ECE219 — Large Scale Data Mining

Name: Lu Ren	UID: 704706181
Name: Tianxue Chen	UID: 004943548
Name: Yuting Tang	UID: 304881011
Name: Xiao Peng	UID: 005033608
Due Date: March 14 2018, 11:59pm	Assignment: HW 5

Part 1: Popularity Prediction

Problem 1.1

Q1:
Description: Calculate the following statistics for each hashtag:
- Average number of tweets per hour
- Average number of followers of users posting the tweets
- Average number of retweets

Solution: Since the tweets are all in sorted order of 'firstpost_date', but in this question we should use the 'citation_date' to evaluate their first posting time. So first we resorted each hashtag by the 'citation_date' of each tweet within it. Then the 'citation_date' of the first and last tweets are considered to calculate total hours, while summarizing the number of retweets and followers of all the tweets to calculate the statistics.

Result:

Table 1. Statistics for each hashtag

Hashtag	**Total Tweets**	**Avg. # Tweets per hour**	**Avg. # of Follower of user**	**Avg. # of Retweets**
# gohawks	188136	325.3716	2203.9318	2.0146
# gopatriots	26232	45.6945	1401.8955	1.4001
# nfl	259024	441.3234	4653.2523	1.5385
# patriots	489713	834.5555	3309.9788	1.7828
# sb49	826951	1419.8879	10267.3168	2.5111
# superbowl	1348767	2302.5004	8858.9747	2.3883

Discussion:
1. Most Tweeted Hashtags per hour: # sb49 and # superbowl
2. Most Followers of Users for Hashtag: # sb49, # superbowl and # nfl
3. All of these six hashtags are composed of tweets that are not re-tweeted or re-tweeted by very few times, thus the average number of retweets are all approximately equal to 2

Q2:
Description: Plot "number of tweets in hour" over time for #superbowl and #NFL (a histogram with 1-hour bins). The tweets are stored in separate files for different hashtags and files are named as tweet_[#hashtag].txt.

Plots:

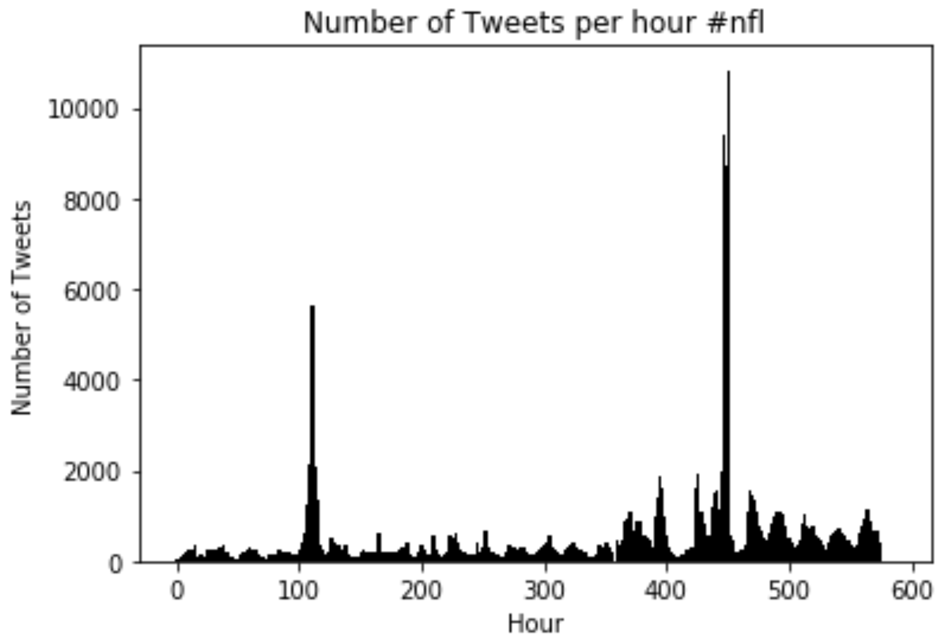


Figure 1. Number of Tweets in hour of #nfl

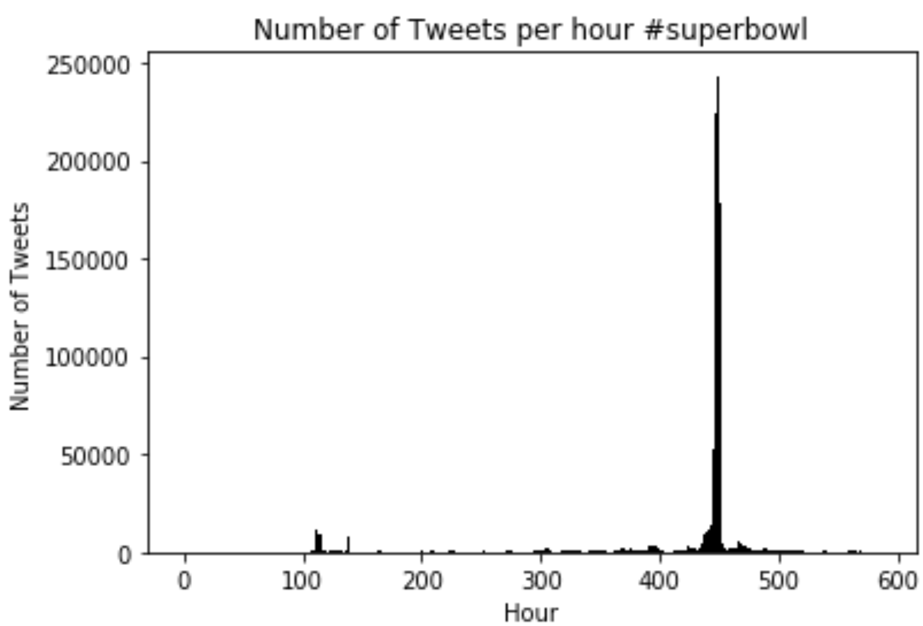


Figure 2. Number of Tweets in hour of #superbowl

Problem 1.2

Description: For each hashtag, fit a linear regression model using the following 5 features to predict number of tweets in the next hour, with features extracted from tweet data in the previous hour.
The features you should use are:

- Number of tweets (hashtag of interest)
- Total number of retweets (hashtag of interest)
- Sum of the number of followers of the users posting the hashtag
- Maximum number of followers of the users posting the hashtag
- Time of the day (which could take 24 values that represent hours of the day with respect to a given time zone)

For each hashtag, you should train a separate model. For each of your models, report your model's RMSE and R-squared measure. Also, analyze the significance of each feature using the t-test and P-value.

