PROJECT-2

POKEMON GO!! ANALYTICS

INTRODUCTION

Background: Pokémon Go! became a very famous augmented reality (AR) game in 2016 summer. In this project, we want to understand the success of the mobile app game by predicting different models.

Project Overview: We have extracted data from HTML files provided and formed a data frame. Using the data frame , created a scatter matrix and found correlation between variables. We then used different regression Algorithms to predict the future values.

WORKFLOW

Data Extraction: From the HTML files provided , we created a function to read the files from folders using os.path.join and os.path.isdir(). Used Beautiful soup library to extract data by web scraping. We identified the unique html tags to scrape IOS and Android variables . Used try and except to handle the improper files and missing values. We store the data in CSV(read_csv) ,JSON(to_json) and Excel(to_excel) format.

Data Frame: We used pandas library to create the data frame using data time as index.

Handling missing values: As we had many blank values , we used front fill (ffill) method which fills the previous value to all the blank spaces.

Data Exploration: We used describe() function to get the different parameter values for each of the 11 variables . Used scatter matrix() to find the variables with any positive or negative correlation. We used numpy module to calculate the Pearson's correlation between the correlated pairs found from scatter matrix. Using matplotlib, we created a time series plot.

Prediction Model:

The models which we used to predict are :

- Linear Regression
- Lasso
- Ridge
- Random Forest

Tensor Flow: We again used web scraping technique and used if-else condition to extract the unique screen shots from both Android and HTML files. Then we used tensor floe to get the tags from all the extracted screen shots.

RESULTS

Describe Function:

s.describe()

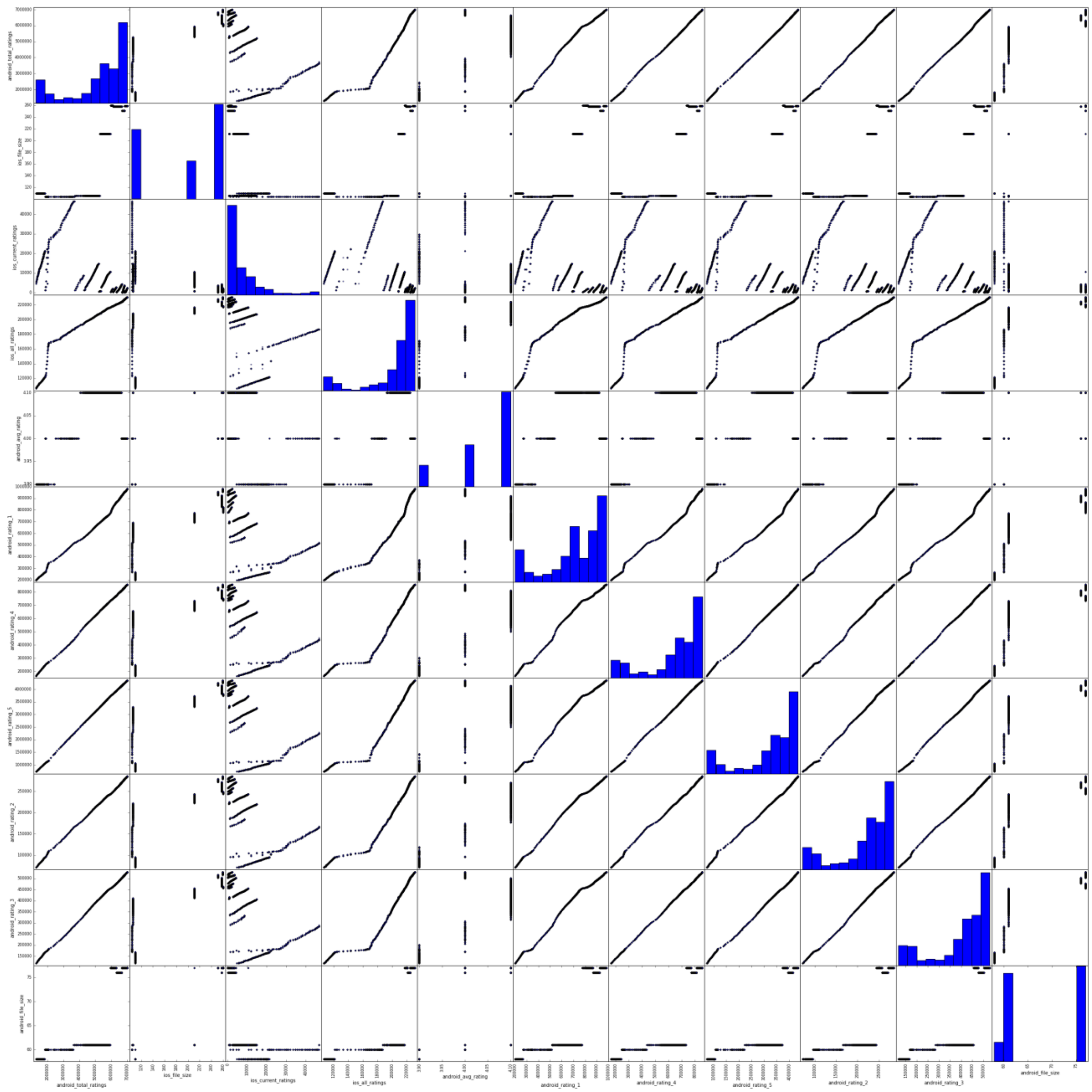
	Unnamed: 0	android_total_ratings	ios_file_size
count	14810.000000	1.481000e+04	14810.000000
mean	7404.500000	5.277341e+06	196.614382
std	4275.423078	1.695718e+06	67.086982
min	0.000000	1.281802e+06	104.000000
25%	3702.250000	4.779210e+06	110.000000
50%	7404.500000	5.790213e+06	211.000000
75%	11106.750000	6.577516e+06	258.000000
max	14809.000000	7.005220e+06	260.000000

