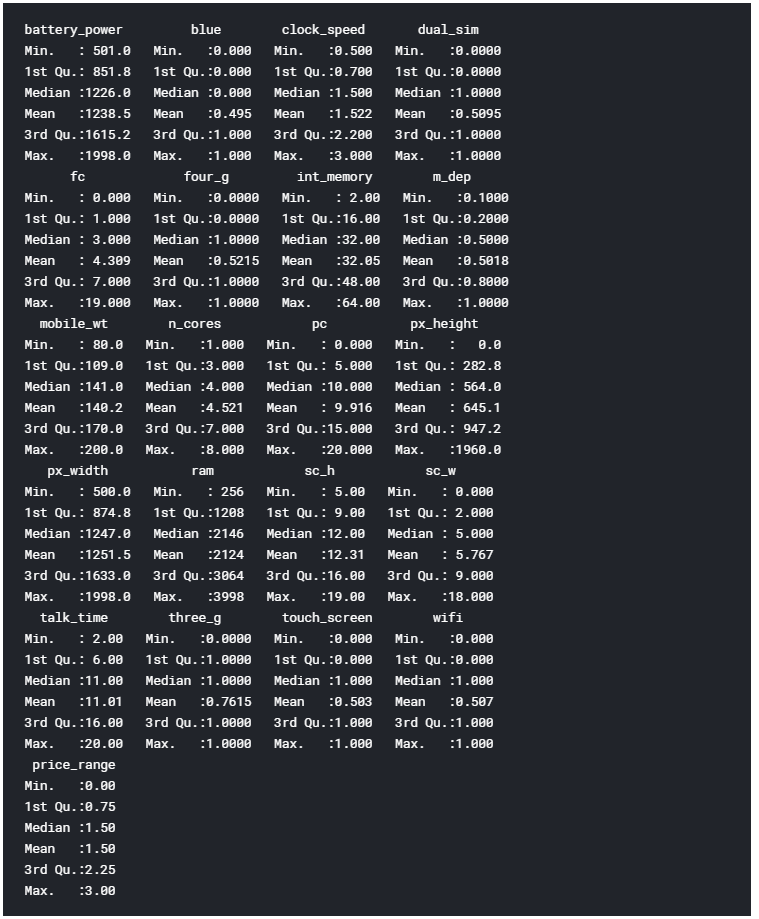


Figure 3: Dataset Summary

Finally, we can now check if our Dataset contains any Not A Numbers (NaNs) value using the code shown below.

# Checking for Missing values

missing\_values <- df %>% summarize\_all(funs(sum(is.na(.))/n()))

missing\_values <- gather(missing\_values, key="feature", value="missing\_pct")

missing\_values %>%

ggplot(aes(x=reorder(feature,-missing\_pct),y=missing\_pct)) +

geom\_bar(stat="identity",fill="red")+

coord\_flip()+theme\_bw()

As we can see from Figure 4, no missing numbers have been found.

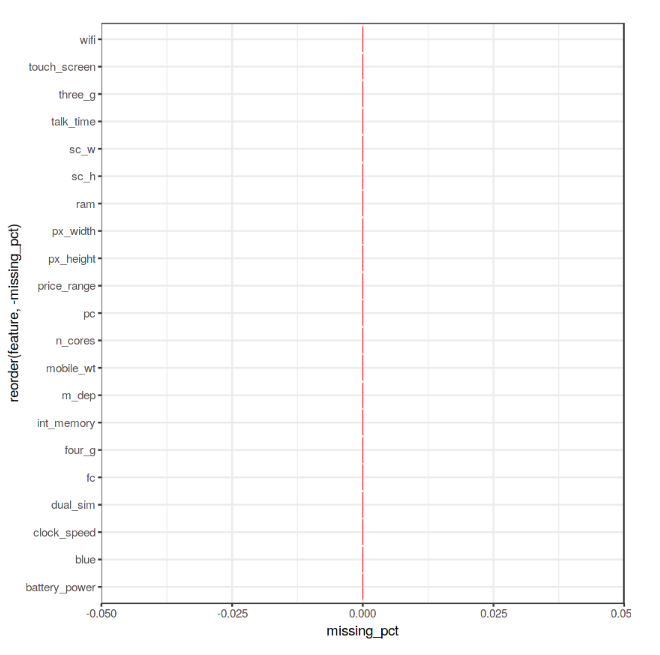


Figure 4: Percentage of NaNs in each feature

## Data Visualization

We can now start our Data Visualization by plotting a Correlation Matrix of our dataset (Figure 5).

