INSY 5378 : Data Science

Group Project - 2

Pokemon Go - Analytics

Team 2

Sushidhar Jayaraman (#1001400523),
Prashanthi Chandrasekaran (#1001415332),
Pradeep Kumar Madhangopal (#1001448700)

Introduction :

Pokemon Go! – an online mobile app game based on augmented reality (AR) became a very famous in 2016 summer. In this project, we would be understanding the success of the mobile app game. The purpose of this project was to learn the below mentioned concepts and implement them accordingly in our analysis.

i) to perform web scraping using *BeautifulSoup*,
ii) to construct a *Pandas dataframe*,
iii) to explore or visualize the numeric data using *Matplotlib* or *seaborn*,
iv) to build machine learning models to predict the app's review counts using *sklearn*.
v) to analyze the app's screenshot images using deep learning concepts with *tensorflow*.

Data Description :

The dataset for this project, was downloaded from the app pages of Pokemon Go! from Google Play Store and Apple App Store from July 21 2016 to October 31 2016:

1) *https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=en*
2) *https://itunes.apple.com/us/app/pok%C3%A9mongo/id1094591345?mt=8*

The webpages were downloaded once every ten minutes, which indicates that there are 144 (=24x6) HTML files for a given day and a given platform. Download the zip file from the following link:

3) *http://diamond.mccombs.utexas.edu/insy5378/pokemon_5378.zip*

Once the ZIP file was extracted, there were 103 date folders under "data" folder. Each date folder contains HTML files downloaded for a 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".

Web scrapping :

As the source data that we need is available on website, we would be performing Web scrapping to obtain it. **Web Scraping** (also termed Screen Scraping, Web Data Extraction, Web Harvesting etc.) is a technical process employed to extract large amounts of data directly from websites whereby the data is extracted and saved to a local file in our computer or to a database in table (spreadsheet) format.

The initial step is to extract various values from the raw HTML files, which would be done using `BeautifulSoup` - a Python module that parses the required data from the overall content. To segregate the 'iOS' files from the android files in the zipped folder, we had to first identify the file names using '*_pokemon_ios.html*'. Then, from the seperated iOS files we extracted three main values, namely

i) the number of customer ratings in the Current Version (identified as *ios_current_ratings*);
ii) the number of customer ratings in All Versions (identified as *ios_all_ratings*); and
iii) the file size in MB (identified as *ios_file_size*).

For instance, the extracted values would be in the format : 4688, 106508, 110 for the value "*2016-07 21/00_00_pokemon_ios.html*" file.

Similarly we extracted android files using '_pokemon_android.html', and then retrieve eight values from those files as mentioned below :

i) the average rating (in the scale between 1.0 and 5.0) (identified by *android_avg_rating*);
ii) the number of total ratings (identified as *android_total_ratings*);
iii) the number of ratings for 1-5 stars, identified as
    a) *android_ratings_1*,
    b) *android_ratings_2*,
    c) *android_ratings_3*,
    d) *android_ratings_4*,
    e) *android_ratings_5*;
iv) the file size in MB (identified as *android_file_size*).

And for example, the extracted numbers should be of the format : 3.9, 1281802, 199974, 71512, 117754, 165956, 726597, 58 for the "2016-07-21/00_00_pokemon_android.html" file.

Data Organization & Exploration :

The next step is to organize the extracted values, so that we can do some data exploration. **Data exploration** is the first step in data analysis and typically involves summarizing the main characteristics of a dataset. Therefore, after extracting the iOS and Android file individually, we need to create a unique Index for our data. Having a unique index value for every record in the dataset is very important for our analysis. And in general, 'Time stamp' attribute is used as the unique attribute. In our dataset, we have a time series data, we will organize the data by *datetime* which is a Python data type.

The timestamp of the iOS files are from replaced from '*%Y-%m-%d%H_%M*' format to '*%Y,%m,%d,%H,%M,%S*'. Similar parsing process was followed for android files as well. An exceptions were handled for files with missing values. The retrieved files were then stored into temporary dictionaries.

