Project Report
On
Predicting Application Rating of Google Play Store

Developed By: -
Astha Patel(15012121012)
Parth Patel(15012121018)
Kavan Patel(16012122004)

Guided By:-
Prof. Umang Shukla

Submitted to
Department of Computer Science & Engineering
U V Patel College of Engineering

Year: 2019

CERTIFICATE

This is to certify that the report entitled **"Predicting Application Rating of Google Play Store"** by Astha Patel(Enrolment No.15012121012), Parth Patel(Enrolment No.15012121018) and Kavan Patel (EnrolmentNo.16012122004) of Ganpat University, towards the fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering, is record of bonafide final year Project work, carried out by them in the CSE(BDA) Department. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for award of any other Degree/Diploma.

Signature of Head of Department:

Name of Head of Department:

Signature of Internal Guide:

Name of Internal Guide: Prof. Umang Shukla

(Office seal)

Place:
Date:

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank **Dr. Hemal Shah,** Head of Department, Computer Science and Engineering, who gave us an opportunity to undertake this project. My grateful thanks to **Prof. Umang Shukla** for his guidance in project work **Predicting Application Rating of Google Play Store**, who despite being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where would have been without his/her help. CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Signature of Students

ASTHA PATEL

(Enrollment No: 15012121012)

PARTH PATEL

(Enrollment No: 15012121018)

KAVAN PATEL

(Enrollment No: 16012122004)

ABSTRACT

A cursory glance at the newspaper reveals the fact that every year, the ratio of the mobile application is rapidly rising. We strongly believe that Data Science can be used for good, that's why we decided to make this contribution. We are considering Google Play Store data-set and with the help of this data-set we will make approximately prediction of application rating; after that, we will create UI based application, which helps Google Play Store mobile application developer while developing the application.

INDEX

LIST OF FIGURES

Figure No **Page No**

GROUP ID: 23

LIST OF TABLES

Table No **Page No**

GROUP ID: 23

CHAPTER: 1 INTRODUCTION

GROUP ID: 23

CHAPTER 1 INTRODUCTION

In today's world, every people loves to use application in their mobile phones so the day-by-day ratio of the mobile application is increasing one of the best examples is Google play store where in July 2013 total number of the application was 1 million which hiked to 2.6 million in December 2019 [1]. It is very helpful for us if we can know approximately rating of application before developing application; here is an example to support this point; first of all let's consider that if you want to develop and publish your application so before developing it you will decide its properties like my application will be free and its category will be gaming after that you will predict rating on the bases of that properties so by this predicted rating you can decide that you should keep this same property for your new application or you need to change it to get a better rating from users.

By considering this above scenario we decided to make a UI based application which helps to prediction rating by taking few parameters in input; to make this application we have first scrap data from Google play store and then perform data cleaning operations on that; after getting clean data we have done feature selection and extraction; then in the last step we applied various model for prediction of rating and then select best fit model.

The main focus of this project is to utilize historical data and make effective use of this data in the development of successful application using various new technologies. Below is the list of the tools and technologies which we have used in this project:-

- Smarten for data cleaning
- Tableau for data visualization
- Anaconda for web scraping (Python 3.6)
- R Studio for rating prediction by creating various models in this we used (name of libraries which we used) library (R programming)
- Trello for project management

GROUP ID: 23

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The project is limited to only Google Play Store because data which is considered for rating prediction is of Google Play Store.

CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Processor	2.0 GHz
RAM	4GB
HDD	40GB

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

Operating System	Any operating system which can support an internet browser.
Programming language	-
Other tools & tech	Internet browser

Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODEL

CHAPTER 4 PROCESS MODEL

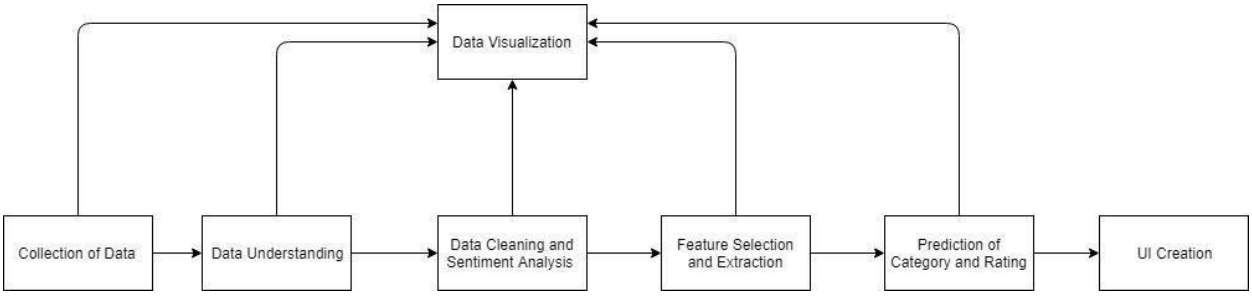

Figure 4.1 Process Model of Project

CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List of Major Activities

Task: - 1 Data Understanding using Visualization
Task: - 2 Pre-Processing of Application Information Data and Visualization
Task: - 3 Web Scraping of Reviews and its Visualization
Task: - 4 Pre-processing of Reviews Data with Sentiment Analysis and its Visualization
Task: - 5 Feature Selection and Extraction along with Visualization
Task: - 6 Implement a various model Rating prediction
Task: - 7 Selection of Best Fit Model
Task: - 8 UI Creation

5.2 Estimated Time Duration in Days

Figure 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

CHAPTER 6 IMPLEMENTATION DETAIL

6.1 Flowchart of Implementation

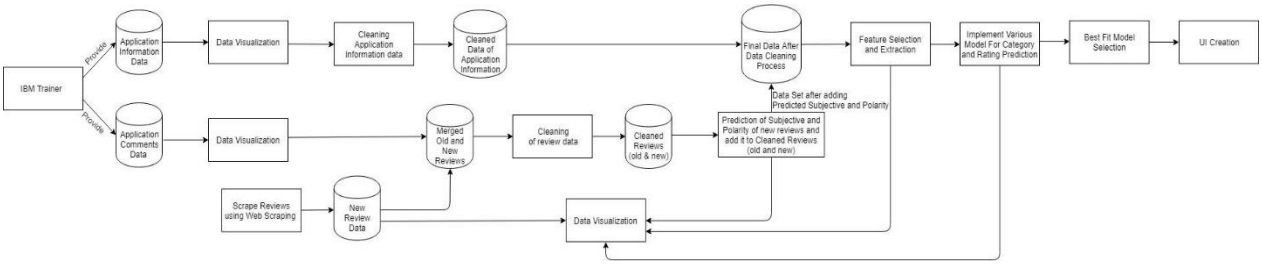

Figure 6.1 Project Implementation Flowchart

6.1.1 Data Collection

We are provided two different data (Application Information Data and Application Comments Data) from IBM trainer.

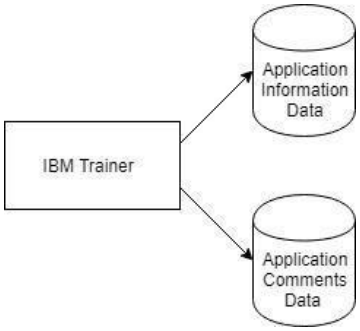

Figure 6.2 Method of Data Collection

GROUP ID: 23

6.1.2 Understanding Data

In this Phase, we have understood the structure of data-set with the help of visualization. In outcome we get to know that in application information data there are total 13 columns which are application name, category, rating, no of reviews, size, installs, type, price, content rating, genres, last update, current version and android version; talking about application comments there are total 5 columns which are application name, review, sentiment, polarity and subjectivity.

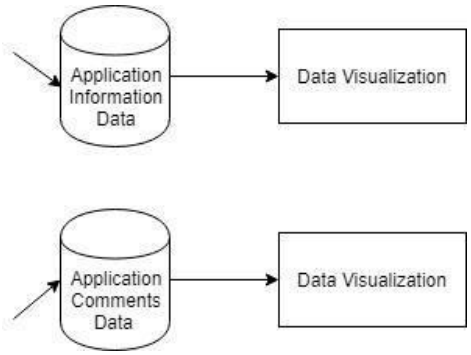

Figure 6.3 Method of Data Understanding

6.1.3 Data Visualization:-

In this phase we have implemented various graph of different phases; phases which are considered are understood data, web scraping of comments, data cleaning, prediction of subjective and polarity, Feature selection and extraction, Model selection for rating prediction. Following are the things which we have implemented in this phase:-

6.1.3.1 Understand Original Application Information Data:-

By making graphs we have to understand the value of every column in the original data set. Below mention is the visualization of every column except android version and last update.

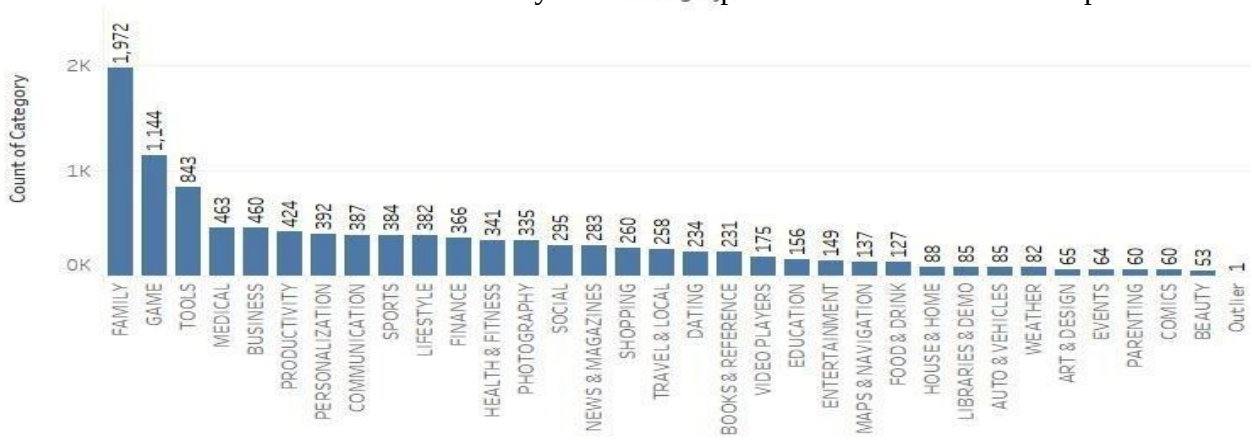

Figure 6.4 Description of Category Column in Application Information Original Data

GROUP ID: 23

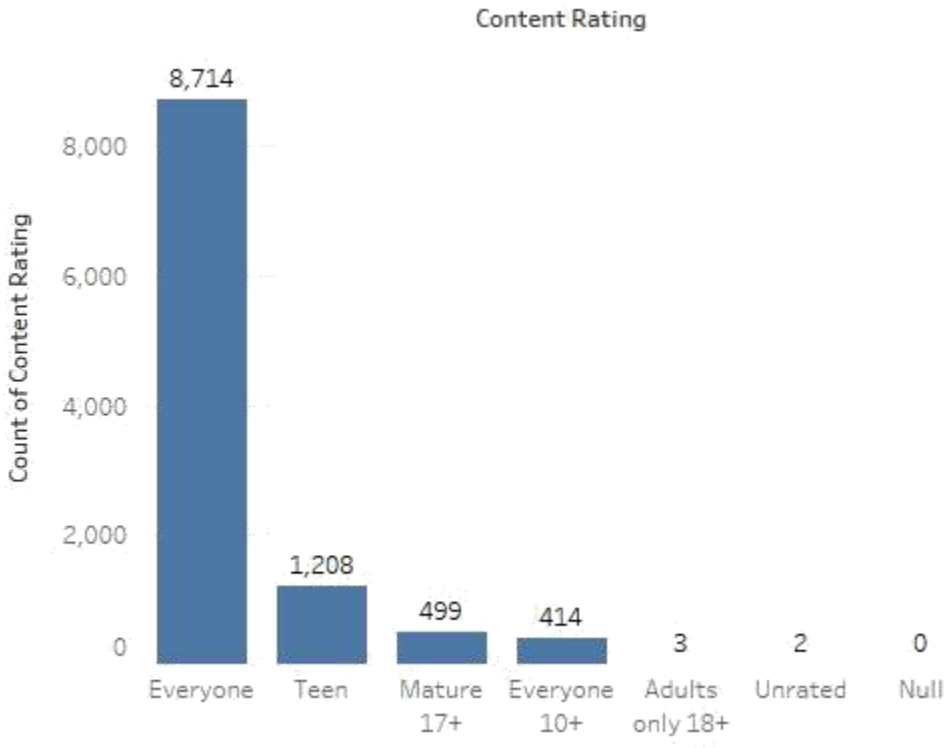

Figure 6.5 Description of Content Rating Column in Application Information Original Data-set

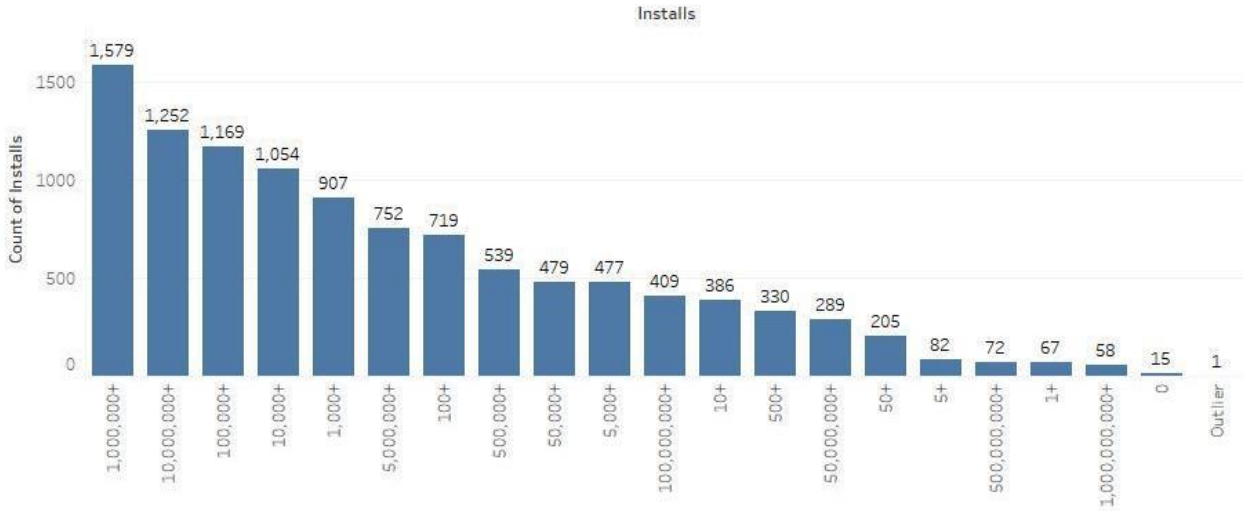

Figure 6.6 Description of Installs Column in Application Information Original Data-set

