insy 5378: project 2

Pokémon Go! Analytics

Kartik Chavan

Roma Patil

Vidyalakshmi Krishnakumar

Table of Contents

[1. Introduction 2](#_Toc481773415)

[2. Data Description: 2](#_Toc481773416)

[3. Project Execution 2](#_Toc481773417)

[3.1 Web Scraping 2](#_Toc481773418)

[3.2 Data Organization 3](#_Toc481773419)

[3.3 Data Exploration 4](#_Toc481773420)

[3.4 Prediction Model: 10](#_Toc481773421)

[3.5 Deep Learning: 14](#_Toc481773422)

[4. References 26](#_Toc481773423)

# 1. Introduction

In this project, we want to understand the success of Pokémon Go, a mobile app game. Pokémon Go is a free-to-play, location-based augmented reality game developed by Niantic for iOS and Android devices. This guide us towards the first step of the project that is, data collection. For this project, the data was in the form of HTML files which have been made available to us and the PLATFORM for these files is either “android” or “iOS”. The purpose of this project is

(1) to do web scraping using BeautifulSoup,

(2) to construct a Pandas dataframe

(3) to explore/visualize the numeric data using matplotlib or seaborn

(4) to use sklearn to build machine learning models to predict the app’s review counts

(5) to analyze the app’s screenshot images using deep learning with tensorflow.

# 2. Data Description:

The data for this project has been obtained by downloading app pages of Pokémon Go! from Google Play Store and Apple App Store from July 21, 2016 to October 31, 2016. The webpages were downloaded every ten minutes. This means that there are 144 HTML files for a given day and a given platform. The HTML files are contained in 103 date folders.

We downloaded the zip file containing the date folders from the following link: [http://diamond.mccombs.utexas.edu/insy5378/pokemon\_5378.zip](http://diamond.mccombs.utexas.edu/insy5378/pokemon_537)

Each date folder contains HTML files downloaded in the specified date. Each HTML file name is formatted as “HH\_MM\_pokemon\_PLATFORM.html”, where HH is hour, MM is minute, and PLATFORM is either “android” or “iOS”.

# 3. Project Execution

## 3.1 Web Scraping

Web scraping is a computer software technique of extracting information from websites. This step illustrates to extract specific values from the downloaded HTML files for Android and iOS files. It can be executed using various Python modules and we have used modules such as BeautifulSoup and Urllib in part of project.

1. **BeautifulSoup:** BeautifulSoup automatically converts incoming documents to Unicode and outgoing documents to UTF-8.
2. **Urllib:** This module provides a high-level interface for fetching data across the World Wide Web. In particular, the [**urlopen()**](https://docs.python.org/2/library/urllib.html#urllib.urlopen) function is similar to the built-in function [**open()**](https://docs.python.org/2/library/functions.html#open), but accepts Universal Resource Locators (URLs) instead of filenames.

Values extracted from all:

1. **iOS Pages includes** :(i) Number Of customer ratings in the Current Version (let’s call it ios\_current\_ratings)

(ii) number of customer ratings in All Versions (ios\_all\_ratings) and

(iii) file size in MB (ios\_file\_size).

For an example, the output we obtained for 2016-07-29 at 20:00:00 for an iOS page

is [20376, 120971, 110] for ios\_current\_ratings, ios\_all\_ratings and ios\_file\_size respectively.

2.**Android Pages includes** :(i) average rating (in the scale between 1.0 and 5.0) (android\_avg\_rating)

(ii)number of total ratings (android\_total\_ratings)

(iii)number of ratings for 1-5 stars (android\_ratings\_1, android\_ratings\_2, …, android\_ratings\_5)

(iv) file size in MB (android\_file\_size)

For an example, the output we obtained for 2016-07-29 at 20:00:00 for an Android page

Is ['3.9', 1799443, 262587, 94159, 164519, 240408, 1037770, '58'] for android\_avg\_rating,android\_ratings\_1,android\_ratings\_2,android\_ratings\_3,android\_ratings\_4, android\_ratings\_5 and android\_file\_size respectively.

## 3.2 Data Organization

Now that we have extracted the values from the HTML, the next step leads to organizing the extracted values which paves the way for data exploration.

As we have time series data, we can organize the data by datetime. The [**datetime**](https://docs.python.org/2/library/datetime.html#module-datetime) module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation. For related functionality, see also the [**time**](https://docs.python.org/2/library/time.html#module-time) and [**calendar**](https://docs.python.org/2/library/calendar.html#module-calendar) modules.

The extracted values from Android and iOS pages were stored in the dictionary. The next step is to convert the dictionary to Pandas dataframe where the index is datetime and columns are names of the extracted 11 iOS/Android values. We imported json and csv to convert the obtained dictionary to json and csv files. Also, we used pd.ExcelWriter to convert the file into excel and saved all the converted files.

## 

## 3.3 Data Exploration

Data Exploration comprises of four steps of execution which are as follows:

1. **Dataframe:** This includes describe() method to find the count/mean/std/min/25%/50%75%/max values for each 11 variables.

