EE219 PROJECT 5
Popularity Prediction on Twitter

Guanchu Ling 904590047
Yingchao Tang 404592020
Yang Guo 304681050

Introduction:

Twitter, with its public discussion model, is a good platform to predict future popularity of a topic or an event. If knowing current and previous tweet activity for a hashtag (#), we can predict if it becomes more prominent and trendy in the future and if yes by how much.

Twitter data is collected by querying popular hash-tags related to the Super Bowl spanning a period starting from 2 weeks before the game to a week after the game. We use these data to train a regression model and then use the model to making predictions for other hash-tags. The test data consists of tweets containing a hash-tag in a specified time window, and we have then used our model to predict the number of tweets containing the hash-tag posted within one hour immediately following the given time window. Finally, we use the knowing data to define our problem and try to implement our idea and show how to work.

Question 1

The dataset contains raw tweets information that was obtained during the 2015 Super Bowl, spanning a period starting from 2 weeks before the event to 1 week after the event. Every tweet is related to one or more of the following 6 hashtags: #GoHawks, #GoPatriots, #NFL, #Patriots, #SB49 and #SuperBowl. Each txt file contains all tweets that have one of the 6 hashtags.

In the original txt files, information is stored as JSON strings and tweets are sorted with respect to their posting time, from the earliest to the latest. Each tweet can be converted

to a Python Dictionary object, which includes information such as the name of the author, the posting time of the tweet, the number of followers of the user, etc.

To get a whole sense of the pattern embedded in the datasets, first we want to find some basic statistical results, including the average number of tweets per hour, the average number of followers, and the average number of retweets.

In order to get the average number of tweets per hour, we must set the hourly time window and then fill in the missing data. The first hourly time window starts at the earliest time of all the posted tweets, then subsequently add later hours after that. The posting time of each tweet can be directly accessed after the JSON string has been loaded.

In order to get the average number of followers, we should know the total number of users. Because one user may post multiple tweets during the designated period, there could be multiple records of the user in the dataset. Therefore, we should use a Python Dictionary to obtain the number of followers of each user. For the case that one user posted multiple tweets, the number of his followers may change over time, so it is reasonable to use the average value of all records.

The results are listed below:

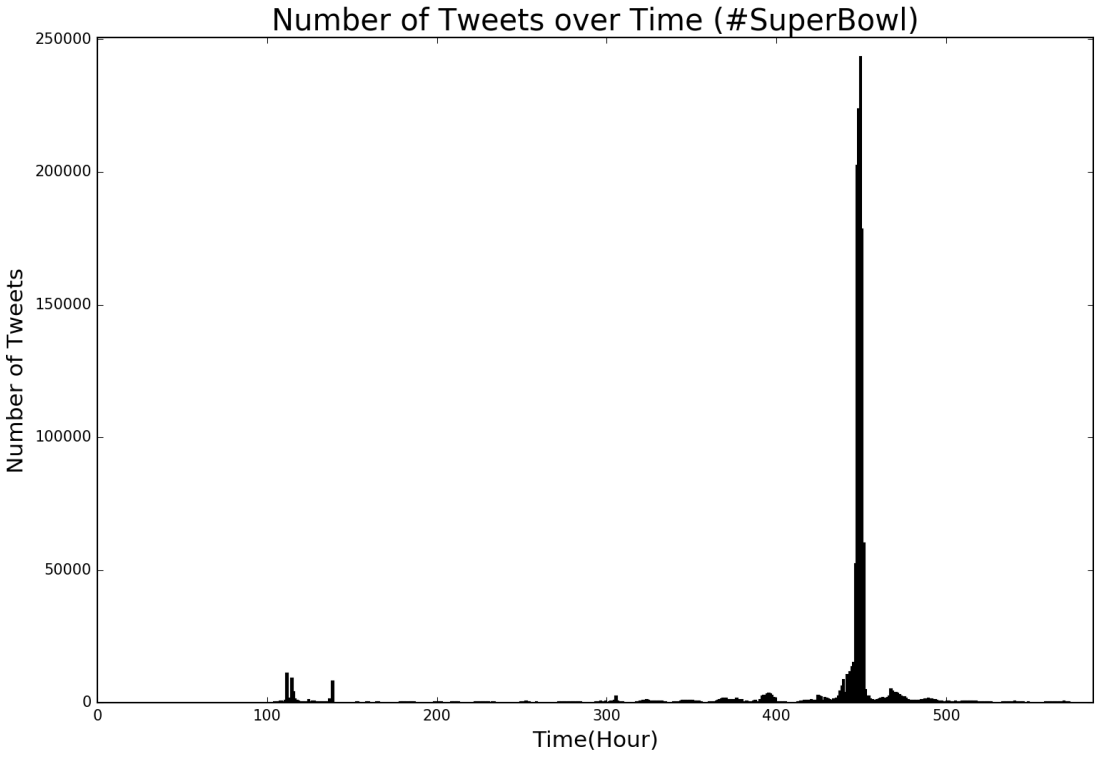
```
Statistics for #GoHawks
    Total number of tweets: 188136
    Total number of users: 77584
    Average number of tweets per hour: 325.371591304
    Average number of followers per user: 1588.18866293
    Average number of followers per tweet: 2203.93176744
    Average number of retweets per tweet: 2.01461708551
```

```
Statistics for #GoPatriots
    Total number of tweets: 26232
    Total number of users: 18087
    Average number of tweets per hour: 45.6945105736
```

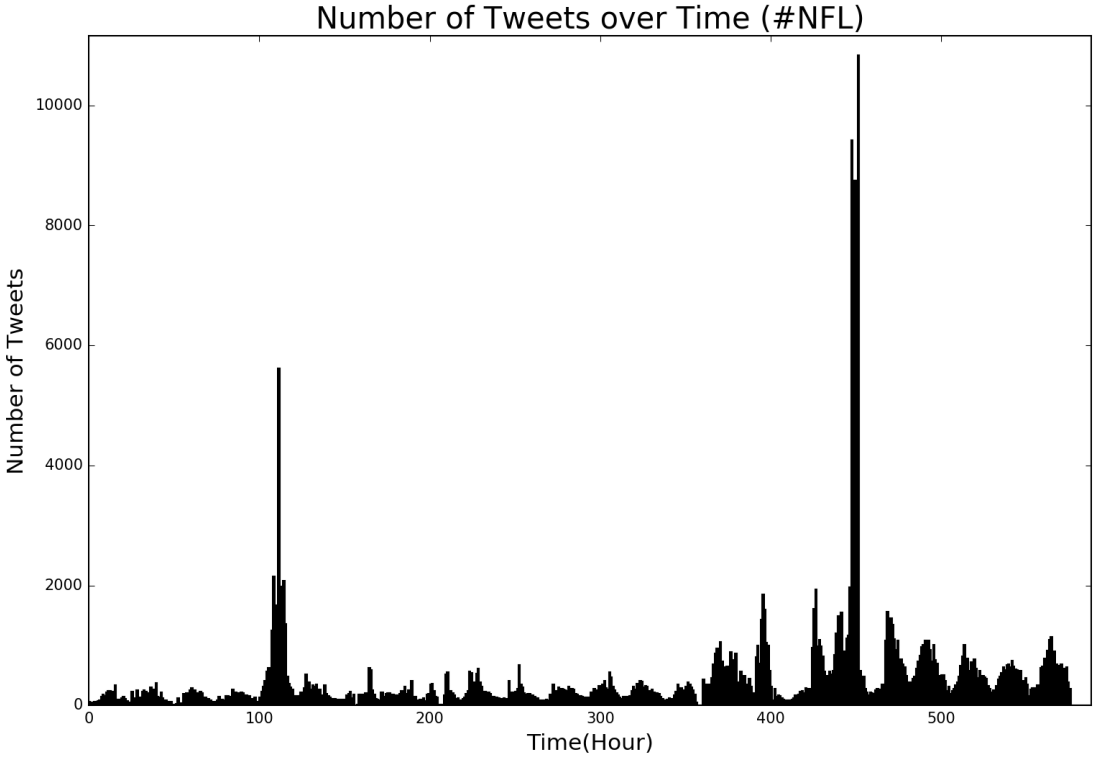
```
    Average number of followers per user: 1294.46936646
    Average number of followers per tweet: 1401.8955093
    Average number of retweets per tweet: 1.40008386703
Statistics for #NFL
    Total number of tweets: 259024
    Total number of users: 75642
    Average number of tweets per hour: 441.323431137
    Average number of followers per user: 4221.07698787
    Average number of followers per tweet: 4653.2522855
    Average number of retweets per tweet: 1.5385331089
Statistics for #Patriots
    Total number of tweets: 489713
    Total number of users: 327326
    Average number of tweets per hour: 834.555509164
    Average number of followers per user: 1695.27106215
    Average number of followers per tweet: 3309.97882842
    Average number of retweets per tweet: 1.78281564917
Statistics for #SB49
    Total number of tweets: 826951
    Total number of users: 590636
    Average number of tweets per hour: 1419.88790749
    Average number of followers per user: 2250.85025774
    Average number of followers per tweet: 10267.3168495
    Average number of retweets per tweet: 2.51114878632
Statistics for #SuperBowl
    Total number of tweets: 1348767
    Total number of users: 693087
    Average number of tweets per hour: 2302.50040188
    Average number of followers per user: 3798.67911709
    Average number of followers per tweet: 8858.97466278
    Average number of retweets per tweet: 2.3882723999
```

The plots of “Number of tweets per hour” for hashtags #SuperBowl and #NFL are

shown below:



Number of tweets over time for #SuperBowl



Number of tweets over time for #NFL

From these plots, we can see that the there are two peaks in the 2015 Super Bowl period. The highest peaks occurred on the Super Bowl day, which makes sense that people post much more tweets during the important event. Another peak occurs approximately 350 hours before Super Bowl, which should be another important event of NFL.

Question 2

Now we want to use a linear regression model to predict the number of tweets in the next hour, given 5 attributes extracted from the tweets in the previous hour. The five attributes are:

```
number of tweets
total number of retweets
sum of the number of followers of the users
maximum number of followers of the users
time of the day
```

Among these 5 attributes, we should notice that the attribute "time of the day" is not numerical-valued but string-valued. In order for the regression algorithm to perform calculation, all string values must be properly converted to numerical values.

A straightforward approach is to assign every string-valued item a corresponding numerical-valued number (i.e. simply use integers 1 – 24 to represent all 24 hours in a day). However, by doing this, these attributes will become ordinal values rather than categorical values. This is obviously incorrect because we cannot say that the 24th hour has more weight than the 1st hour.