corr <- round(cor(df), 8)

ggcorrplot(corr)

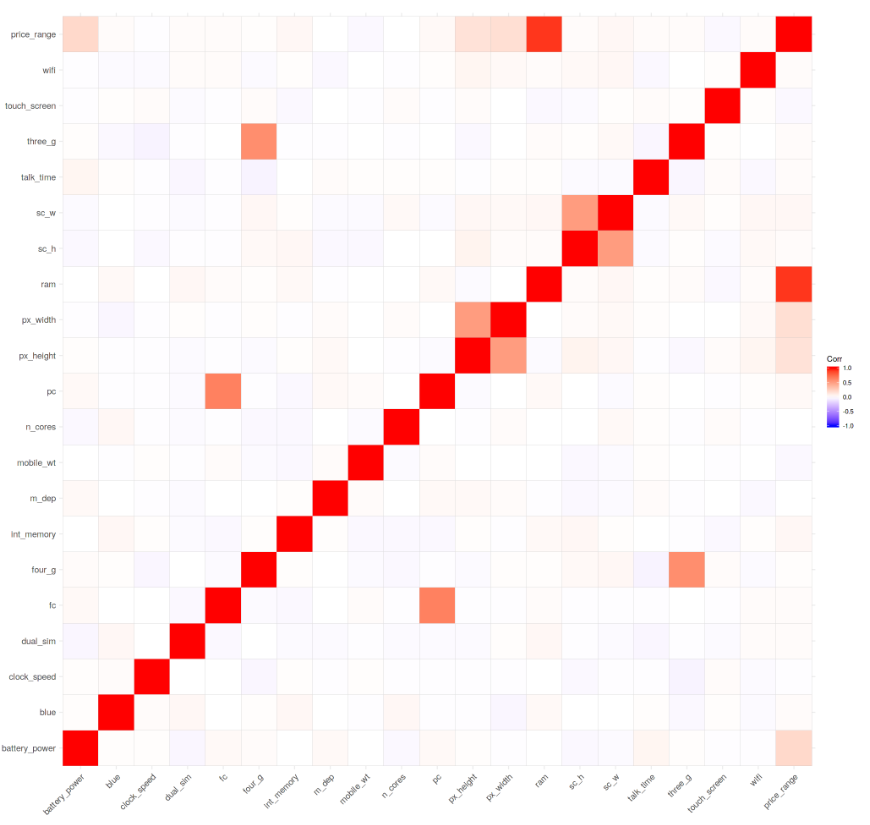


Figure 5: Correlation Matrix

Successively, we can start analysing individual features using Bar and Box plots. Before creating these plots, we need though to first convert the considered features from Numeric to Factor (this allow us to bin our data and then plot the binned data).

df$blue <- as.factor(df$blue)

df$dual\_sim <- as.factor(df$dual\_sim)

df$four\_g <- as.factor(df$four\_g)

df$price\_range <- as.factor(df$price\_range)

We can now create 3 Bar Plots by storing them in there different variables (p1, p2, p3) and then add them to ***grid.arrange()*** to create a subplot. In this case, I decided to examine the Bluetooth, Dual Sim and 4G features. As we can see from Figure 6, a slight majority of mobiles considered in this Dataset does not support Bluetooth, is Dual Sim and has 4G support.