	ios_current_ratings	ios_all_ratings	android_avg_rating
count	14810.000000	14810.000000	14810.000000
mean	7428.748751	202847.008845	4.046550
std	9113.271931	33368.819746	0.071877
min	29.000000	106508.000000	3.900000
25%	1865.000000	201533.000000	4.000000
50%	3676.000000	215355.000000	4.100000
75%	9609.000000	223336.000000	4.100000
max	46692.000000	230601.000000	4.100000

	android_rating_1	android_rating_4	android_rating_5	android_rating_2
count	14810.000000	14810.000000	1.481000e+04	14810.000000
mean	720980.905199	651181.763268	3.277477e+06	221147.682039
std	227579.456259	202624.070942	1.085623e+06	61577.895162
min	199974.000000	165956.000000	7.265970e+05	71521.000000
25%	627242.000000	596010.000000	2.977746e+06	204299.000000
50%	752846.000000	716201.000000	3.633064e+06	240452.000000
75%	909636.000000	804331.000000	4.099775e+06	267621.000000
max	982631.000000	856213.000000	4.352574e+06	285115.000000

	android_rating_3	android_file_size
count	14810.000000	14810.000000
mean	406554.050574	67.968265
std	119815.222861	8.191596
min	117754.000000	58.000000
25%	373913.000000	61.000000
50%	447650.000000	61.000000
75%	496153.000000	77.000000
max	528687.000000	77.000000

Scatter Matrix:



<u>Pearson Coefficient Correlation:</u>

```
#Pearson's Coefficient
correlation1 = np.corrcoef(df2['android_total_ratings'],df2['ios_all_ratings'])
correlation2 = np.corrcoef(df2['android_total_ratings'],df2['android_rating_1'])
correlation3 = np.corrcoef(df2['android_total_ratings'],df2['android_rating_4'])
correlation4 = np.corrcoef(df2['android_total_ratings'],df2['android_rating_5'])
correlation5 = np.corrcoef(df2['android_total_ratings'],df2['android_rating_2'])
correlation6 = np.corrcoef(df2['android_total_ratings'],df2['android_rating_3'])
correlation7 = np.corrcoef(df2['ios_all_ratings'],df2['android_rating_1'])
correlation8 = np.corrcoef(df2['ios_all_ratings'],df2['android_rating_2'])
correlation9 = np.corrcoef(df2['ios_all_ratings'],df2['android_rating_3'])
correlation10 = np.corrcoef(df2['ios_all_ratings'],df2['android_rating_4'])
correlation11 = np.corrcoef(df2['ios_all_ratings'],df2['android_rating_5'])
correlation12 = np.corrcoef(df2['android_rating_1'],df2['android_rating_2'])
correlation13 = np.corrcoef(df2['android_rating_1'],df2['android_rating_3'])
correlation14 = np.corrcoef(df2['android_rating_1'],df2['android_rating_4'])
correlation15 = np.corrcoef(df2['android_rating_1'],df2['android_rating_5'])
correlation16 = np.corrcoef(df2['android_rating_4'],df2['android_rating_3'])
correlation17 = np.corrcoef(df2['android_rating_4'],df2['android_rating_2'])
correlation18 = np.corrcoef(df2['android_rating_4'],df2['android_rating_5'])
correlation19 = np.corrcoef(df2['android_rating_5'],df2['android_rating_2'])
correlation20 = np.corrcoef(df2['android_rating_5'],df2['android_rating_3'])
correlation21 = np.corrcoef(df2['android_rating_2'],df2['android_rating_3'])
```

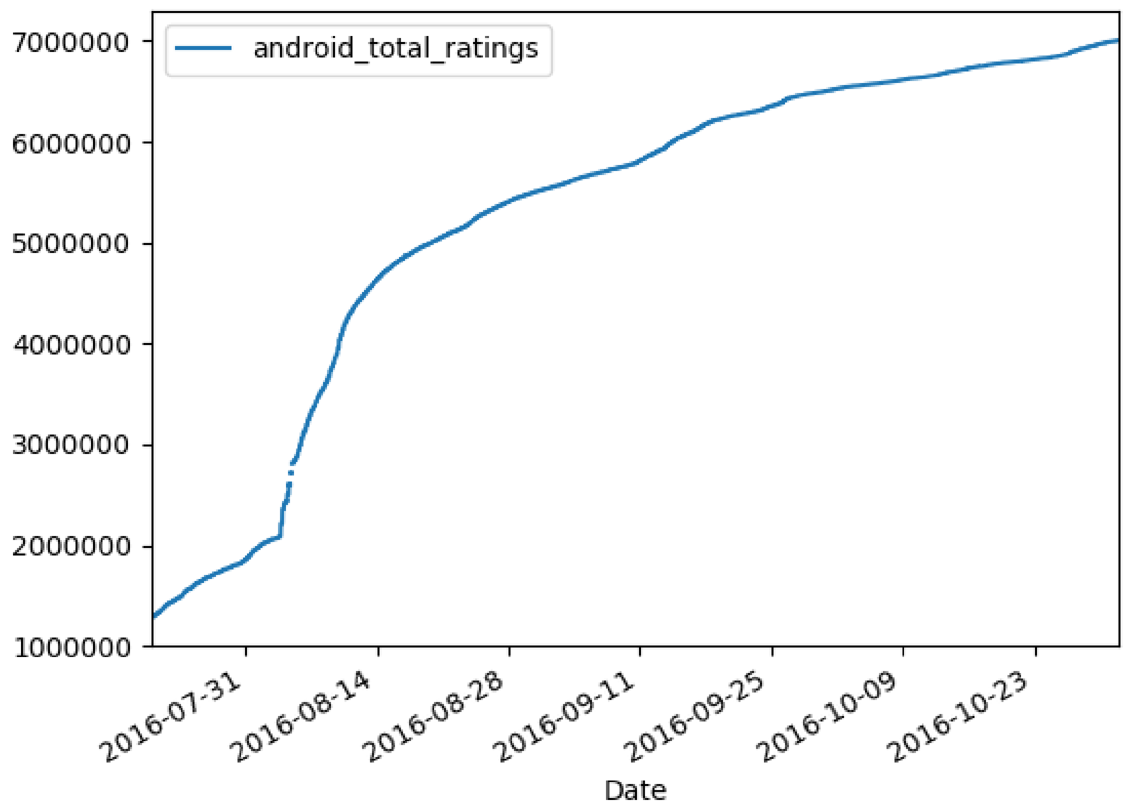
```
In [5]: correlation1
Out[5]:
array([[ 1.        ,  0.96291116],
       [ 0.96291116,  1.        ]])

In [6]: correlation2
Out[6]:
array([[ 1.       ,  0.9947341],
       [ 0.9947341,  1.       ]])

In [7]: correlation3
Out[7]:
array([[ 1.        ,  0.99972081],
       [ 0.99972081,  1.        ]])

In [8]: correlation4
Out[8]:
array([[ 1.        ,  0.99983935],
       [ 0.99983935,  1.        ]])

In [9]: correlation5
Out[9]:
array([[ 1.        ,  0.99966088],
       [ 0.99966088,  1.        ]])

In [10]: correlation6
Out[10]:
array([[ 1.        ,  0.99957579],
       [ 0.99957579,  1.        ]])

In [11]: correlation7
Out[11]:
array([[ 1.        ,  0.95010733],
       [ 0.95010733,  1.        ]])
```