Further, we converted the dictionary into a *`Pandas dataframe`* where the index was *`datetime`* and columns are names of the extracted eleven iOS and Android values. Then saved the *`dataframe`* into three formats namely : JSON, CSV, and Excel file formats under the names *`data.json`*, *`data.csv`* and *`data.xlsx`*.

Now that we have `Pandas dataframe` ready, we started to explore the data.

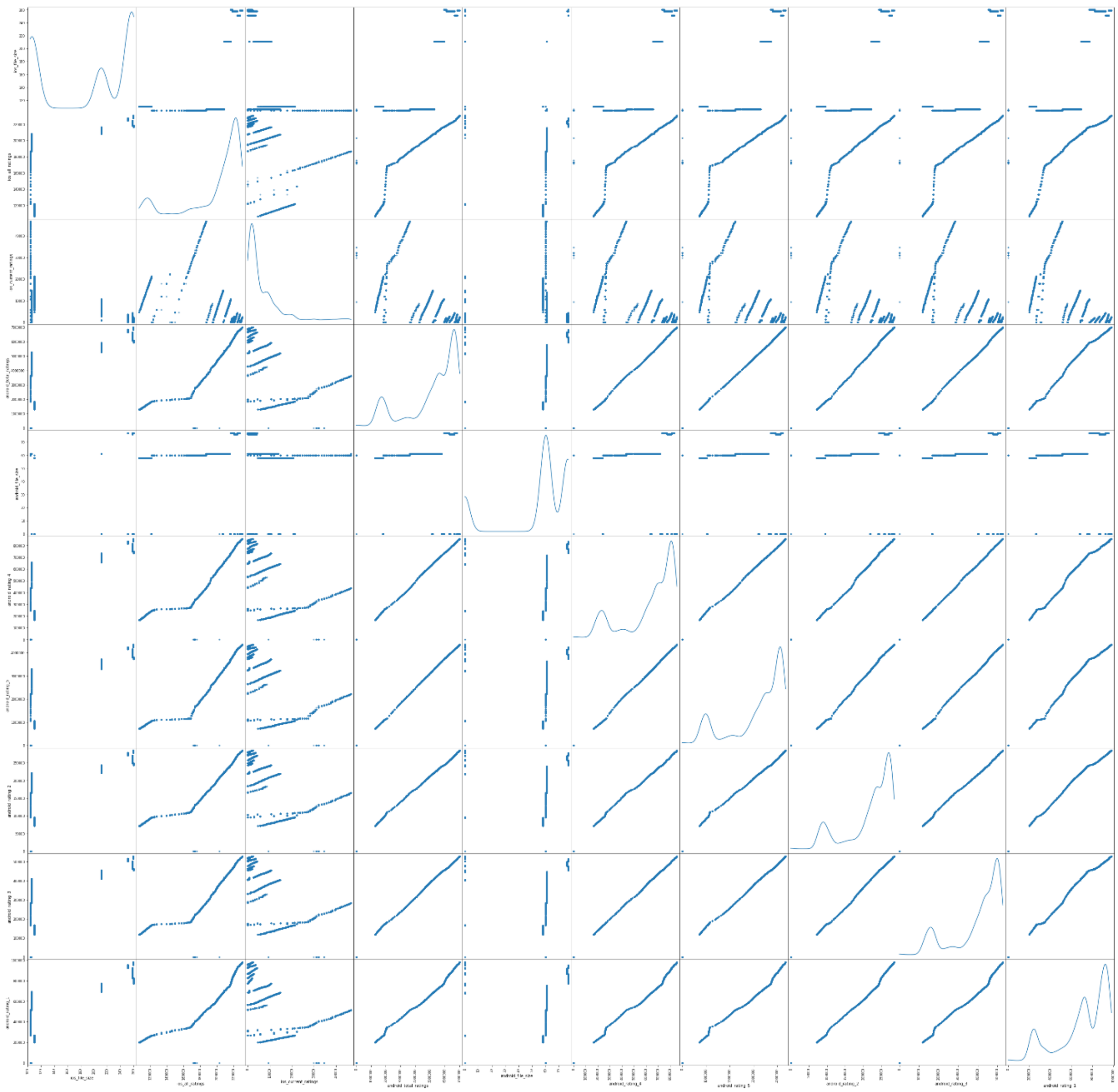
i) Used *`describe()`* method to find the count, mean, standard deviation, mode, minimum, 25%, 50%, 75%, maximum values for each of the eleven variables.

```
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
ios_file_size	14810.0	1.967181e+02	6.716468e+01	104.0	110.0	211.0	258.0	260.0
ios_all_ratings	14810.0	2.028599e+05	3.335211e+04	106508.0	201533.0	215355.0	223336.0	230601.0