GROUP ID: 23

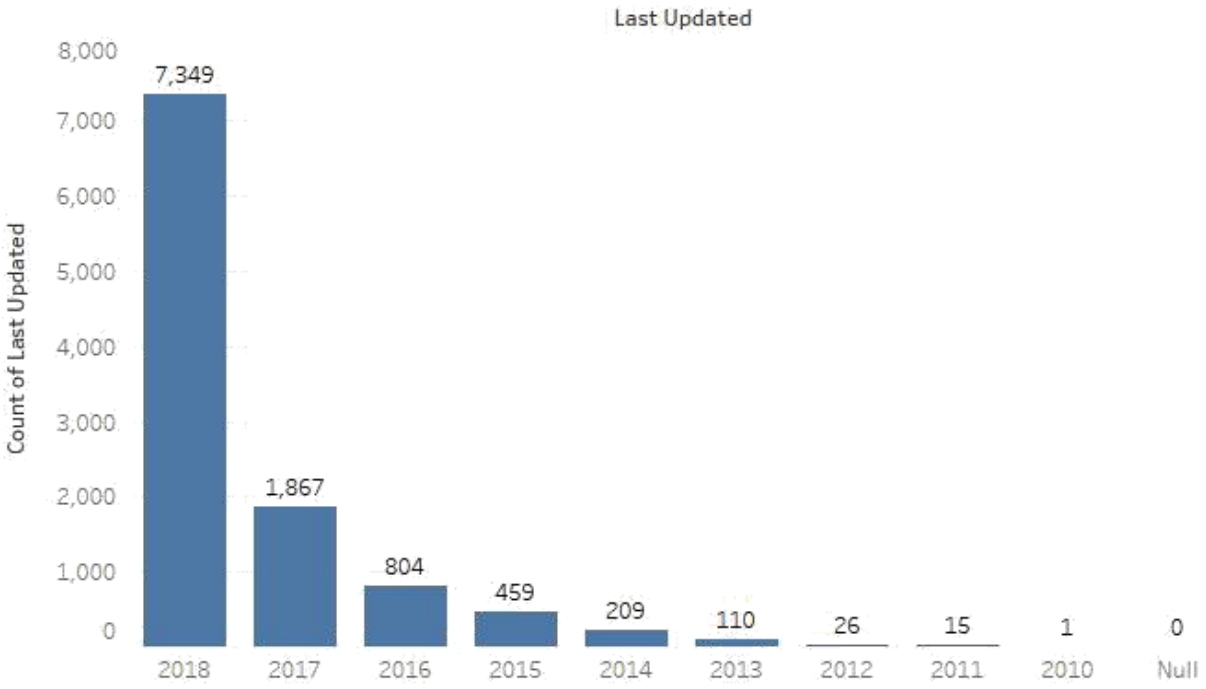

Figure 6.7 Description of Last Updated Column in Application Information Original Data-set

Figure 6.8 Description of Rating Column in Application Information Original Data-set

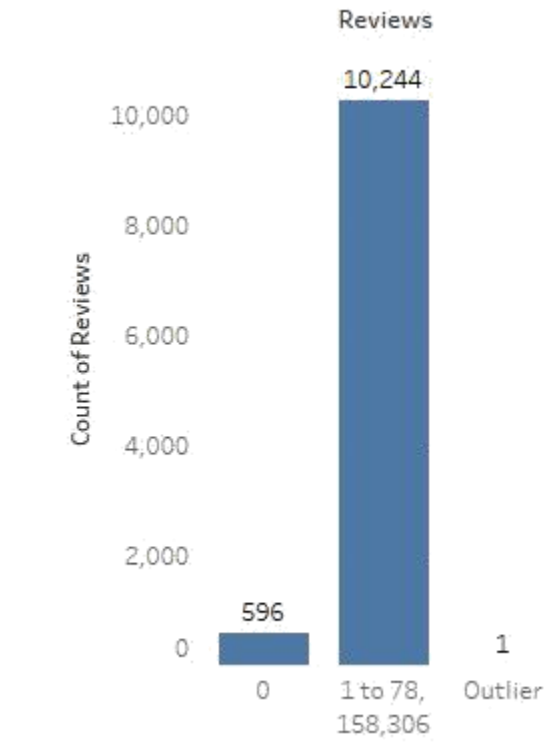

Figure 6.9 Description of Reviews Column in Application Information Original Data-set

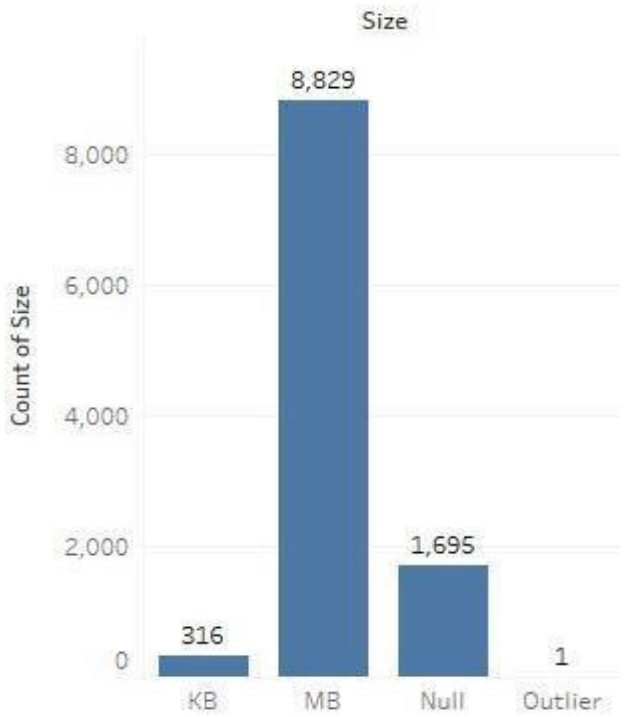

Figure 6.10 Description of Size Column in Application Information Original Data-set

GROUP ID: 23

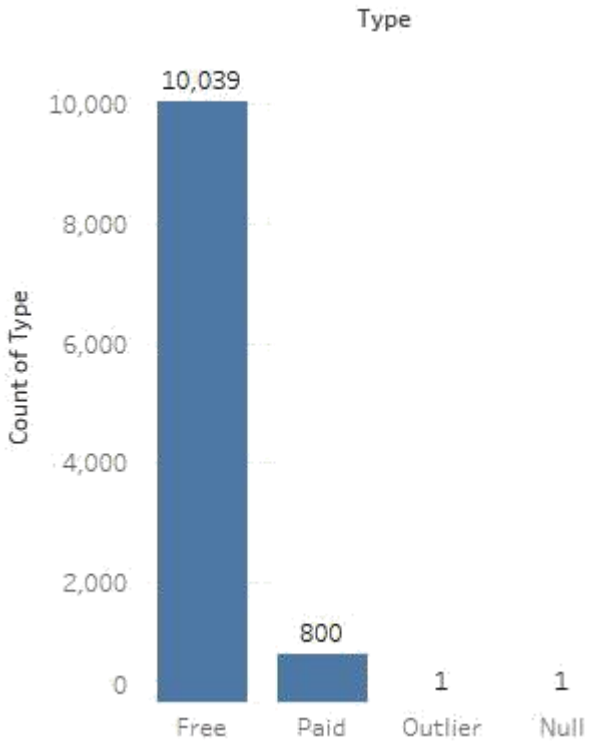

Figure 6.11 Description of Type Column in Application Information Original Data-set

6.1.3.2 Understand Original Application Review Data:-

By making graphs we have to understand the value of every column in the original data set of reviews. Below mention is the visualization from which we have carry out meaningful information which help us in pre-processing of this data-set.

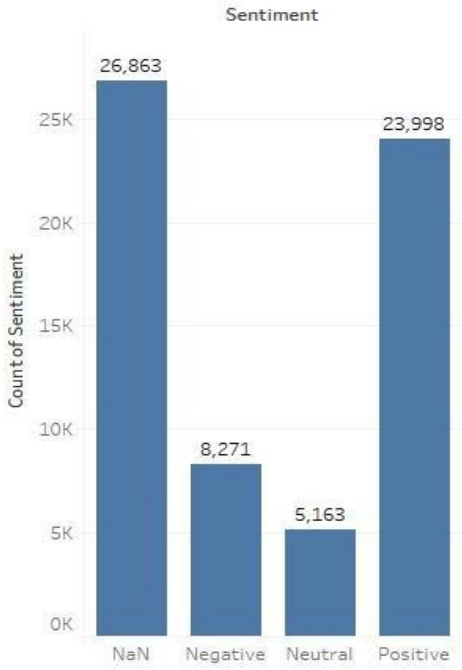

Figure 6.12 Description of Sentiment Column in Application Reviews Original Data-set

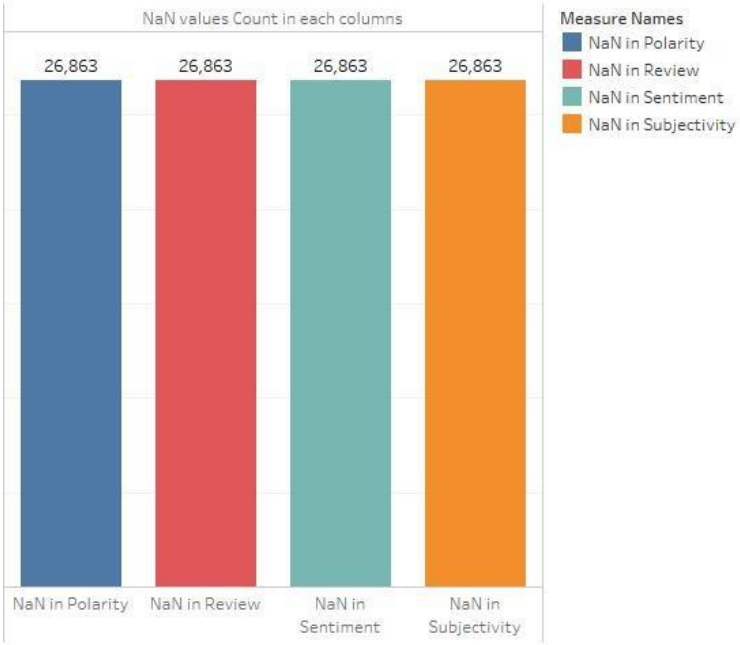

Figure 6.13 Description of Total NaN values in all Column of Application Reviews Original Data-set

GROUP ID: 23

6.1.4 Data Cleaning and Sentiment Analysis:-

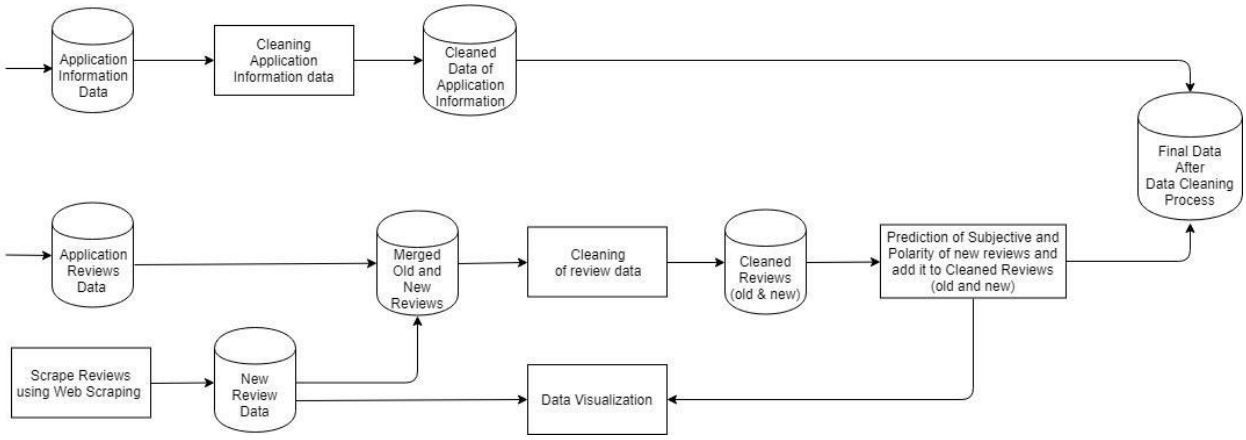

Figure 6.14 Method of Data Cleaning and Sentiment Analysis

6.1.4.1 Data Cleaning of Application Information Data:-

Following task are done in this phase using Smarten:-

- Remove all rows with all same values.
- Find out NaN values in every column by table graph in smarten
- Remove rows which are not appropriate like category name 1.9 row
- Remove rows who's rating was NaN, installs were 0 and review was 0
- Select size column which only has values in Mb; remove M from it and then convert it into kb
- Place NaN in size column where there was varies with device
- In review, column replace NaN where there is varies with device
- Remove + from Installs column
- Remove $ from price column
- Remove and up from android version column and round value to by 1 after.
- Separate data category wise
- In the size column, we place mean of specific categories field where there was NaN value Rating column is our Output parameter so we decided to predict the value of NaN for that, due to that we can get high accuracy -> in category separated file first we place mean of specific category where there was NaN value and then we do the same thing for median and make new CSV file for every category.
- Feature selection for model creation.[3][4][5]
  Selected algorithm: - Boruta feature Selection
  We have used Boruta feature selection because of various reasons which are listed below:-
    - It works well for both regression and classification problem.
    - It takes multi-variable relationship into consideration.
    - This algorithm is an advance improvement on random forest variable importance measure and this method is getting popular in today's world.

GROUP ID: 23

- It use all the features which are relevant to the output variable which is also known as all-relevant variable selection method. Where other feature selection methods only consider subset of features which yields a minor error so due to that other algorithm are minimal optimal method as compare to boruta.
- Interactions between variables are also handled.