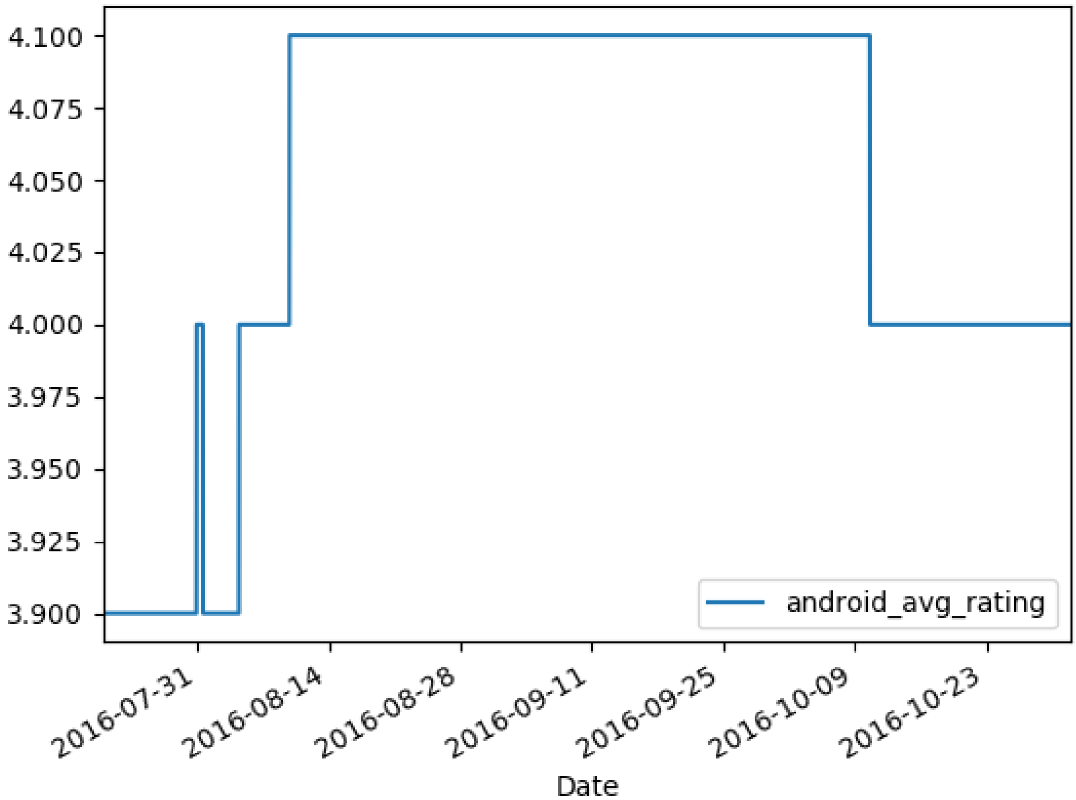
```
In [12]: correlation8
Out[12]:
array([[ 1.        ,  0.96759541],
       [ 0.96759541,  1.        ]])

In [13]: correlation9
Out[13]:
array([[ 1.        ,  0.96295606],
       [ 0.96295606,  1.        ]])

In [14]: correlation10
Out[14]:
array([[ 1.        ,  0.96230442],
       [ 0.96230442,  1.        ]])

In [15]: correlation11
Out[15]:
array([[ 1.       ,  0.9641064],
       [ 0.9641064,  1.       ]])

In [16]: correlation12
Out[16]:
array([[ 1.        ,  0.99425805],
       [ 0.99425805,  1.        ]])

In [17]: correlation13
Out[17]:
array([[ 1.        ,  0.99242733],
       [ 0.99242733,  1.        ]])

In [18]: correlation14
Out[18]:
array([[ 1.        ,  0.99305048],
       [ 0.99305048,  1.        ]])
```

```
In [19]: correlation15
Out[19]:
array([[ 1.        ,  0.99285061],
       [ 0.99285061,  1.        ]])

In [20]: correlation16
Out[20]:
array([[ 1.        ,  0.99989139],
       [ 0.99989139,  1.        ]])

In [21]: correlation17
Out[21]:
array([[ 1.       ,  0.9994067],
       [ 0.9994067,  1.       ]])

In [22]: correlation18
Out[22]:
array([[ 1.        ,  0.99968367],
       [ 0.99968367,  1.        ]])

In [23]: correlation19
Out[23]:
array([[ 1.       ,  0.9994579],
       [ 0.9994579,  1.       ]])

In [24]: correlation20
Out[24]:
array([[ 1.        ,  0.99959113],
       [ 0.99959113,  1.        ]])

In [25]: correlation21
Out[25]:
array([[ 1.        ,  0.99949371],
       [ 0.99949371,  1.        ]])
```