In fact, what we want to do is denote every categorical value differently while keep their weights treated equally. Therefore, One-Hot Encoding is utilized for this purpose. For every categorical attribute, the complete set of all possible values across the whole dataset will first be obtained by iterating the dataset; then for every possible value of this attribute, a new Boolean attribute will be added to the dataset to denote whether or

not the item belongs to this category. For example, the attribute "10th hour of the day" will be added to the dataset. If a tweet was posted during the 10th hour of the day, the value under this attribute will be 1; otherwise it will be 0. In this sense, for all 24 possible values in "time of the day", there will be totally 24 new attributes added to the dataset.

Since we are applying the linear regression analysis to the data, we should evaluate the training accuracy of the linear model. We choose to use "Coefficient of Determination" (often called "R-squared") to measure the accuracy quantitatively. R-squared is a number that indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. For linear regression, the value of R-squared is the square of correlation coefficient of the sample. It is calculated as:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

where SS_{res} is residual sum of squares, SS_{tot} is total sum of squares. If the linear model has high accuracy, SS_{res} will be small and thus R^2 will be closer to 1. Therefore, the value of R-squared can be used as the representation of the accuracy of the model.

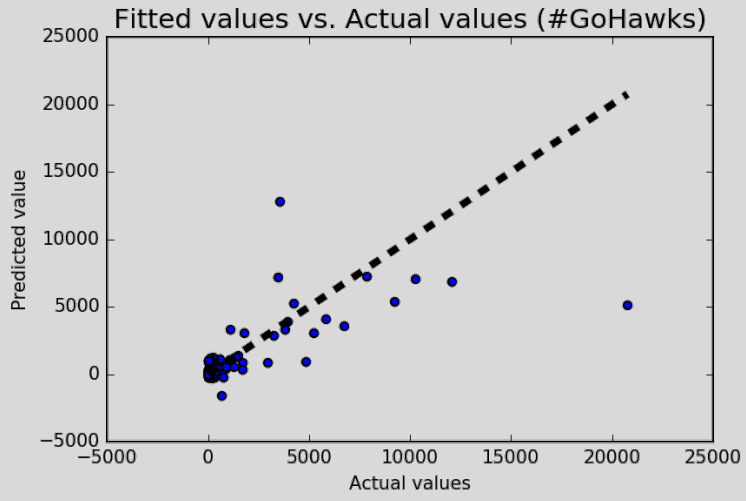
The results of linear regression prediction are listed below:

#GoHawks

R-squared: 0.532

Results of t-test:

	coef	std err	t	P>|t|
number_of_tweets	1.4237	0.164	8.659	0.000
number_of_retweets	-19.8237	3.889	-5.097	0.000
sum_of_followers	-0.0003	8.51e-05	-3.270	0.001
max_number_of_followers	0.0002	0.000	1.126	0.261
0th_hour	16.9938	187.031	0.091	0.928
1th_hour	12.1762	186.821	0.065	0.948
2th_hour	3.2168	186.790	0.017	0.986
3th_hour	23.4880	190.666	0.123	0.902
4th_hour	34.2407	190.662	0.180	0.858
5th_hour	52.2219	190.884	0.274	0.785
6th_hour	120.2952	190.978	0.630	0.529
7th_hour	142.3044	190.957	0.745	0.456
8th_hour	205.6231	193.187	1.064	0.288
9th_hour	197.4692	191.701	1.030	0.303
10th_hour	324.2885	191.838	1.690	0.092
11th_hour	215.3796	191.699	1.124	0.262
12th_hour	17.7893	192.423	0.092	0.926
13th_hour	253.5218	192.556	1.317	0.189
14th_hour	999.8690	193.175	5.176	0.000
15th_hour	-243.6088	198.152	-1.229	0.219
16th_hour	199.9072	193.022	1.036	0.301
17th_hour	172.4385	193.656	0.890	0.374
18th_hour	-130.1525	192.812	-0.675	0.500
19th_hour	-16.4555	197.652	-0.083	0.934
20th_hour	157.2451	193.179	0.814	0.416
21th_hour	85.4406	193.508	0.442	0.659
22th_hour	26.5137	191.449	0.138	0.890
23th_hour	-7.8748	190.873	-0.041	0.967

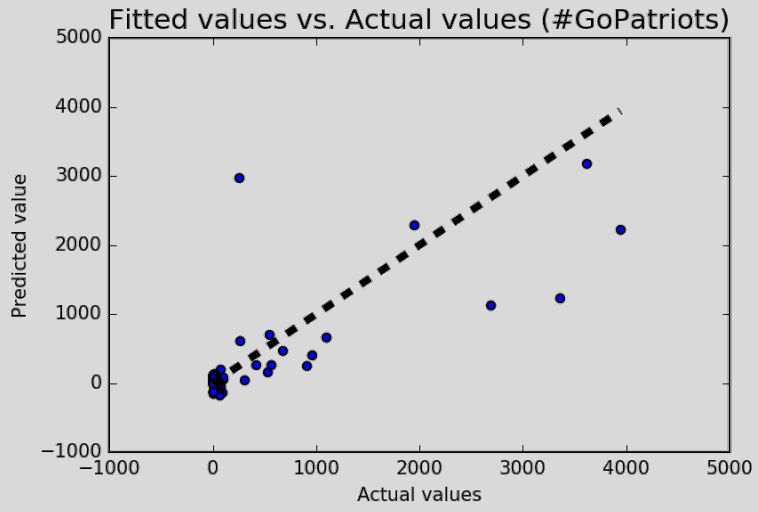


#GoPatriots

R-squared: 0.627

Results of t-test:

	coef	std err	t	P>|t|
number_of_tweets	0.2511	0.203	1.236	0.217
number_of_retweets	-10.9993	3.836	-2.867	0.004
sum_of_followers	0.0006	0.000	3.230	0.001
max_number_of_followers	-0.0007	0.000	-3.548	0.000
0th_hour	3.1339	39.392	0.080	0.937
1th_hour	3.7415	39.391	0.095	0.924
2th_hour	6.1516	39.394	0.156	0.876
3th_hour	9.5014	39.413	0.241	0.810
4th_hour	10.9079	39.429	0.277	0.782
5th_hour	11.4425	39.409	0.290	0.772
6th_hour	9.7617	39.410	0.248	0.804
7th_hour	20.9183	39.428	0.531	0.596
8th_hour	15.4460	39.403	0.392	0.695
9th_hour	23.8400	39.557	0.603	0.547
10th_hour	14.1391	39.437	0.359	0.720
11th_hour	33.6158	40.157	0.837	0.403
12th_hour	119.8666	39.450	3.038	0.002
13th_hour	49.6551	39.945	1.243	0.214
14th_hour	-22.2959	40.144	-0.555	0.579
15th_hour	112.0844	39.919	2.808	0.005
16th_hour	80.0945	39.711	2.017	0.044
17th_hour	-122.2721	39.740	-3.077	0.002
18th_hour	13.7698	39.708	0.347	0.729
19th_hour	7.9766	39.415	0.202	0.840
20th_hour	7.8640	39.419	0.199	0.842
21th_hour	6.7570	39.440	0.171	0.864
22th_hour	2.7549	39.393	0.070	0.944
23th_hour	2.9905	40.239	0.074	0.941

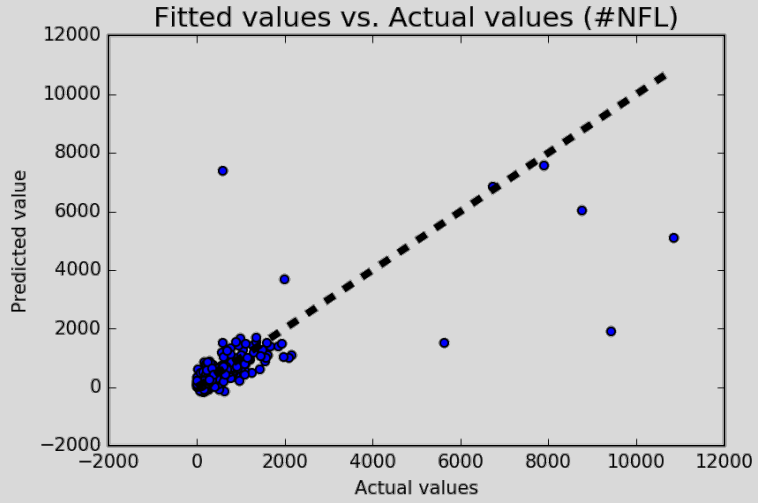


#NFL

R-squared: 0.585

Results of t-test:

	coef	std err	t	P>|t|
number_of_tweets	0.4752	0.103	4.601	0.000
number_of_retweets	-1.5832	2.807	-0.564	0.573
sum_of_followers	7.315e-05	2.65e-05	2.756	0.006
max_number_of_followers	-9.432e-05	3.7e-05	-2.547	0.011
0th_hour	56.3083	116.806	0.482	0.630
1th_hour	67.4735	116.873	0.577	0.564
2th_hour	76.3124	116.595	0.655	0.513
3th_hour	92.3850	116.643	0.792	0.429
4th_hour	83.8884	116.783	0.718	0.473
5th_hour	169.5487	116.801	1.452	0.147
6th_hour	196.7151	117.237	1.678	0.094
7th_hour	149.7344	117.061	1.279	0.201
8th_hour	166.4218	117.790	1.413	0.158
9th_hour	120.6821	117.350	1.028	0.304
10th_hour	237.9457	117.583	2.024	0.043
11th_hour	244.0414	120.708	2.022	0.044
12th_hour	83.3442	120.386	0.692	0.489
13th_hour	220.2633	121.927	1.807	0.071
14th_hour	634.0510	121.109	5.235	0.000
15th_hour	15.7214	123.964	0.127	0.899
16th_hour	269.0316	122.193	2.202	0.028
17th_hour	218.6963	125.524	1.742	0.082
18th_hour	283.3371	122.493	2.313	0.021
19th_hour	-198.0234	122.052	-1.622	0.105
20th_hour	86.8392	120.082	0.723	0.470
21th_hour	81.4536	119.888	0.679	0.497
22th_hour	33.0973	119.432	0.277	0.782
23th_hour	63.9559	119.234	0.536	0.592