Code:-

```
rm(list=ls())
f<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\all category data with mean_rating\\final\\mean_Google_.csv")

a<-f[,c(2,4:8,10:11,15)]   #a<-data[,c(2,4:8,10:11,15)]
str(f)


library(Boruta)
boruta_output <- Boruta(f$Rating ~ ., data=f)
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")
```

Figure 6.15 Code of Feature Selection

Output:-

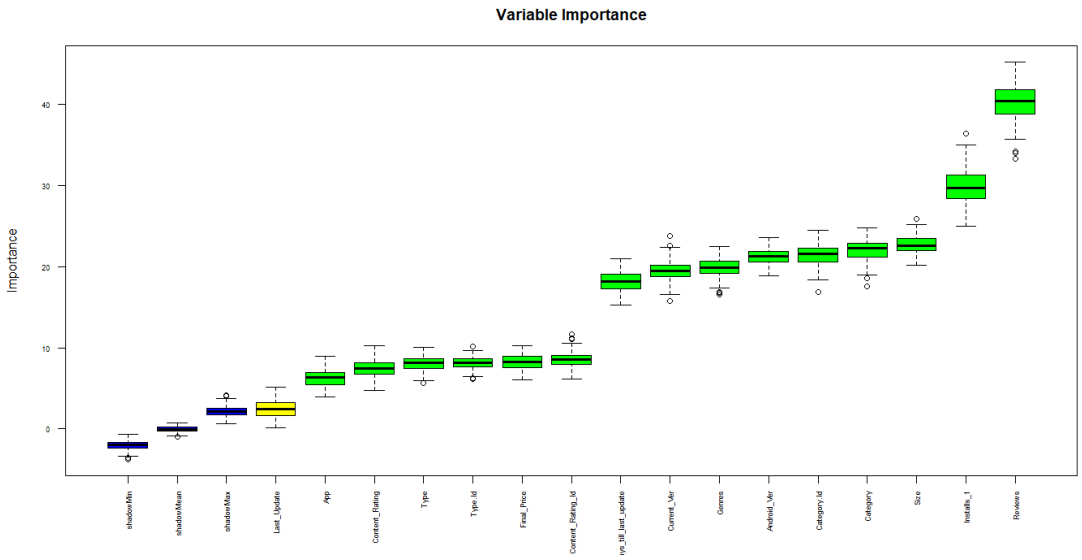

Figure 6.16 Plot of Feature Selection

Selected variable for all model are:-

- Category Name
- Reviews
- Size
- Installs
- Type
- Final Price
- Content Rating
- No of days till last update

- Prediction of NaN value of rating by applying a machine learning model. (Imputation) [2] Given below are the code and outcome of machine learning models applied on mean and median data-set which are used for rating prediction.

  - Multiple Linear Regression

```
rm(list=ls())
data<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\all category data with mean_rating\\final\\mean_Google_.csv")

a<-data[,c(2,4:8,10:11,15)]
results <- fastDummies::dummy_cols(a)
x<-results[,c(2:5,7,9:41,43,45:48)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]

rf <- lm(Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)
library(Metrics)
mse(test$Rating, p1)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))

postResample(pred = p1, obs = test$Rating)
```

Figure 6.17 Multiple Linear Regression on Mean Data

```
> mean((test$Rating - p1)^2)
[1] 0.2303739
```

Figure 6.18 Mean Square Error of Multiple Linear Regression on Mean Data

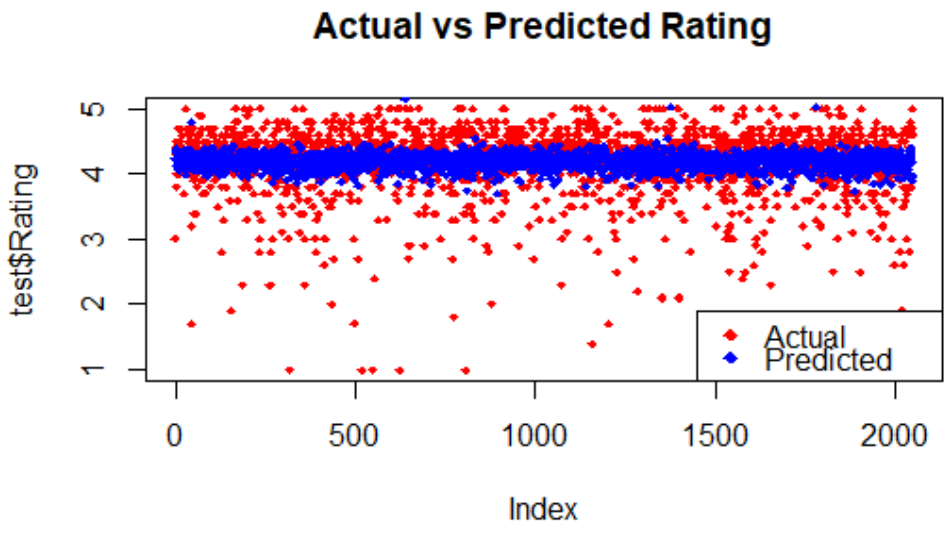

Figure 6.19 Plot of Multiple Linear Regression on Mean Data

GROUP ID: 23

```
results <- fastDummies::dummy_cols(a)
x<-results[,c(2:5,7,9,10:49)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]

rf <- lm(Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)
library(Metrics)
mse(test$Rating, p1)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))

postResample(pred = p1, obs = test$Rating)
```

Figure 6.20 Multiple Linear Regression on Median Data

```
> mse(test$Rating, p1)
[1] 0.2467404978
>
```

Figure 6.21 Mean Square Error of Multiple Linear Regression on Median Data

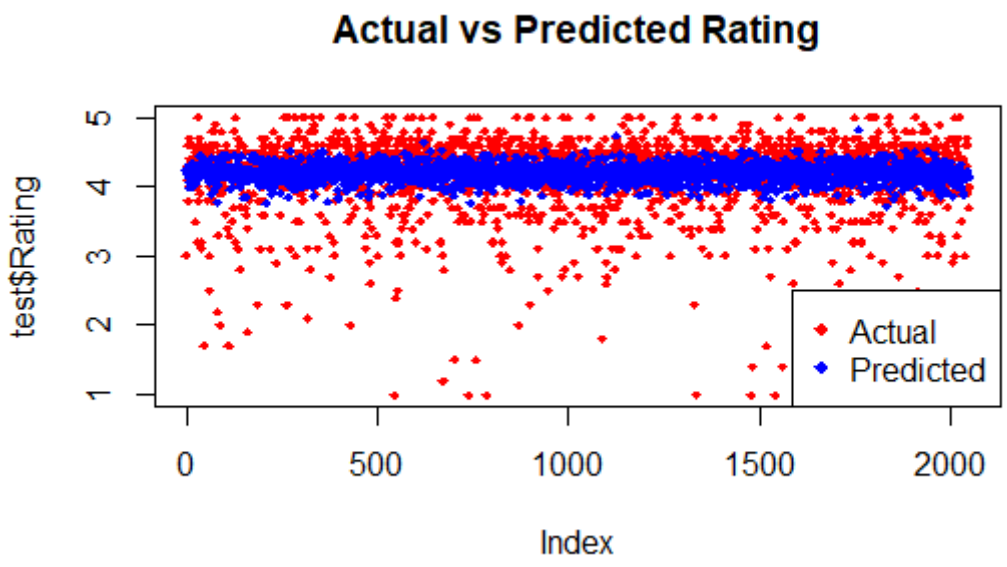

Figure 6.22 Plot of Multiple Linear Regression on Median Data

GROUP ID: 23

- Multiple Polynomial Regression

```
rm(list=ls())
f<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\all category data with mean_rating\\final\\mean_Google_.csv")
a<-f[,c(2,4:8,10:11,15)]   #a<-data[,c(2,4:8,10:11,15)]
str(f)
w <- a[,c(1:9)]
results <- fastDummies::dummy_cols(a)
w<-results[,c(2:5,7,9:41,43,45:48)]
set.seed(222)
ind <- sample(2, nrow(w), replace = T, prob = c(0.7, 0.3))
train <- w[ind==1,]
test <- w[ind==2,]

rf <- lm(train$Rating ~ poly(Reviews+Size+Installs_1+Final_Price+No_of_days_till_last_update+Category_FAMILY+Categor
library(caret)
p1 <- predict(rf, test)
x1<- mean((test$Rating-p1)^2)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))

postResample(pred = p1, obs = test$Rating)
```

Figure 6.23 Polynomial Regression on Mean Data

```
> x1<- mean((test$Rating-p1)^2)
> x1
[1] 0.2407948
```

Figure 6.24 Mean Square Error of Polynomial Regression on Mean Data

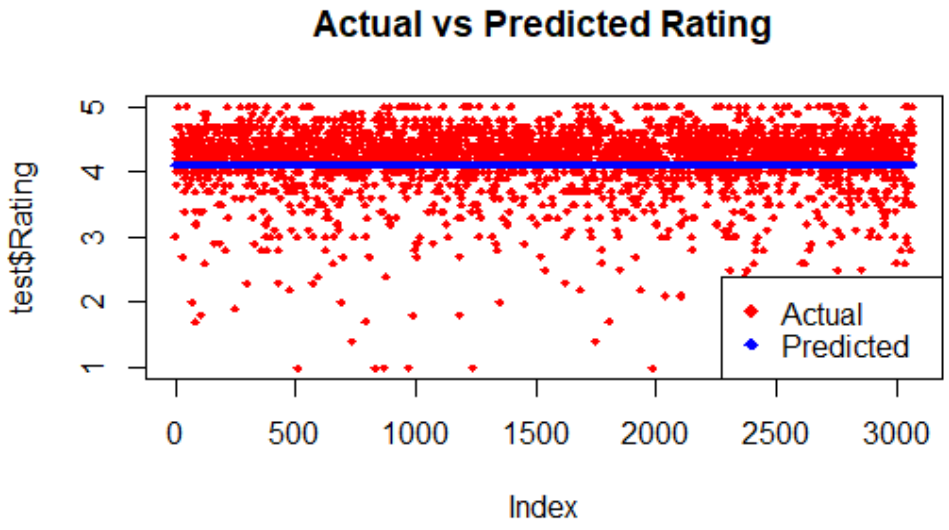

Figure 6.25 Plot of Polynomial Regression on Mean Data

```
f<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\Data Cleaning of Application Informat
a<-f[,c(2:9,12)]   #a<-data[,c(2,4:8,10:11,15)]
str(f)
w <- a[,c(1:9)]
results <- fastDummies::dummy_cols(a)
w<-results[,c(2:5,7,9,10:49)]
set.seed(222)
ind <- sample(2, nrow(w), replace = T, prob = c(0.7, 0.3))
train <- w[ind==1,]
test <- w[ind==2,]

rf <- lm(train$Rating ~ poly(Reviews+Size+Installs_1+Final_Price+No_of_days_till_last_updat
library(caret)
p1 <- predict(rf, test)
x1<- mean((test$Rating-p1)^2)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.26 Polynomial Regression on Median Data

```
> x1
[1] 0.2944918567
```

Figure 6.27 Mean Square Error of Polynomial Regression on Median Data

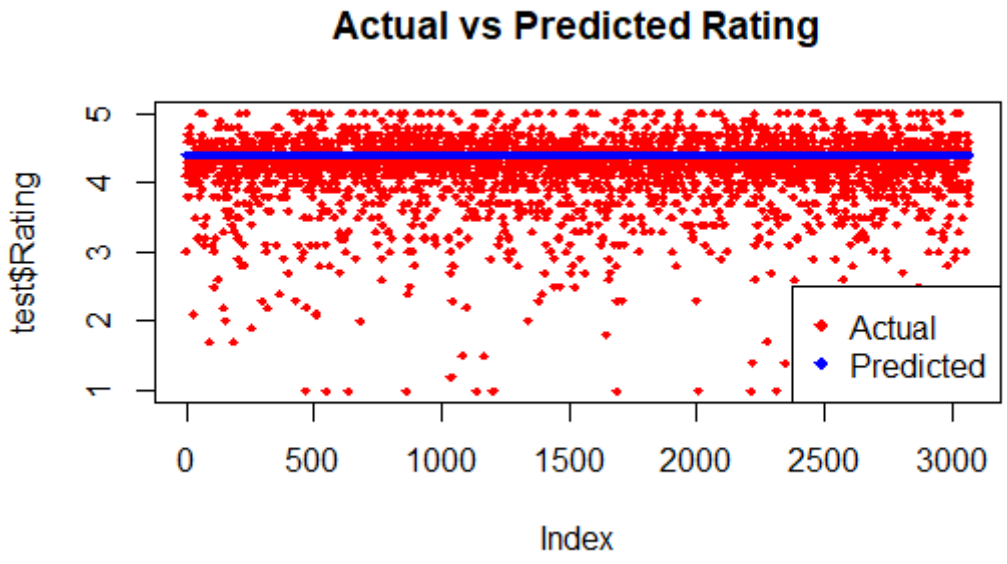

Figure 6.28 Plot of Polynomial Regression on Median Data

GROUP ID: 23

- Random Forest

```
rm(list=ls())
data<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\all category data with mean_rating\\final\\mean_Google_.csv")
#data_NaN<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\all category data with nan only\\Final all Nan values in rating

#a1<-data_NaN[,c(2:9,12)]
#results1 <- fastDummies::dummy_cols(a1)
#x1<-results1[,c(2:5,7,9,10:49)]

a<-data[,c(2,4:8,10:11,15)]
results <- fastDummies::dummy_cols(a)
x<-results[,c(2:5,7,9:49)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]
#rf <- randomForest(Rating ~ Category.Id + Reviews + Size + Installs_1 + Android_Ver + No_of_days_till_last_update + Content_
library(randomForest)
rf <- randomForest(Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, test)
#x1$Rating <- p1
#a1$Rating <- p1
#data_NaN$Rating <- p1

#write.csv(data_NaN, file = "File_info_clean.csv",row.names=FALSE)

mean((test$Rating - p1)^2)
postResample(pred = p1, obs = test$Rating)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.29 Random Forest on Mean Data

```
> mean((test$Rating - p1)^2)
[1] 0.2016968
```

Figure 6.30 Mean Square Error of Random Forest on Mean Data

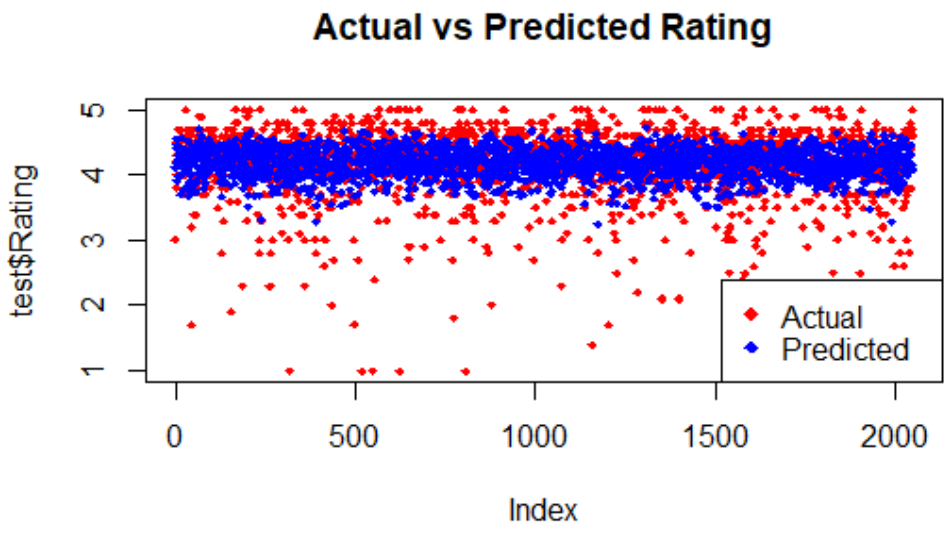

Figure 6.31 Plot of Random Forest on Mean Data

GROUP ID: 23

```
rm(list=ls())
data<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\Data Cleaning of Application Infor

a<-data[,c(2:9,12)]
results <- fastDummies::dummy_cols(a)
x<-results[,c(2:5,7,9,10:49)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]
#rf <- randomForest(Rating ~ Category.Id + Reviews + Size + Installs_1 + Android_Ver + No_o
library(randomForest)
rf <- randomForest(Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)
postResample(pred = p1, obs = test$Rating)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
```