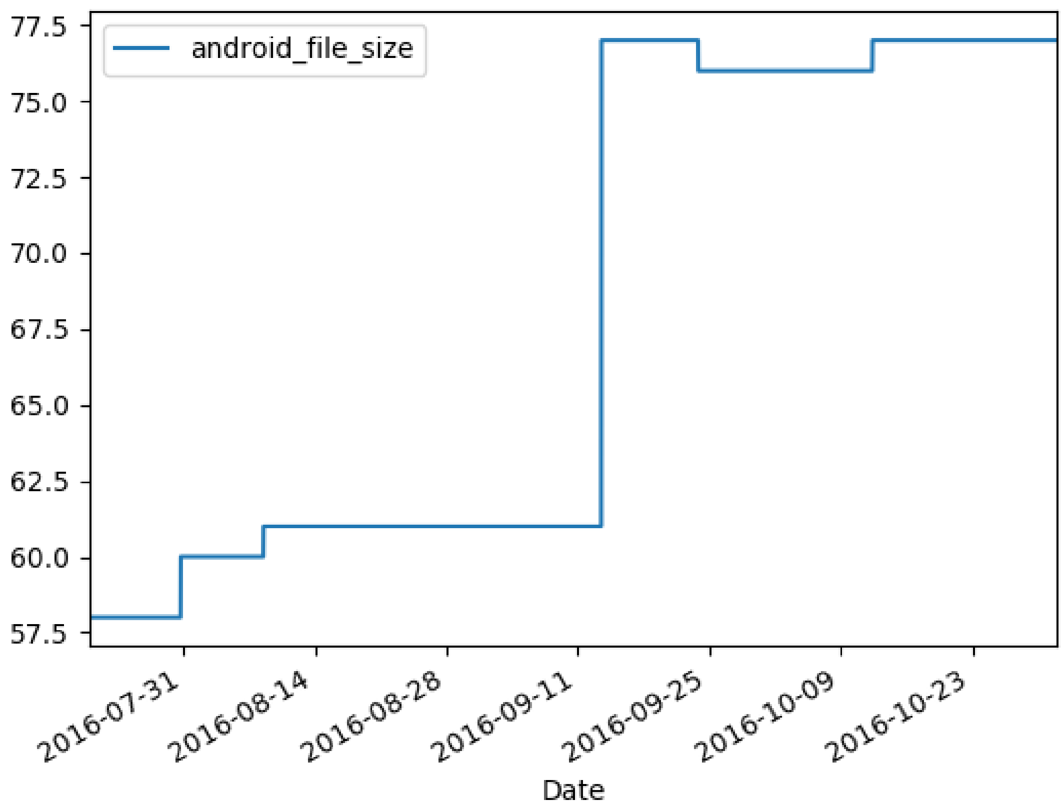
Time Series:

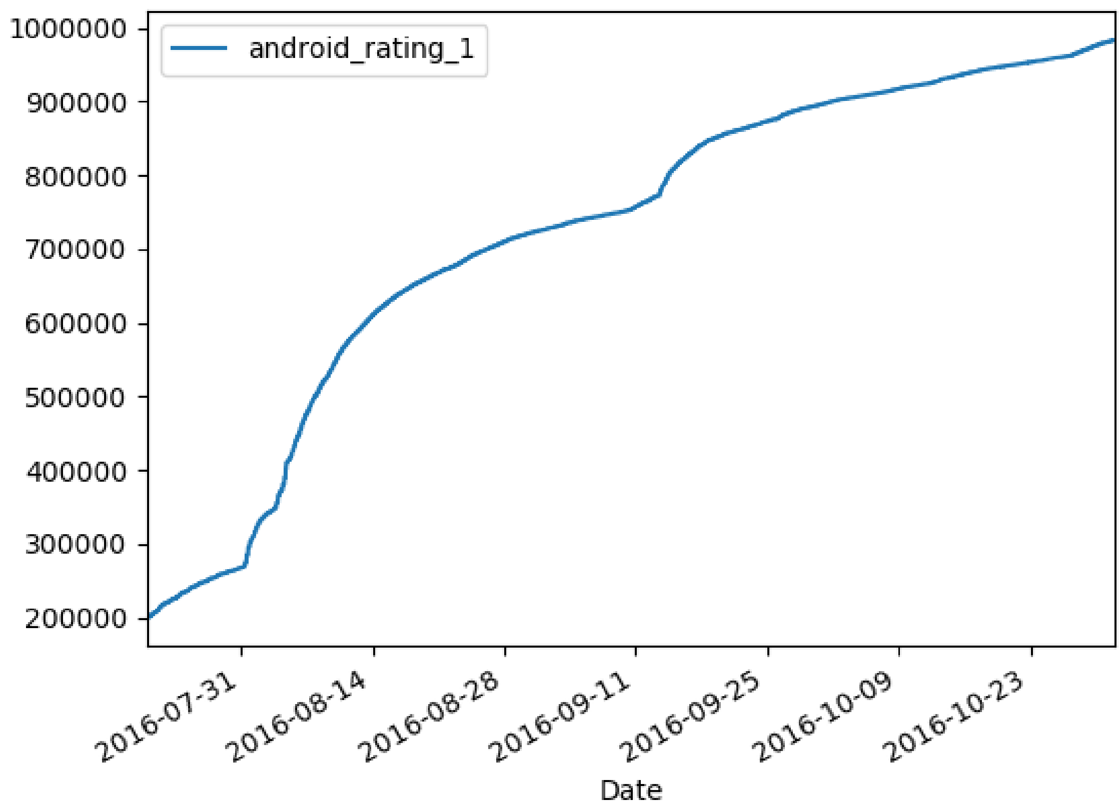
Below are time series graph for each of the 11 variables