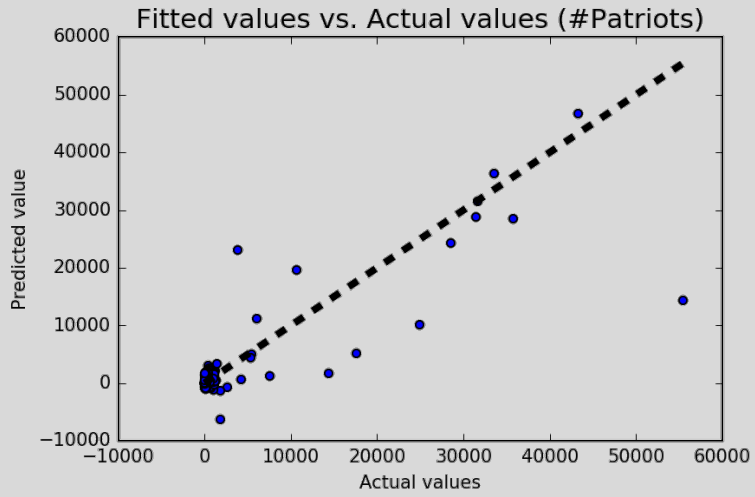


#Patriots

R-squared: 0.723

Results of t-test:

	coef	std err	t	P>|t|
number_of_tweets	0.8230	0.039	21.051	0.000
number_of_retweets	-19.7359	3.548	-5.562	0.000
sum_of_followers	3.505e-05	2.54e-05	1.382	0.168
max_number_of_followers	0.0001	9.69e-05	1.402	0.161
0th_hour	59.0273	472.067	0.125	0.901
1th_hour	47.4428	471.927	0.101	0.920
2th_hour	50.7466	471.983	0.108	0.914
3th_hour	77.6890	472.291	0.164	0.869
4th_hour	79.4322	473.004	0.168	0.867
5th_hour	76.4594	474.558	0.161	0.872
6th_hour	70.7354	473.728	0.149	0.881
7th_hour	21.2432	476.106	0.045	0.964
8th_hour	96.7734	473.981	0.204	0.838
9th_hour	500.0589	474.386	1.054	0.292
10th_hour	1805.4470	474.871	3.802	0.000
11th_hour	-44.4154	487.337	-0.091	0.927
12th_hour	-306.7086	489.203	-0.627	0.531
13th_hour	31.0503	488.518	0.064	0.949
14th_hour	435.8040	491.708	0.886	0.376
15th_hour	-78.9151	492.324	-0.160	0.873
16th_hour	-463.0496	498.382	-0.929	0.353
17th_hour	1203.5731	486.460	2.474	0.014
18th_hour	-92.8105	490.764	-0.189	0.850
19th_hour	-841.7744	484.449	-1.738	0.083
20th_hour	271.5677	491.369	0.553	0.581
21th_hour	68.8244	483.349	0.142	0.887
22th_hour	95.4799	482.402	0.198	0.843
23th_hour	28.7491	482.046	0.060	0.952

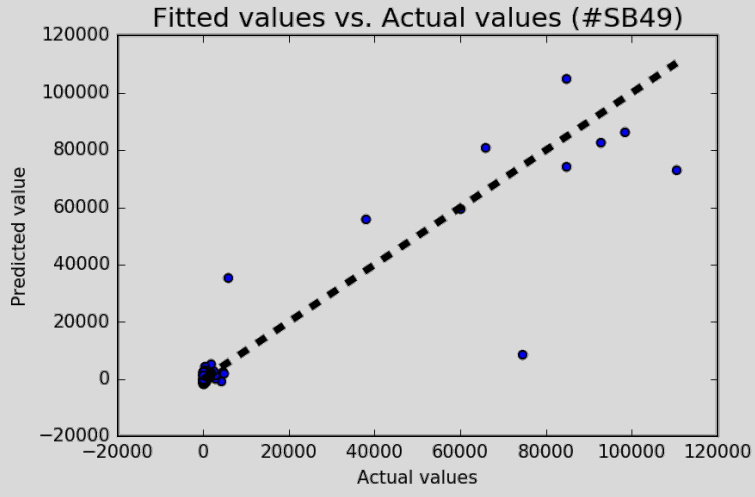


#SB49

R-squared: 0.857

Results of t-test:

	coef	std err	t	P>|t|
number_of_tweets	0.9414	0.030	31.904	0.000
number_of_retweets	-75.1845	10.554	-7.124	0.000
sum_of_followers	3.355e-06	4.49e-06	0.748	0.455
max_number_of_followers	0.0002	4.15e-05	4.185	0.000
0th_hour	-70.8379	780.105	-0.091	0.928
1th_hour	-102.1442	782.115	-0.131	0.896
2th_hour	-164.2185	782.418	-0.210	0.834
3th_hour	-197.2752	782.395	-0.252	0.801
4th_hour	-256.9267	786.731	-0.327	0.744
5th_hour	2494.5636	785.829	3.174	0.002
6th_hour	1319.3125	785.055	1.681	0.093
7th_hour	-1295.7993	807.914	-1.604	0.109
8th_hour	-1039.8350	803.523	-1.294	0.196
9th_hour	30.3122	819.470	0.037	0.971
10th_hour	-182.7876	809.576	-0.226	0.821
11th_hour	205.5069	804.837	0.255	0.799
12th_hour	-430.8907	817.501	-0.527	0.598
13th_hour	-1031.0163	804.730	-1.281	0.201
14th_hour	-1397.9380	797.865	-1.752	0.080
15th_hour	-65.5140	799.750	-0.082	0.935
16th_hour	-50.2012	799.358	-0.063	0.950
17th_hour	-292.5488	803.164	-0.364	0.716
18th_hour	-59.0724	795.882	-0.074	0.941
19th_hour	-174.8924	797.157	-0.219	0.826
20th_hour	-136.8856	795.658	-0.172	0.863
21th_hour	-6.9382	795.166	-0.009	0.993
22th_hour	40.7550	795.142	0.051	0.959
23th_hour	41.0264	795.259	0.052	0.959

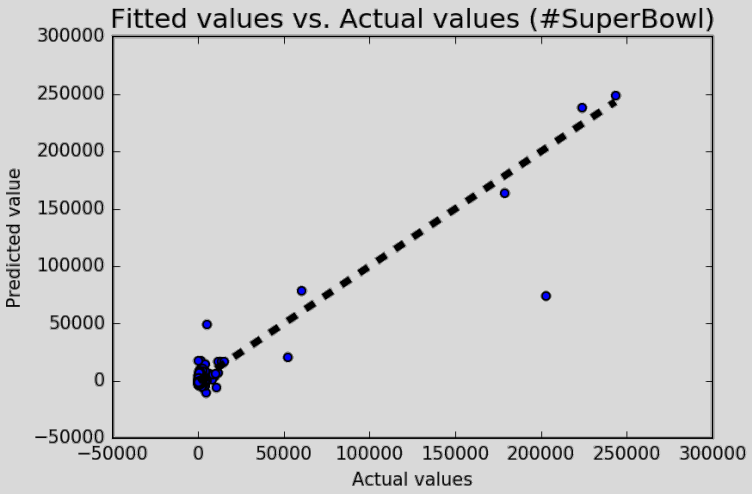


#SuperBowl

R-squared: 0.870

Results of t-test:

	coef	std err	t	P>|t|
number_of_tweets	2.2978	0.084	27.221	0.000
number_of_retweets	-3.9616	1.741	-2.275	0.023
sum_of_followers	-0.0002	1.2e-05	-19.193	0.000
max_number_of_followers	0.0011	0.000	9.488	0.000
0th_hour	-1152.9726	1329.978	-0.867	0.386
1th_hour	-523.1336	1320.899	-0.396	0.692
2th_hour	-320.5041	1321.701	-0.242	0.808
3th_hour	-754.8762	1327.757	-0.569	0.570
4th_hour	-325.6118	1322.931	-0.246	0.806
5th_hour	-656.1878	1335.849	-0.491	0.623
6th_hour	-210.0972	1330.078	-0.158	0.875
7th_hour	-707.7970	1337.028	-0.529	0.597
8th_hour	-50.0254	1340.333	-0.037	0.970
9th_hour	-659.1034	1337.665	-0.493	0.622
10th_hour	-954.4856	1376.758	-0.693	0.488
11th_hour	-460.5540	1373.900	-0.335	0.738
12th_hour	-182.9047	1365.986	-0.134	0.894
13th_hour	981.7322	1375.887	0.714	0.476
14th_hour	5249.6969	1390.505	3.775	0.000
15th_hour	-2006.0965	1395.997	-1.437	0.151
16th_hour	-1450.4399	1388.073	-1.045	0.297
17th_hour	24.4359	1389.325	0.018	0.986
18th_hour	-2880.8218	1370.936	-2.101	0.036
19th_hour	-3352.1905	1369.294	-2.448	0.015
20th_hour	-1327.1132	1361.904	-0.974	0.330
21th_hour	-1105.3829	1355.967	-0.815	0.415
22th_hour	-799.4634	1351.176	-0.592	0.554
23th_hour	-1423.8574	1358.102	-1.048	0.295



From all the results, we can see that larger training datasets yield better accuracy. For example, the dataset for "#GoPatriots" contains about 26000 tweets, and the R-squared value is as low as 0.627. In comparison, the dataset for "#SuperBowl" contains about 1340000 tweets, and the corresponding R-squared value is as high as 0.870.

The P-values of attributes "number of tweets", "total number of retweets", "sum of number of followers" and "max number of followers" are very close to zero, which means that these four attributes are critical and they are important during the regression analysis. In comparison, the attribute "time of the day" (which has been split into 24 sub-attributes by One-hot Encoding) is much less important because the P-values of this attribute is very large and close to 1. Therefore, it is reasonable to exclude "time of the day" without impairing the accuracy of the model.

Question 3

In this question, in order to design a better regression model with higher accuracy, we try to add 5 new attributes to the previous model. Now the modified training attributes are:

num_tweets: total number of tweets within an hour

num_retweets: total number of retweets within an hour

sum_followers: sum of the number of followers of the users

max_followers: maximum number of followers of the users

time_of_day: one of all 24 hours of the day

num_URLs: total number of URLs included by the tweets

num_authors: total number of authors involved in an hour

num_mentions: total number of mentions in the tweets

ranking_score: sum of ranking scores of all tweets in an hour

num_hashtags: total number of hashtags in the tweets

After adding attributes and extracting data from files, we apply the same linear regression model on the dataset, just as that in the previous question. Then we also perform t-test and use P-values to determine the importance of the attributes. The results are listed below:

#GoHawks

R-squared: 0.724

Results of t-test:

	coef	std err	t	P>|t|
num_tweets	-67.2442	4.190	-16.047	0.000
num_retweets	13.4966	3.547	3.805	0.000
sum_followers	-0.0003	6.86e-05	-4.390	0.000
max_followers	0.0002	0.000	1.234	0.218
num_URLs	7.5628	1.508	5.016	0.000
num_authors	4.5277	0.745	6.079	0.000
num_mensions	2.8669	0.468	6.130	0.000
ranking_score	13.5304	0.838	16.141	0.000
num_hashtags	0.3588	0.329	1.091	0.276
0th_hour	54.3914	144.659	0.376	0.707
1th_hour	-18.1857	144.375	-0.126	0.900
2th_hour	16.6854	144.484	0.115	0.908
3th_hour	14.8236	147.270	0.101	0.920
4th_hour	-21.3342	147.266	-0.145	0.885
5th_hour	-4.0843	147.593	-0.028	0.978
6th_hour	-90.3962	148.223	-0.610	0.542
7th_hour	-131.8647	149.518	-0.882	0.378
8th_hour	-180.9324	152.608	-1.186	0.236
9th_hour	-242.7575	152.420	-1.593	0.112
10th_hour	-276.5702	154.535	-1.790	0.074
11th_hour	-475.3761	155.993	-3.047	0.002
12th_hour	-452.4758	153.927	-2.940	0.003
13th_hour	-175.3495	152.959	-1.146	0.252
14th_hour	575.9658	153.188	3.760	0.000
15th_hour	-289.4265	154.147	-1.878	0.061
16th_hour	-67.9202	151.560	-0.448	0.654
17th_hour	163.8076	152.011	1.078	0.282
18th_hour	-156.5957	149.677	-1.046	0.296
19th_hour	12.6608	155.421	0.081	0.935
20th_hour	-2.0205	151.888	-0.013	0.989
21th_hour	-52.0524	150.618	-0.346	0.730
22th_hour	7.2817	148.972	0.049	0.961
23th_hour	-2.1436	147.790	-0.015	0.988

Most significant attributes:

num_tweets

num_retweets

sum_followers

num_URLs

num_authors

num_mentions

ranking_score

#GoPatriots

R-squared: 0.894

Results of t-test:

	coef	std err	t	P>|t|
num_tweets	-8.0301	2.727	-2.944	0.003
num_retweets	-43.2781	2.608	-16.592	0.000
sum_followers	-0.0020	0.000	-9.179	0.000
max_followers	0.0019	0.000	8.756	0.000
num_URLs	13.5502	0.853	15.892	0.000
num_authors	-6.6753	0.593	-11.248	0.000
num_mensions	3.1152	0.402	7.740	0.000
ranking_score	2.2835	0.472	4.836	0.000
num_hashtags	1.8346	0.315	5.823	0.000
0th_hour	-13.4494	21.067	-0.638	0.523
1th_hour	-0.9546	21.060	-0.045	0.964
2th_hour	-11.1825	21.073	-0.531	0.596
3th_hour	1.2885	21.079	0.061	0.951
4th_hour	-11.0857	21.099	-0.525	0.600
5th_hour	-12.2316	21.084	-0.580	0.562
6th_hour	-30.7719	21.122	-1.457	0.146
7th_hour	-17.9052	21.133	-0.847	0.397
8th_hour	-23.6883	21.136	-1.121	0.263
9th_hour	-28.8476	21.277	-1.356	0.176
10th_hour	-50.5863	21.388	-2.365	0.018
11th_hour	-15.3564	21.620	-0.710	0.478
12th_hour	66.9282	21.341	3.136	0.002
13th_hour	11.5140	21.887	0.526	0.599
14th_hour	-39.9123	21.823	-1.829	0.068
15th_hour	37.0422	21.613	1.714	0.087
16th_hour	-6.6506	21.945	-0.303	0.762
17th_hour	-5.8325	21.765	-0.268	0.789
18th_hour	-2.3024	21.274	-0.108	0.914
19th_hour	-4.3360	21.109	-0.205	0.837
20th_hour	6.1273	21.077	0.291	0.771
21th_hour	15.0085	21.090	0.712	0.477
22th_hour	-5.5452	21.062	-0.263	0.792
23th_hour	-5.2516	21.517	-0.244	0.807

Most significant attributes:

num_tweets

num_retweets

sum_followers

max_followers

num_URLs

num_authors

num_mentions

ranking_score

num_hashtags

#NFL

R-squared: 0.765

Results of t-test:

	coef	std err	t	P>|t|
num_tweets	-1.1317	1.496	-0.757	0.450
num_retweets	-9.9461	2.517	-3.951	0.000
sum_followers	-2.376e-05	2.42e-05	-0.980	0.327
max_followers	1.302e-05	3.21e-05	0.405	0.686
num_URLs	-0.4606	0.174	-2.648	0.008
num_authors	-4.4403	0.343	-12.962	0.000
num_mensions	2.9480	0.644	4.574	0.000
ranking_score	0.2665	0.307	0.867	0.386
num_hashtags	1.1542	0.083	13.882	0.000
0th_hour	-85.0332	91.043	-0.934	0.351
1th_hour	-76.5635	90.406	-0.847	0.397
2th_hour	-68.3849	89.549	-0.764	0.445
3th_hour	20.6908	89.562	0.231	0.817
4th_hour	67.9376	90.042	0.755	0.451
5th_hour	172.6458	89.976	1.919	0.056
6th_hour	236.0467	91.980	2.566	0.011
7th_hour	180.6493	93.876	1.924	0.055
8th_hour	163.9958	96.150	1.706	0.089
9th_hour	120.3897	95.002	1.267	0.206
10th_hour	187.8101	94.409	1.989	0.047
11th_hour	144.3960	99.966	1.444	0.149
12th_hour	95.9948	99.206	0.968	0.334
13th_hour	92.6347	97.977	0.945	0.345
14th_hour	530.2282	98.621	5.376	0.000
15th_hour	50.4429	99.130	0.509	0.611
16th_hour	18.9676	96.550	0.196	0.844
17th_hour	189.4597	99.700	1.900	0.058
18th_hour	119.1781	97.274	1.225	0.221
19th_hour	-59.1401	96.239	-0.615	0.539
20th_hour	-50.2045	95.079	-0.528	0.598
21th_hour	-64.2906	95.030	-0.677	0.499
22th_hour	-167.6198	93.279	-1.797	0.073
23th_hour	-131.1583	93.127	-1.408	0.160

Most significant attributes:

num_retweets

num_URLs

num_authors

num_mentions

num_hashtags

#Patriots

R-squared: 0.823

Results of t-test:

	coef	std err	t	P>|t|
num_tweets	-63.0835	4.992	-12.636	0.000
num_retweets	-27.0591	2.945	-9.188	0.000
sum_followers	0.0004	5.7e-05	6.938	0.000
max_followers	-0.0006	0.000	-5.621	0.000
num_URLs	-5.0346	1.739	-2.895	0.004
num_authors	2.2108	0.994	2.224	0.027
num_mensions	7.0053	0.955	7.339	0.000
ranking_score	11.0880	0.949	11.687	0.000
num_hashtags	3.6577	0.413	8.852	0.000
0th_hour	197.8735	381.113	0.519	0.604
1th_hour	-65.4957	382.060	-0.171	0.864
2th_hour	-36.3362	380.504	-0.095	0.924
3th_hour	-156.8004	380.059	-0.413	0.680
4th_hour	-114.6024	381.599	-0.300	0.764
5th_hour	-164.1821	383.748	-0.428	0.669
6th_hour	-331.3956	388.221	-0.854	0.394
7th_hour	-669.8187	392.247	-1.708	0.088
8th_hour	-756.3629	394.680	-1.916	0.056
9th_hour	-313.3698	405.646	-0.773	0.440
10th_hour	972.2628	391.872	2.481	0.013
11th_hour	-843.7053	401.738	-2.100	0.036
12th_hour	-711.3069	403.419	-1.763	0.078
13th_hour	-581.7998	403.619	-1.441	0.150
14th_hour	-313.4766	407.543	-0.769	0.442
15th_hour	-562.1471	405.120	-1.388	0.166
16th_hour	-76.3642	404.922	-0.189	0.850
17th_hour	140.2104	401.606	0.349	0.727
18th_hour	345.9975	409.073	0.846	0.398
19th_hour	-362.4667	400.155	-0.906	0.365
20th_hour	330.5370	402.736	0.821	0.412
21th_hour	-309.1792	397.389	-0.778	0.437
22th_hour	151.0046	393.973	0.383	0.702
23th_hour	225.2162	392.846	0.573	0.567

Most significant attributes:

num_tweets

num_retweets

sum_followers

max_followers

num_URLs

num_mentions

ranking_score

num_hashtags

#SB49

R-squared: 0.902

Results of t-test:

	coef	std err	t	P>|t|
num_tweets	-62.3620	7.510	-8.304	0.000
num_retweets	38.2823	16.974	2.255	0.025
sum_followers	0.0002	2.08e-05	9.917	0.000
max_followers	-0.0004	6.51e-05	-6.221	0.000
num_URLs	5.7079	1.908	2.992	0.003
num_authors	-2.3755	0.971	-2.446	0.015
num_mensions	3.0465	1.066	2.857	0.004
ranking_score	11.9501	1.512	7.906	0.000
num_hashtags	2.4295	0.546	4.447	0.000
0th_hour	-165.0774	648.509	-0.255	0.799
1th_hour	48.1939	650.880	0.074	0.941
2th_hour	-685.9166	652.544	-1.051	0.294
3th_hour	-831.8146	655.672	-1.269	0.205
4th_hour	-880.0948	661.754	-1.330	0.184
5th_hour	1536.0553	658.248	2.334	0.020
6th_hour	234.3266	669.661	0.350	0.727
7th_hour	-1771.5668	674.098	-2.628	0.009
8th_hour	-1310.1479	668.042	-1.961	0.050
9th_hour	-885.5791	683.618	-1.295	0.196
10th_hour	-713.8312	678.722	-1.052	0.293
11th_hour	-334.9490	675.902	-0.496	0.620
12th_hour	-418.1016	682.315	-0.613	0.540
13th_hour	-797.8293	670.214	-1.190	0.234
14th_hour	-289.9327	676.372	-0.429	0.668
15th_hour	-252.1275	667.650	-0.378	0.706
16th_hour	296.4913	666.397	0.445	0.657
17th_hour	396.5588	670.308	0.592	0.554
18th_hour	139.2750	662.873	0.210	0.834
19th_hour	276.7323	663.295	0.417	0.677
20th_hour	221.8280	661.866	0.335	0.738
21th_hour	78.6283	660.838	0.119	0.905
22th_hour	33.6887	661.351	0.051	0.959
23th_hour	-60.6663	661.336	-0.092	0.927