Figure 6.32 Random Forest on Median Data

```
> mean((test$Rating - p1)^2)
[1] 0.2164535408
```

Figure 6.33 Mean Square Error of Random Forest on Median Data

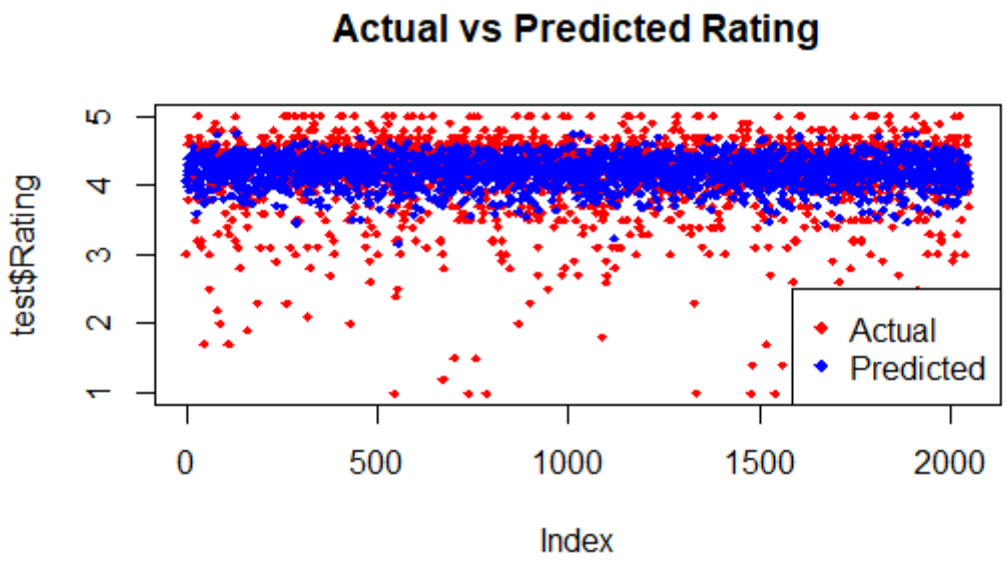

Figure 6.34 Plot of Random Forest on Median Data

GROUP ID: 23

- Support Vector Machine

```
rm(list=ls())
data<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\all category data with mean_rating\\final\\mean_Google_.csv")

a<-data[,c(2,4:8,10:11,15)]
results <- fastDummies::dummy_cols(a)

set.seed(222)
ind <- sample(2, nrow(results), replace = T, prob = c(0.7, 0.3))
train <- results[ind==1,]
test <- results[ind==2,]

library(e1071)
rf <- svm(Rating ~ ., data = train)
k <- as.matrix(train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.35 Support Vector Machine on Mean Data

```
> mean((test$Rating - p1)^2)
[1] 0.2195853
```

Figure 6.36 Mean Square Error of Support Vector Machine on Mean Data

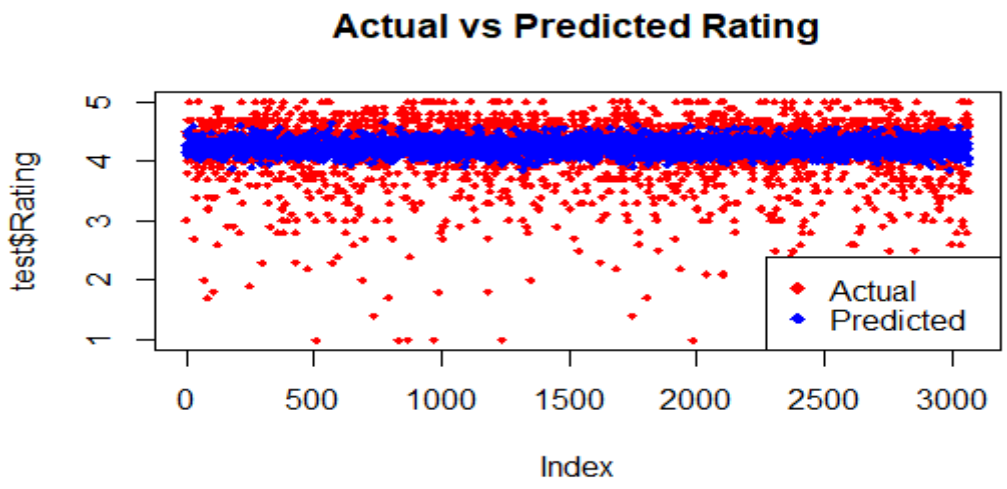

Figure 6.37 Plot of Support Vector Machine on Mean Data

GROUP ID: 23

```
data<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\Data Cleaning of Application Infor

a<-data[,c(2:9,12)]
results <- fastDummies::dummy_cols(a)
results<- results[,c(2:5,7,9,10:49)]
set.seed(222)
ind <- sample(2, nrow(results), replace = T, prob = c(0.7, 0.3))
train <- results[ind==1,]
test <- results[ind==2,]

library(e1071)
rf <- svm(Rating ~ ., data = train)
k <- as.matrix(train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
```

Figure 6.38 Support Vector Machine on Median Data

```
> mean((test$Rating - p1)^2)
[1] 0.2501026009
```

Figure 6.39 Mean Square Error of Support Vector Machine on Median Data

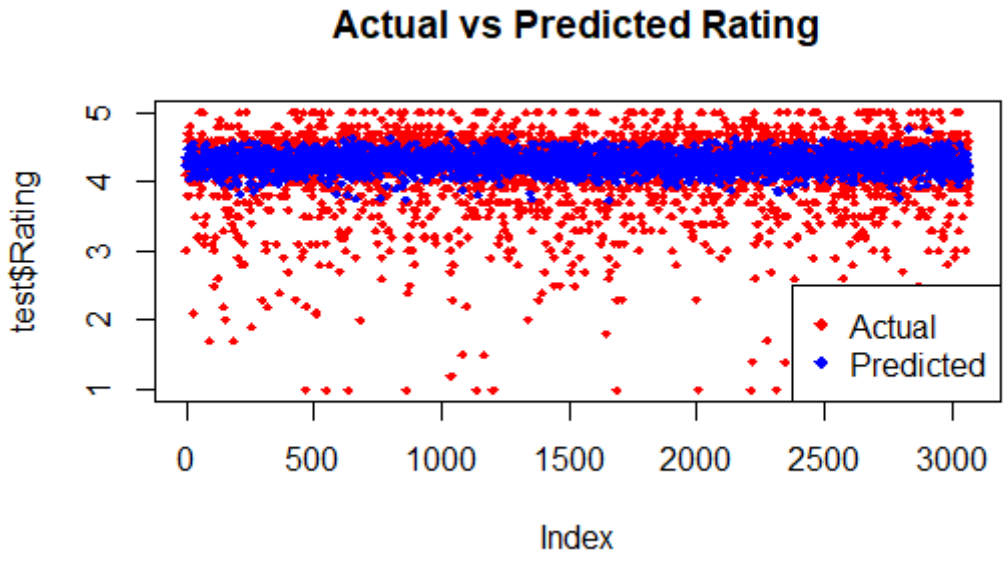

Figure 6.40 Plot of Support Vector Machine on Median Data

GROUP ID: 23

- Neural Network

```
library(keras)
library(mlbench)
library(dplyr)
library(magrittr)
library(neuralnet)
f<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\Data Cleaning of Application Information\\all category data wit
str(f)
a<-f[,c(2,4:8,10:11,15)]   #a<-data[,c(2,4:8,10:11,15)]
str(a)
results <- fastDummies::dummy_cols(a)
colnames(results)
w <- results[,c(2:5,7,9,10:49)]

n <- neuralnet(w$Rating ~ Reviews+Size+Installs_1+Final_Price+No_of_days_till_last_update+Category_FAMILY+Category_VI
               , data = w, hidden = c(10,5), linear.output = FALSE, lifesign = 'full',rep = 1)
plot(n,
     col.hidden = 'darkgreen',
     col.hidden.synapse = 'darkgreen',
     show.weights = F,
     information = F,
     fill = 'lightblue')

w <- as.matrix(w)
dimnames(w) <- NULL
set.seed(222)
ind <- sample(2, nrow(w), replace = T, prob = c(0.7, 0.3))
training <- w[ind==1,2:46]
test <- w[ind==2,2:46]
trainingtarget <- w[ind==1,1]
testingtarget <- w[ind==2,1]

m <- colMeans(training)
s <- apply(training, 2, sd)
training <- scale(training, center = m, scale = s)
test <- scale(test, center = m, scale = s)
```

```
-------------------------------------------------------------------------------
model <- keras_model_sequential()
model %>%
  layer_dense(units = 10, activation = 'relu', input_shape =c(45)) %>%
  layer_dense(units = 5, activation = 'relu') %>%
  layer_dense(units = 1)

model %>% compile(loss= 'mse',
                  optimizer = 'rmsprop',
                  metrics = 'mae')

mymodel <- model %>%
  fit(training,
      trainingtarget,
      epochs = 100,
      batch_size = 32,
      validation_split = 0.2)

model %>% evaluate(test, testingtarget)
pred <- model %>% predict(test)
mean((testingtarget-pred)^2)
plot(testingtarget,pred)
postResample(pred = pred, obs = testingtarget)
```

Figure 6.41 Neural Network on Mean Data

```
> mean((testingtarget-pred)^2)
[1] 0.2173983437
```

Figure 6.42 Mean Square Error of Neural Network on Mean Data

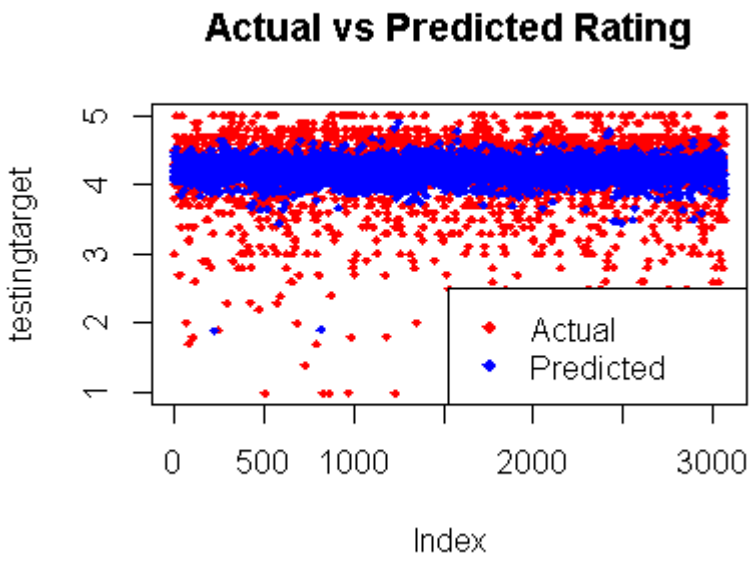

Figure 6.43 Plot of Neural Network on Mean Data

```
library(dplyr)
library(magrittr)
library(neuralnet)
f<-read.csv("C:\\Users\\elegant\\Downloads\\data_set\\Data Cleaning of Application Information\\all categ
str(f)
a<-f[,c(2:9,12)]   #a<-data[,c(2,4:8,10:11,15)]
str(a)
results <- fastDummies::dummy_cols(a)
colnames(results)
w <- results[,c(2:5,7,9,10:49)]
w[is.na(w)]<-0
n <- neuralnet(Rating ~ Reviews+Size+Installs_1+Final_Price+No_of_days_till_last_update+Category_FAMILY+Ca
               , data = w, hidden = c(10,5), linear.output = FALSE, lifesign = 'full',rep = 1)


plot(n,
     col.hidden = 'darkgreen',
     col.hidden.synapse = 'darkgreen',
     show.weights = F,
     information = F,
     fill = 'lightblue')

w <- as.matrix(w)
dimnames(w) <- NULL
set.seed(222)
ind <- sample(2, nrow(w), replace = T, prob = c(0.7, 0.3))
training <- w[ind==1,2:46]
test <- w[ind==2,2:46]
trainingtarget <- w[ind==1,1]
testingtarget <- w[ind==2,1]

m <- colMeans(training)
s <- apply(training, 2, sd)
training <- scale(training, center = m, scale = s)
test <- scale(test, center = m, scale = s)
```

```
------------------------------------------------------------------------
  model <- keras_model_sequential()
model %>%
  layer_dense(units = 100, activation = 'relu', input_shape =c(45)) %>%
  layer_dropout(rate=0.4)%>%
  layer_dense(units = 50, activation = 'relu') %>%
  layer_dropout(rate=0.3)%>%
  layer_dense(units = 20, activation = 'relu') %>%
  layer_dropout(rate=0.2)%>%
  layer_dense(units = 1)

model %>% compile(loss= 'mse',
                  #optimizer = 'rmsprop',
                  optimizer = optimizer_rmsprop(lr = 0.001),
                  metrics = 'mae')

mymodel <- model %>%
  fit(training,
      trainingtarget,
      epochs = 100,
      batch_size = 32,
      validation_split = 0.2)

model %>% evaluate(test, testingtarget)
pred <- model %>% predict(test)
m1 <- mean((testingtarget-pred)^2)
sqrt(m1)
plot(testingtarget, pred)
```

Figure 6.44 Neural Network on Median Data