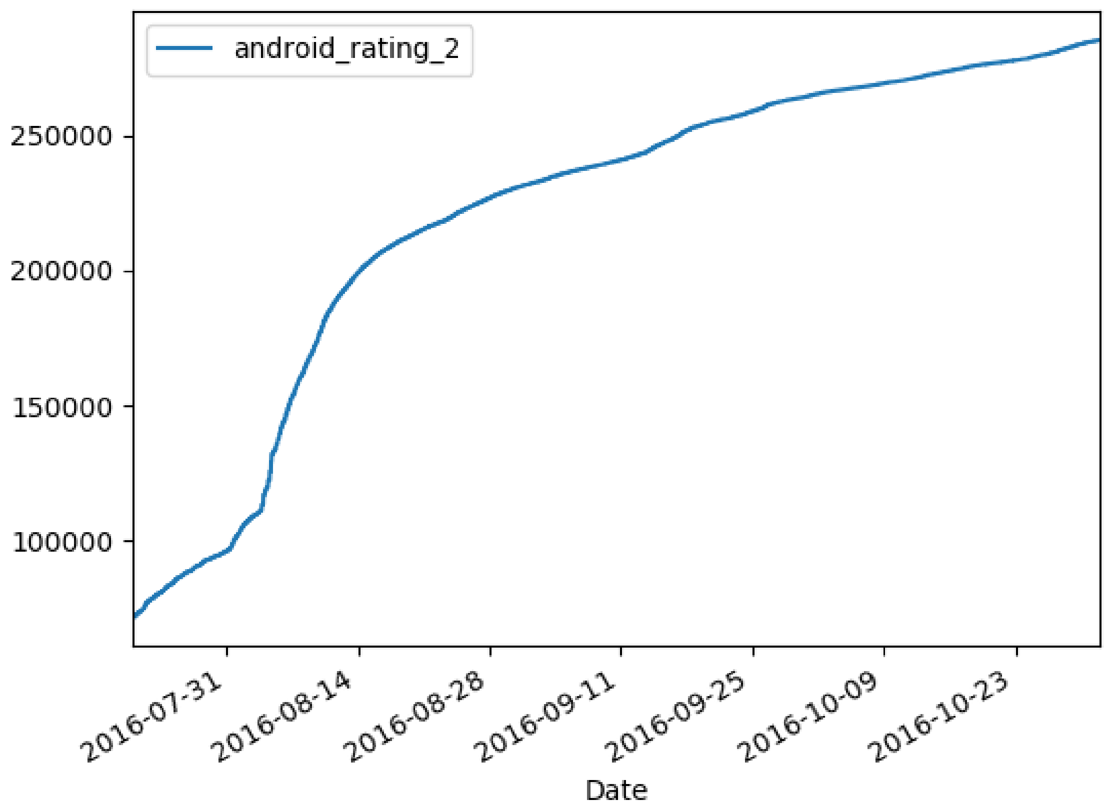


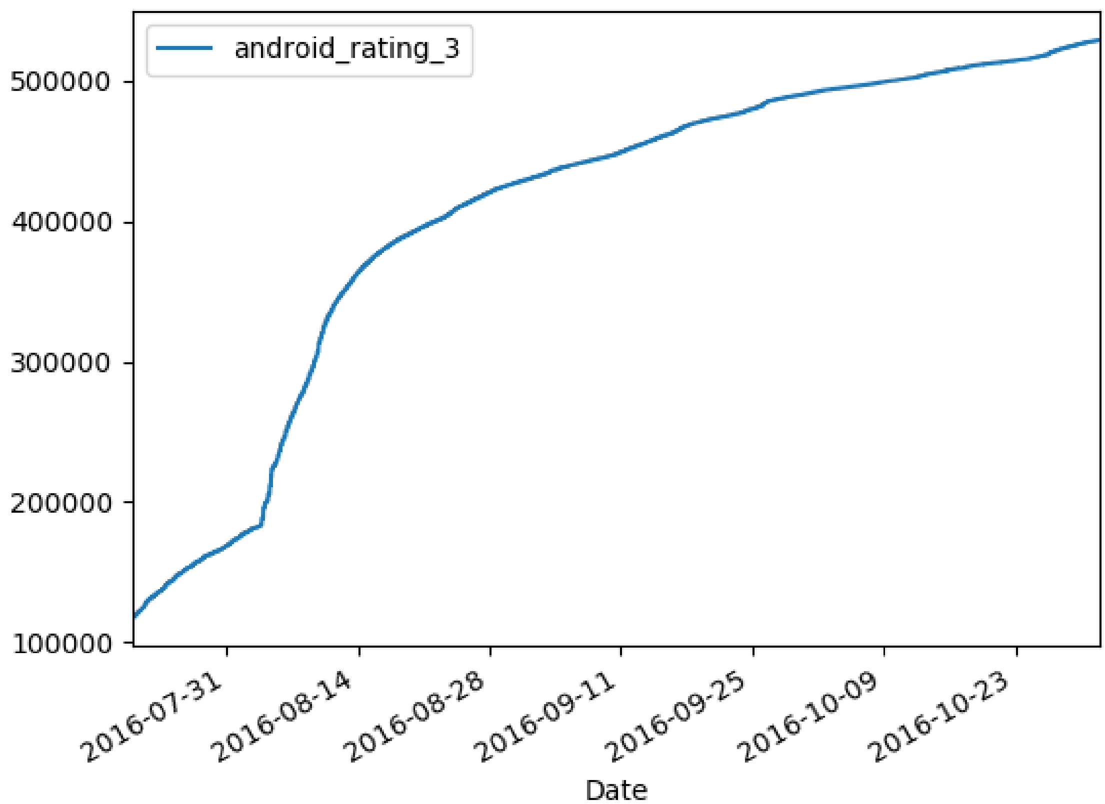


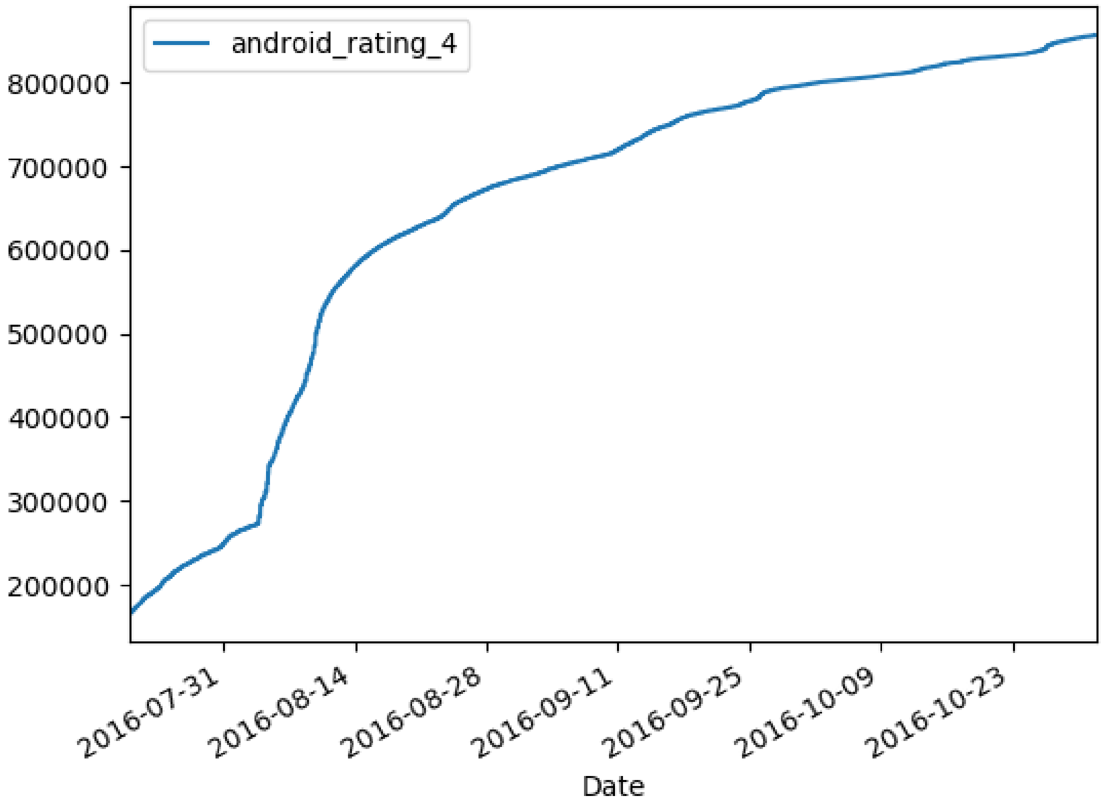
4.100
4.075
4.050
4.025
4.000
3.975
3.950
3.925
3.900
android_avg_rating
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date