Solutions: A total of 5 features were used to create a linear regression model. We use a one-hour time window to extract features from the data and the number of samples we get is equal to the number of total hours for each hashtag. And then split the features into independent variables and dependent variable. Since the dependent variable (i.e. number of tweets per hour) of the first hour window cannot be predicted by the features from previous hour and the independent variables from the last hour window has no corresponding "next hour number of tweets" to predict, we removed the first value of dependent variable (y) and the last sample of independent variable (x's).

Result:

Table 2. Model RMSE and R-squared for each hashtag

Hashtag	**RMSE**	**R-squared**
#gohawks	949.9897	0.4618
#gopatriots	195.3543	0.6070
#nfl	584.7333	0.6288
#patriots	2371.9722	0.6009
#sb49	4009.9151	0.7939
#superbowl	6532.7680	0.7412

Table 3. P-value and t-value for each attribute

Hashtag	**# of retweets**		**Σ of # of followers**		**Max. followers**		**Time of the day**	
	P-value	**t-test**	**P-value**	**t-test**	**P-value**	**t-test**	**P-value**	**t-test**
#gohawks	8.02297712e-01	-0.25049046	4.0490e-22	10.0820695	3.4845e-06	-4.68589648	3.1520e-04	3.62452617
#gopatriots	0.03476319	2.11617046	0.0520287	1.94697345	0.01098856	-2.55141694	0.1948119	1.29800121
#nfl	1.93165905e-01	1.30277767	6.36378098e-13	7.35855625	2.91731799e-07	-5.18948027	1.62046863e-07	5.18948027
#patriots	6.06990063e-44	15.14886184	1.54019553e-01	-1.42733535	2.11543764e-02	2.31148636	2.95375483e-01	-1.04734448

#sb49	9.27513771e-104	26.86006675	1.41021889e-048	-16.12526047	1.50985409e-011	6.88507301	5.11812706e-002	-1.95401539
#superbowl	2.95321701e-49	16.25663128	9.43909136e-09	-5.82544577	1.04023906e-04	3.90799848	6.07052626e-01	-0.51456512

Discussion:

According to the definition of p-values and t-test, a predictor that has a low p-value is more likely to be a meaningful addition or the model and the larger the absolute value of t, the less likely that the actual value of the parameter could be zero. It ca be seen that the most contributing feature towards the linear regression model in all hashtag is the number of retweets and sum of the number of followers.

The R-squared value is used to evaluate the performance of linear regression model. It can be seen that for most hashtag, the R-squared value is approximately equal to 0.6, which represents the accuracy is not very high. It can be attribute to the window size of one-hour, since in the initial hours the average number of tweets are pretty low and building a model for these sparse features is more difficult.

Problem 1.3

Description:
Design a regression model using any features form the papers you find or other new features you may find useful for this problem. Fit your model on the data of each hashtag and report RMSE and significance of variables.
For each of the top 3 features in your measurements, draw a scatter plot of predicting (number of tweets for next hour) versus value of that feature, using all the samples you have extracted, and analyze it.

Solution:
A linear regression model with 4 more features (ranking score, favorite count, impression count and number of unique users tweeting) was create in this part. A detailed description of features considered in this model as as follows:

- Number of tweets (hashtag of interest)
- Total number of retweets —— (metrics/citations/total)
- Sum of the number of followers of the users posting the hashtag —— (author/followers)
- Maximum number of followers of the users posting the hashtag
- Time of the day (which could take 24 values that represent hours of the day with respect to a given time zone)
- Ranking Score —— (metrics/ranking_score) —— Measures the number of times a user is served a Promoted Tweet either in time-line or on search
- Impression Count —— (tweet/favorite_count) - Number of tweets that favorites by users
- Number of Users per hour - (tweet/user/id) - Counted number of users posting per hour

Results:
We deal with the features by the same method as mentioned in problem 1.2. The model was tested and the results obtained are as follows:

Table 4. Model RMSE and R-squared for each hashtag

Hashtag	RMSE	R-squared
#gohawks	899.5147400931043	0.541380307327
#gopatriots	165.4322594378894	0.706207583841
#nfl	491.67179108094444	0.688286751017
#patriots	2391.801502456121	0.701234299336
#sb49	4021.6568490217733	0.838509984346
#superbowl	6366.544402530947	0.872934026504

P-value and t-value for each feature(order of features are the same as listed in the "Solution" part)

#gohawks

R-squares measure : 0.541218691949
RMSE : 902.5959282427425
p values : [4.86083194e-17 3.43155185e-07 3.97424364e-02 4.80913495e-01
1.85485426e-28 9.31022302e-14 1.76679515e-05 2.62824681e-01]
t-test : [-8.65983082 -5.15903336 2.06114265 0.70529707 11.69223101
7.63893933 4.3293392 1.12084774]

#gopatriots

R-squares measure : 0.706101992477
RMSE : 167.14501658830417
p values : [6.39986531e-03 1.11827772e-06 1.57440332e-03 2.22888602e-01
5.63225355e-01 7.18323810e-38 3.17094895e-07 5.92885942e-01]
t-test : [2.73675096 4.92325792 -3.17601762 1.22022273 -0.57839823
-13.87093663 -5.17493436 0.53496242]

#nfl

R-squares measure : 0.688286751017
RMSE : 491.67179108094444
p values : [3.93380946e-02 6.90655932e-01 6.55252718e-01 7.25274160e-01
1.17335625e-20 1.24058507e-44 7.34035733e-01 1.23126298e-01]
t-test : [-2.06531889 0.39816453 0.4467087 -0.35158965 9.68481152
-15.30357616 -0.33992602 1.54403854]

#patriots

R-squares measure : 0.701234299336
RMSE : 2391.801502456121
p values : [1.49431138e-04 5.90188383e-01 2.98671302e-02 9.50630300e-01
6.97297879e-28 6.02529853e-01 5.58271696e-01 4.23124001e-01]
t-test : [-3.81738775 0.5388637 2.17723123 -0.06194217 11.5435339
0.52105237 -0.5857507 0.80158234]

#sb49

R-squares measure : 0.838488352702
RMSE : 4035.4478106167744
p values : [7.14147065e-02 1.38888156e-01 7.21476180e-08 4.01413076e-01
1.52691536e-24 1.83368933e-02 1.54824340e-01 8.31772869e-01]
t-test : [-1.80617478 1.48200339 5.45692714 -0.83972344 10.71418118
-2.36550563 -1.42457221 0.21252507]

