Google Play Project Report - CIS 545

Gabriella Bongiovanni & Syed Shaheryar Qadir

Google Play Project Report

This project aims to prescribe a specific price point at which an app should be priced in order to maximize the number of installs.

Two variables from the dataset will be used in this prescription, Price and Installs.

Data consists of over 10000 rows and was found on Kaggle.

```r
# to install package and call the library
#install.packages("RMySQL")
library(RMySQL)

## Warning: package 'RMySQL' was built under R version 3.5.2

## Loading required package: DBI

# to connec to the database hosted with the login credentials
con = dbConnect(MySQL(),
                user = 'root', password ="",
                dbname = 'googleplaystore', host = 'localhost')

# to send a query to get all rows from table : 'googleplaystore'
result = dbSendQuery(con,"SELECT*FROM googleplaystore")

#to establish a data frame with the retrieved result
googledata = fetch(result, n = -1)
#print(googledata)

#tidy data
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

clean.googledata = googledata %>% select(Category, Rating, Reviews, Installs,
Price)
```

In order to obtain an accurate prescription, first the data will be tidied. The original dataset had several rows in different formats which created difficulties for R to understand the numeric values and instead treated the values as characters rather than numeric.

```
#clean the data

#to specify value type to each column of the dataset
clean.googledata = na.omit(clean.googledata)

clean.googledata$Category = as.character(clean.googledata$Category)
clean.googledata$Rating = as.numeric(clean.googledata$Rating)
clean.googledata$Reviews = as.integer(clean.googledata$Reviews)
clean.googledata$Installs = as.numeric(clean.googledata$Installs)
clean.googledata$Price = as.numeric(clean.googledata$Price)
print(str(clean.googledata))

## 'data.frame':    10841 obs. of  5 variables:
##  $ Category: chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN"
"ART_AND_DESIGN" ...
##  $ Rating  : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
##  $ Reviews : int  159 967 87510 215644 967 167 178 36815 13791 121 ...
##  $ Installs: num  1e+04 5e+05 5e+06 5e+07 1e+05 5e+04 5e+04 1e+06 1e+06
1e+04 ...
##  $ Price   : num  0 0 0 0 0 0 0 0 0 0 ...
## NULL
```

Once the data is tidied, it will be randomly split by R to create a testing and training set.

```
# to define the training and the testing set
#install.packages("caTools")
library(caTools)

## Warning: package 'caTools' was built under R version 3.5.2

set.seed(100)
split <- sample.split(clean.googledata$Price,SplitRatio = 0.8)
training_set<- subset(clean.googledata,split==TRUE)
test_set<- subset(clean.googledata,split==FALSE)
```

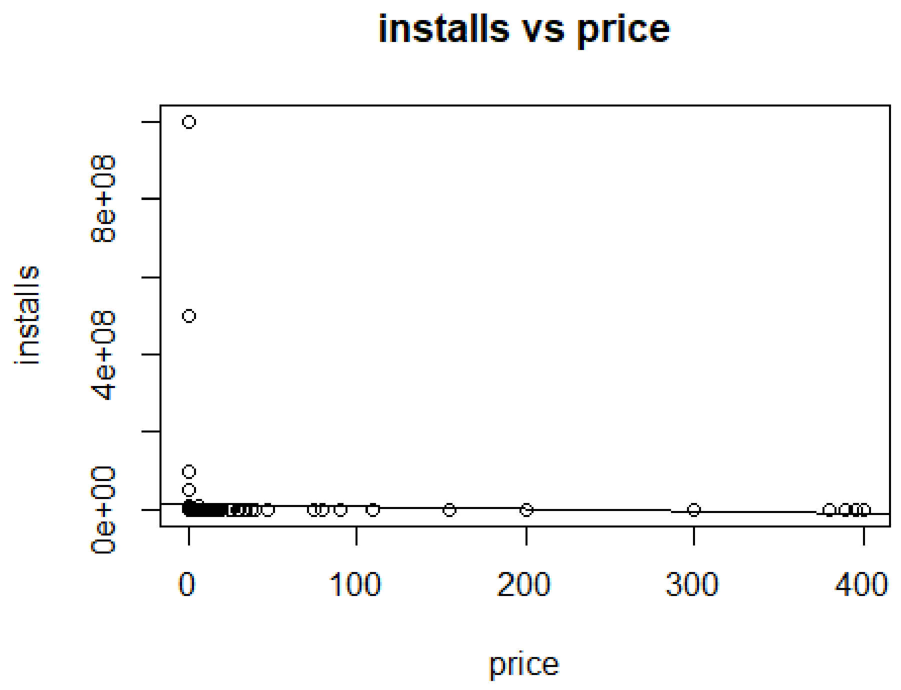
To see if a linear model can be used, the correlation between Price and Installs must first be determined. If the correlation is not significant, an alternate method must be used.

```
#see if there is a linear correlation between Price of app and Installs
cor(clean.googledata$Price,clean.googledata$Installs)

## [1] -0.01168837
```

According to the function, there is no linear correlation.

Below the data can be visualized, and perhaps a trend can be spotted. It is evident from the plot that the highest number of installs happened when the price was lowest.

```
model <- glm(Installs ~ Price, data = clean.googledata)

plot(x = clean.googledata$Price, y = clean.googledata$Installs,
   xlab = "price",
    ylab = "installs",
     main = "installs vs price",
   abline(model)
)
```



####Since the inear model is not signifcant, next a non linear model will be attempted. ####Below is the code for a non-linear least squares model.

```
m2<-nls(Installs ~ Price*a + b, data = clean.googledata, start = c(a=0,b=0))

summary(m2)

##
## Formula: Installs ~ Price * a + b
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a   -62312      51203  -1.217    0.224
## b 15526924     818285  18.975   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85020000 on 10839 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 6.076e-11
```

This model proved much more successful and the p-value is less than 0.05.

Due to this model's accuracy, it will be used to predict the number of installs if a company was to start selling an app and priced it at a certain point.

```
#set target to obtain prediction
target = data.frame(Price = 0)
result = predict(m2,target)
cat("Number of installs if priced at $0:", result)

## Number of installs if priced at $0: 15526924

target = data.frame(Price = 0.5)
result = predict(m2,target)
cat("Number of installs if priced at $0.50:", result)

## Number of installs if priced at $0.50: 15495768

target = data.frame(Price = 1)
result = predict(m2,target)
cat("Number of installs if priced at $1:", result)

## Number of installs if priced at $1: 15464612

target = data.frame(Price = 10)
result = predict(m2,target)
cat("Number of installs if priced at $10:", result)

## Number of installs if priced at $10: 14903805
```

As proven in this predictive model, the higher an app is priced, the number of installs will be negatively affected.

If a company's goal is to maximize the number installs, the prescribed price according to this model is $0.