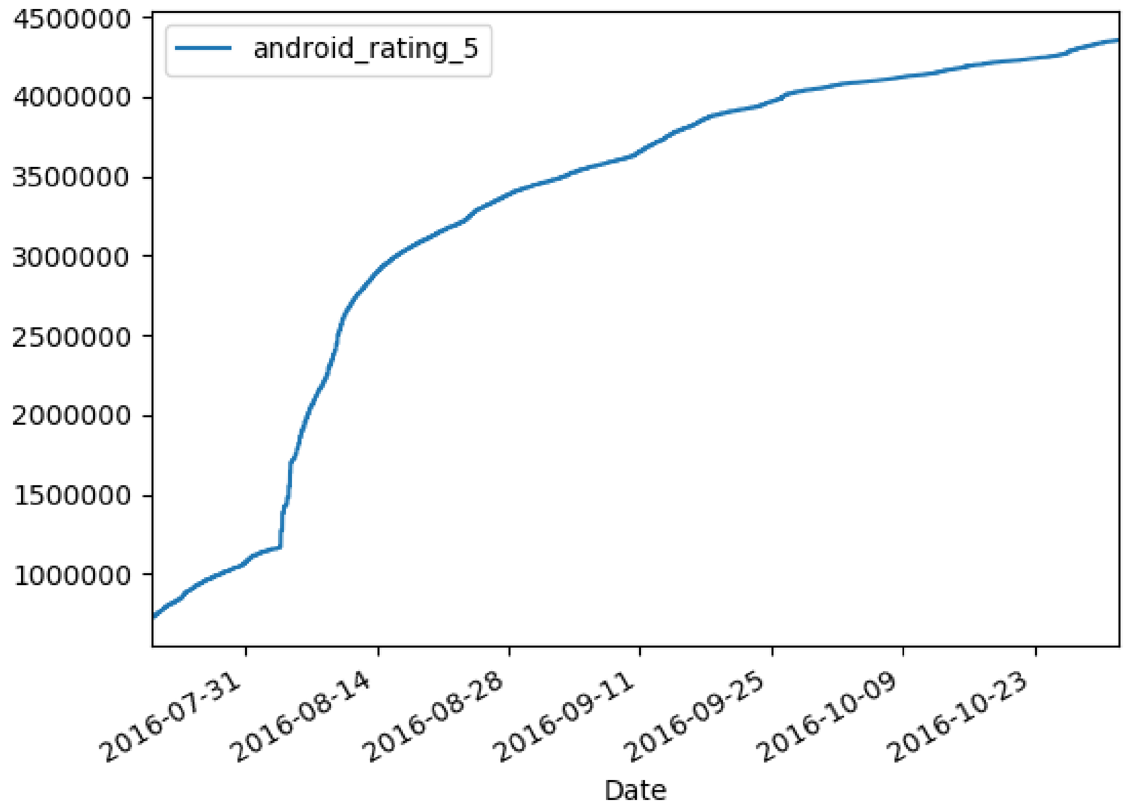

android_file_size
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date


android_rating_1
1000000
900000
800000
700000
600000
500000
400000
300000
200000
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date