#superbowl

R-squares measure : 0.872933772074
RMSE : 6371.899503865502
p values : [9.17059056e-02 3.20920681e-01 5.10414718e-25 7.37432447e-01
5.80154806e-53 1.64011069e-06 6.35499891e-02 1.75270652e-01]
t-test : [-1.68928055 0.99342216 10.83269811 -0.33541702 17.02425596

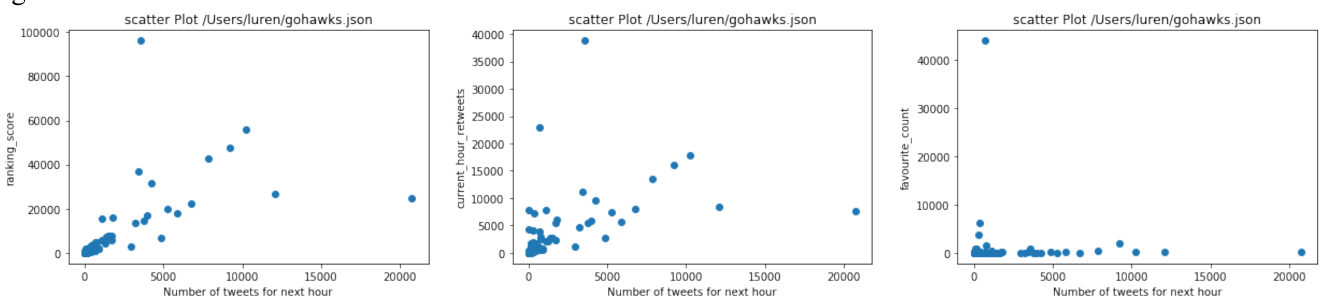
-4.8436094 -1.85890166 -1.35712962]

Discussion:

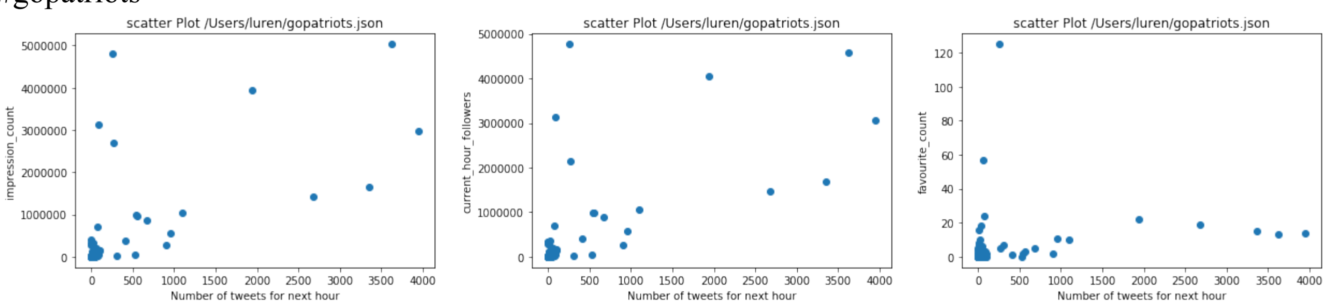
From the above result we can see that after adding new features, the R-squared value of the linear regression model increases significantly, which means the performance of prediction improves quite a lot compared to only involve 5 features in problem 1.2. Thus, the new features are meaningful additions.

Plots:

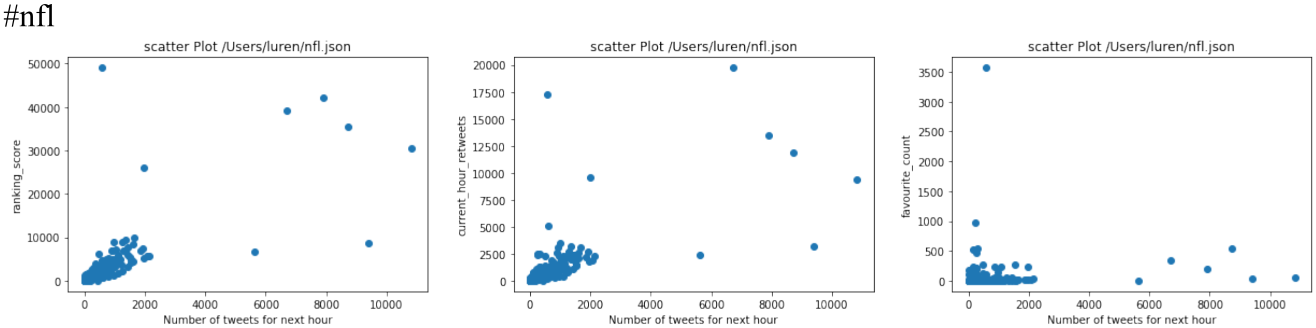
#gohawks



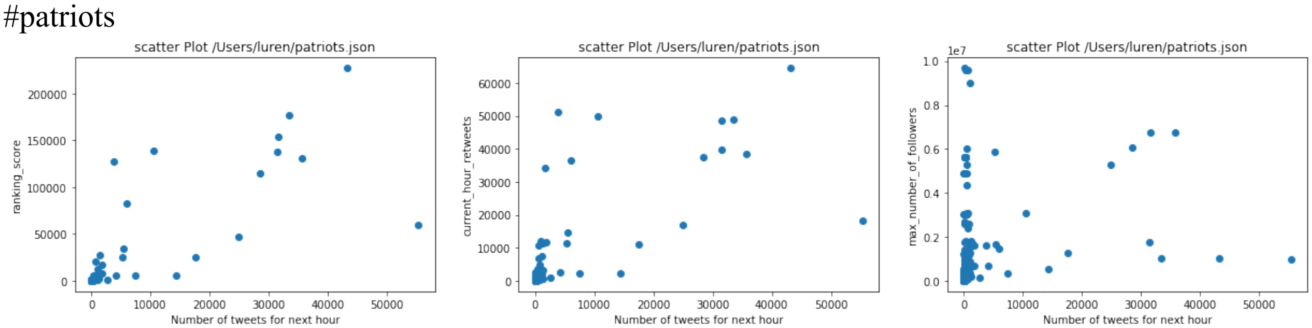
#gopatriots



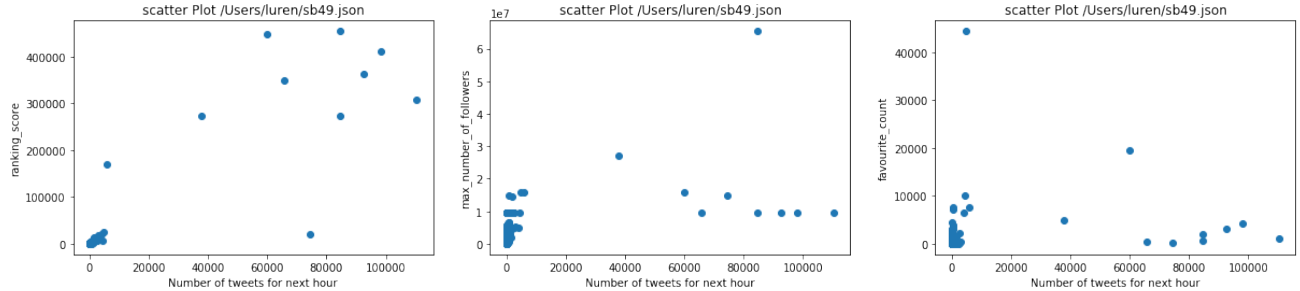
#nfl



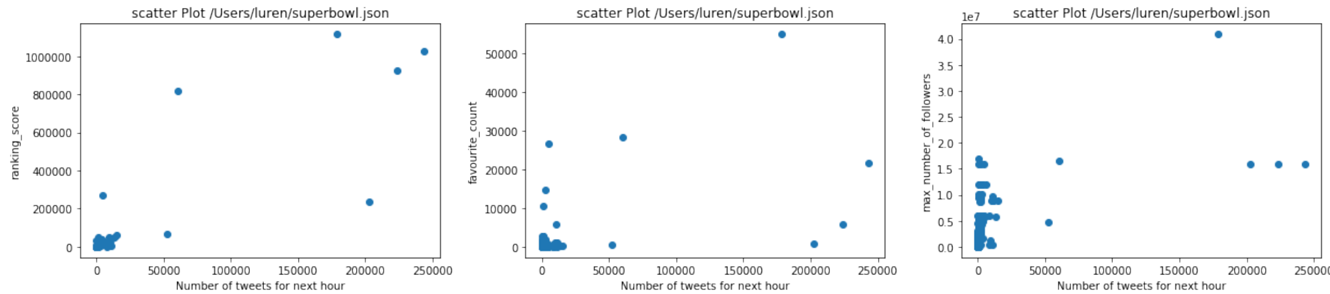
#patriots



#sb49



#superbowl



Discussion:
As the number of tweets can be very sparse when it goes to some very large value, it will influence the linear relationship between top features and target values. We can see that the features we use have different influence on different hashtag. For gohawks, gopatiots, nfl and patriots, we can see a clear linear relationship between the number of tweets for next hour and the value of top features. However, for sb49 and superbowl, the top 3 features don't perform as well. It can be explained by that as the number of tweets can be very sparse when it goes to some very large value, it will influence the linear relationship between top features and target values.

Problem 1.4

Q1:
Description: For each hashtag, report the average cross-validation errors for the 3 different models.
Note that you should do the 90-10% splitting for each model within its specific time window. I.e. Only use data within one of the 3 periods above for training and testing each time, so for each period you will run 10 tests.
Also, aggregate the data of all hashtags, and train 3 models (for the intervals mentioned above) to predict the number of tweets in the next hour on the aggregated data.

Solution:
In this question, linear regression model, random forest model and SVM model is used to predict the number of tweets for three time-periods in each hashtag and the performance is compared according to the results.

Result:
#gohawks

Time period	Linear regression		Random Forest	SVM
	R-squared	RMSE	RMSE	RMSE
before	0.38911834512	770.585251684	453.639902473	499.098731346
between	0.87883277681	29751.2478133	3180.92523884	6550.96616575
after	0.93143503659	14380.3584417	50.4877008065	41.5668817952

 # gopatriots

Time period	Linear regression	Random Forest	SVM

	R-squared	RMSE	RMSE	RMSE
before	0.77036394444	38.9203899253	27.8582114043	30.5257518178
between	0.96943198371	4168.73899362	612.245189186	1977.70914843
after	0.93334814981	9.79870201874	5.26441177913	5.35472033384

#nfl

Time period	Linear regression		Random Forest	SVM
	R-squared	RMSE	RMSE	RMSE
before	0.52512878322	208.98782393	211.054778305	309.488027946
between	0.92613983980	16655.2641211	1697.43695713	5978.46578313
after	0.84148033785	143.523487658	156.862274028	574.149918017

#patriots

Time period	Linear regression		Random Forest	SVM
	R-squared	RMSE	RMSE	RMSE
before	0.46621397129	841.768541744	808.270842872	566.312577506
between	0.91886891667	30647.9898439	17057.9604853	22637.8231049
after	0.90516145512	163.417665798	128.707629591	165.774155956

#sb49

Time period	Linear regression		Random Forest	SVM
	R-squared	RMSE	RMSE	RMSE
before	0.84143520012	71.4383097527	87.1331238808	148.849181531
between	0.82204000995	143239.145853	33484.1974593	42728.2930238
after	0.89359588811	202.049088633	181.05800651	373.507031171

#superbowl

Time period	Linear regression		Random Forest	SVM
	R-squared	RMSE	RMSE	RMSE
before	0.35381567889	667.777378198	660.047602016	827.348516912
between	0.92145998295	439992.541589	54437.0773237	183020.048
after	0.86937085943	468.523005289	299.175797961	835.238478927

Discussion:
It can be seen clearly that due to the Between time period having only 12 one-hour window, the number of instances in this time-period is to create a model is very low. Hence the model for this time period has a very high RMSE. Since the before time period and after time period have the greater number of instances, the models perform much better and gives the lowest RMSE.

According to the R-squared value of linear regression model, we can see that after split the data into different time period, the performance of prediction has improved quite a lot compared to use all the data as a whole to train a linear regression model.

Among the three models we use, random Forest regression model has the lowest RMSE value, thus Random Forest regression model performs the best.

Q2:

Description: Perform the same evaluations on your combined model and compare with models you trained for individual hashtags.

Solution:
According to the result of previous problem, we use random forest model to predict the number of tweets for the aggregated data.

Result:

Time period	RMSE
Before	1664.56053354
Between	92598.0738588
After	486.481342756

Discussion:
Average RMSE for 6 individual hashtags

Time period	RMSE
Before	374.6673435
Between	18411.64043
After	136.9259701

From the comparison between combined model and models for individual hashtags, we can see that models based on dataset of individual hashtags perform much better. It is consistent with our common sense since different hashtags can have different characteristics and features in number of tweets, such that build different models can better track their trends.