```
> m1 <- mean((testingtarget-pred)^2)
> m1
[1] 0.2392144054
```

Figure 6.45 Mean Square Error of Neural Network on Median Data

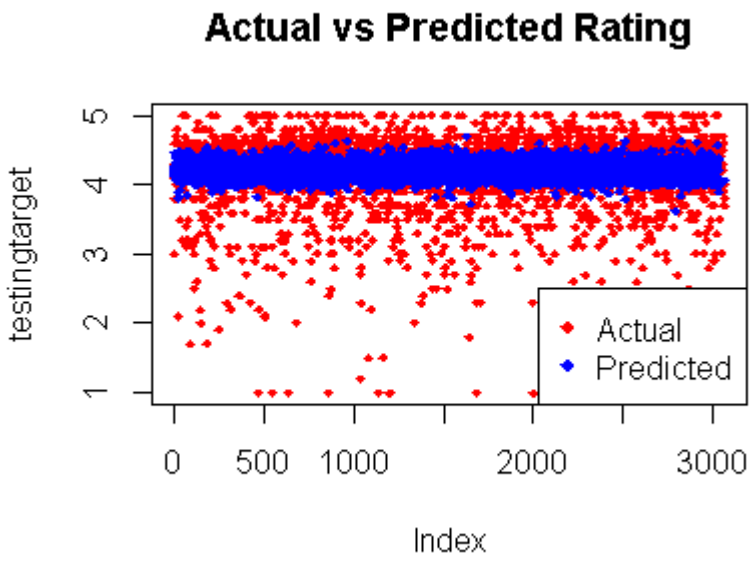

Figure 6.46 Plot of Neural Network on Median Data

GROUP ID: 23

- Replace NaN of rating with the help of the model which get low error; while using random forest on mean replaced data set we got less error as compare to other algorithm. Given below is the code for it:-

```
library(randomForest)

rm(list=ls())
data<-read.csv("mean_Google_.csv")
data_NaN<-read.csv("Google_mean_only_nan_.csv")

a1<-data_NaN[,c(2:9,12)]
results1 <- fastDummies::dummy_cols(a1)
x1<-results1[,c(2:5,7,9,10:49)]

a<-data[,c(2,4:8,10:11,15)]
results <- fastDummies::dummy_cols(a)
x<-results[,c(2:5,7,9:49)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]
#rf <- randomForest(Rating ~ Category.Id + Reviews + Size + Installs_1 + Android_Ver + No_of_days_till_last_update+ Content_Rating_Id +
library(randomForest)
rf <- randomForest(Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, x1)
x1$Rating <- p1
a1$Rating <- p1
data_NaN$Rating <- p1

write.csv(data_NaN, file = "File_info_clean.csv",row.names=FALSE)

mean((test$Rating - p1)^2)
postResample(pred = p1, obs = test$Rating)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.47 Random Forest for NAN Values of Rating

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	App	Category	Rating	Reviews	Size	Installs_1	Type	Final_Price	Content_Ra	Genres	Last_Updat	No_of_day	Current_Ve	Android_Ver
2	Word Search Tab 1 FR	FAMILY	3.531093978	0	1020	50	Paid	1.04	Everyone	Puzzle	#########	2516	1.1	3
3	BS Films	FAMILY	4.237525836	1	24000	100	Free	0	Everyone	Entertainm	#########	380	2.0.0	4.3
4	Latest 2018 Home Designs	HOUSE_AND_HOME	4.155193318	174	8700	10000	Free	0	Everyone	House & Hc	#########	154	1.1	2.3
5	BAR-B-Q Recipes	FOOD_AND_DRINK	4.2316242	0	3600	100	Free	0	Everyone	Food & Drir	#########	331	1	4
6	FK Vardar	SPORTS	4.102536098	2	26000	10	Free	0	Everyone	Sports	#########	310	1	4.1
7	Caprock Santa Fe Credit Unic	FINANCE	4.118060723	0	2900	100	Free	0	Everyone	Finance	#########	279	3.0.6	4.1
8	ec-Work	PRODUCTIVITY	4.475774687	3	26000	100	Free	0	Everyone	Productivit	#########	171	1.12	5
9	AF Nutrition - Integratori	HEALTH_AND_FITNESS	4.39736622	6	19000	100	Free	0	Everyone	Health & Fi	#########	147	1.3	4.1
10	FE IP	TOOLS	3.75660995	3	12000	1000	Free	0	Everyone	Tools	#########	426	1.51	3
11	CP Parquet	LIFESTYLE	4.410405899	2	1200	100	Free	0	Everyone	Lifestyle	#########	960	1.0.2	4
12	BL ONLINE PERSONAL TRAIN	HEALTH_AND_FITNESS	4.274806958	0	41000	5	Free	0	Everyone	Health & Fi	#########	485	BL ONLINE	4.1
13	P Icon Pack	FAMILY	4.194955302	0	21000	10	Paid	0.99	Everyone	Entertainm	#########	164	1.0.6	4
14	Voyance du pyromancien	FAMILY	4.231823746	22	16000	5000	Free	0	Everyone	Entertainm	#########	167	1.0.3	4.1
15	Bu Hangi Uygulama ?	FAMILY	4.312093398	1	24000	10	Free	0	Everyone	Puzzle	#########	296	3.2.6z	4
16	FK Utenis Utena	SPORTS	4.051192792	0	26000	5	Free	0	Everyone	Sports	#########	309	1	4.1
17	AJ Wallpapers	PERSONALIZATION	4.321374077	0	3900	100	Free	0	Everyone	Personaliza	#########	235	2	4
18	Ek-yatri: Travel where you B	TRAVEL_AND_LOCAL	4.126339509	0	29000	1	Free	0	Everyone	Travel & Lo	#########	148	1.0.0	4.4
19	CQ Ukraine	PRODUCTIVITY	4.186443487	0	9100	10	Free	0	Everyone	Productivit	#########	185	1.17.0	4.1
20	TuenMun BM	TOOLS	3.697632767	6	47000	1000	Free	0	Everyone	Tools	#########	589	1.2	4
21	DT Freight	PRODUCTIVITY	4.213925201	0	54000	1	Free	0	Everyone	Productivit	#########	393	8	4
22	Guia Bike DF	SPORTS	4.301017346	29	14000	100	Free	0	Everyone	Sports	#########	312	22	5
23	ExtendedCare Virtual Care R	MEDICAL	4.210577938	0	18000	5	Free	0	Everyone	Medical	#########	150	1	5.1
24	Ay Sabz Gunbad Waly	VIDEO_PLAYERS	4.095731947	4	3900	1000	Free	0	Everyone	Video Play	#########	472	1.1	4

Figure 6.48 NaN Rating Prediction Using Random Forest

GROUP ID: 23

6.1.4.2 Visualization of Application Information Clean Data:-

By making graphs we can compare both application information data before cleaning and after cleaning; by seeing a visualization of application information clean data we can also cross verify that if there is any unclean or noisy data is still there or not.

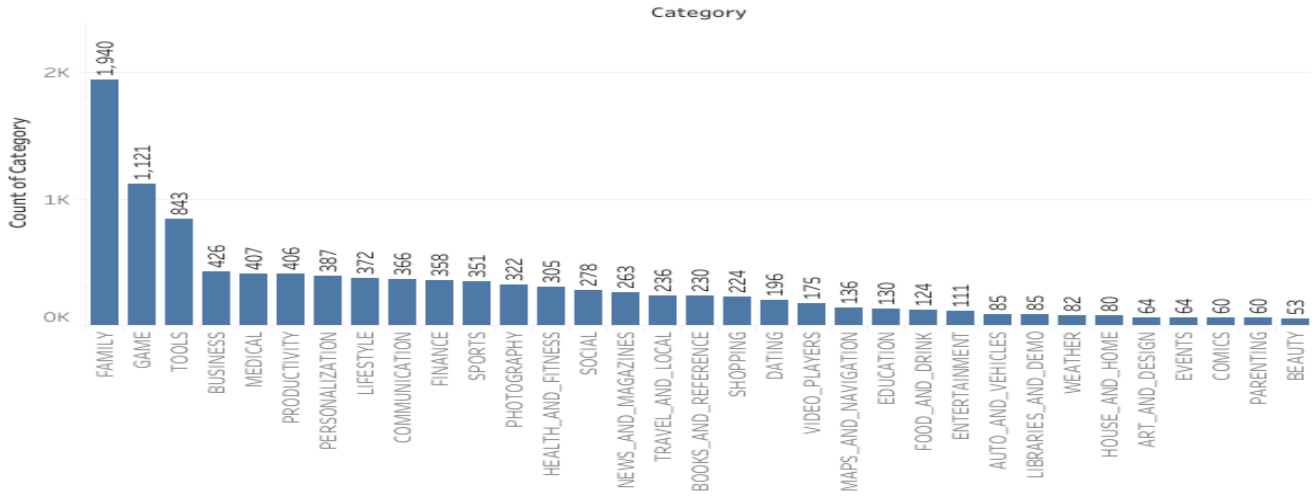

Figure 6.49 Description of Category Column in Application Information of Clean Data

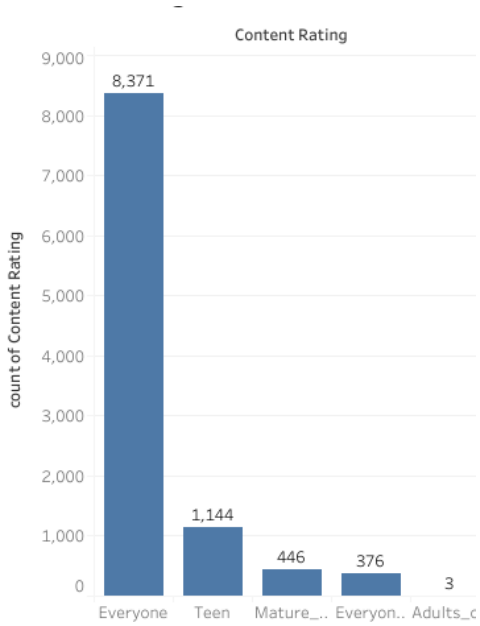

Figure 6.50 Description of Content Rating Column in Application Information of Clean Data-set

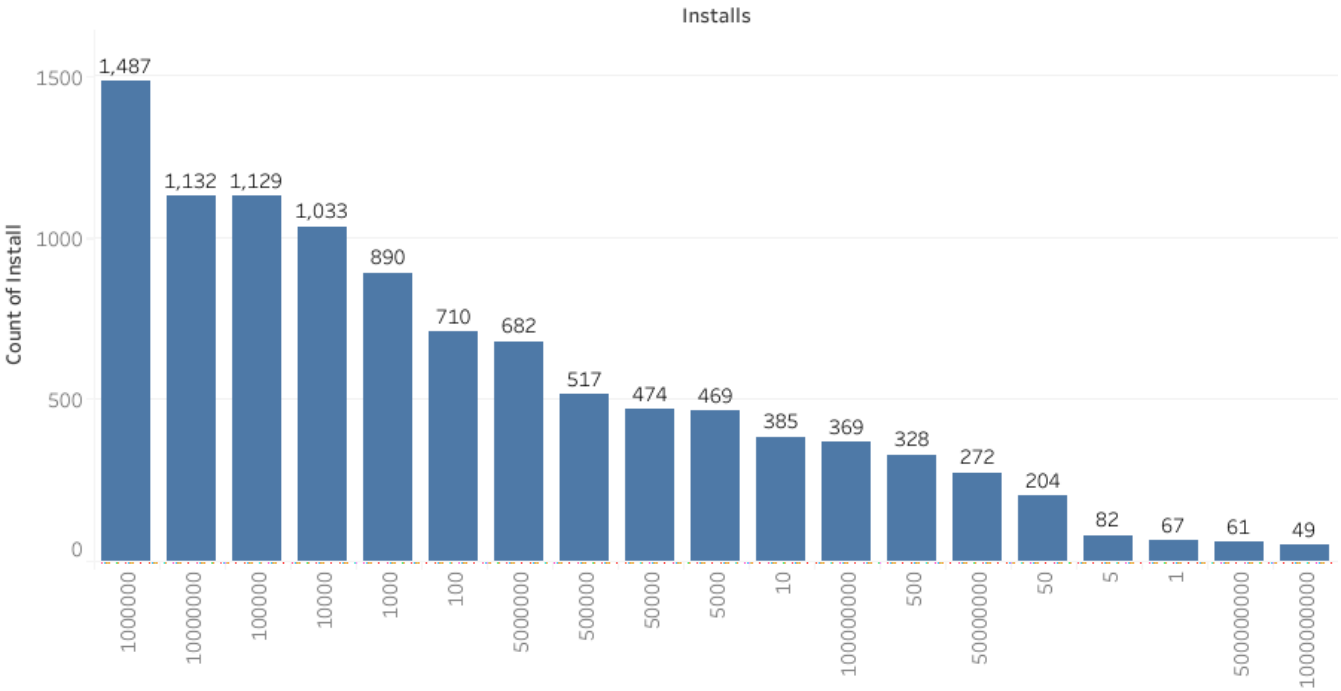

Figure 6.51 Description of Installs Column in Application Information of Clean Data-set

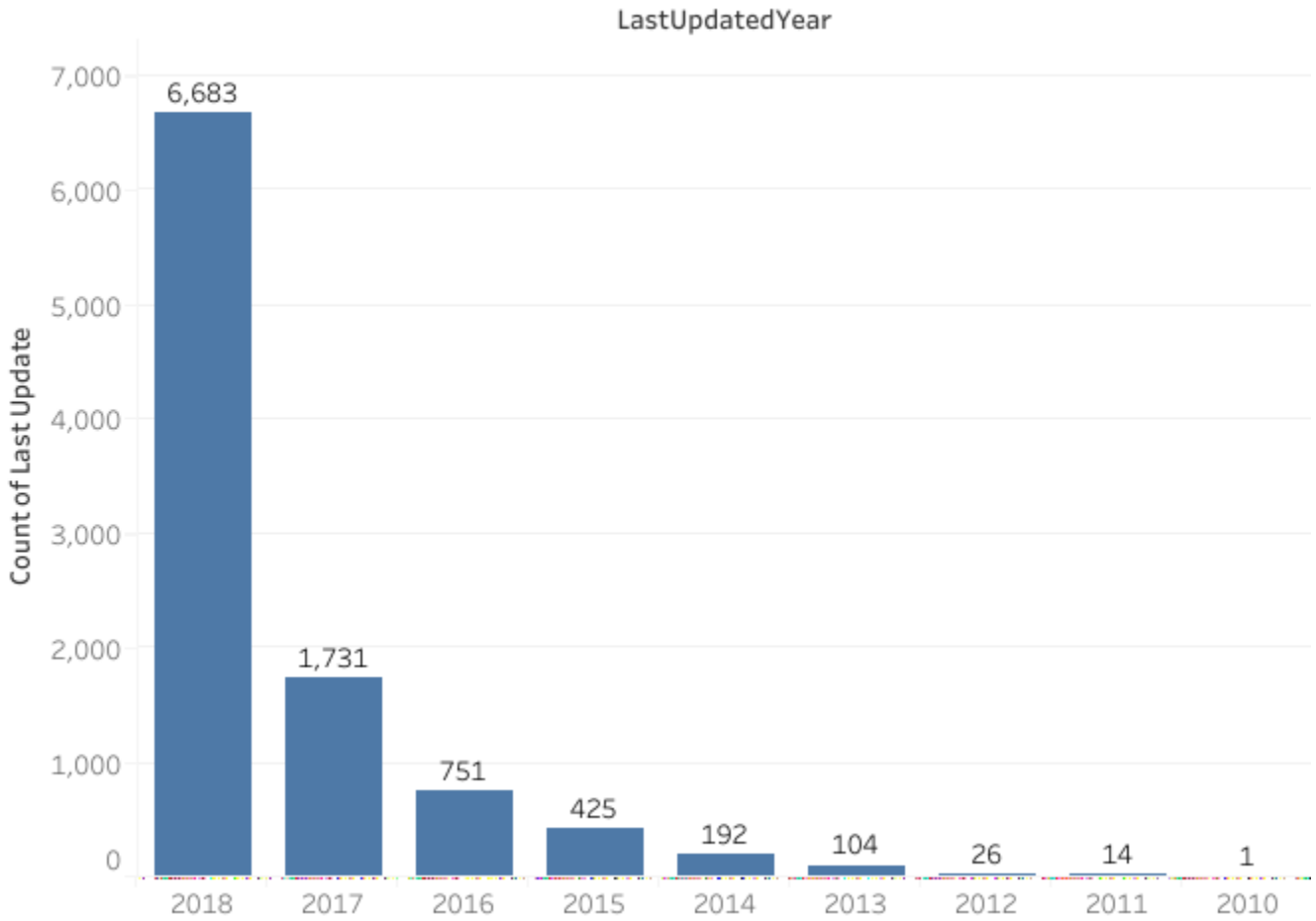

Figure 6.52 Description of Last Updated Column in Application Information of Clean Data-set