# Bar Chart Subplots

p1 <- ggplot(df, aes(x=blue, fill=blue)) +

theme\_bw() +

geom\_bar() +

ylim(0, 1050) +

labs(title = "Bluetooth") +

scale\_x\_discrete(labels = c('Not Supported','Supported'))

p2 <- ggplot(df, 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'))

p3 <- ggplot(df, aes(x=four\_g, fill=four\_g)) +

theme\_bw() +

geom\_bar() +

ylim(0, 1050) +

labs(title = "4 G") +

scale\_x\_discrete(labels = c('Not Supported','Supported'))

grid.arrange(p1, p2, p3, nrow = 1)

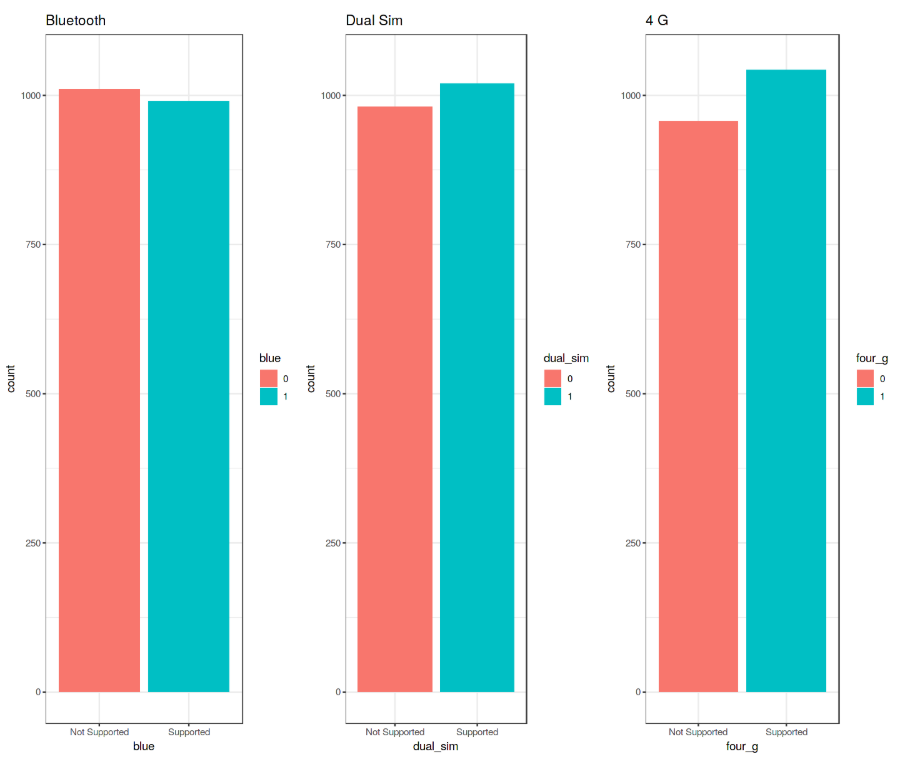


Figure 6: Bar Plot Analysis

These plots have been created using R ***ggplot2*** library. When calling the ***ggplot()*** function, we create a coordinate system on which we can add layers on top of it [2].

The first argument we give to the ***ggplot()*** function is the dataset we are going to use and the second one is instead an aesthetic function in which we define the variables we want to plot. We can then go on adding other additional arguments such us defining a desired geometric function (eg. barplot, scatter, boxplot, histogram, etc…), adding a plot theme, axis limits, labels, etc…

Taking our analysis a step further, we can now calculate the precise percentages of the difference between the different cases using the ***prop.table()*** function. As we can see from the resulting output (Figure 7), 50.5% of the considered mobile devices do not support Bluetooth, 50.9% is Dual Sim and 52.1% has 4G.