Following displays the obtained output:

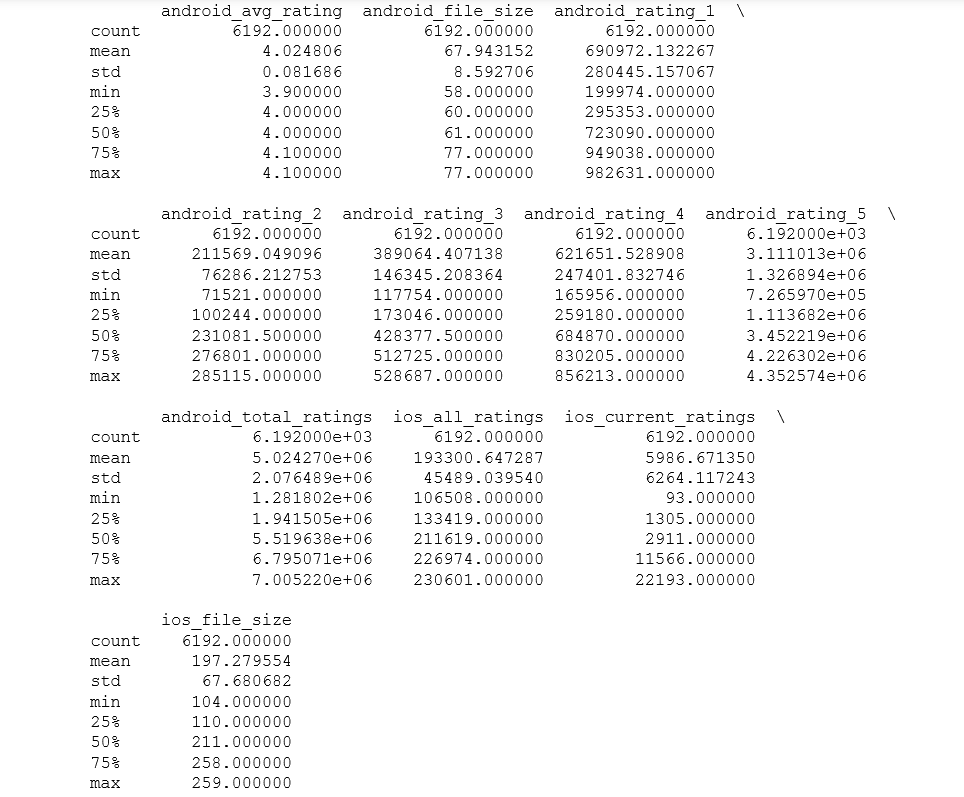


Fig a: Dataframe

2.**Scatter Matrix:** We used scatter\_matrix() method to find pairs of variables with high correlations (either positive or negative).

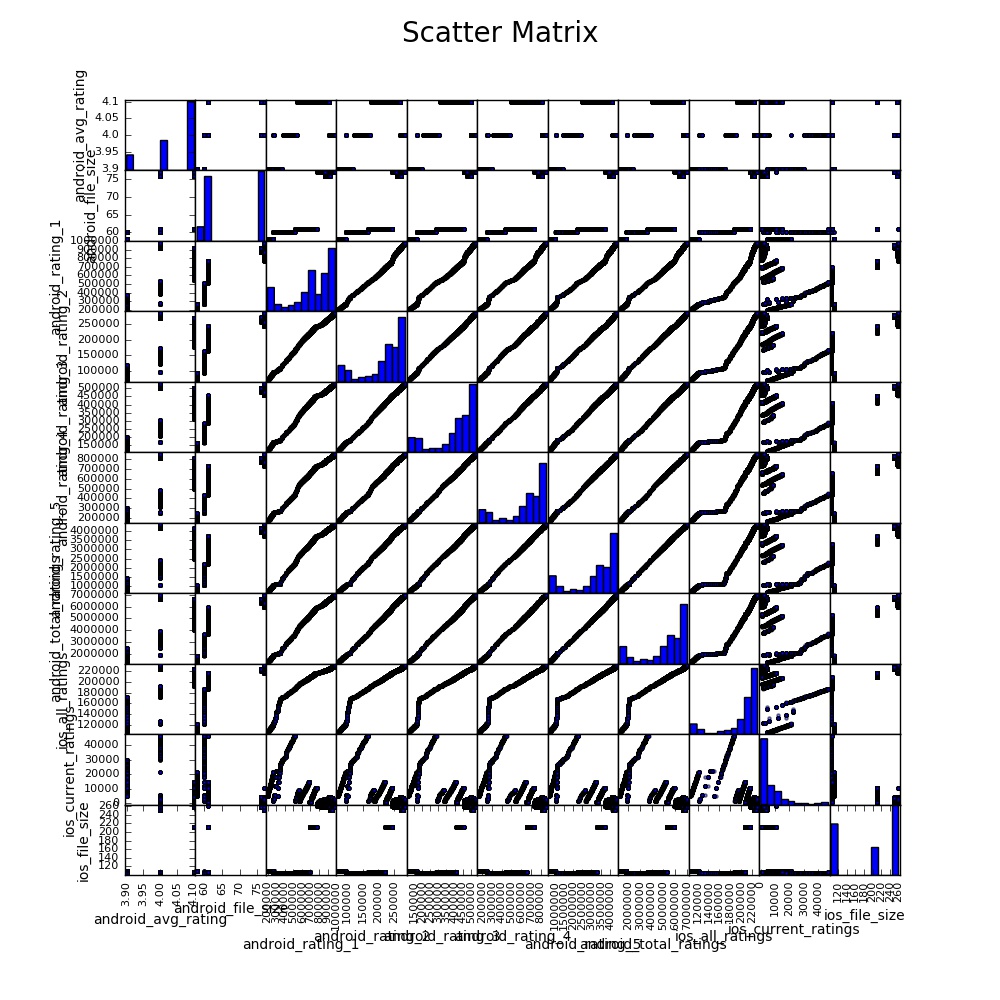


Fig b: Scatter Matrix

3.**Correlation Coefficients:**

To calculate the Pearson’s correlation coefficients for identified pairs. We used corrcoef() function in numpy module for this step.