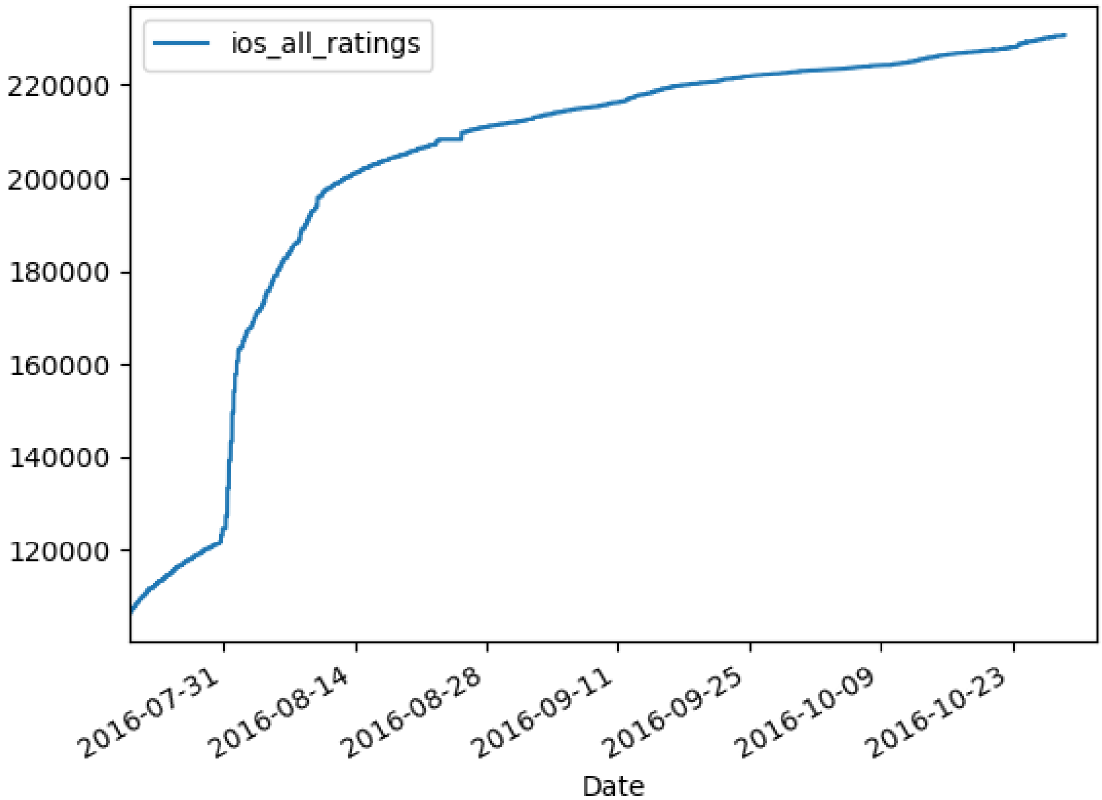

android_rating_2
250000
200000
150000
100000
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date


android_rating_3
500000
400000
300000
200000
100000
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date

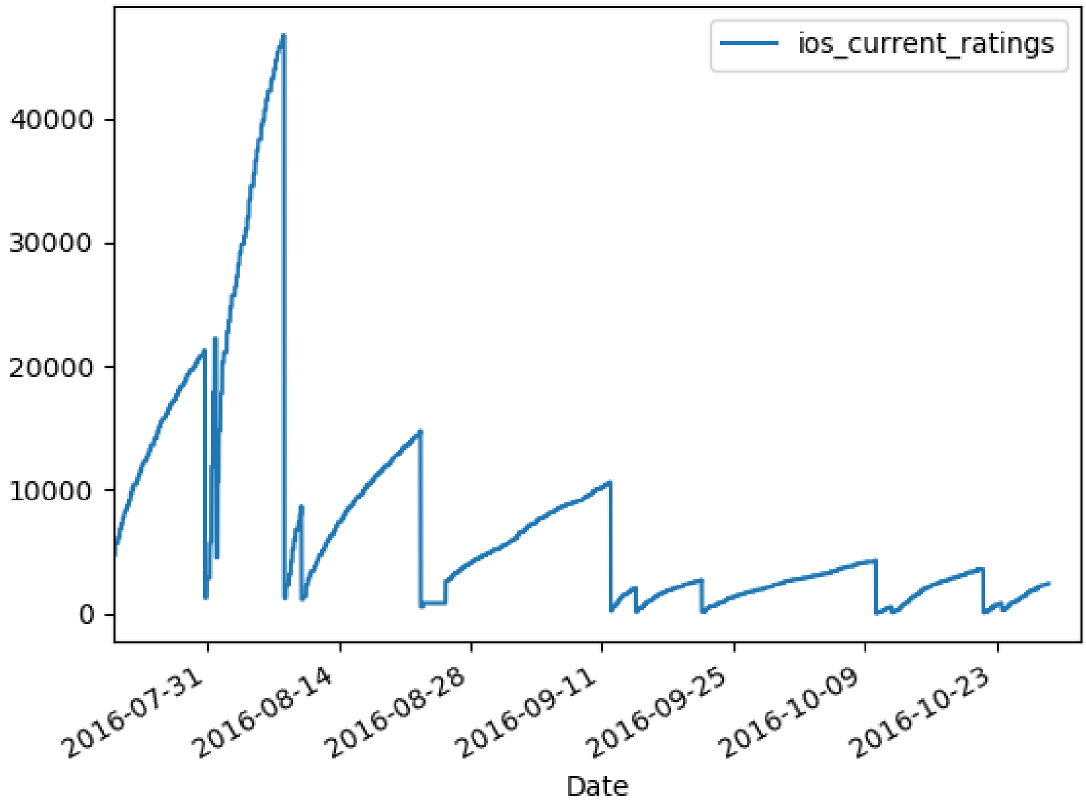

android_rating_4
800000
700000
600000
500000
400000
300000
200000
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date