prop.table(table(df$blue)) # cell percentages

prop.table(table(df$dual\_sim)) # cell percentages

prop.table(table(df$four\_g)) # cell percentages

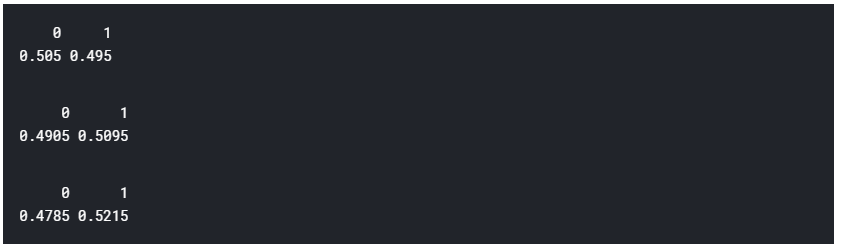


Figure 7: Classes Distribution Percentage

We can now go on creating 3 different Box Plots using the same technique used before. In this case, I decided to examine how having more battery power, phone weight and RAM (Random Access Memory) can affect mobiles prices. In this Dataset, we are not given the actual phone prices but a price range indicating how high the price is (four different levels from 0 to 3).

# Bar Chart Subplots

p1 <- ggplot(df, aes(x=price\_range, y = battery\_power, color=price\_range)) +

geom\_boxplot(outlier.colour="red", outlier.shape=8,

outlier.size=4) +

labs(title = "Battery Power vs Price Range")

p2 <- ggplot(df, aes(x=price\_range, y = mobile\_wt, color=price\_range)) +

geom\_boxplot(outlier.colour="red", outlier.shape=8,

outlier.size=4) +

labs(title = "Phone Weight vs Price Range")

p3 <- ggplot(df, aes(x=price\_range, y = ram, color=price\_range)) +

geom\_boxplot(outlier.colour="red", outlier.shape=8,

outlier.size=4) +

labs(title = "RAM vs Price Range")

grid.arrange(p1, p2, p3, nrow = 1)

The results are summarised in Figure 8. Increasing Battery Power and RAM consistently lead to an increase in Price. Instead, more expensive phones seem to be overall more lightweight. In the RAM vs Price Range plot have interestingly been registred some outliers values in the overall distribution.

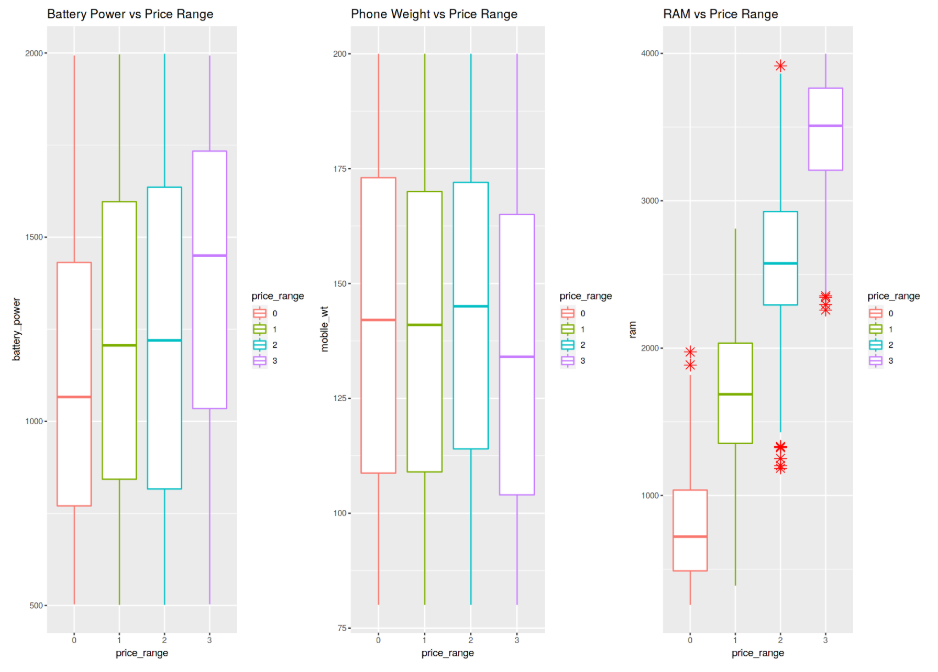


Figure 8: Box Plot Analysis

Finally, we are now going to examine the distribution of camera quality in Megapixels for both the Front and Primary cameras (Figure 9). Interestingly, the Front camera distribution seems to follow an exponentially decaying distribution while the Primary camera roughly follows a uniform distribution. If you are interested in finding out more about Probability Distributions, you can find more information [here](https://towardsdatascience.com/probability-distributions-in-data-science-cce6e64873a7).