Most significant attributes:

num_tweets

sum_followers

max_followers

num_URLs

num_mentions

ranking_score

num_hashtags

#SuperBowl

R-squared: 0.943

Results of t-test:

	coef	std err	t	P>|t|
num_tweets	-111.7023	4.908	-22.761	0.000
num_retweets	48.3817	3.474	13.925	0.000
sum_followers	-0.0001	1.1e-05	-10.690	0.000
max_followers	-7.19e-06	9.41e-05	-0.076	0.939
num_URLs	2.4611	0.608	4.045	0.000
num_authors	15.7667	0.785	20.073	0.000
num_mensions	-13.8255	0.833	-16.606	0.000
ranking_score	22.3118	0.990	22.547	0.000
num_hashtags	1.7899	0.250	7.147	0.000
0th_hour	650.9831	887.874	0.733	0.464
1th_hour	1004.6848	881.765	1.139	0.255
2th_hour	172.0183	879.944	0.195	0.845
3th_hour	112.2364	884.318	0.127	0.899
4th_hour	663.0132	882.862	0.751	0.453
5th_hour	423.9448	890.243	0.476	0.634
6th_hour	-346.4221	887.821	-0.390	0.697
7th_hour	-1179.6425	895.216	-1.318	0.188
8th_hour	-833.1994	897.610	-0.928	0.354
9th_hour	-708.9208	897.564	-0.790	0.430
10th_hour	-865.3510	921.721	-0.939	0.348
11th_hour	-158.8165	922.615	-0.172	0.863
12th_hour	-27.3412	917.905	-0.030	0.976
13th_hour	803.5519	918.861	0.875	0.382
14th_hour	3179.5406	929.410	3.421	0.001
15th_hour	-239.1008	938.458	-0.255	0.799
16th_hour	1537.7769	934.802	1.645	0.101
17th_hour	1407.0925	932.097	1.510	0.132
18th_hour	-330.7317	926.100	-0.357	0.721
19th_hour	48.6008	933.891	0.052	0.959
20th_hour	-350.2424	908.766	-0.385	0.700
21th_hour	566.5183	906.024	0.625	0.532
22th_hour	574.2600	901.899	0.637	0.525
23th_hour	564.9300	907.577	0.622	0.534

Most significant attributes:

num_tweets

num_retweets

sum_followers

num_URLs

num_authors

num_mentions

ranking_score

num_hashtags

Analysis on each hashtag:

#GoHawks: A linear distribution could be seen in the scatter plots reflecting a good relationship between the three features.

#GoPatriots: Almost identical scatter plots with clustering towards the region of the origin

#NFL: A linear relationship can be seen for features and similar proportionality of the three features.

#Patriots: Linear relationship for the three features and extremely similar distributions.

#SB49: Similar analysis to #Patriots

#SuperBowl: Clustered region with a very small linear deviation. Large number of instances fits a better regression model hence the higher accuracy.

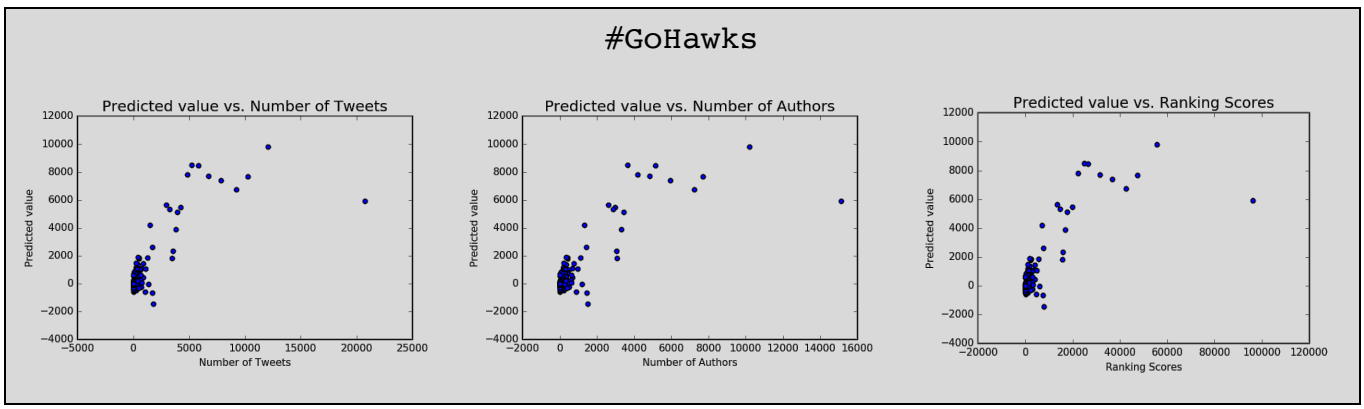
Compared to the results in previous question, we can see that the accuracy of the model is substantially improved, as indicated in the table below:

	#GoHawks	#GoPatriots	#NFL	#Patriots	#SB49	#SuperBowl
Previous R^2	0.532	0.627	0.585	0.723	0.857	0.870
Current R^2	0.724	0.894	0.765	0.823	0.902	0.943

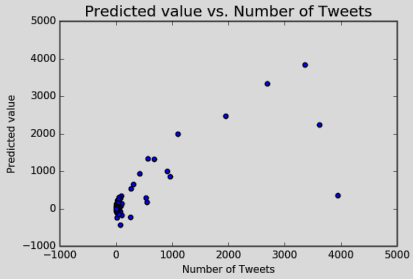
As seen from the above results we have a significant increase in the accuracy of the model for each of the hash-tag, this can be attributed to features that are not sparse and have a well-defined distribution through-out the period of the SuperBowl. Metrics employed in the tweet-data have been used to model the importance of the tweet for a given window frame thereby increasing the accuracy. In order to better visualize the contribution of the features in the model a scatter plot was created of the Top 3 features for hash-tags. Since the initial hours have less number of tweets, all of the graphs exhibit clustering of values near low of tweets/hour.

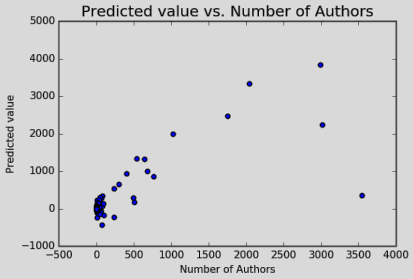
The improvement in accuracy is easy to interpret: adding more attributes provides more information for the regression model. Since the dimension of the training model increases, the target value can be predicted from more aspects. As a result, the regression model fit better and performs more accurately.

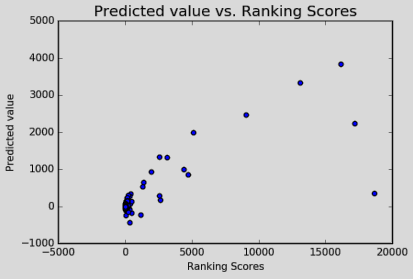
Also we know that the 3 most significant attributes are “num_tweets”, “num_authors”, “ranking_scores”. The plots of predictants versus feature values are shown below:



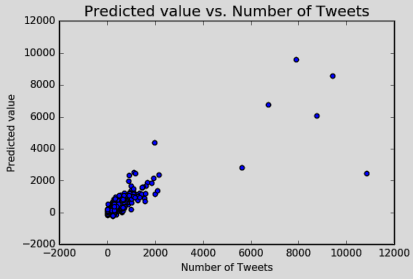
#GoPatriots

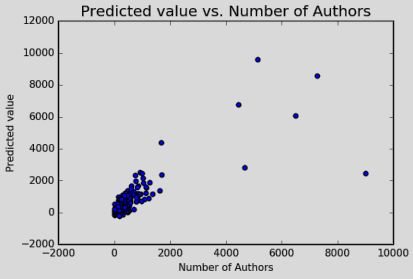


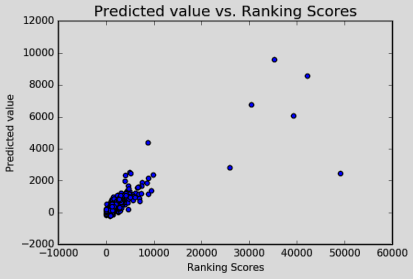




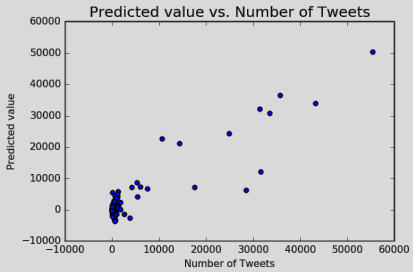
#NFL

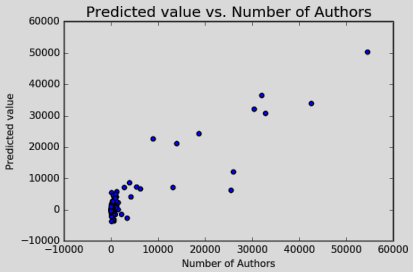


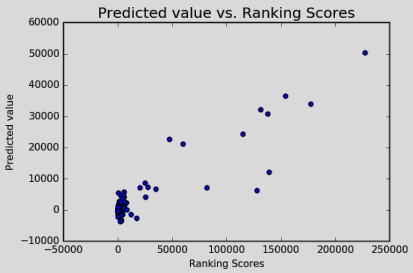




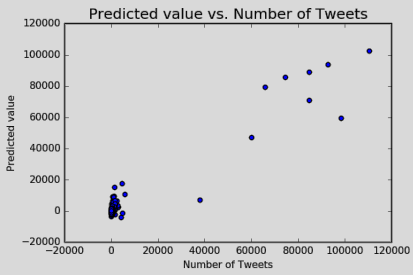
#Patriots

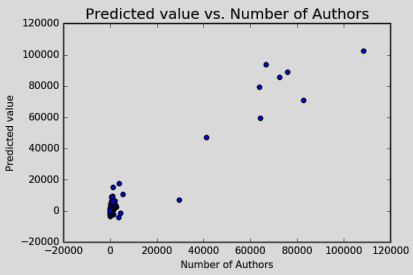


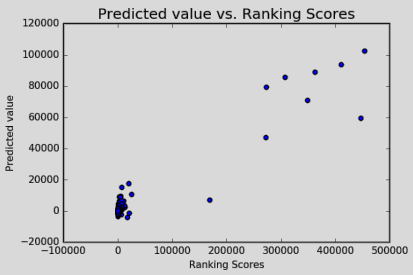




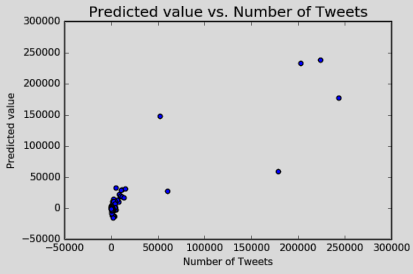
#SB49

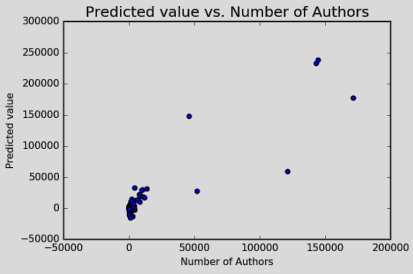


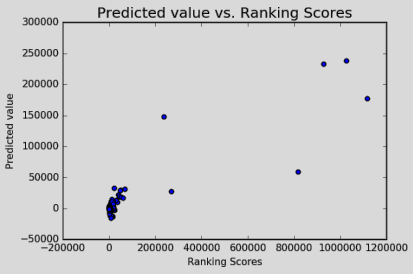




#SuperBowl







From the plots above, we can see that within each hashtag, the patterns are very similar with respect to different attributes. This indicates that the selected attributes have equal importance on the accuracy of the regression model.