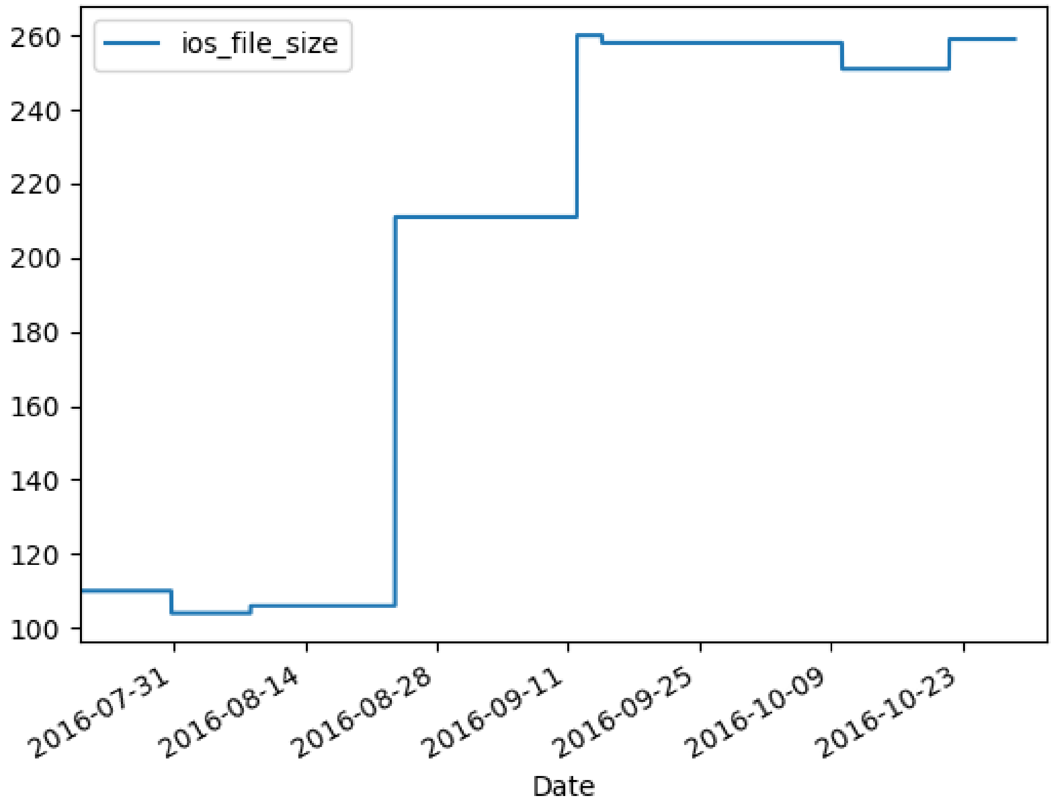
android_rating_5
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date



ios_all_ratings
220000
200000
180000
160000
140000
120000
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date



ios_current_ratings
40000
30000
20000
10000
0
2016-07-31
2016-08-14
2016-08-28
2016-09-11
2016-09-25
2016-10-09
2016-10-23
Date



Prediction models:

Linear Regression :
Android_total_ratings = 7005220
IOS_total_ratings = 225650.92877735

Lasso Regression:
Android_total_ratings = 7013087.03581924
IOS_total_ratings = 225650.92877818

Ridge :
Android_total_ratings = 7005219.99999986
Ios_total_ratings = 225650.92840677

Random Forest:
Android_total_ratings = 7004878.9
IOS_total_ratings = 230592.635

TENSOR FLOW:

Here are the outputs for the 22 unique image file after scraping:

- **Android Files:**

file1- and_screen1.jpg
monitor (score = 0.34349)
screen, CRT screen (score = 0.12488)
desktop computer (score = 0.07624)
web site, website, internet site, site (score = 0.05729)
television, television system (score = 0.02949)

file2- and_screen1.jpg
lawn mower, mower (score = 0.17193)
golf ball (score = 0.11031)
croquet ball (score = 0.08029)
mountain tent (score = 0.03014)
bow (score = 0.02765)

file3- and_screen1.jpg

web site, website, internet site, site (score = 0.49719)
monitor (score = 0.07830)
notebook, notebook computer (score = 0.05803)
iPod (score = 0.03292)
desktop computer (score = 0.02498)

file4- and_screen1.jpg
web site, website, internet site, site (score = 0.59586)
comic book (score = 0.03351)
iPod (score = 0.02989)
screen, CRT screen (score = 0.02483)
television, television system (score = 0.02017)

file5- and_screen1.jpg
web site, website, internet site, site (score = 0.62170)
television, television system (score = 0.08697)
monitor (score = 0.04946)
screen, CRT screen (score = 0.03223)
hand-held computer, hand-held microcomputer (score = 0.02868)

- **IOS Files:**

file1-ios_screen1
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)

file2-ios_screen2
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)

file3-ios_screen3
ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, tra

sh barrel, trash bin (score = 0.15498)
joystick (score = 0.06405)
cannon (score = 0.03585)
maraca (score = 0.02727)
pedestal, plinth, footstall (score = 0.02715)

file4-ios_screen4
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)

file5-ios_screen5
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)

file6-ios_screen6
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)

file7-ios_screen7
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)

file8-ios_screen8
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)

file9-ios_screen9
web site, website, internet site, site (score = 0.11637)
laptop, laptop computer (score = 0.08080)
notebook, notebook computer (score = 0.05349)
joystick (score = 0.04791)
monitor (score = 0.04169)

file10-ios_screen10
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)

file11-ios_screen11
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)

file12-ios_screen12
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)

file 13-ios_screen13
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)

file14-ios_screen14
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)

file15-ios_screen15
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)

file16-ios_screen16
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)

file17-ios_screen17
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)