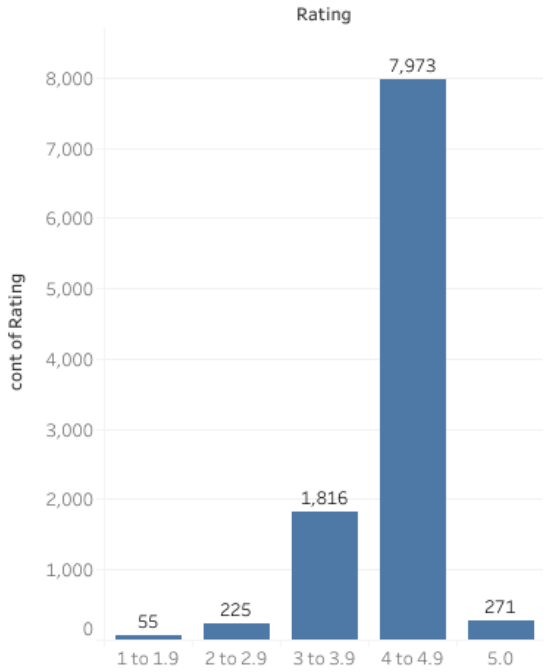

Figure 6.53 Description of Rating Column in Application Information of Clean Data-set

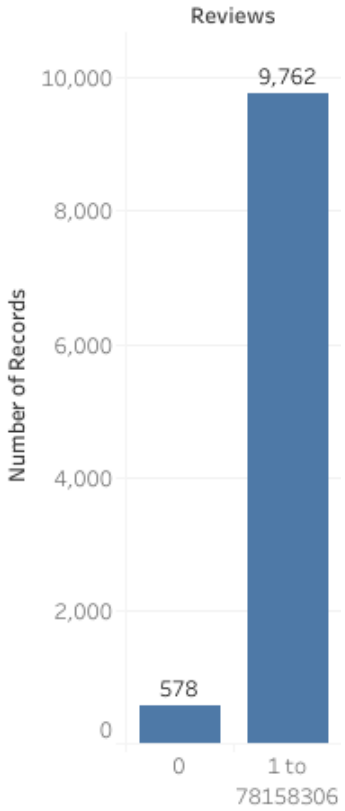

Figure 6.54 Description of Reviews Column in Application Information of Clean Data-Set

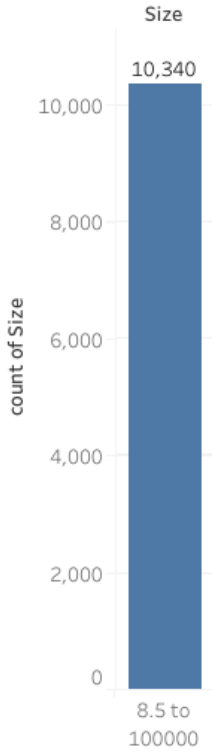

Figure 6.55 Description of Size Column in Application Information of Clean Data-set

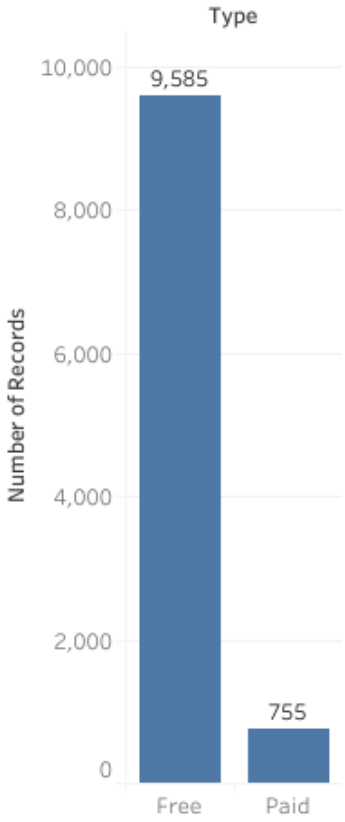

Figure 6.56 Description of Type Column in Application Information of Clean Data-set

6.1.4.3 Data Cleaning and Visualization of Original Application Review Data:-

The original reviews data was having NaN values in columns of Translated Review, Sentiment, Polarity and Subjectivity. As it was not useful for sentiment analysis so we removed it and give below is its visualization.

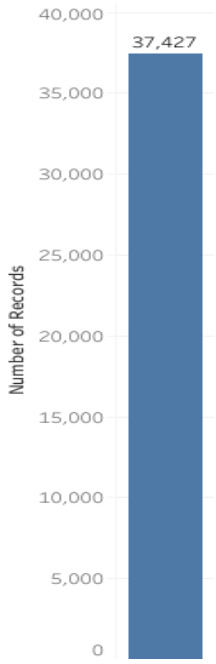

Figure 6.57: Number of Original Reviews after removing NaN data

6.1.4.4 Web Scraping of New Reviews:-

Now we are having only 37,427 no of reviews so due to that we decided to scrap the review of application from play store for that we have written a python script using selenium and pandas for scraping comments from play store; following is the process, which we have implemented through python code:-

- First, it will go to https://play.google.com/store/apps
- Then in the search bar it will search a name of the application
- After searching application name it will click on application name which matches to the text which was searched in search bar by clicking on it application page will be open.
- In the next step, it will click on full review button in the comments section due to that we can see the whole comment.
- Finally, in the last step we fetch the first few comments by considering the most helpful first filter.

GROUP ID: 23

Given below are the screenshot of code and outcome:-

```
from selenium.webdriver.common.keys import Keys
from time import sleep
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
import sys
import pandas as pd



all_apps_name = pd.read_csv("appname2.csv",encoding='utf-8')

all_apps_name.drop_duplicates()

reviews_fetched_apps = pd.read_csv("googleplaystore_user_reviews.csv")

reviews_fetched_apps.drop_duplicates()

c=all_apps_name.App.isin(reviews_fetched_apps.App).astype(int)
l=0
n=0
o=1001
#Change the number given below for changing the apps.
apps_name = all_apps_name.iloc[5333:10341,]

driver = webdriver.Chrome('C:\\Users\\admin\\Desktop\\play\\chromedriver.exe')

reviews_write = pd.DataFrame()
```

```
for index,row in apps_name.iterrows():
    try:
        print("app:",row['App'])
        if(c[o]==0):



            driver.get("https://play.google.com/store/apps")
            search_box = driver.find_element_by_class_name("gbqfif")
            search_box.send_keys(row['App'])
            search_box_btn = driver.find_element_by_class_name("gbqfb")
            search_box_btn.click()

            wait =WebDriverWait(driver,10)
            elem = wait.until(EC.element_to_be_clickable((By.LINK_TEXT,row['App'])))
            print("Script reached here.")
            app_name = driver.find_element_by_partial_link_text(row['App']).click()


            #print("Script reached here2. ")
            driver.get(driver.current_url+'&showAllReviews=true')


            #full_review_button =  driver.find_elements(By.CLASS_NAME,'LkLjZd.ScJHi.OzU4dc')
            #full_review_button = driver.find_elements(By.PARTIAL_LINK_TEXT,'Full Review')

            #full_review_button = driver.find_elements(By.XPATH,'//*[@id="fcxH9b"]/div[4]/c-wiz/div/div[2]/div

            full_review_button = driver.find_elements(By.CSS_SELECTOR,'button.LkLjZd.ScJHi.OzU4dc')
```

GROUP ID: 23

```
            for i in reviews:
                print("length:",len(reviews))
                print("Reviews:",i.text.translate(non_bmp_map))
                print("n:",n)
                df_temp = pd.DataFrame.from_records({'App':[row['App']],'Review':[i.text.translate(non_bmp_map)]})
                reviews_write = reviews_write.append(df_temp,)
                print("reviews:",reviews_write)

            l+=1
            print("l:",l)
            print("o:",o)
        o+=1
    except:
            l+=1
            print("l:",l)
            o+=1
            print("o:",o)
            continue

reviews_write.to_csv('Reviews_Scraped.csv',sep=',',encoding='utf-8',index=False)
```

Figure 6.58 Code for Review Scraping from Play Store

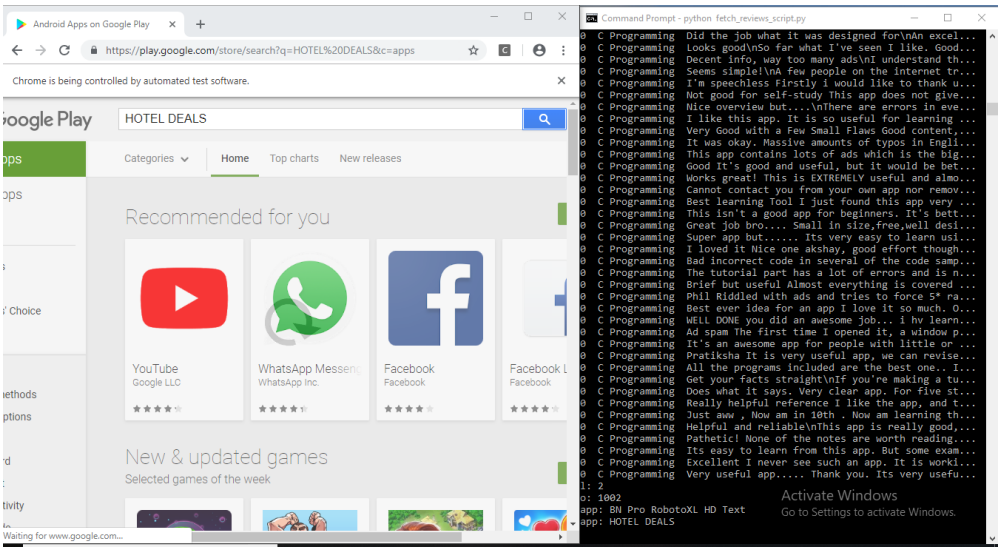

Figure 6.59 Result of Review Scraping from Play Store

6.1.4.5 Visualization of Scraped Reviews:-

By this visualization we get clear idea that number of reviews which we have scraped are 155,213.

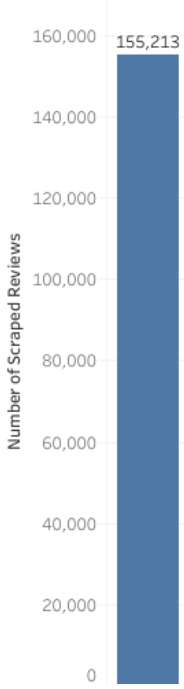

Figure 6.60: Number of Scraped Reviews

6.1.4.6 Merging and Visualization of Original Application Reviews Data and Scraped Reviews:-

We have merge Original Application Review and Scraped Reviews dataset directly in excel only. By this visualization we get to know that total number of reviews are 192,640 after adding both Scraped Reviews and Original Reviews clean review.

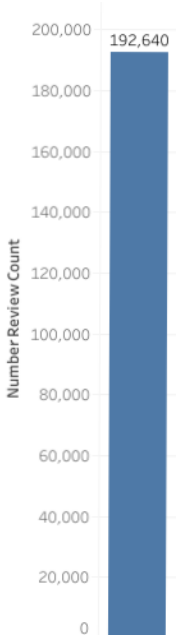

Figure 6.61: Number of Reviews After merging Original Application Reviews and Scraped Reviews

6.1.4.7 Cleaning, Sentiment Analysis with Visualization of Merged Reviews:-

For both Cleaning and sentiment analysis we have used tm library in R. Given below is the sample code for it. We have used tm.plugin.sentiment package. First Step is to calculated sentiment subjectivity and sentiment polarity for all reviews. Given below is its code.

```
library("openxlsx")
w1<-read.xlsx("C:\\Users\\Downloads\\data_set\\Data Cleaning of Reviews\\Scraped Reviews\\Final_Scraped_Reviews.xlsx",1)

library(tm)
library(devtools)
library(tm.plugin.sentiment)

corpus <- Corpus(VectorSource(w1$Review))
n1 <- TermDocumentMatrix(corpus)

sub <- subjectivity(n1)
pol <- polarity(n1)

w1$sub <- sub
w1$pol <- pol

write.xlsx(w1, file = "R_review.xlsx",row.names=FALSE)
```

Figure 6.62: Code to Calculating Sentiment Polarity and Subjectivity

Next Step is to calculated average polarity for each application and then assigned sentiment as positive, negative and neutral on the basis of polarity score. If polarity score is 0 then sentiment is neutral, polarity score is greater than 0 and less than or equal to 1 then sentiment is positive and if polarity score is between -1 to 0 then sentiment is negative. Given Below is its code and visualization.

```
library("openxlsx")
w1<-read.xlsx("C:\\Users\\Downloads\\data_set\\Data Cleaning of Reviews\\File Ger
length(unique(w1$App))
w1<-aggregate(w1,by = list(w1$App),FUN = mean)
w1 <- w1[,c(1,3,4)]
colnames(w1) <- c("App", "Sub","Pol")

w1$Sentiment<-ifelse(w1$Pol > 0,"Positive",
                     ifelse(w1$Pol == 0,"Neutral",
                            ifelse(w1$Pol < 0,"Negative","Null")))

write.xlsx(w1, file = "Agg_R_review.xlsx",row.names=FALSE)
```