ios_current_ratings	14810.0	7.428749e+03	9.113272e+03	29.0	1865.0	3676.0	9609.0	46692.0
android_total_ratings	14810.0	5.277341e+06	1.695718e+06	1281802.0	4779210.0	5790213.0	6577516.0	7005220.0
android_avg_rating	14810.0	4.046550e+00	7.187742e-02	3.9	4.0	4.1	4.1	4.1
android_file_size	14810.0	6.796826e+01	8.191596e+00	58.0	61.0	61.0	77.0	77.0
android_rating_4	14810.0	6.511818e+05	2.026241e+05	165956.0	596010.0	716201.0	804331.0	856213.0
android_rating_5	14810.0	3.277477e+06	1.085623e+06	726597.0	2977746.0	3633064.0	4099775.0	4352574.0
android_rating_2	14810.0	2.211477e+05	6.157790e+04	71521.0	204299.0	240452.0	267621.0	285115.0
android_rating_3	14810.0	4.065541e+05	1.198152e+05	117754.0	373913.0	447650.0	496153.0	528687.0
android_rating_1	14810.0	7.209809e+05	2.275795e+05	199974.0	627242.0	752846.0	909636.0	982631.0

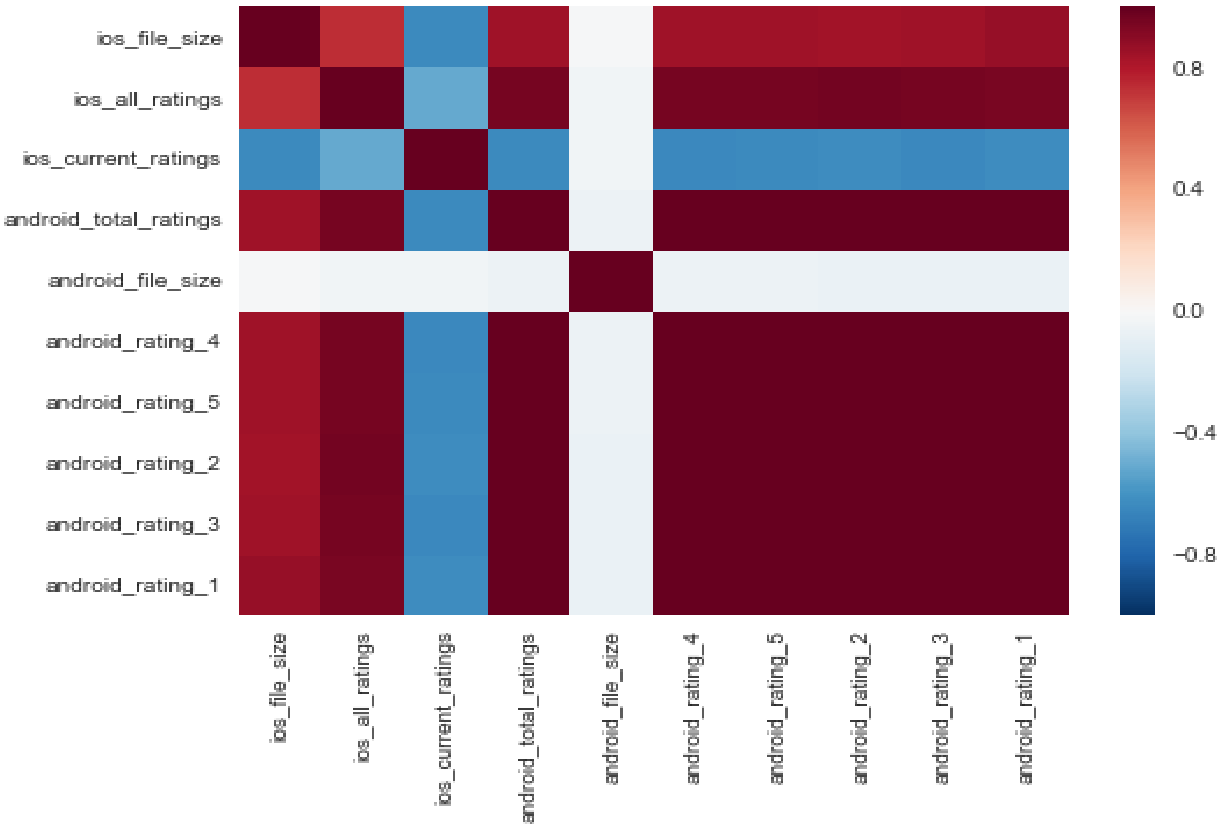
ii) Used `scatter_matrix()` method to find pairs of variables with high correlations (either positive or negative).

scatter_matrix(data_df, figsize=(50,50), diagonal = 'kde')
plt.savefig('image.jpeg')
plt.show()



i) There is a very **High Positive correlation** between the five android_rating attributes, within one and another.
ii) And there is a very **Low Negative correlation** with the ios-current-rating.

The **Heat map** below also defines the correlations between the attributes :



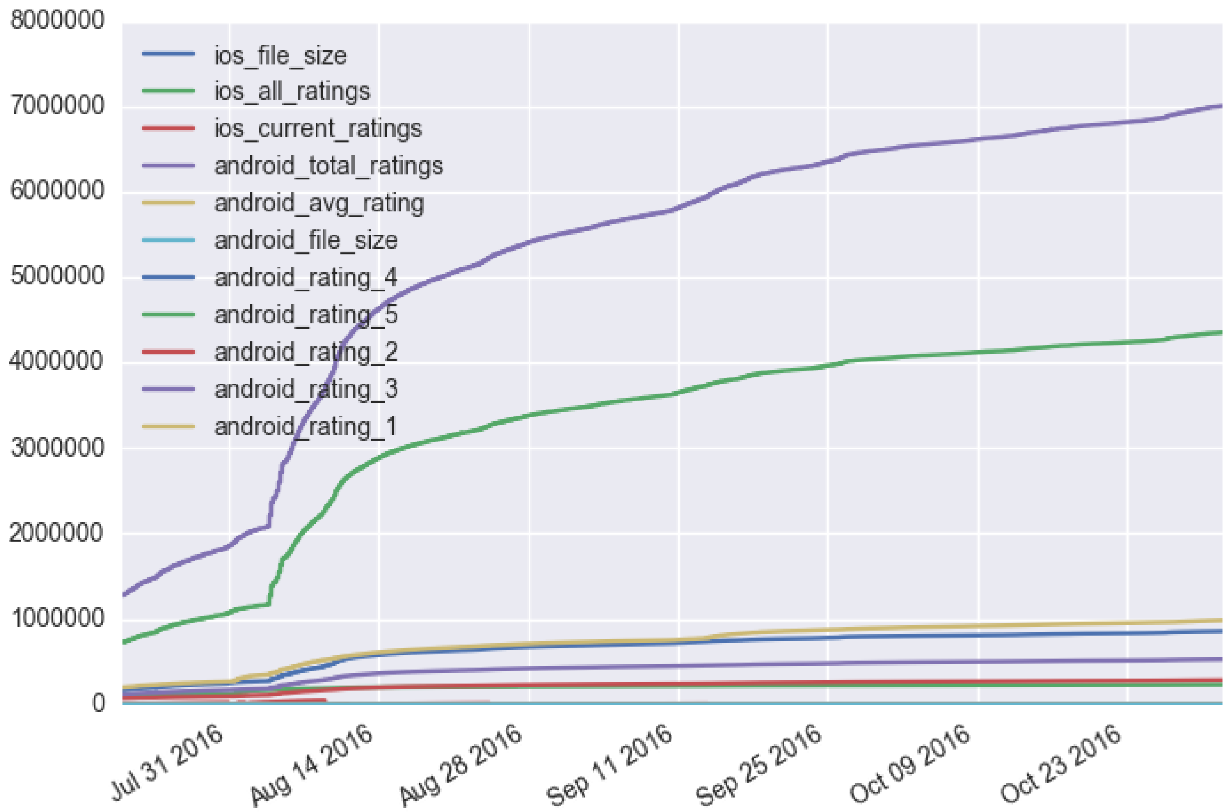
iii) Identified pairs, then calculated the ***Pearon's correlation coefficients***. We used `corrcoef()` function in `numpy` module for this.