data = data.frame(MagaPixels = c(df$fc, df$pc),

Camera = rep(c("Front Camera", "Primary Camera"),

c(length(df$fc), length(df$pc))))

ggplot(data, aes(MagaPixels, fill = Camera)) +

geom\_bar(position = 'identity', alpha = .5)

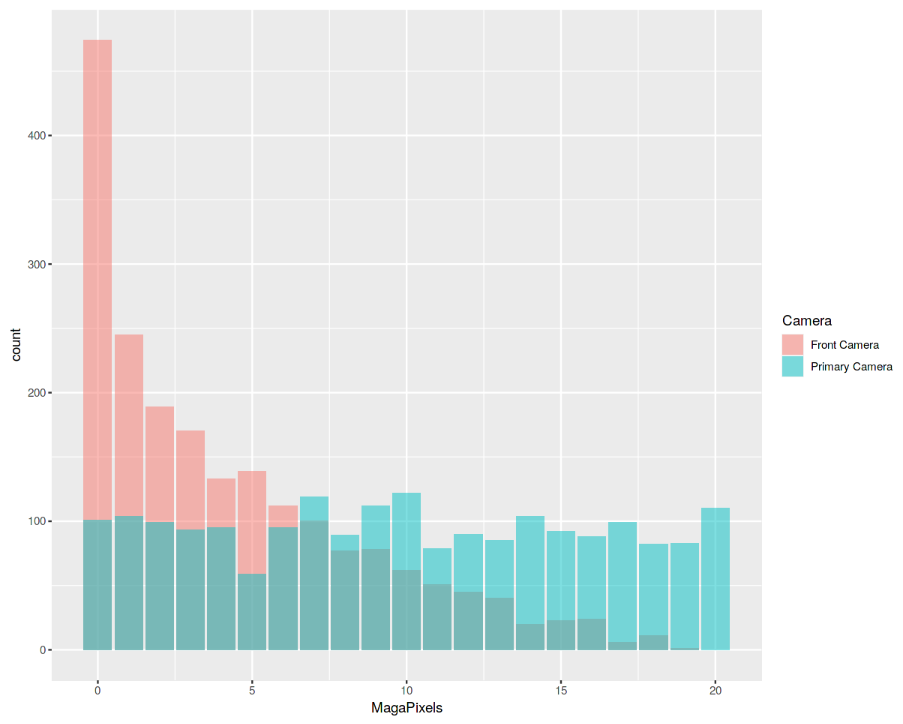


Figure 9: Histogram Analysis

## Machine Learning

In order to perform our Machine Learning analysis, we need first to convert our Factor variables in Numeric form and then divide our dataset into train and test sets (75:25 ratios). Lastly, we divide the train and test sets into features and labels (***price\_range***).

df$blue <- as.numeric(df$blue)

df$dual\_sim <- as.numeric(df$dual\_sim)

df$four\_g <- as.numeric(df$four\_g)

df$price\_range <- as.numeric(df$price\_range)

## 75% of the sample size

smp\_size <- floor(0.75 \* nrow(df))

# set the seed to make our partition reproducible

set.seed(123)

train\_ind <- sample(seq\_len(nrow(df)), size = smp\_size)

train <- df[train\_ind, ]

test <- df[-train\_ind, ]

x\_train <- subset(train, select = -price\_range)

y\_train <- train$price\_range

x\_test <- subset(test, select = -price\_range)

y\_test <- test$price\_range

It’s now time to train our Machine Learning model. In this example, I decided to use [Support Vector Machines (SVM)](https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c) as our multiclass classifier. Using R ***summary()*** we can then inspect the parameters of our trained model (Figure 10).

model <- svm(x\_train, y\_train, type = 'C-classification',

kernel = 'linear')

print(model)

summary(model)