Following is the result obtained for Correlation Coefficients for Android and iOS:

(I)  
android\_rating\_1 & android\_rating\_2: 0.994269754176  
android\_rating\_1 & android\_rating\_3: 0.992442844685  
android\_rating\_1 & android\_rating\_4: 0.993062742559  
android\_rating\_1 & android\_rating\_5: 0.992862531873  
android\_rating\_1 & android\_total\_ratings: 0.994743322502  
android\_rating\_1 & ios\_all\_ratings: 0.950066135104  
  
(II)  
android\_rating\_2 & android\_rating\_3: 0.999493959679  
android\_rating\_2 & android\_rating\_4: 0.999406679108  
android\_rating\_2 & android\_rating\_5: 0.999457879345  
android\_rating\_2 & android\_total\_ratings: 0.99966102058  
android\_rating\_2 & ios\_all\_ratings: 0.967512495911  
  
(III)  
android\_rating\_3 & android\_rating\_4: 0.999891575729  
android\_rating\_3 & android\_rating\_5: 0.999591350882  
android\_rating\_3 & android\_total\_ratings: 0.999576599907  
android\_rating\_3 & ios\_all\_ratings: 0.962863624613  
  
(IV)  
android\_rating\_4 & android\_rating\_5: 0.999683661904  
android\_rating\_4 & android\_total\_ratings: 0.99972102069  
android\_rating\_4 & ios\_all\_ratings: 0.962216883202  
  
(V)  
android\_rating\_5 & android\_total\_ratings: 0.999839520338  
android\_rating\_5 & ios\_all\_ratings: 0.964005740125  
  
(VI)  
ios\_all\_ratings & android\_total\_ratings: 0.962819781846  
  
(VII)  
ios\_file\_size & ios \_current\_rating: -0.666963045082  
ios\_all\_ratings & ios \_current\_rating: -0.527430099664  
android\_total\_ratings & ios \_current\_rating: -0.655849670038  
android\_rating\_5 & ios \_current\_rating: -0.655603070386  
android\_rating\_4 & ios \_current\_rating: -0.664581716948  
android\_rating\_3 & ios \_current\_rating: -0.659767604663  
android\_rating\_2 & ios \_current\_rating: -0.645628952946  
android\_rating\_1 & ios \_current\_rating: -0.64563552713  
  
(VIII)  
android\_rating\_1 & android\_file\_size: 0.826194337129  
android\_rating\_2 & android\_file\_size: 0.768179466127  
android\_rating\_3 & android\_file\_size: 0.763519161738  
android\_rating\_4 & android\_file\_size: 0.769198559651  
android\_rating\_5 & android\_file\_size: 0.768071374068  
android\_total\_ratings & android\_file\_size: 0.776367884478

(IX)

ios\_all\_ratings & android\_file\_size: 0.667405812658  
ios\_current\_ratings & android\_file\_size: -0.586650887614  
  
(X)  
android\_rating\_1 & android\_avg\_rating: 0.557795132855  
android\_rating\_2 & android\_avg\_rating: 0.625004144412  
android\_rating\_3 & android\_avg\_rating: 0.631453122937  
android\_rating\_4 & android\_avg\_rating: 0.631496575698  
android\_rating\_5 & android\_avg\_rating: 0.636737094165  
android\_total\_ratings & android\_avg\_rating: 0.625279387378  
ios\_all\_ratings & android\_avg\_rating: 0.698174766571  
ios\_current\_ratings & android\_avg\_rating: -0.402332062457  
ios\_file\_size & android\_rating\_5: 0.84801244846