Additionally, the patterns have some sort of linearity, with respect to each attribute. This means that the application of linear regression is quite reasonable and suitable.

Question 4

In this question, we are asked to perform cross validation on the dataset. We will use the same model as previously designed. After linear regression, we evaluate the accuracy of the model by calculating average prediction error:

$$Average\ Prediction\ Error = \left|N_{predicted} - N_{real}\right|$$

The results are listed below:

```
Average prediction error for #GoHawks: 401.040104176
Average prediction error for #GoPatriots: 66.4462748984
Average prediction error for #NFL: 234.942842166
Average prediction error for #Patriots: 979.049767991
Average prediction error for #SB49: 1446.34988841
Average prediction error for #SuperBowl: 3229.99047053
```

Next, we will create different regression model for different periods of time which are:

1. Before the hashtags become very active.
2. When the hashtags become very active.
3. After the hashtags pass their high-activity time.

Again, we use the same regression model to predict the results for these different periods. Average prediction error is then calculated to evaluate the accuracy of the model for different periods. The results are listed below:

```
#GoHawks
    Average prediction error before Super Bowl: 158.032826001
    Average prediction error during Super Bowl: 2114.475
    Average prediction error after Super Bowl: 27.0477279782
    Total average prediction error: 169.850094808

#GoPatriots
    Average prediction error before Super Bowl: 11.2305355536
    Average prediction error during Super Bowl: 1048.00833333
    Average prediction error after Super Bowl: 3.75521554611
    Total average prediction error: 31.220802728

#NFL
    Average prediction error before Super Bowl: 112.437542795
    Average prediction error during Super Bowl: 2670.42083333
    Average prediction error after Super Bowl: 161.107968434
    Total average prediction error: 175.923585296

#Patriots
    Average prediction error before Super Bowl: 197.917613577
    Average prediction error during Super Bowl: 17112.1125
    Average prediction error after Super Bowl: 109.03492559
    Total average prediction error: 523.251814188

#SB49
    Average prediction error before Super Bowl: 44.2858246736
    Average prediction error during Super Bowl: 26596.9916667
    Average prediction error after Super Bowl: 111.150250635
    Total average prediction error: 602.478614467

#SuperBowl
    Average prediction error before Super Bowl: 265.452488486
```

```
Average prediction error during Super Bowl: 68980.9166667
Average prediction error after Super Bowl: 298.785380566
Total average prediction error: 1677.86392046
```

It is easy to find that the error for the second period (during Super Bowl) is much larger than the other two periods (before and after Super Bowl). There are 3 reasons for this:

1. The data during the second period is collected between 8:00am and 8:00pm on 2/1/2015, only 10 hours. In comparison, the data during the other two periods contains several hundred hours. We can see that the amount of data during the second period is very small, it cannot provide enough training data for the model. Therefore, the fit of the model is bad and the error is large.

2. During the Super Bowl event, there were unusually large number of users posting tweets, the regularity is much harder to predict. The bursts of tweets often happen in a very short period of time, making it difficult to use simple linear model to fit.

3. Since the absolute number of tweets during the second period is very large, even a small value of relative error will result in large absolute error.

Question 5

In this question, we will use our trained model to predict new test datasets. Each file in the test data contains a hashtag's tweets for a 6-hour window. The number of tweets in the next hour window can be predicted using the data from previous hour window, but we can only evaluate the error using 5 data points, because we do not have the actual number of tweets in the 7th hour.

Another issue we should consider is that there are 6 different training datasets (6 different hashtags) and we can have 6 different models to predict the testing data. Since the test data comprise of all the hashtags mixed we need to apply only those models that fit appropriately. An alternative approach is to apply all the models and check the

error of predicted values of the first 6 hours to estimate the performance of the 7th hour. The predicted values for the 7th hour is provided below and the value with the least error with respect to the 6 hours data is highlighted indicating the estimated predicted value.

The results are listed below:

sample1_period1.txt

Best training dataset:
#SuperBowl

	Predicted	Actual
0	127.341725	82.0
1	85.795261	68.0
2	89.462271	94.0
3	97.960537	171.0
4	164.474315	178.0
5	136.199193	NaN

Average prediction error:
30.8479727911

sample2_period2.txt

Best training dataset:
#SuperBowl

	Predicted	Actual
0	10260.70	9361.0
1	10327.10	10374.0
2	14595.00	20066.0
3	75725.95	81958.0
4	81001.75	82923.0
5	89028.05	NaN

Average prediction error:
2914.18

sample3_period3.txt

Best training dataset:
#NFL

	Predicted	Actual
0	521.167614	550.0
1	534.817614	610.0
2	646.031250	888.0
3	757.833864	616.0
4	556.507614	523.0
5	507.471364	NaN

Average prediction error:
104.265

sample4_period1.txt

Best training dataset:
#GoHawks

	Predicted	Actual
0	260.282048	257.0
1	234.380020	236.0
2	267.229742	266.0
3	231.759649	267.0
4	263.213098	201.0
5	184.061465	NaN

Average prediction error:
20.7170438973

sample5_period1.txt

Best training dataset:
#GoPatriots

	Predicted	Actual
0	436.40000	508.0
1	425.35000	353.0
2	351.55000	362.0
3	387.95000	281.0
4	193.10000	213.0
5	96.29875	NaN

Average prediction error:
56.25

sample6_period2.txt

Best training dataset:
#Patriots

	Predicted	Actual
0	13343.50	12931.0
1	40409.45	60619.0
2	33617.00	52699.0
3	33669.25	41019.0
4	31447.90	37307.0
5	31331.95	NaN

Average prediction error:
10582.58

```
sample7_period3.txt

Best training dataset:
#Patriots

    Predicted  Actual
0  103.123869   102.0
1   84.601032    66.0
2   56.066550    60.0
3   53.500580    55.0
4   50.190729   120.0
5  100.397817     NaN

Average prediction error:
18.9934083213
```

```
sample8_period1.txt

Best training dataset:
#GoHawks

   Predicted  Actual
0  65.353358    72.0
1  62.160708    56.0
2  43.642159    41.0
3  28.242817    11.0
4  37.279317     NaN

Average prediction error:
8.17308153614
```

```
sample9_period2.txt

Best training dataset:
#NFL

   Predicted  Actual
0    1661.50  1734.0
1    1584.10  1619.0
2    1554.40  1582.0
3    1774.90  1857.0
4    2593.85  2790.0
5    2979.80     NaN

Average prediction error:
82.65
```

```
sample10_period3.txt

Best training dataset:
#GoHawks

   Predicted  Actual
0     60.100    54.0
1     61.300    68.0
2     56.600    62.0
3     61.125    58.0
4     61.125    61.0
5     61.650     NaN

Average prediction error:
4.29
```

From the results, we can see that the training accuracy is relatively high. Only those testing data with extremely large values will yield larger errors. However, if we consider relative errors, we still can draw the conclusion that the models achieved high accuracy.

Question 6

In the previous 5 questions, we mainly focused on training a model to do prediction on the popularity of a topic. But if we take a look at the structure of a single tweet the JSON file provided, we could see it contains much more information than the features we explored before. From the website of twitter API for developers, we found that there is some useful information we could use to do the user location prediction. First of all, user the <user> tag, there is a child element called location, which contains the user-defined location for this account's profile, but the information is not necessarily a parse-

able string containing the true location information. And there is a child element under the tweet object names <highlight>, which is simply the content of the current tweet, excluding the original content if it's a retweet, but including the URLs which links to some special webpages, as shown below:

```
"location":"San Francisco, CA"
```

So our basic idea for this problem is retrieve the related tweets from the pool firstly, using some related keywords such as the city names in Washington state and Massachusetts state, which will be explained afterwards. And based on those tweets content and location information we get, we train a classification model based on the textual content of the tweets, optimize the dimension reduction procedure and try different classification methods on the dataset. Based on the results we got from project 2, Support Vector Machine and Logistic Regression have much better performance compared with the Naïve Bayes and other classifier. So we mainly focus on SVM and LR classifier for this problem.

6.1 Prepare the dataset

Before training the classification model, we need to retrieve the related dataset from the tweet pool we got in the JSON files. As illustrated before, the <location> element in the object is not necessarily semantic, so we need to come up with a keyword-set to do the string matching between the <location> text with our keyword-set. We use the main city names as well as two state names to do this job. Here are our keyword-set:

WA = ["Washington", "WASHINGTON", "WA", "Aberdeen", "ABERDEEN", "Anacortes", "ANACORTES", "Auburn", "AUBURN"…]

MA = ["Massachusetts", "MASSACHUSETTS", "MA", "Abington", "ABINGTON", "Adams", "ADAMS", "Amesbury", "AMESBURY", "Amherst", "AMHERST"…]

And one thing to mention is that, the keyword 'Washington' might cause some mistakes due to the confusion between 'Washington' and 'Washington DC', so we specifically

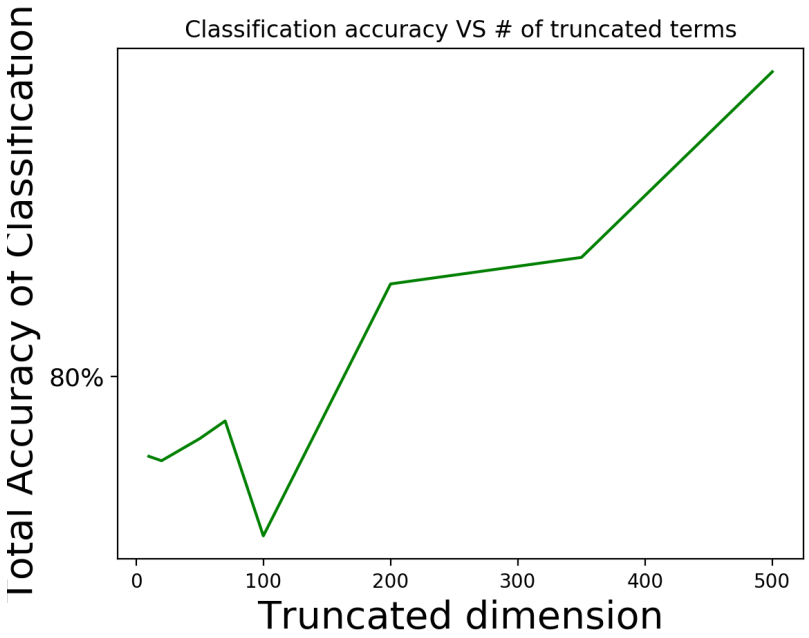
excluded the location entries that contains the ‘DC’ or ‘D.C.’ as well. Finally, we got totally 54916 related tweets, within which 22191 are labeled 1 for Washington and 32725 for Massachusetts.