```
In [199]: df.corr()
```

Out[199]:

	ios_file_size	ios_all_ratings	ios_current_ratings	android_total_ratings	android_avg_rating	android_file_size	android_rati
ios_file_size	1.000000	0.738353	-0.666968	0.851887	0.359255	0.857121	0.849747
ios_all_ratings	0.738353	1.000000	-0.527437	0.962817	0.698174	0.666907	0.962216
ios_current_ratings	-0.666968	-0.527437	1.000000	-0.655855	-0.402308	-0.586349	-0.664586
android_total_ratings	0.851887	0.962817	-0.655855	1.000000	0.625244	0.775928	0.999721
android_avg_rating	0.359255	0.698174	-0.402308	0.625244	1.000000	0.192362	0.631472
android_file_size	0.857121	0.666907	-0.586349	0.775928	0.192362	1.000000	0.768741
android_rating_4	0.849747	0.962216	-0.664586	0.999721	0.631472	0.768741	1.000000
android_rating_5	0.848019	0.964005	-0.655608	0.999839	0.636709	0.767616	0.999684
android_rating_2	0.843736	0.967511	-0.645634	0.999661	0.624979	0.767726	0.999407
android_rating_3	0.847385	0.962864	-0.659771	0.999576	0.631440	0.763044	0.999891
android_rating_1	0.871197	0.950040	-0.645636	0.994734	0.557689	0.825864	0.993050

iv) Used `matplotlib` to create time series graphs for each of the eleven variables. As the files are collected in every 10 minutes, there are multiple values for a given date. The graph below is a combined time series for both the iOS and Android parameters. Thus, the X-axis incorporates dates and times.



Prediction Model :

At this point, we are quite familiar with the data. Now let's build a machine learning model on the success of Pokemon Go! app. People often use the number of ratings (ios_all_ratings and android_total_ratings) as a proxy of app success.

1) Built two best regression models (one for iOS and one for Android) using ***sklearn*** cross validation. Tried to add/remove variables among the eleven attributes. Created our own variables where ever necessary. Also tried various algorithms in the module: like - **LinearRegression**, **Ridge**, **Lasso**, etc. (http://scikitlearn.org/stable/modules/linear_model.html)
2) Then submitted our predicted values of ios_all_ratings and android_total_ratings for **2016/11/01 11:50 PM**

For predicting the ios_all_ratings and android_total_ratings, we used Linear Regression. Since android_rating_1, android_rating_2, android_rating_3, android_rating_4 and android_rating_5 are *highly co-related* , using these would give a *false prediction*. So we used android_avg_rating and android_avg_rating for predicting average rating.

For ios, file size and current ratings were used to predict ios_all_ratings. The following was the equation that we obtained :

> ***('Coefficients: \n', array([[11677600.77820515, 140730.36350031]]))***
>
> ***y = 11677600.78 x1 + 140730.36 x2 -51543199.2*** *(android equation)*
>
> ***('Coefficients: \n', array([[3.45677092e+02, -2.30676397e-01]]))***
>