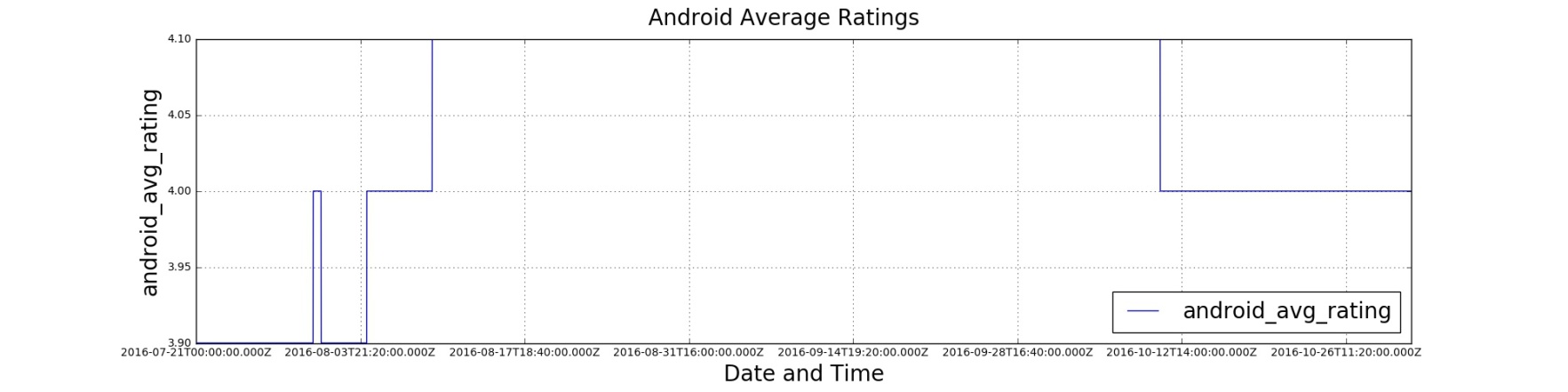
4**. Time Series Graphs:**

We used matplotlib tool to plot time series graphs for each of the 11 variables, using our judgement on which variables to combine for these graphs. Since there was huge difference in range of android data values and iOS data values, we did not combine them. But we combined all android ratings into a single plot.

In these plots, we observed that there is no pattern in variation of android\_avg\_rating, android\_file\_size, ios\_current\_ratings, or iOS file size over changing time.

But we also observed reliable trends in variation of android\_ratings and iOS\_all\_ratings over changing time.

For iOS we observed that, as the file size increased dramatically from 100 to 200MBs, the ratings decreased significantly. May be because of heavy application, people refrained themselves from playing the game on iOS.