Problem 1.5

Description: Report the model you use. For each test file, provide your predictions on the number of tweets in the next hour.

Results: During this part, we use the best model found in problem 1.4: Random Forest.
The predict result for the number of tweets in the next hour is:

<table>
<tr><td></td><td>File_name</td><td>Predict tweets number</td><td>Ground truth</td><td>RMSE</td></tr>
<tr><td rowspan="4">Before 8 am.</td><td>sample1_period1.txt</td><td>181</td><td>178</td><td rowspan="3">2.5</td></tr>
<tr><td>sample4_period1.txt</td><td>295</td><td>203</td></tr>
<tr><td>sample5_period1.txt</td><td>271</td><td>211</td></tr>
<tr><td>sample8_period1.txt</td><td>50</td><td>12</td><td>37.6</td></tr>
<tr><td rowspan="3">Between 8 am. and 8 pm.</td><td>sample2_period2.txt</td><td>293990</td><td>82892</td><td rowspan="3">211097.5</td></tr>
<tr><td>sample6_period2.txt</td><td>310178</td><td>37279</td></tr>
<tr><td>sample9_period2.txt</td><td>249775</td><td>2791</td></tr>
<tr><td rowspan="3">After 8 pm.</td><td>sample3_period3.txt</td><td>984</td><td>524</td><td rowspan="3">460.4</td></tr>
<tr><td>sample7_period3.txt</td><td>32</td><td>121</td></tr>
<tr><td>sample10_period3.txt</td><td>49</td><td>62</td></tr>
</table>

Discussion:
Overall, the model is train on the aggregate of the training data for all hashtags. The training data is sorted by 'firstpost_date'. The reason why using firstpost_date instead of using citation_date is that the citation_date is too sparse to be predict.
In specific, the sample8_period1.txt has only 5 hours. A separated model with sliding 4-hour window was trained to predict the number.

From the prediction result, we can observe that the number of tweets between 8 am. and 8 pm. has the largest RMSE. The reasons can be anticipated as below:

1. Time between 8 am and 8 pm is the most activate time. Since testset and trainset has different hashtags, it may have different popularity. Thus, using a popular hashtag to predict an unpopular hashtag may cause incorrect prediction.
2. The testdata is sorted by firstpost_date with relatively sparse citation_date. However, the firstpost_date may be not the best way to represent testdata features.
3. From the results in problem 1.4, we can see that hours between 8 am and 8 pm do have larger RMSE compared to other time period. So this kind of result is coherent to our expectation on some degree.

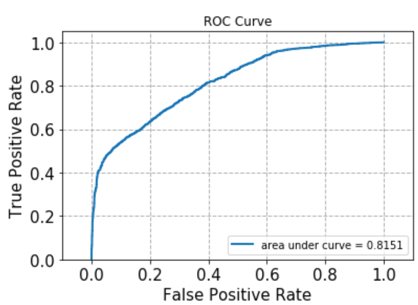
Part 2: Fan Base Prediction

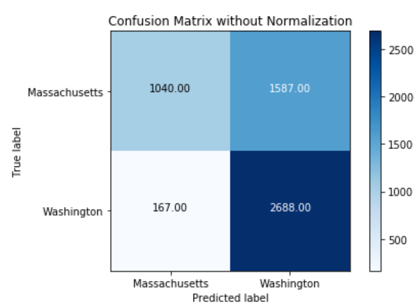
Requirement: Train a binary classifier to predict the location of the author of a tweet (Washington or Massachusetts), given only the textual content of the tweet (using the techniques you learnt in project 1). Try different classification algorithms (at least 3) in your submission. For each, plot ROC curve, report confusion matrix, and calculate accuracy, recall and precision.

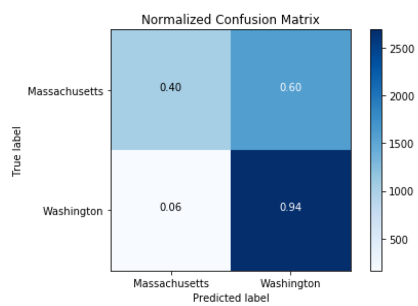
Results:

Soft Margin SVM Classifier (SVC):

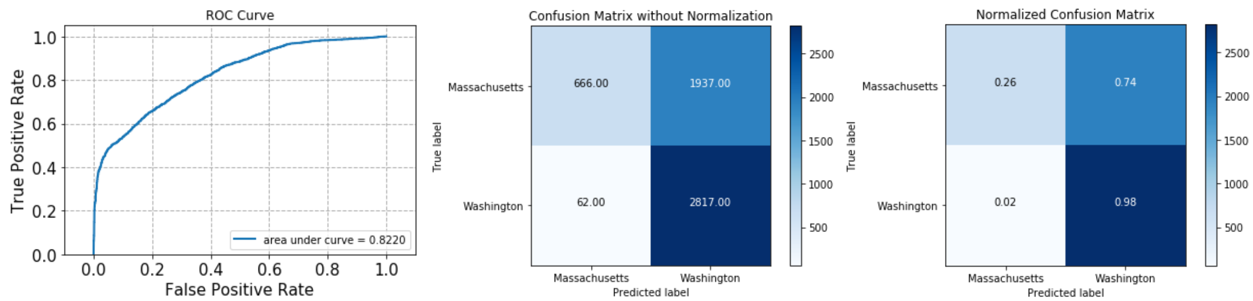
```
Confusion Matrix without Normalization
[[1040 1587]
 [ 167 2688]]
Normalized Confusion Matrix
[[ 0.4   0.6 ]
 [ 0.06  0.94]]
Accuracy: 0.680043779642
Recall: 0.941506129597
Precision: 0.628771929825
```