Figure 6.63: Code to Calculating Average Polarity and Assigning Sentiment based on Application Name

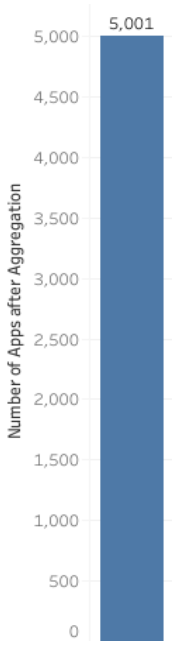

Figure 6.64: No. of Application after Aggregation

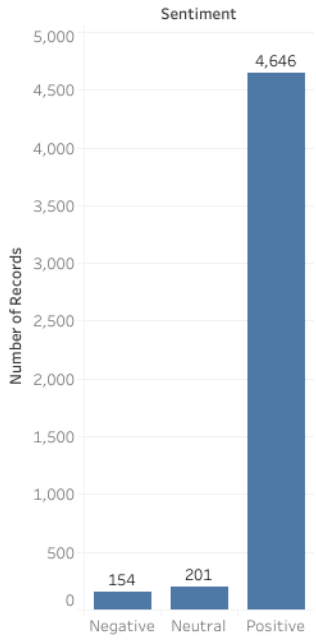

Figure 6.65: Overall Sentiment of Applications

6.1.4.8 Merging Application Reviews and Information Data and its Visualization:-

Now we are having both clean data of application information and its reviews so to make final prediction we have join both the dataset on the bases of application name. Given below is its code and its visualizations.

```
library("openxlsx")
df1 <- read.xlsx("C:\\Users\\Downloads\\data_set\\Data Cleaning of Reviews\\File Generated on the bases of R Script\\Agg_R_review.xlsx")
df2 <- read.csv("C:\\Users\\Downloads\\data_set\\Data Cleaning of Application Information\\cleaned data of application information\\Final Cleaned Data.csv")
df3 <- merge(x = df1, y = df2, by = "App", all.x = TRUE)
w2<- na.omit(df3)
#x <- unique( w1[ , 1])
#library(dplyr)
#w2 <- distinct(w1,App, .keep_all= TRUE)
write.xlsx(w2, file = "Final_Data_Both_Review_and_Information.xlsx",row.names=FALSE)
```

Figure 6.66: Code to join Cleaned Application Review and Information Data based on Application Name

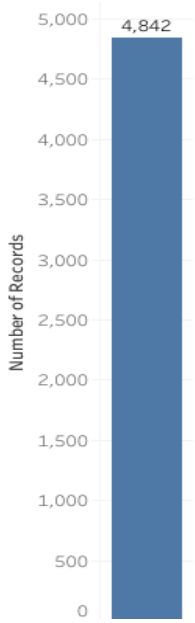

Figure 6.67: Number of Applications in Final Clean Data

In this number of application get reduced as compare to number of application in clean review dataset because there are some application which are removed from application information during cleaning process and for scraping we have taken application name from original dataset which was given us.

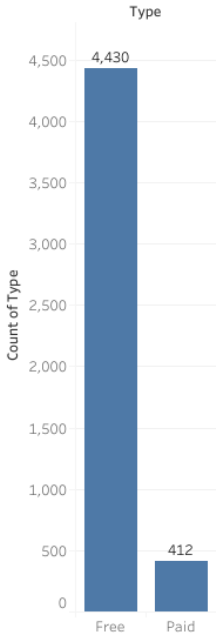

Figure 6.68: Number of Free and Paid Application in Final Clean Data

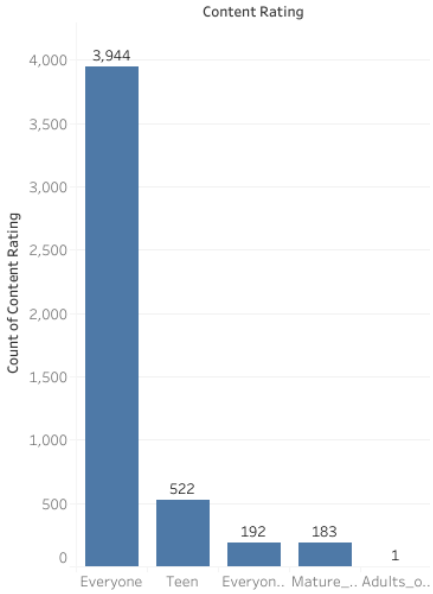

Figure 6.69: Content Rating Wise Number of Application in Final Clean Data

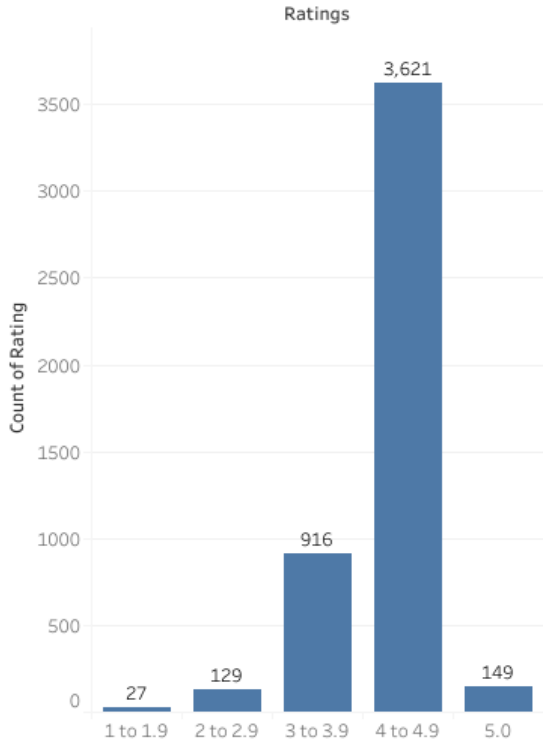

Figure 6.70: Rating wise Number of Application in Final Clean Data

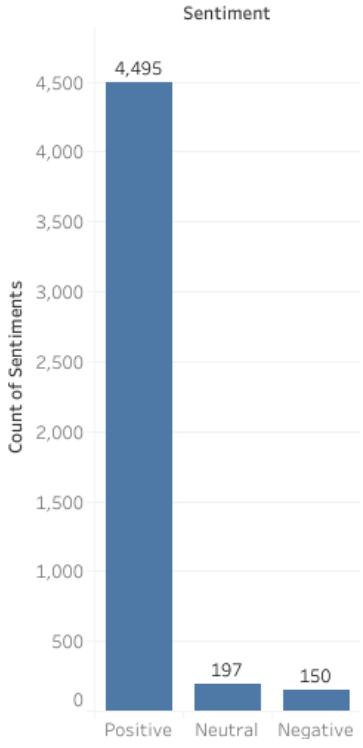

Figure 6.71: Sentiment wise Number of Application in Final Clean Data

GROUP ID: 23

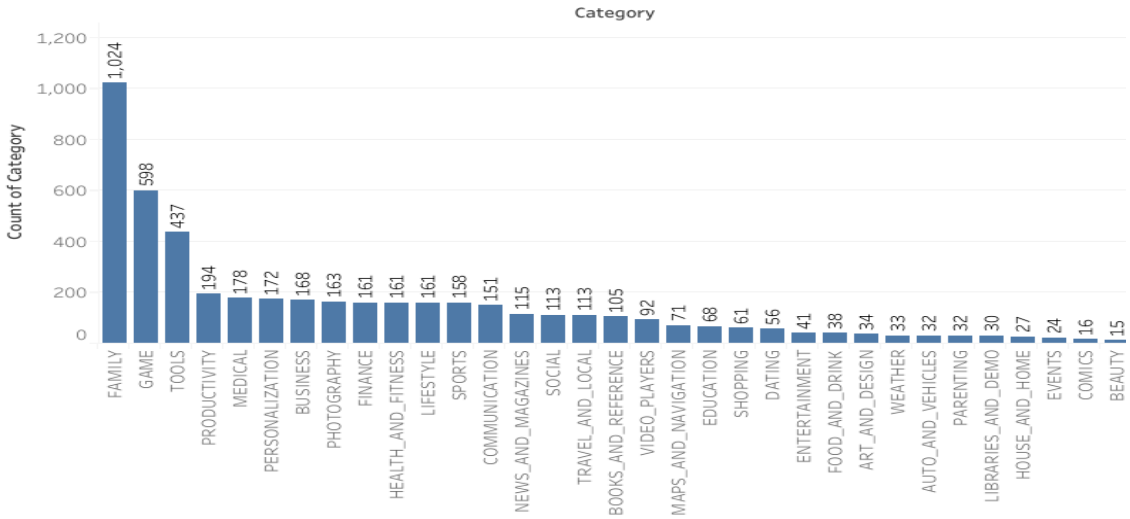

Figure 6.72: Category Wise Number of Application in Final Clean Data

6.1.4.9 Feature Selection in Final Clean Data:-

Selected algorithm: - Boruta feature Selection

The reason for using Boruta feature selection algorithm and it code is given in section 6.1.4.1 and figure 6.15, respectively so kindly, refer it for more information on Boruta feature selection algorithm. Given below is the outcome of this feature selection

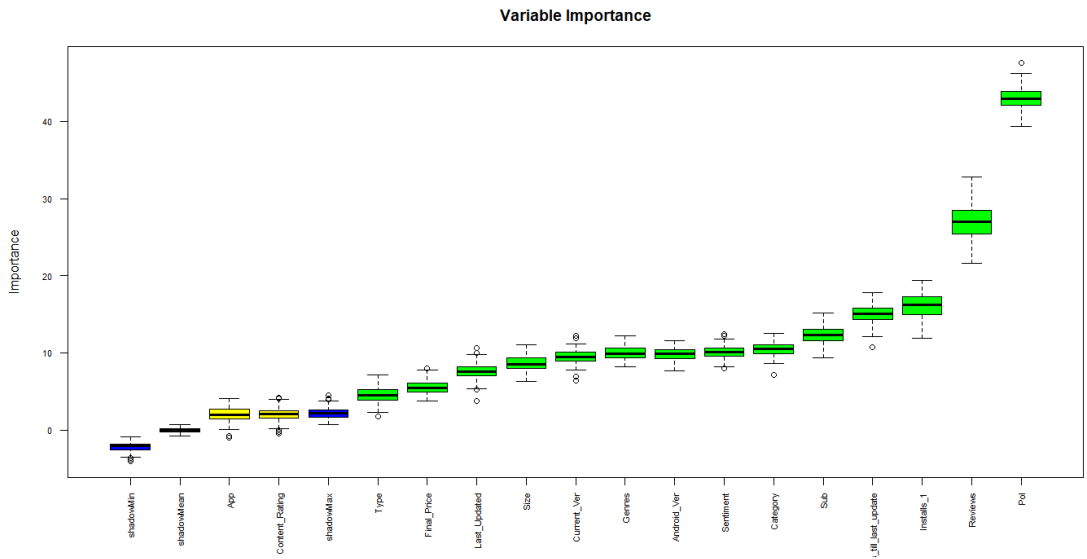

Figure 6.73: Outcome of Feature Selection based on Final Clean Data

Selected features for all model are:-

- Category Name
- Installs
- Content Rating
- Sentiment
- Reviews
- Type
- Polarity
- Size
- Final Price
- No of days till last update

6.1.4.10 Prediction of Rating on clean data:-

Here, we have used total 5 algorithms for prediction of rating and selected a best fit algorithm on the basis of low mean square error; features which are confirmed through the Boruta algorithm and our basic logic are used as an input parameters for prediction.

Following are the list of algorithms along with its code and outcome:-

- Multiple Linear Regression

```
rm(list=ls())
library("openxlsx")
data<-read.xlsx("C:\\Users\\Downloads\\data_set\\Final Clean and Merge Data of Information and Review\

a<-data[,c(3:12,15)]
results <- fastDummies::dummy_cols(a)
x<-results[,c(1,4:7,9,11:13,15:46,48,50:53)]
#x<-results[,c(1:2,5:8,10,12:14,16:47,49,51:54)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]

rf <- lm(train$Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)
library(Metrics)
mse(test$Rating, p1)


postResample(pred = p1, obs = test$Rating)
```

Figure 6.74: Multiple Linear Regression Code for final Clean Data

```
> mean((test$Rating - p1)^2)
[1] 0.2188624
```

Figure 6.75: Mean Square Error of Multiple Linear Regression for Final Clean Data

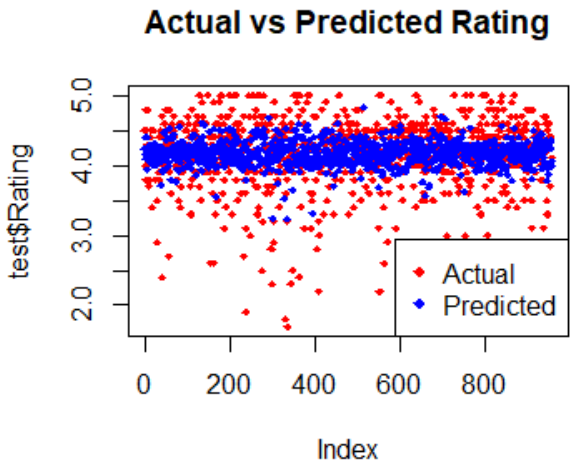

Figure 6.76: Plot of Multiple Linear Regression for Final Clean Data

GROUP ID: 23

- Multiple Polynomial Regression

```
rm(list=ls())
#f<-read.csv("C:\\Users\\Downloads\\data_set\\Data Cleaning of Application Information\\all category data with median_r

library(openxlsx)
data<-read.xlsx("C:\\Users\\Downloads\\data_set\\Final Clean and Merge Data of Information and Review\\Final_Data_Both_


#a<-f[,c(2:9,12)]
a<-data[,c(3:12,15)]
#str(f)
w <- a[,c(1:9)]
results <- fastDummies::dummy_cols(a)
w<-results[,c(1,4:7,9,11:12,14:46,48,50:53)]
set.seed(222)
ind <- sample(2, nrow(w), replace = T, prob = c(0.7, 0.3))
train <- w[ind==1,]
test <- w[ind==2,]

rf <- lm(train$Rating ~  poly( Pol+Reviews+Size + Installs_1 + Final_Price + No_of_days_till_last_update + Sentiment_P

library(caret)
p1 <- predict(rf, test)
x1<- mean((test$Rating-p1)^2)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.77: Polynomial Regression Code for Final Clean Data