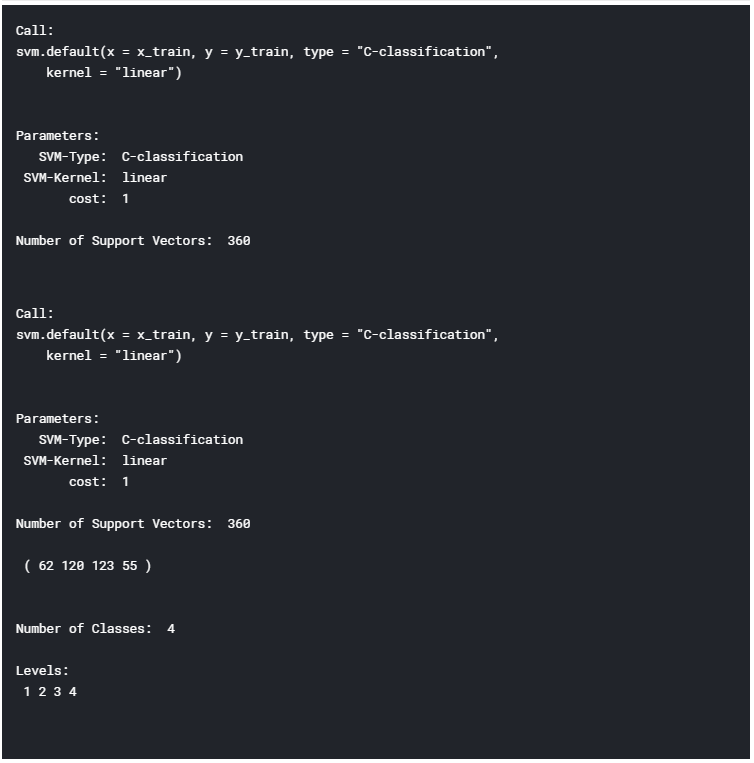


Figure 10: Machine Learning Model Summary

Finally, we can now test our model making some predictions on the test set. Using R ***confusionMatrix()*** function we can then get a complete report of our model accuracy (Figure 11). In this case, an Accuracy of 96.6% was registered.

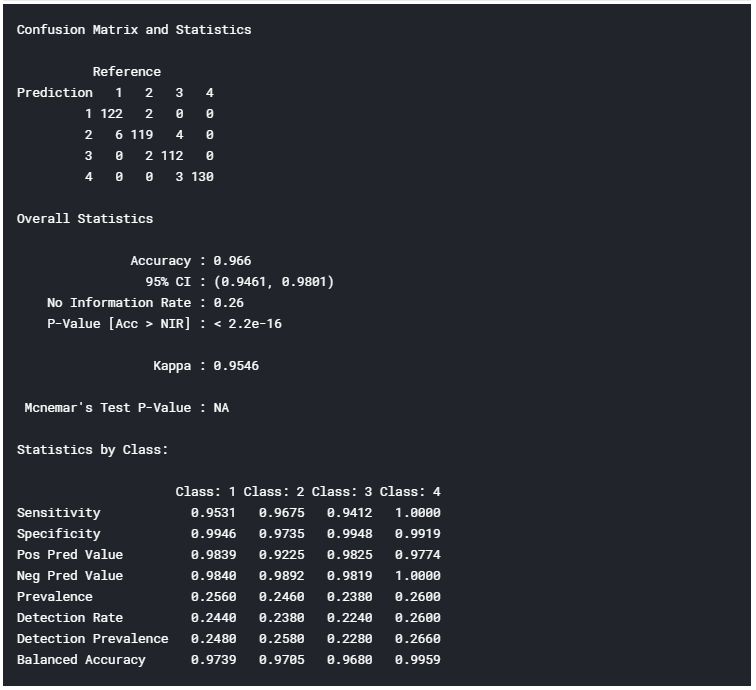


Figure 11: Model Accuracy Report

*I hope you enjoyed this article, thank you for reading!*