> ***y = 345.68 x1 + -0.23 x2 + 136565.84*** *(iOS equation)*

This equation is analogous to **y=mx+c**. Here, x1 and x2 are the predictors. We have been asked to predict the rating for 2016/11/01 11:50 PM. So with the last timestamp we have, we iterate backwards so that we have values to substitute for x1 and x2 and we can have nearly perfect prediction.

- ***Android Total Rating* predicted for the date 2016/11/01 11:50 PM: *6003441.89807***

- ***iOS Total Rating* predicted for the date 2016/11/01 11:50 PM: *978575.487992***

Linear Regression:

1) Mean squared error: 1000299765865.80
2) Variance score: -7167400.06
3) Root Mean squared error: 1000149.87

We also built models using ridge and lasso and got the following results:

Rigde Algorithm:

1) Ridge Algorithm: coefficient of determination R^2 for android: 0.83752937667808958
2) Ridge Algorithm: coefficient of determination R^2 for iOS: 0.54697801880264885
3) Mean squared error: 980990094498.22
4) Variance score: -7029041.38
5) Root Mean squared error: 990449.44

Lasso Algorithm:

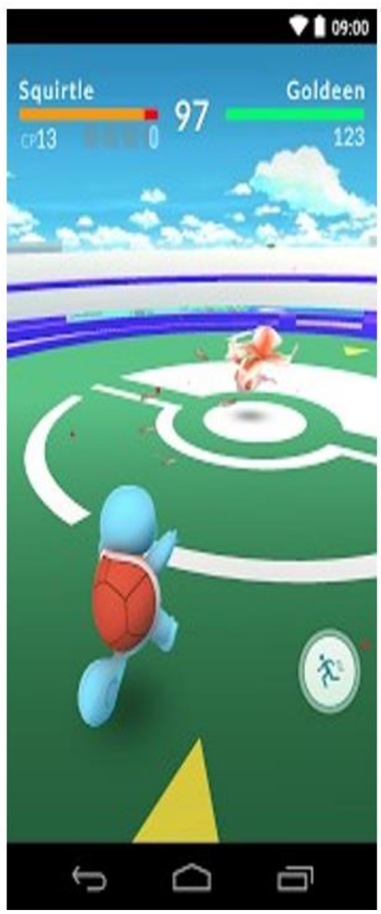
1) Lasso Algorithm: coefficient of determination R^2 for android: 0.83757178583919656
2) Lasso Algorithm: coefficient of determination R^2 for iOS: 0.54697801880264885
3) Mean squared error: 24812098.76
4) Variance score: -47729.26
5) Root Mean squared error: 4981.17

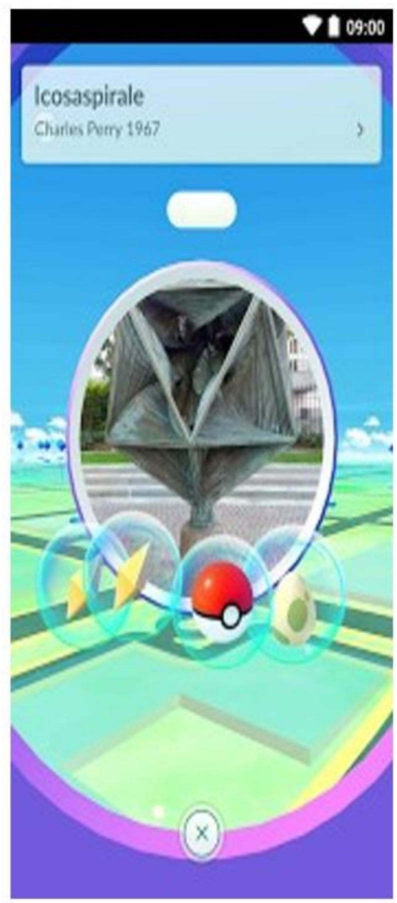
Deep Learning :

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural network. In our project, we want to understand the screenshot Images of the app.

i) Identified all unique screenshots from iOS and Android pages. More over, there are multiple images in each app page.
ii) Downloaded the screenshot images from iOS and Android webpages.
iii) For each image, used `tensorflow` to extract the tags with the corresponding probabilities.

Sample Images :





Android Image Probability values :

Image :	Tag Description :	Probabilities :
1.jpg	lawn mower, mower	0.252659321
	golf ball	0.127368331
	croquet ball	0.060719773
	bow	0.052845191
	steel arch bridge	0.022924505
2.jpg	web site, website, internet site, site	0.629976332
	television, television system	0.085679717
	monitor	0.043532189
	screen, CRT screen	0.033237603
	hand-held computer, hand-held microcomputer	0.019567024
3.jpg	web site, website, internet site, site	0.559773982
	iPod	0.043711599
	comic book	0.030220011
	screen, CRT screen	0.025605576
	monitor	0.021581972
4.jpg	web site, website, internet site, site	0.442224413
	monitor	0.065088287
	notebook, notebook computer	0.055768777
	home theater, home theatre	0.03170713
	television, television system	0.029129347
5.jpg	monitor	0.237243131
	screen, CRT screen	0.08305151
	web site, website, internet site, site	0.060117662
	garbage truck, dustcart	0.060014669
	desktop computer	0.04502682

iOS Image Probability values :