Dataset distribution

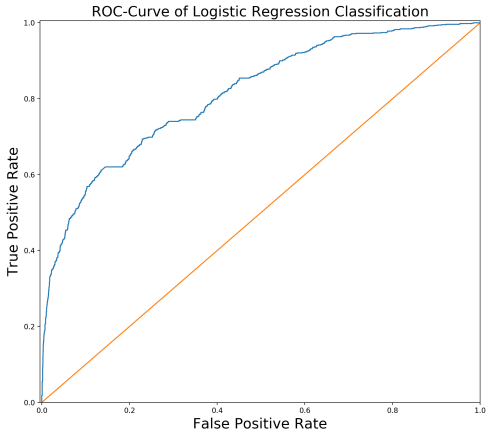
	Number of tweets	**Number of unique <location>**
Washington(WA)	22191	3862
Massachusetts(MA)	32725	6303
Total	54916	10165

6.2 Logistic Regression

After the data retrieval, we converted the content pool into a TF-IDF matrix, and the matrix contains the TD-IDF scores over 3416 terms and 54916 documents. As we have proved in project 2, the highly sparse TF-IDF might not be a good choice to train the classification model. So we reduced the dataset dimension using SVD and also explored how the dimension would influence our classification performance. Below are the results of Logistic Regression. And to fasten the optimization, we use last 4000 of the 54916 entries as the test sample:



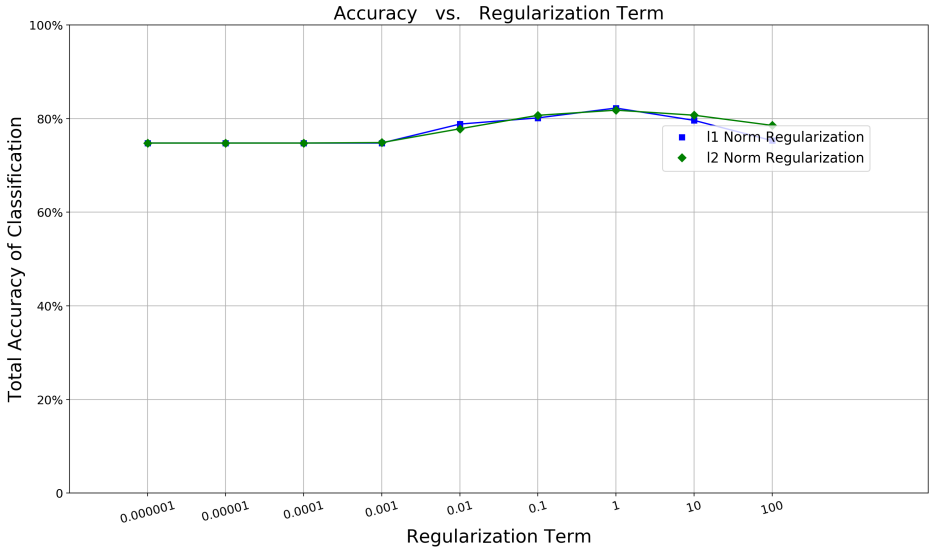
We can see that, as the number of truncated terms increases, the classification accuracy increases as well, which is different from the conclusion we got in project 2. And we also explored the classification performance without the dimension reduction:



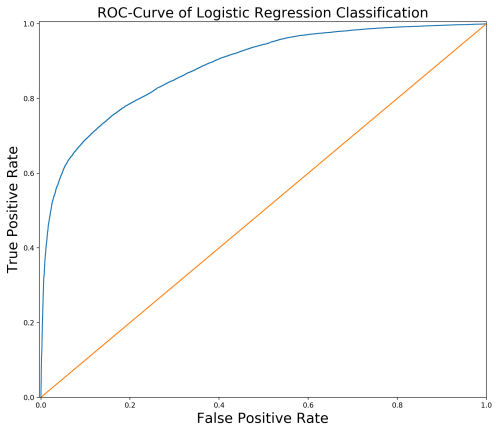

```
             Classification report:
=========================================================
Classifier: LR with Regularization & Penalty
=========================================================
               precision    recall  f1-score   support

   Washington       0.83      0.96      0.89      2991
Massachusetts       0.76      0.40      0.53      1009

  avg / total       0.81      0.82      0.80      4000
=========================================================

Confusion Matrix:
==============
[[2866  125]
 [ 603  406]]
==============

Total accuracy:
0.818
```

The dimension reduction does not improve the classification performance as we expected in Logistic Regression. One possible reason is that, compared with the long textual documents we classified in project 2, the tweet content is much shorter and containing much less information. So the dimension reduction might cause information or feature loss in this case.

And we also tried the Logistic Regression with different penalty term and regularization, here is the results:



we could see that, the Logistic Regression has best performance as for classification accuracy with l1 penalty function and regularization term of 1, while the accuracy variation over both the regularization term and penalty function is not very considerable. Based on the optimization result, we perform the LR-classification on the whole dataset using 5-fold cross validation, and here is the results:



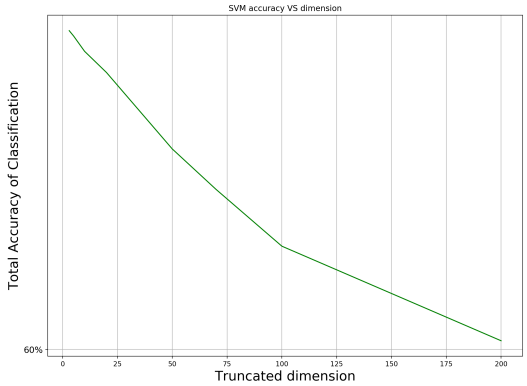

```
             Classification report:
=============================================================
Classifier: LR(l1 penalty) - Cross Validation
=============================================================
               precision    recall  f1-score   support

   Washington       0.74      0.86      0.79     32725
Massachusetts       0.73      0.55      0.63     22191

  avg / total       0.73      0.73      0.73     54916
=============================================================

Confusion Matrix:
==============
[[28162  4563]
 [ 9991 12200]]
==============

Total accuracy:
0.734977055867
```

Overall, the Logistic Regression classifier has accuracy of 0.734, and the low recall of classifying tweets from Massachusetts still need to be optimized.

6.3 Support Vector Machine

In this part, we explored the classification performance of SVM. And from project 2 we know that the calculation complexity of SVM of much higher than the Logistic Regression, so in order to fasten the optimization procedure, we use a 5000-entry-containing random subset of the original dataset to optimize the truncated dimension and the Kernel coefficient gamma.

Firstly, we use truncated SVD to reduce the dimension of the sample dataset, here is the classification accuracy with different dimension:

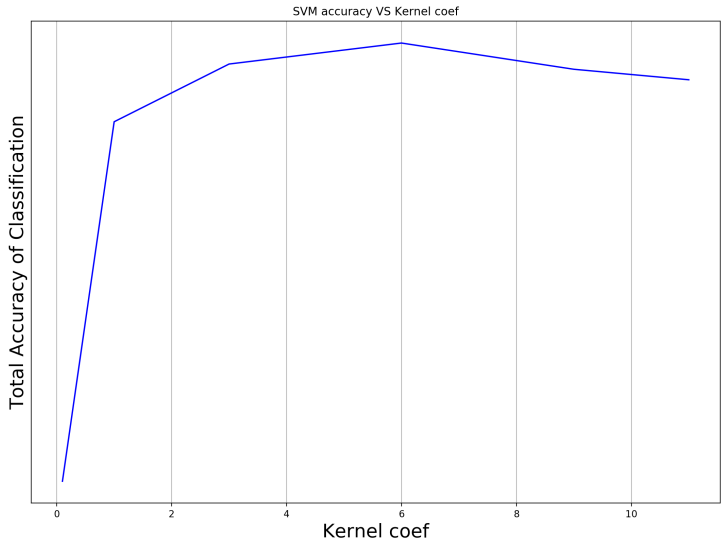



```
             Classification report:
=========================================================
Classifier: SVM
=========================================================
               precision    recall  f1-score   support

   Washington       0.73      0.95      0.83      5925
Massachusetts       0.88      0.50      0.63      4075

  avg / total       0.79      0.77      0.75     10000
=========================================================

Confusion Matrix:
==============
[[5644  281]
 [2050 2025]]
==============

Total accuracy:
0.7669
```

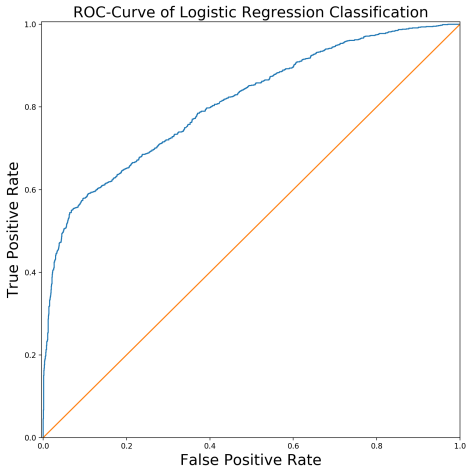
we could see that, SVM performance is very different from the Logistic Regression, where the classification accuracy is much higher when the dataset dimension is small. When dataset is truncated to 3-dimension, which means classification only depends on

three terms, the accuracy is 0.7516 as shown above. And same as the Logistic Regression, the recall for Massachusetts located tweet classify is very low.

We also tried to optimize the SVM classifier by tuning the Kernel coefficient, results:



the variance of classification accuracy over the different Kernel coefficient is considerably small, while we could pick 6 as the optimized value. And giving the optimization results: truncated to 3-dimension with 6 as gamma, here is our classification results for the overall dataset:



```
             Classification report:
=========================================================
Classifier: SVM(Cross Validation)
=========================================================
               precision    recall  f1-score   support

   Washington       0.74      0.95      0.83     32725
Massachusetts       0.87      0.51      0.65     22191

  avg / total       0.79      0.77      0.76     54916
=========================================================

Confusion Matrix:
==============
[[31083  1642]
 [10837 11354]]
==============

Total accuracy:
0.772762036565
```

We could see that, the SVM classification has better accuracy comparing with the Logistic Regression. And it performs very well in classifying tweets posted by users from Washington state.

Question 7

Problem Proposal:

It is important for Twitter Company to deploy the proper amount of computing facility and storage resource at a given time. For example, at early times when Twitter has just been created and began service on the internet, only a small amount of users were using this service. Thus, Twitter did not need to setup too many servers to support its service. It would be a wasting of money if too many computing resources are deployed. However, nowadays twitter has hundreds of millions of users posting a huge amount of tweets every day; there is no doubt that Twitter should use a lot more servers and storages to support its service. Twitter would become vulnerable if it does not increase the computing facilities.