```
> x1<- mean((test$Rating-p1)^2)
> x1
[1] 0.2605608
```

Figure 6.78: Mean Square Error of Polynomial Regression for Final Clean Data

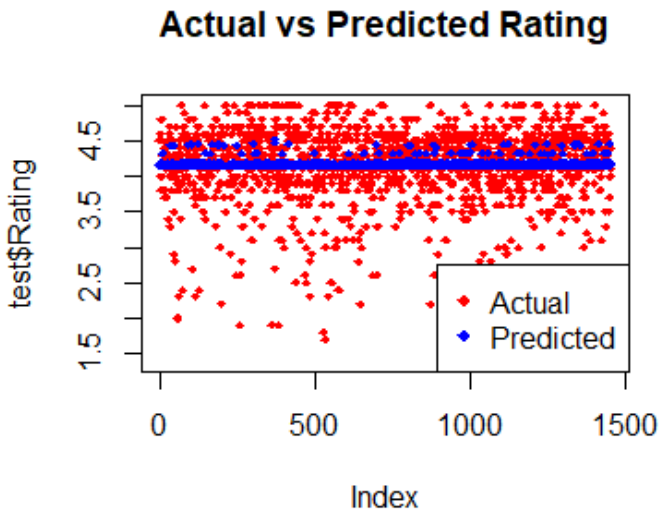

Figure 6.79: Plot of Polynomial Regression for Final Clean Data

GROUP ID: 23

- Random Forest

```
rm(list=ls())
library("openxlsx")
data<-read.xlsx("C:\\Users\\Downloads\\data_set\\Final Clean and Merge Data of Information and Review\\Final_Data_Both
#data_NaN<-read.csv("C:\\Users\\Downloads\\data_set\\all category data with nan only\\Final all Nan values in rating\'

#a1<-data_NaN[,c(2:9,12)]
#results1 <- fastDummies::dummy_cols(a1)
#x1<-results1[,c(2:5,7,9,10:49)]

a<-data[,c(3:12,15)]
results <- fastDummies::dummy_cols(a)
x<-results[,c(1,4:7,9,11:54)]

set.seed(222)
ind <- sample(2, nrow(x), replace = T, prob = c(0.8, 0.2))
train <- x[ind==1,]
test <- x[ind==2,]
#rf <- randomForest(Rating ~ Category.Id + Reviews + Size + Installs_1 + Android_Ver + No_of_days_till_last_update + (
library(randomForest)
rf <- randomForest(Rating ~ ., data = train)
library(caret)
p1 <- predict(rf, test)
#x1$Rating <- p1
#a1$Rating <- p1
#data_NaN$Rating <- p1

#write.csv(data_NaN, file = "File_info_clean.csv",row.names=FALSE)

mean((test$Rating - p1)^2)
postResample(pred = p1, obs = test$Rating)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.80: Random Forest Code for Final Clean Data

```
> mean((test$Rating - p1)^2)
[1] 0.1878426
```

Figure 6.81: Mean Square Error of Random Forest for Final Clean Data

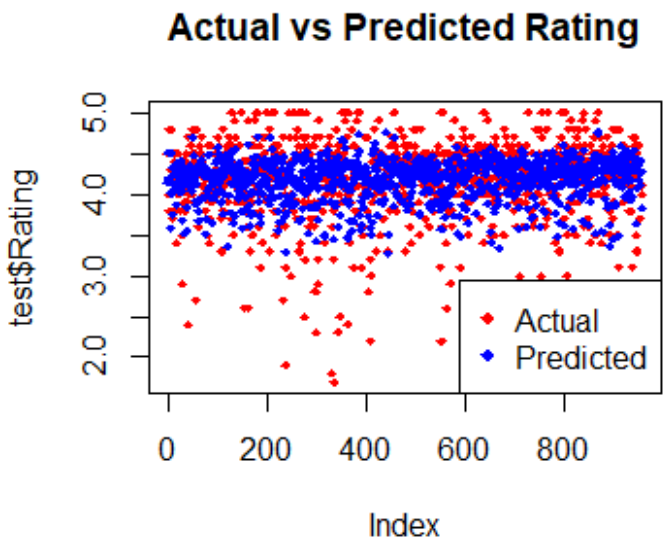

Figure 6.82: Plot of Random Forest for Final Clean Data

GROUP ID: 23

- Support Vector Machine

```
rm(list=ls())
library("openxlsx")
data<-read.xlsx("C:\\Users\\Downloads\\data_set\\Final Clean and Merge Data of Information and Review\\Final_Dat

a<-data[,c(3:12,15)]
results <- fastDummies::dummy_cols(a)
a<-results[,c(1,4:7,9,11:54)]

set.seed(222)
ind <- sample(2, nrow(results), replace = T, prob = c(0.7, 0.3))
train <- a[ind==1,]
test <- a[ind==2,]

library(e1071)
rf <- svm(Rating ~ ., data = train)
k <- as.matrix(train)
library(caret)
p1 <- predict(rf, test)
mean((test$Rating - p1)^2)

plot(test$Rating,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(p1,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))

postResample(pred = p1, obs = test$Rating)
```

Figure 6.83: Support Vector Machine Code for Final Clean Data

```
> mean((test$Rating - p1)^2)
[1] 0.2463433
```

Figure 6.84: Mean Square Error of Support Vector Machine for Final Clean Data

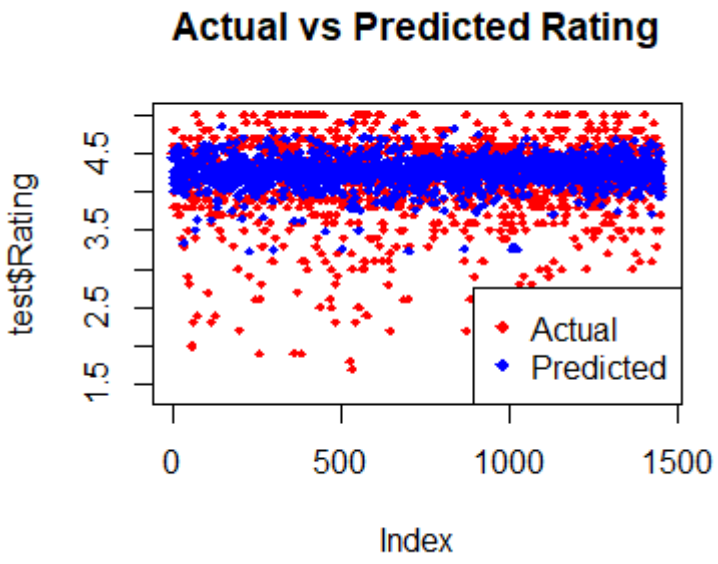

Figure 6.85: Plot of Support Vector Machine for Final Clean Data

GROUP ID: 23

- Neural Network

```
rm(list=ls())
library(keras)
library(mlbench)
library(dplyr)
library(magrittr)
library(neuralnet)
library("openxlsx")
f<-read.xlsx("C:\\Users\\Downloads\\data_set\\Final Clean and Merge Data of Information and Review\\Final_Data_Both_Rev
str(f)
a<-f[,c(3:12,15)]   #a<-data[,c(2,4:8,10:11,15)]
str(a)
results <- fastDummies::dummy_cols(a)
colnames(results)
w <- results[,c(1,4:7,9,11:54)]
w[is.na(w)]<-0
n <- neuralnet(Rating ~ Pol+Reviews+Size+Installs_1+Final_Price+No_of_days_till_last_update+Sentiment_Positive+Sentimer
               , data = w, hidden = c(10,5), linear.output = FALSE, lifesign = 'full',rep = 1)
plot(n,
     col.hidden = 'darkgreen',
     col.hidden.synapse = 'darkgreen',
     show.weights = F,
     information = F,
     fill = 'lightblue')

w <- as.matrix(w)
dimnames(w) <- NULL
set.seed(222)
ind <- sample(2, nrow(w), replace = T, prob = c(0.7, 0.3))
training <- w[ind==1,c(1,3:50)]
test <- w[ind==2,c(1,3:50)]
trainingtarget <- w[ind==1,2]
testingtarget <- w[ind==2,2]
```

```
m <- colMeans(training)
s <- apply(training, 2, sd)
training <- scale(training, center = m, scale = s)
test <- scale(test, center = m, scale = s)
-------------------------------------------------------------------------------
library(keras)
library(tensorflow)
model <- keras_model_sequential()
model %>%
  layer_dense(units = 5, activation = 'relu', input_shape =c(49)) %>%
  layer_dense(units = 1)

model %>% compile(loss= 'mse',
                  optimizer = 'rmsprop',
                  metrics = 'mae')

mymodel <- model %>%
  fit(training,
      trainingtarget,
      epochs = 100,
      batch_size = 32,
      validation_split = 0.2)

model %>% evaluate(test, testingtarget)
pred <- model %>% predict(test)
mean((testingtarget-pred)^2)
plot(testingtarget,pred)
postResample(pred = pred, obs = testingtarget)

plot(testingtarget,col='red',main='Actual vs Predicted Rating',pch=18,cex=0.7)
points(pred,col='blue',pch=18,cex=0.7)
legend('bottomright',legend=c('Actual','Predicted'),pch=18,col=c('red','blue'))
```

Figure 6.86: Neural Network Code for Final Clean Data

```
> mean((testingtarget-pred)^2)
[1] 0.2465095941
```

Figure 6.87: Mean Square Error of Neural Network for Final Clean Data

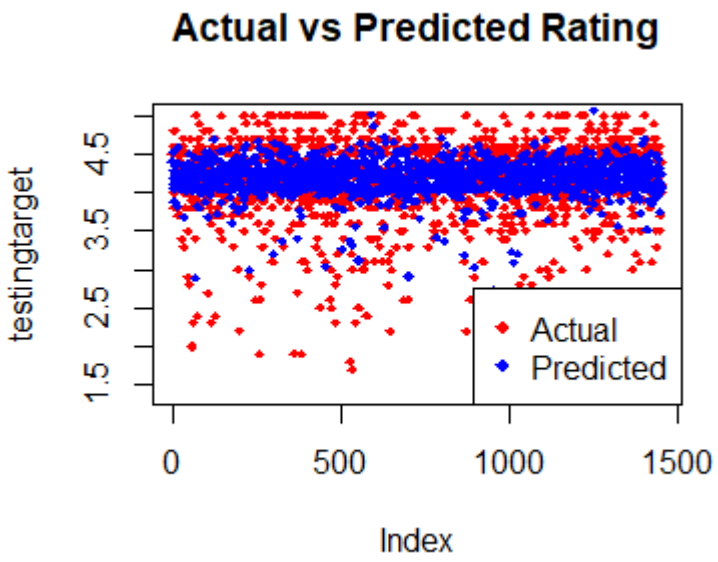

Figure 6.88: Plot of Neural Network for Final Clean Data

6.1.4.11 Selection of Best Fit Model with Visualization:-

We have predict rating by various model which are discuss above but in random forest we have very less mean square error as compare to other mode which is clearly seen in below visualization so we have consider random forest as our final rating prediction.

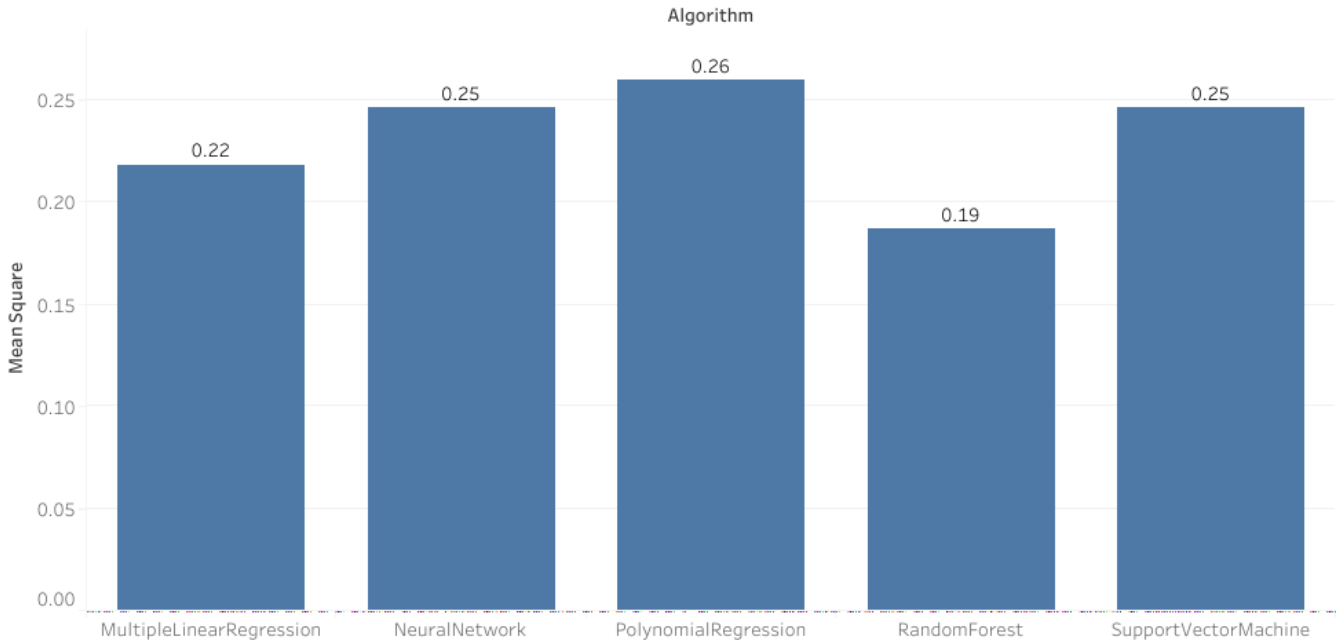

Figure 6.89: Comparison of All Algorithm Mean Square Error

CHAPTER: 7 CONCLUSION AND FUTURE WORK

CHAPTER 7 CONCLUSION AND FUTURE WORK

Conclusion

To reiterate my views, as we all know that the ratio of mobile application development has rapidly increased in the last few years. Before developing a mobile application if we can know the approximate rating with the help of its properties which we are going to consider while development then by this we can conclude that will our application will become a successful app in future? If not then we can also know what kind of properties will help to make it popular in the real world. We are developing a UI based application which will predict approximate rating with that help of properties of application and this UI based application will work only on the application which is going to launch on Google Play Store.

Future work

Till now we have clean application properties data and on the other side, we have also scrap reviews from Google Play Store. Moreover, we have also done data visualization for original data-set. Following are the task which we are going to implement now.

- Pre-processing of reviews
- Sentiment analysis
- Merging of cleaned application properties and reviews data.
- Feature selection

CHAPTER: 8 REFERENCES

CHAPTER 8 REFERENCES

1) https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/
2) https://towardsdatascience.com/the-tale-of-missing-values-in-python-c96beb0e8a9d
3) https://www.listendata.com/2017/05/feature-selection-boruta-package.html
4) https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/
5) https://www.machinelearningplus.com/machine-learning/feature-selection/