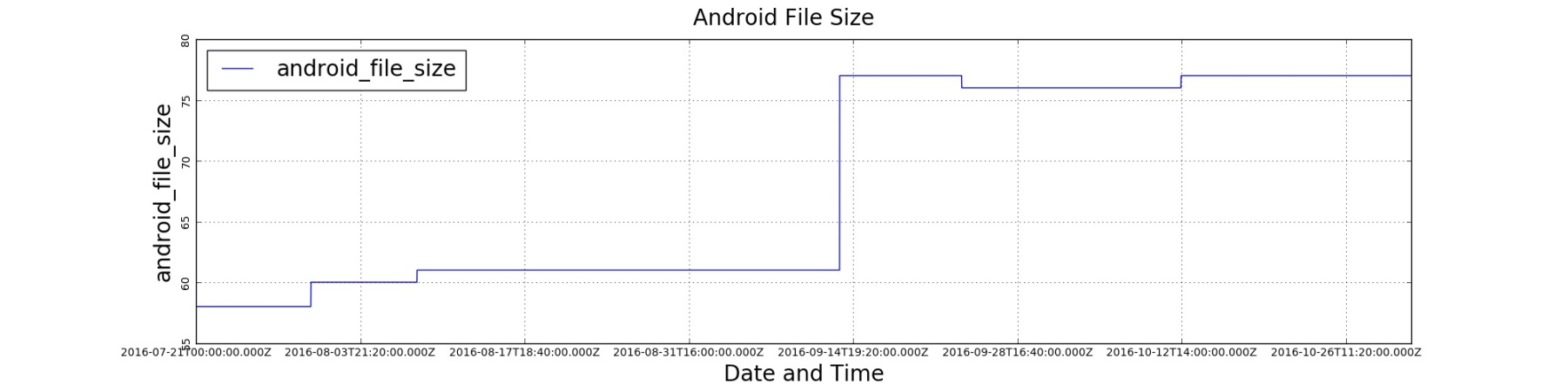
Fig.1: Android Average Ratings

Fig.2: Android File Size

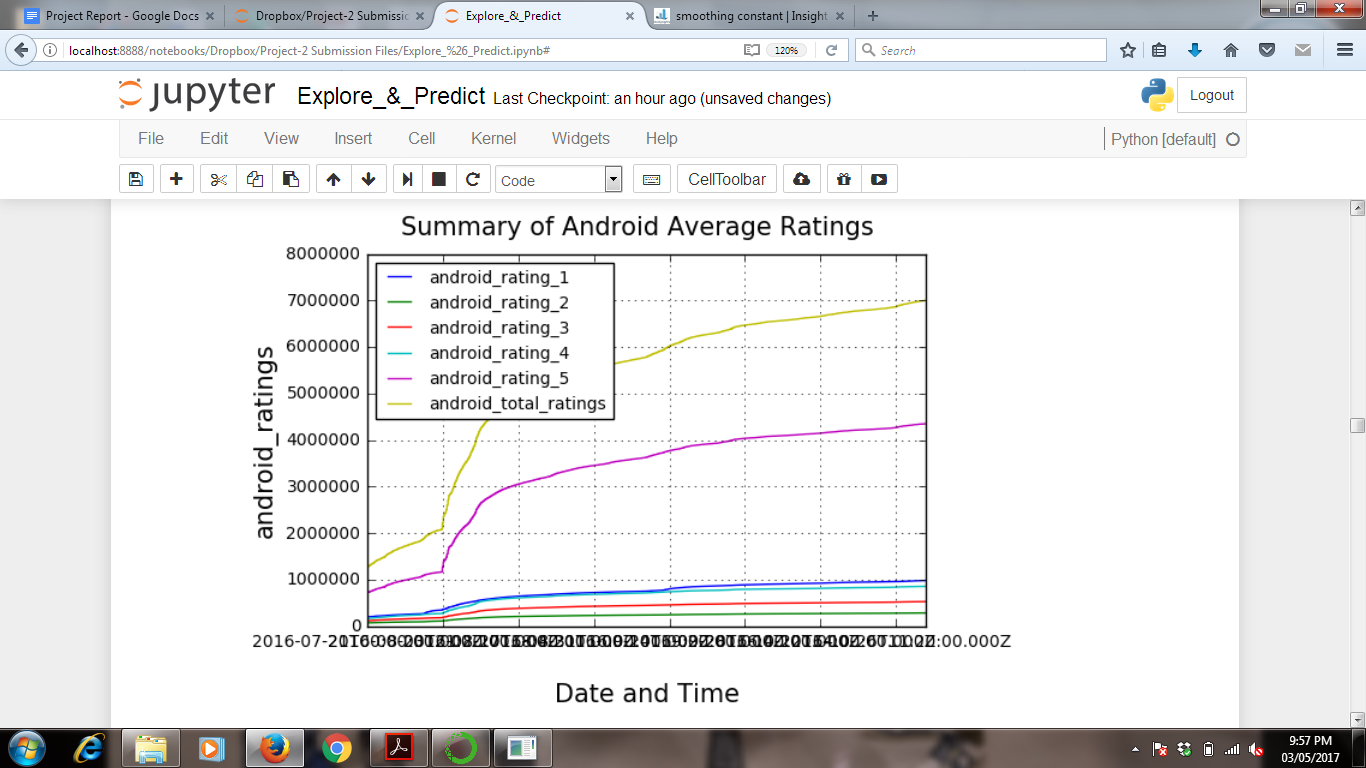


Fig.3: Summary of Android Average Ratings

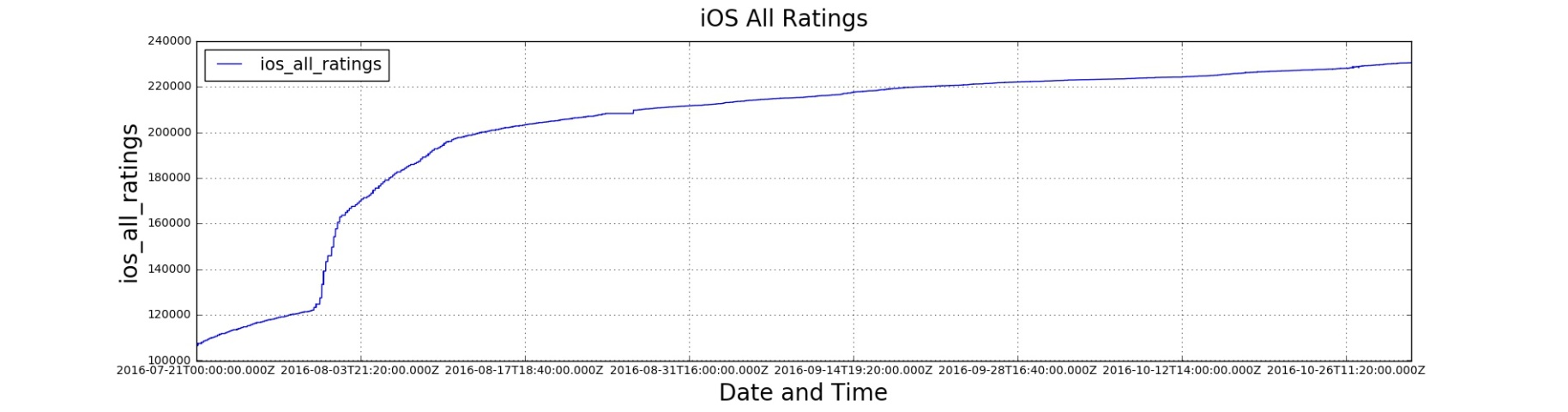


Fig. 4: iOS Ratings

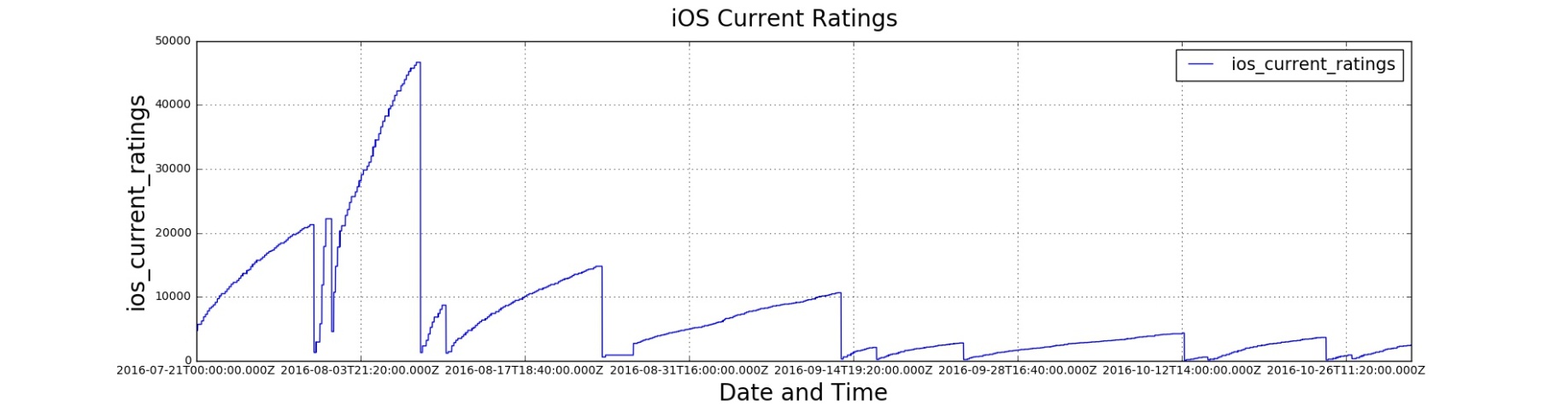


Fig. 6: iOS Current Ratings

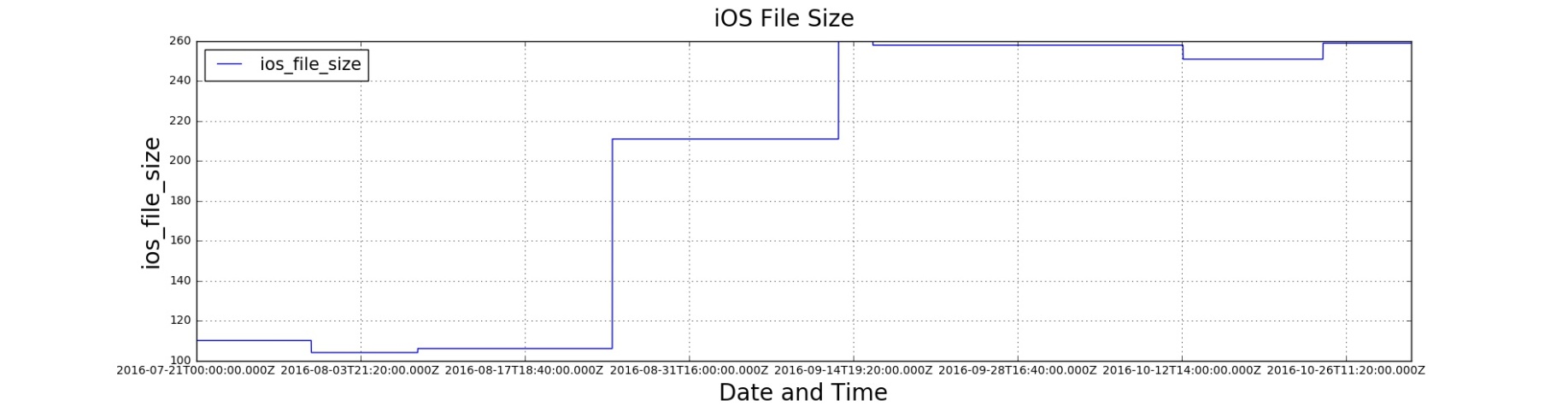


Fig. 7: iOS File Size

## 3.4 Prediction Model:

1.The project required us to build two regression models (one for Android and one for iOS).

We saved all 11 variables in from dataframe as <class 'pandas.core.series.Series'> as follows-

x1→ android\_avg\_rating

x2→ android\_file\_size

x3→ android\_rating\_1

x4→ android\_rating\_2

x5→ android\_rating\_3

x6→ android\_rating\_4

x7→ android\_rating\_5

x8→ ios\_current\_ratings

x9→ ios\_file\_size

We have selected the best prediction model based on following factors:

**i)Margin of Error:** Model with lower margin of error is better

**ii)Coefficient of Determination(R2):** Model with higher coefficient of determination is better.

**iii)Number of variables**: Less the number of variables, the better

Also, we tried to include android variable into iOS model and vice versa.

First, we found the best regression model in each algorithm for both android and iOS.

Following are the three algorithms we performed:

**(A)Linear Regression:**

**i)Android:** After determining a better tradeoff between margin of error and coefficient of determination for several possible combinations which had potential of better modeling, we came up with the model **x1, x3, x4, x7**

linear\_model\_android= 274043.903264 + -63033.6315064 \* android\_avg\_ratings + 0.863898605236 \* android\_rating\_1 2.76580030916 \* android\_rating\_2 1.22773922108 \* android\_rating\_5

This model gave the following R2 and MOE values-

MSE of Android= 7131.83448305

R2-Value for Android= 0.999981817313

**ii)iOS:** After determining a better tradeoff between margin of error and coefficient of determination for several possible combinations which had potential of better modeling, we came up with the model:**x1, x2, x4, x5**.

linear\_model\_ios= -195393.664095 + 69701.4131445 \* android\_avg\_ratings + -563.912893956 \* android\_file\_size + 3.62456807554 \* android\_rating\_2 + -1.59141031078 \* android\_rating\_3

This model gave the following R2 and MOE values-

MSE of iOS= 2308.44370807

R2-Value for iOS= 0.984876027446

**(B)Ridge:**

**i)Android:** After determining a better tradeoff between margin of error and coefficient of determination for several possible combinations which had potential of better modeling, we came up with the model:**x1, x3, x4, x7**.

ridge\_model\_android= 258690.342634 + -59228.3571872 \*android\_avg\_ratings + 0.86989974555 \*android\_rating\_1 2.77284413237 \*android\_rating\_2 1.22593036947 \*android\_rating\_5