It will be good if we can predict future usage of Twitter, given some attributes. Because in this way we can make plans on how to increase the computing and storage resources. In order to make this prediction, we start from predicting future usage of Twitter on

single users. Specifically, we want to know the number of tweets a user may post since the time that the account was registered.

Problem Solution:

The task is to predict the total number of tweets posted by an account, given other attributes of the account. The attributes used in the model are listed below. The reasons why these attributes are selected are also explained briefly.

time_since_register: The total time passed since the Twitter account was created. It is reasonable to guess that the number of posted tweets has a positive relationship with time_since_register, as long as the user has been using this account.

default_profile: Indicates whether the user has altered the theme or background of his user profile. If the theme or background of user profile is altered, it is possible that this user is more active than those who do not alter their profile. Such a user may post more tweets.

favourites_count: The number of tweets this user has favorited in the account's lifetime. If a user has high favourites_count, it is possible that this user is more active than those with low favourites_count. Such a user may post more tweets.

followers_count: The number of followers this account currently has. If a user has many followers, it is likely that the user would post more tweets than whose who have fewer followers.

friends_count: The number of users this account is following (AKA their "followings"). If a user is following a lot of other users, it is likely that this user is more active than those following only a few other users. Thus there should be some relationship between friends_count and the number of tweets

posted by this user.

geo_enabled: Indicates whether the user has enabled the possibility of geotagging their Tweets. If a user is unwilling to share his location when posting a tweet, it means that the user cares about his privacy. So it is possible that this user posts fewer tweets than those who are willing to share location.

is_translator: Indicates whether the user is a participant in Twitter's translator community. This attribute may uncover the language preferences of the user. It is reasonable to believe that users speaking different languages have different activeness.

listed_count: The number of public lists that this user is a member of. This attribute could indicate the activeness of the user. If the user is a member of many public lists, he is possible to post more tweets than those with fewer public lists.

protected: Indicates whether this user has chosen to protect their Tweets. If a user choose to protect his tweets, it means that the user cares about his privacy and do not want public to see part of his tweets. So there should be some relationship between this attribute and the number of posted tweets.

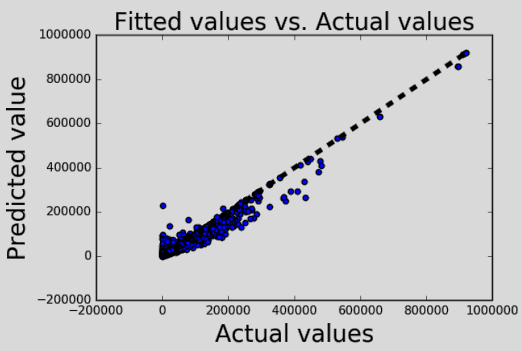
verified: Indicates whether account has been verified by the user. A verified account is more stable. A user is more likely to use a verified account for a long time. So it is possible that the user will post more tweets.

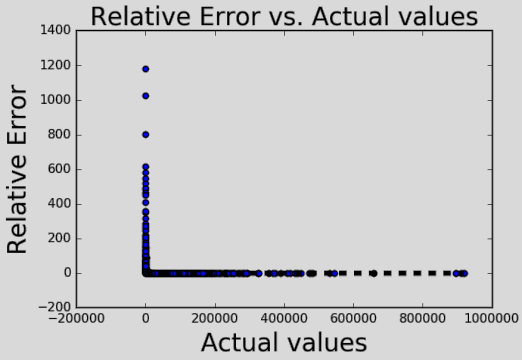
Problem Results:

After extracting all the data needed and use random forest regression to train the model, the results using different datasets are shown below:

#GoHawks

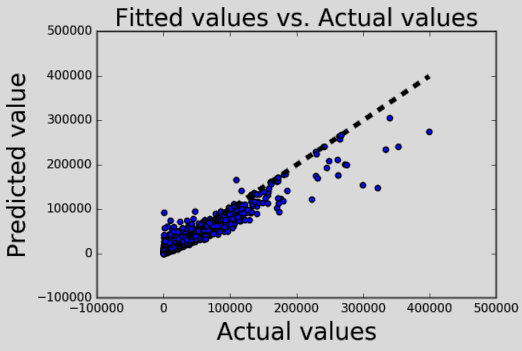
Average prediction error: 718.038126746

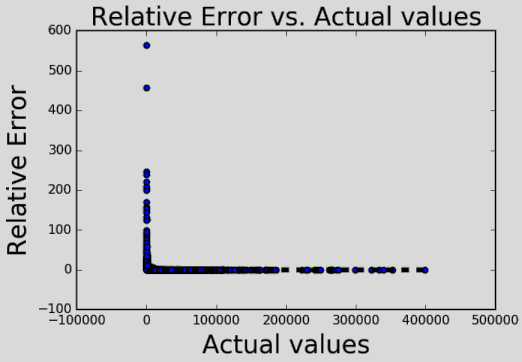




#GoPatriots

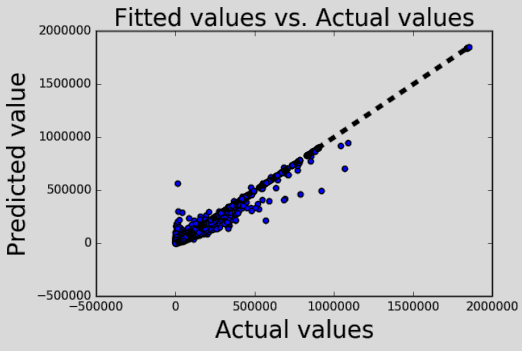
Average prediction error: 1593.68426749

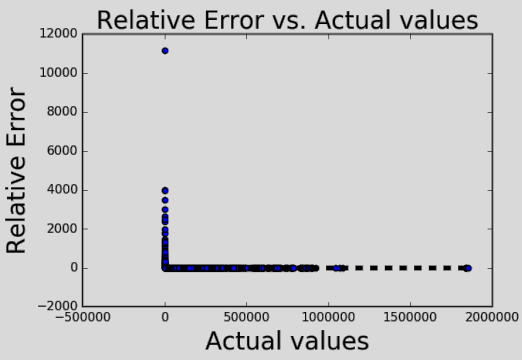




#NFL

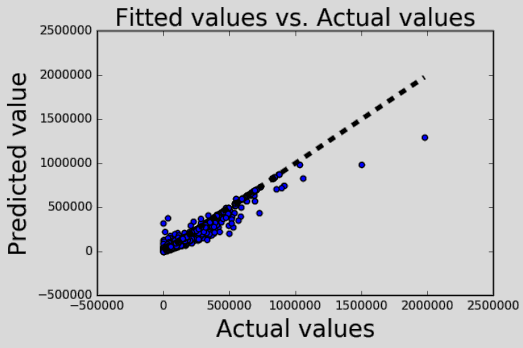
Average prediction error: 903.522689991

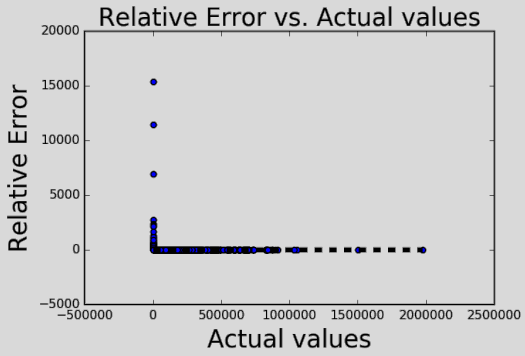




#Patriots

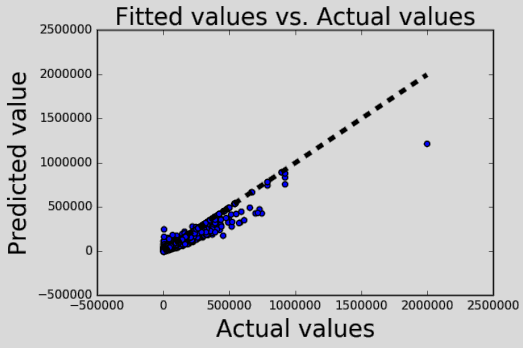
Average prediction error: 1397.99271089

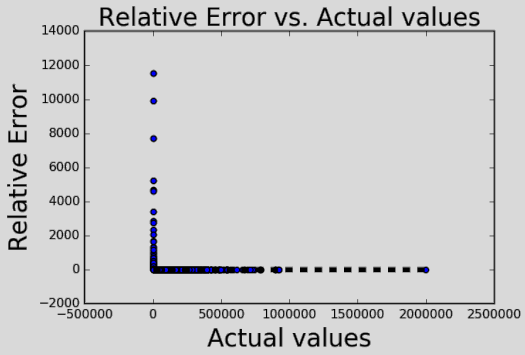




#SB49

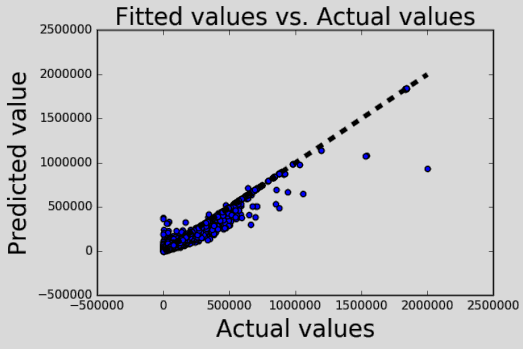
Average prediction error: 1315.72159172

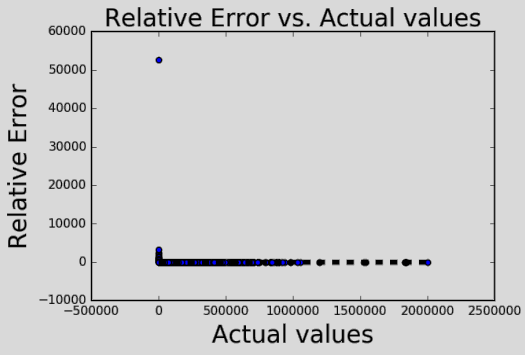




#SuperBowl

Average prediction error: 1117.18616405





We can see from the results that the predicted values and actual values show very strong linearity. Larger dataset has higher accuracy. Also, most predicted values have very low relative error. We can conclude that the model works well with high accuracy.

References:

1. On the Real-time Prediction Problems of Bursting Hashtags in Twitter.

Shoubin Kong, Qiaozhu Mei, Ling Feng, Zhe Zhao, Fei Ye

2. Wikipedia - Super Bowl XLIX

https://en.wikipedia.org/wiki/Super_Bowl_XLIX

3. Wikipedia – Support Vector Machine

https://en.wikipedia.org/wiki/Support_vector_machine

4. Wikipedia – Receiver operating characteristic

https://en.wikipedia.org/wiki/Receiver_operating_characteristic