Image :	Tag Description :	Probabilities :
1.jpg	maze, labyrinth	0.245949954
	comic book	0.131160408
	web site, website, internet site, site	0.038981959
	monitor	0.024550997
	book jacket, dust cover, dust jacket, dust wrapper	0.022834243
2.jpg	web site, website, internet site, site	0.84876883
	menu	0.014709988
	washer, automatic washer, washing machine	0.005761398
	slot, one-armed bandit	0.004649503
	hand-held computer, hand-held microcomputer	0.003371626
3.jpg	aircraft carrier, carrier, flattop, attack aircraft carrier	0.185934395
	pool table, billiard table, snooker table	0.034626596
	wing	0.032510813
	pole	0.020514756
	magnetic compass	0.014414731
4.jpg	ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, trash barrel, trash bin	0.114966638
	joystick	0.047080744
	pedestal, plinth, footstall	0.038930431
	maraca	0.038405225
	cannon	0.036177531
5.jpg	web site, website, internet site, site	0.236590773
	envelope	0.088390186
	Band Aid	0.024608353
	piggy bank, penny bank	0.024148384
	pinwheel	0.023388693
6.jpg	laptop, laptop computer	0.597294569
	web site, website, internet site, site	0.065704942
	monitor	0.044525035
	notebook, notebook computer	0.041533194
	screen, CRT screen	0.023493774
7.jpg	web site, website, internet site, site	0.301225454
	safety pin	0.020946817
	toilet seat	0.020777836

	washer, automatic washer, washing machine	0.017361175
	carton	0.01455635
10.jpg	laptop, laptop computer	0.128781945
	web site, website, internet site, site	0.110864304
	joystick	0.065062352
	notebook, notebook computer	0.056731239
	jellyfish	0.026855864
11.jpg	web site, website, internet site, site	0.585088432
	television, television system	0.040827841
	monitor	0.016358094
	notebook, notebook computer	0.01298201
	digital clock	0.009441537
12.jpg	web site, website, internet site, site	0.936489046
	envelope	0.003425678
	analog clock	0.003385809
	screen, CRT screen	0.003341293
	monitor	0.00265502
13.jpg	space shuttle	0.170907691
	racer, race car, racing car	0.064216323
	scoreboard	0.060315702
	joystick	0.042732585
	airliner	0.032772947
14.jpg	web site, website, internet site, site	0.125778198
	maze, labyrinth	0.068146244
	comic book	0.044193037
	joystick	0.037850235
	monitor	0.037364747
15.jpg	fountain	0.156278402
	carousel, carrousel, merry-go-	0.070300639
	submarine, pigboat, sub, U-boat	0.044049773
	seashore, coast, seacoast, sea-coast	0.029817538
	comic book	0.028290268
16.jpg	web site, website, internet site, site	0.257633179
	envelope	0.16587767
	binder, ring-binder	0.069541492
	tray	0.054265086
	packet	0.020124281
17.jpg	web site, website, internet site, site	0.41079402
	monitor	0.083957359
	television, television system	0.072067894
	comic book	0.066413172
	Teapot	0.050743733

Reference :

1. *Tensorflow library code reference Dr. Gene Moo Lee.*
2. *Sklearn tutorial – Linear Regression, Ridge, Lasso*