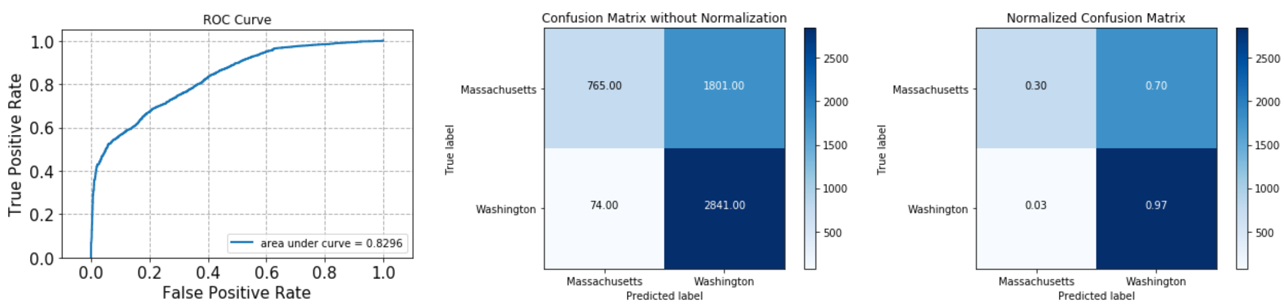




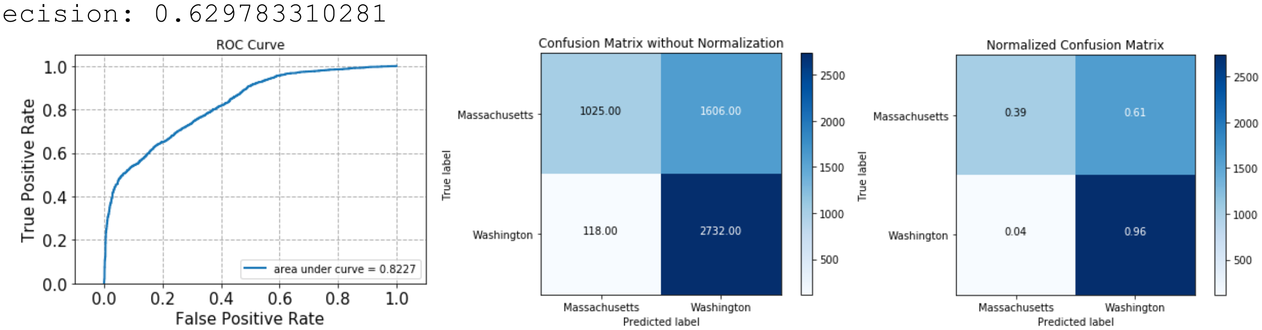

```
------------------------------
Confusion Matrix without Normalization
[[ 666 1937]
 [   62 2817]]
Normalized Confusion Matrix
[[ 0.26  0.74]
 [ 0.02  0.98]]
Accuracy: 0.635352061291
Recall: 0.978464744703
Precision: 0.592553639041
```



Confusion Matrix without Normalization
```
[[ 765 1801]
 [  74 2841]]
```
Normalized Confusion Matrix
```
[[ 0.3   0.7 ]
 [ 0.03  0.97]]
```
Accuracy: 0.657909140668
Recall: 0.97461406518
Precision: 0.612020680741

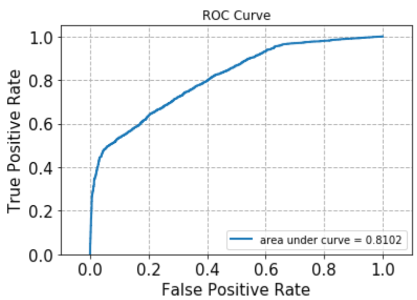


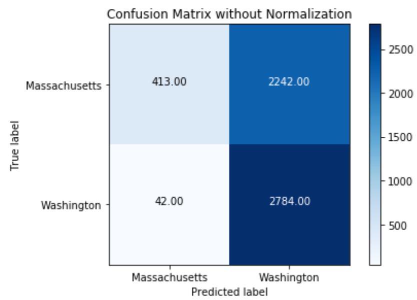
Confusion Matrix without Normalization
```
[[1025 1606]
 [ 118 2732]]
```
Normalized Confusion Matrix
```
[[ 0.39  0.61]
 [ 0.04  0.96]]
```
Accuracy: 0.685458857873
Recall: 0.958596491228
Precision: 0.629783310281

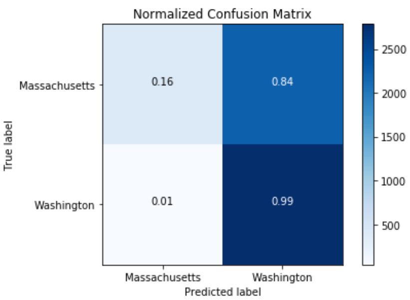


Confusion Matrix without Normalization
```
[[ 413 2242]
 [  42 2784]]
```
Normalized Confusion Matrix

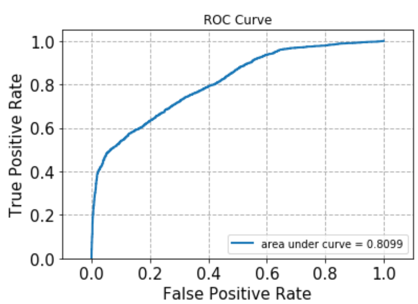
```
[[ 0.16  0.84]
 [ 0.01  0.99]]
Accuracy: 0.583287721219
Recall: 0.985138004246
Precision: 0.553919617986
```

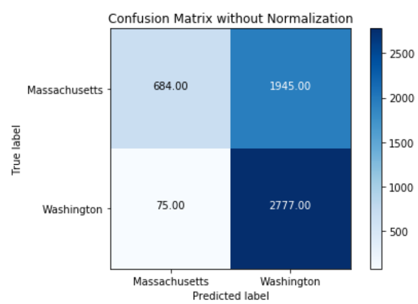


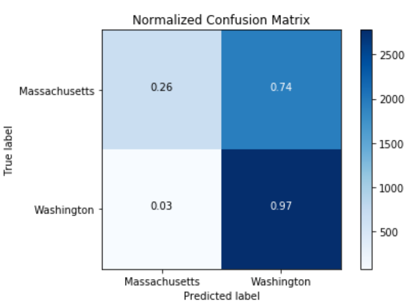




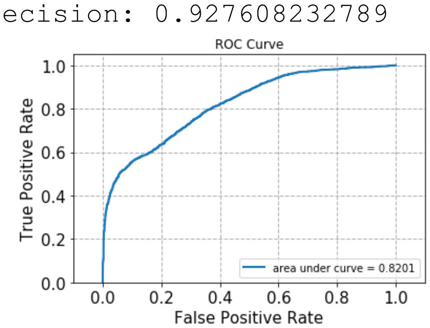

```
------------------------------
Confusion Matrix without Normalization
[[ 684 1945]
 [  75 2777]]
Normalized Confusion Matrix
[[ 0.26  0.74]
 [ 0.03  0.97]]
Accuracy: 0.631454114213
Recall: 0.973702664797
Precision: 0.588098263448
```