This model gave the following R2 and MOE values-

MSE of Android= 7137.84073183

R2-Value for Android= 0.999981802

**ii)iOS:** After determining a better tradeoff between margin of error and coefficient of determination for several possible combinations which had potential of better modeling, we came up with the model:**x1, x2, x4, x5**.

ridge\_model\_android= -185981.830786 + 67460.9314929 \*android\_avg\_ratings + -576.749856743 \*android\_file\_size 3.61701182707 \*android\_rating\_2 -1.58600552805 \*android\_rating\_3

This model gave the following R2 and MOE values-

MSE of iOS= 2311.46386511

R2-Value for iOS= 0.984856240621

**(C)Lasso:**

**i)Android:** After determining a better tradeoff between margin of error and coefficient of determination for several possible combinations which had potential of better modeling, we came up with the model:**x1, x3, x7, x9**.

lasso\_model\_android= 337396.585788 + -55546.3619183 \* android\_avg\_ratings + 1.02771776825 \* android\_rating\_1 + 1.35224462187 \* android\_rating\_5 + -41.5810248656 \* ios\_file\_size

This model gave the following R2 and MOE values-

MSE of Android= 10246.7342264

R2-Value for Android= 0.999973875843

**ii)iOS:** After determining a better tradeoff between margin of error and coefficient of determination for several possible combinations which had potential of better modeling, we came up with the model:**x1, x2, x4, x5**.

lasso\_model\_ios= -155855.333291 + 62572.9731236 \* android\_avg\_ratings + -491.747809921 \* android\_file\_size + 2.39351747802 \* android\_rating\_2 + -0.960161544314 \* android\_rating\_3

This model gave the following R2 and MOE values-

MSE of iOS= 3039.90378159

R2-Value for iOS= 0.980083802261

After finding best regression model in each algorithm for both android and iOS, we came up with the best regression model for Android and iOS as follows:

**Best Regression model for Android**: The best regression model for Android **x1, x3, x4, x7** as it has lowest margin of error and highest Coefficient of Determination amongst all models obtained for android.

This model gave the following R2 and MOE values-

MSE of Android= 7131.83448305

R2-Value for Android= 0.999981817313

**Best Regression model for iOS:** The best regression model for iOS is **x1, x2, x4, x5** as it has lowest margin of error and highest Coefficient of Determination amongst all models obtained for iOS.

This model gave the following R2 and MOE values-

MSE of iOS= 2308.44370807

R2-Value for iOS= 0.984876027446

Both models have been obtained from Linear Regression algorithm.

Also, we have not considered the factor of ‘Number of variables’ for best regression model as all the best models which we obtained from various algorithms (LinearRegression, Ridge, Lasso) have four variables.

2.For prediction of ios\_all\_ratings and android\_total\_ratings, we defined function forecast for forecasting of 11 variables using Smoothing Constant Method.

The predicted values of ios\_all\_ratings and android\_total\_ratings for 2016/11/01 11:50 PM are as follows:

i)Using Linear Model:

android\_total\_ratings prediction: 7001939.99562

ios\_total\_ratings prediction: 238090.330747

ii)Using Ridge Model:

android\_total\_ratings prediction: 7003099.47752

ios\_total\_ratings prediction: 232215.978683

. . .

iii)Using Lasso Model:

android\_total\_ratings prediction: 7000053.7738

ios\_total\_ratings prediction: 231374.787206

. . .

## 

## 3.5 Deep Learning:

To understand the retrieval of unique screenshots from iOS and Android pages we performed the following tasks:

1)The data consisted of multiple images in each app pages which contained few unique screenshots. Hence, the first step was to identify of all unique screenshots from iOS and Android pages. We used URL to distinguish between different images. First, we performed web scraping using BeautifulSoup to extract all images from HTML files. To extract unique screenshots from these we collected all the links in a list and converted into a set as set gives an output of unique values. We performed the above mentioned steps for both iOS and Android pages.

2)Next, we downloaded all the unique images and saved them.

3)For each image, we used tensorflow to extract the tags with the corresponding probabilities scores.

We obtained 17 unique images for Android pages and 5 unique images for iOS pages. Hence, we obtained a total of 22 unique images.

Following are the Images and Tags:



1) web site, website, internet site, site (score = 0.89077)

menu (score = 0.00364)

monitor (score = 0.00185)

screen, CRT screen (score = 0.00184)

analog clock (score = 0.00177)



(2) web site, website, internet site, site (score = 0.42241)

comic book (score = 0.03248)

carousel, carrousel, merry-go-round, roundabout, whirligig (score = 0.02089)

fountain (score = 0.01781)

safety pin (score = 0.01440)



(3) web site, website, internet site, site (score = 0.60886)

television, television system (score = 0.05665)

monitor (score = 0.01996)

notebook, notebook computer (score = 0.01607)

iPod (score = 0.01180)



(4) aircraft carrier, carrier, flattop, attack aircraft carrier (score = 0.09968)

pole (score = 0.03657)

wing (score = 0.02655)

lakeside, lakeshore (score = 0.02437)

magnetic compass (score = 0.02396)



(5) comic book (score = 0.19361)

maze, labyrinth (score = 0.19330)

web site, website, internet site, site (score = 0.05236)

monitor (score = 0.02957)

book jacket, dust cover, dust jacket, dust wrapper (score = 0.02767)



(6) Error: Invalid JPEG data, size 64687



(7) laptop, laptop computer (score = 0.49859)

web site, website, internet site, site (score = 0.10646)

monitor (score = 0.06384)

screen, CRT screen (score = 0.02985)

notebook, notebook computer (score = 0.02801)



(8) laptop, laptop computer (score = 0.08080)

notebook, notebook computer (score = 0.05349)

joystick (score = 0.04791)

monitor (score = 0.04169)



(9) Error: Invalid JPEG data, size 119319



(10) Error: Invalid JPEG data, size 76714



(11) fountain (score = 0.20303)

carousel, carrousel, merry-go-round, roundabout, whirligig (score = 0.08314)

comic book (score = 0.05171)

toyshop (score = 0.03343)

monitor (score = 0.03227)



(12) space shuttle (score = 0.23042)

joystick (score = 0.05992)

racer, race car, racing car (score = 0.05626)

scoreboard (score = 0.04957)

airliner (score = 0.04576)



(13) Error: Invalid JPEG data, size 39763



(14) ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, trash barrel, trash bin (score = 0.15498)

joystick (score = 0.06405)

cannon (score = 0.03585)

maraca (score = 0.02727)

pedestal, plinth, footstall (score = 0.02715)



(15) web site, website, internet site, site (score = 0.22753)

envelope (score = 0.09163)

Band Aid (score = 0.03712)

pinwheel (score = 0.02946)

airship, dirigible (score = 0.02486)



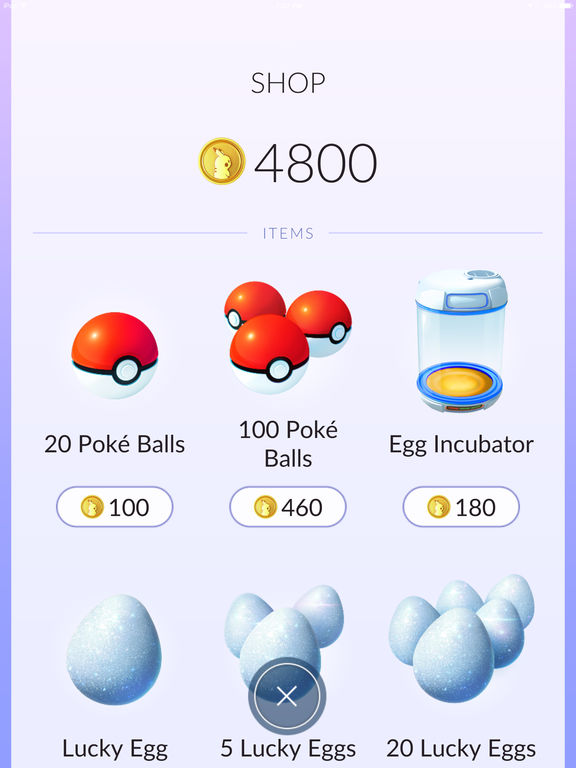
(16) web site, website, internet site, site (score = 0.36619)

safety pin (score = 0.02004)

sunglasses, dark glasses, shades (score = 0.01677)

toilet seat (score = 0.01562)

washer, automatic washer, washing machine (score = 0.01438)



(17) web site, website, internet site, site (score = 0.88357)

menu (score = 0.00803)

slot, one-armed bandit (score = 0.00404)

washer, automatic washer, washing machine (score = 0.00371)

hand-held computer, hand-held microcomputer (score = 0.00296)



(18) Error: Invalid JPEG data, size 45993



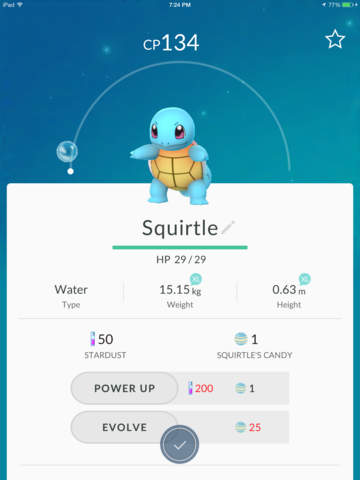
(19) web site, website, internet site, site (score = 0.12342)

maze, labyrinth (score = 0.07149)

comic book (score = 0.04789)

joystick (score = 0.04421)

television, television system (score = 0.03758)



(20) web site, website, internet site, site (score = 0.94092)

analog clock (score = 0.00367)

envelope (score = 0.00291)

monitor (score = 0.00225)

screen, CRT screen (score = 0.00217)



(21) web site, website, internet site, site (score = 0.36779)

envelope (score = 0.16914)

binder, ring-binder (score = 0.05812)

tray (score = 0.01764)

monitor (score = 0.01721)



(22) web site, website, internet site, site (score = 0.58624)

monitor (score = 0.07197)

television, television system (score = 0.05955)

comic book (score = 0.04756)

teapot (score = 0.01425)

# 4. References

* <http://stackoverflow.com/questions/1207457/convert-a-unicode-string-to-a-string-in-python-containing-extra-symbols>
* <http://www.pythonforbeginners.com/python-on-the-web/web-scraping-with-beautifulsoup>
* <https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/>
* <https://docs.python.org/2/library/urllib.html>
* <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_excel.html>
* <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html>
* <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html>
* <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_excel.html>
* <http://stackoverflow.com/questions/7125009/how-to-change-legend-size-with-matplotlib-pyplot>