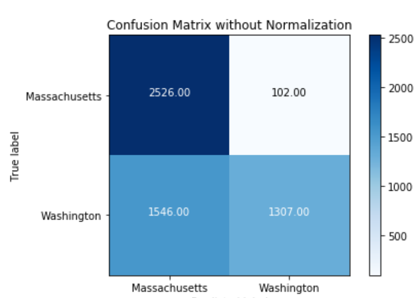


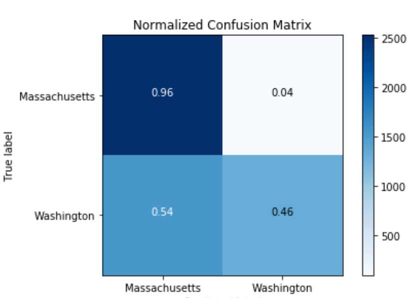




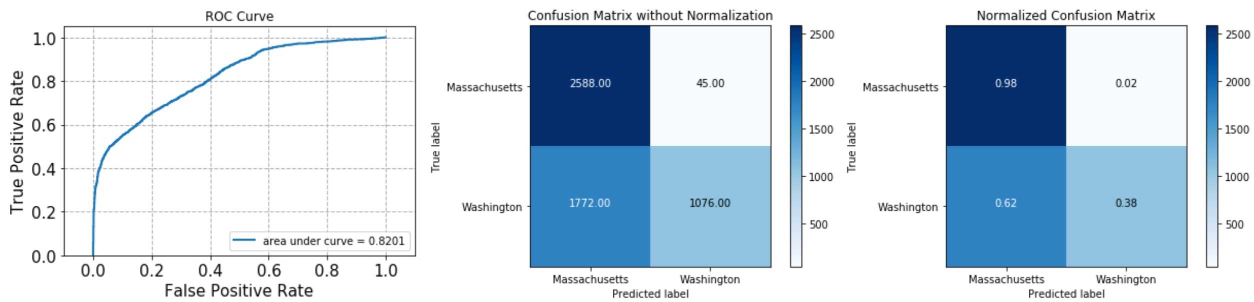

```
------------------------------
Confusion Matrix without Normalization
[[2526  102]
 [1546 1307]]
Normalized Confusion Matrix
[[ 0.96  0.04]
 [ 0.54  0.46]]
Accuracy: 0.699324940704
Recall: 0.458114265685
Precision: 0.927608232789
```



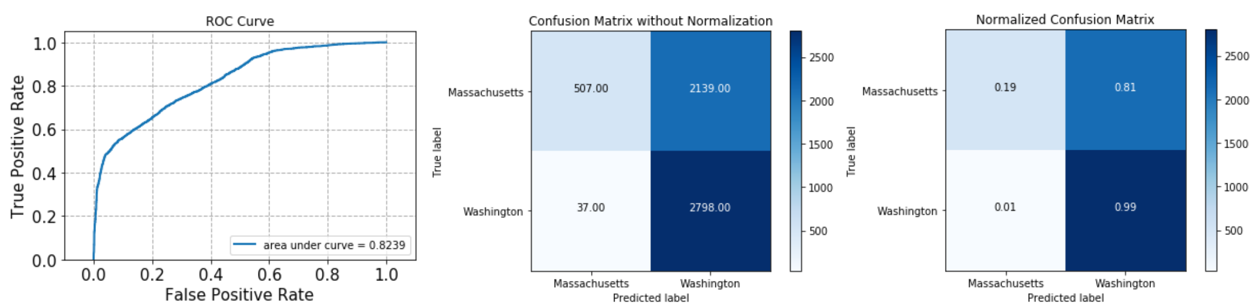




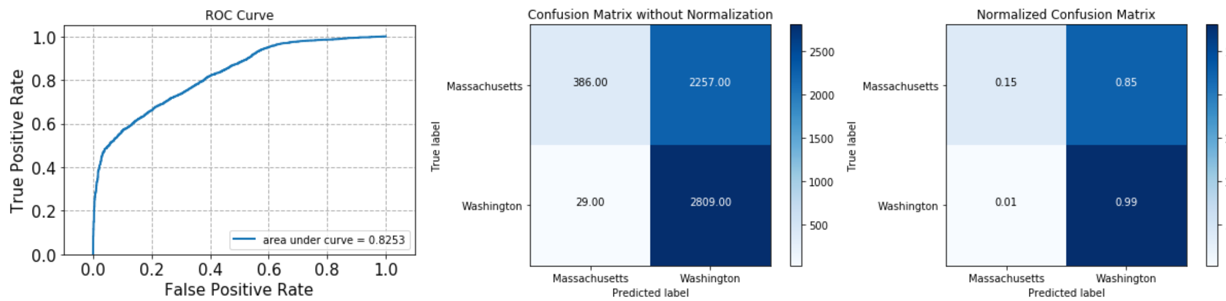
```
------------------------------
Confusion Matrix without Normalization
[[2588   45]
 [1772 1076]]
Normalized Confusion Matrix
[[ 0.98  0.02]
 [ 0.62  0.38]]
Accuracy: 0.66849115125
Recall: 0.377808988764
Precision: 0.959857270294
```




```
------------------------------
Confusion Matrix without Normalization
[[ 507 2139]
 [  37 2798]]
Normalized Confusion Matrix
[[ 0.19  0.81]
 [ 0.01  0.99]]
Accuracy: 0.602992154716
Recall: 0.986948853616
Precision: 0.566740935791
```



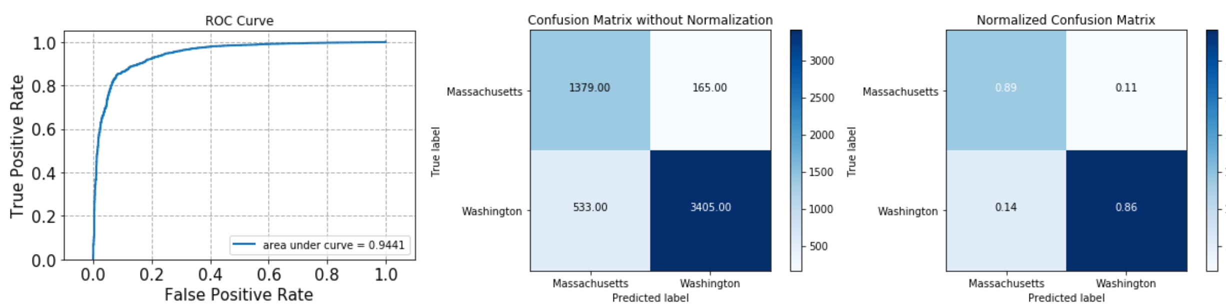

```
------------------------------
Confusion Matrix without Normalization
[[ 386 2257]
 [  29 2809]]
Normalized Confusion Matrix
[[ 0.15  0.85]
 [ 0.01  0.99]]
Accuracy: 0.582922824302
Recall: 0.989781536293
Precision: 0.554480852744
```



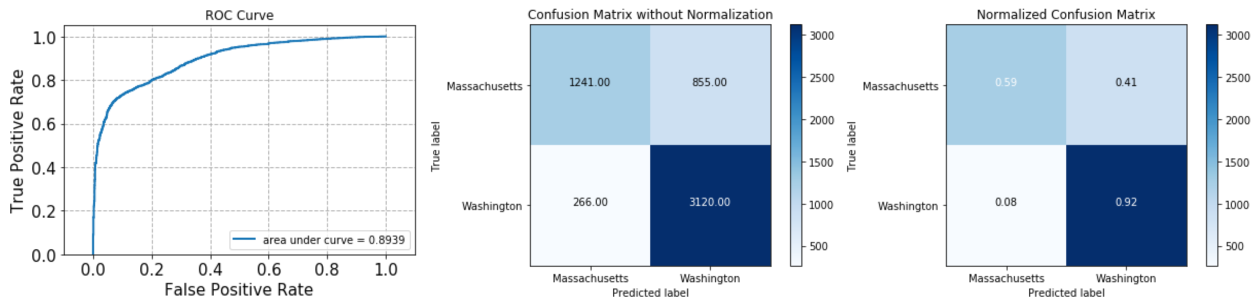

```
------------------------------

The average accuracy is 0.6427
The average recall is 0.8625
The average precision is 0.6614
```

MultinomialNB:

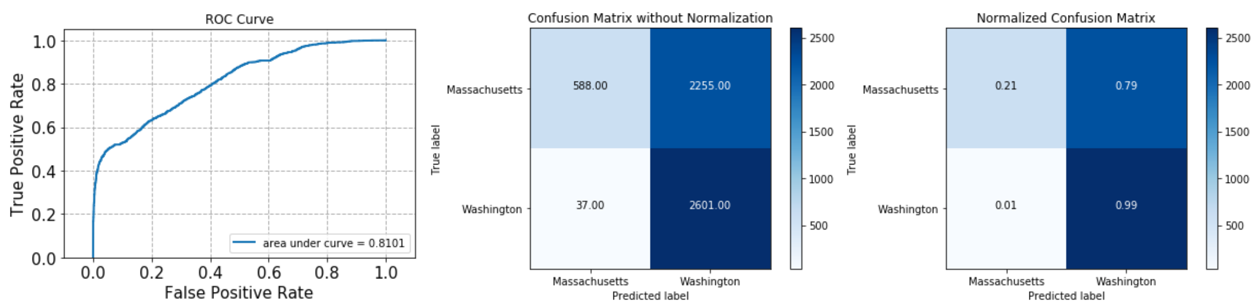
```
Confusion Matrix without Normalization
[[1379  165]
 [ 533 3405]]
Normalized Confusion Matrix
[[ 0.89  0.11]
 [ 0.14  0.86]]
Accuracy: 0.872674206494
Recall: 0.864652107669
Precision: 0.953781512605
```



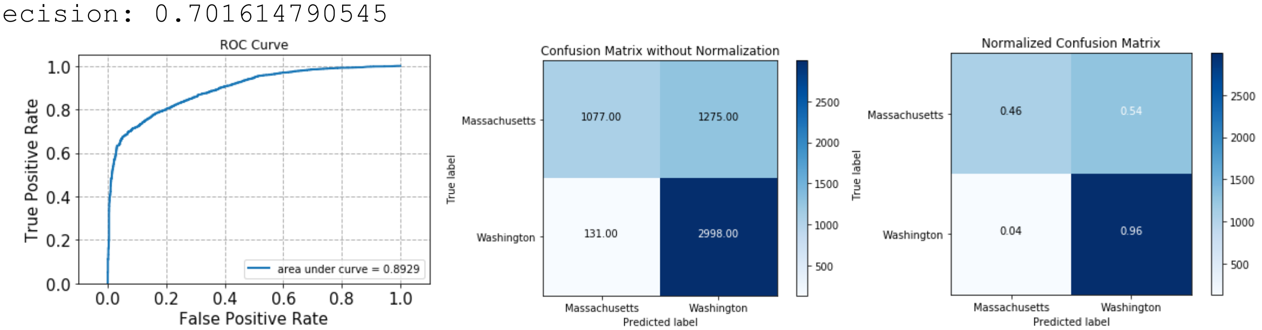

```
------------------------------
Confusion Matrix without Normalization
[[1241  855]
 [ 266 3120]]
Normalized Confusion Matrix
[[ 0.59  0.41]
 [ 0.08  0.92]]
Accuracy: 0.795512586647
Recall: 0.921441228588
Precision: 0.784905660377
```




```
-----------------------------
Confusion Matrix without Normalization
[[ 588 2255]
 [  37 2601]]
Normalized Confusion Matrix
[[ 0.21  0.79]
 [ 0.01  0.99]]
Accuracy: 0.581828133552
Recall: 0.985974222896
Precision: 0.535626029654
```

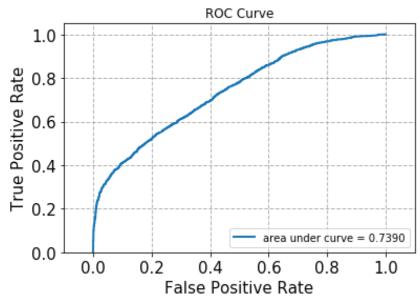


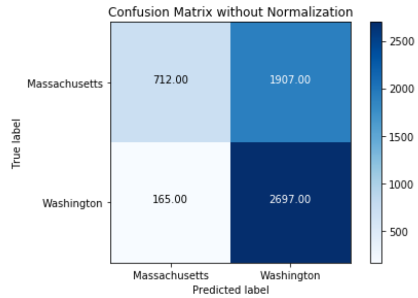

```
-----------------------------
Confusion Matrix without Normalization
[[1077 1275]
 [ 131 2998]]
Normalized Confusion Matrix
[[ 0.46  0.54]
 [ 0.04  0.96]]
Accuracy: 0.743477467615
Recall: 0.958133589006
Precision: 0.701614790545
```

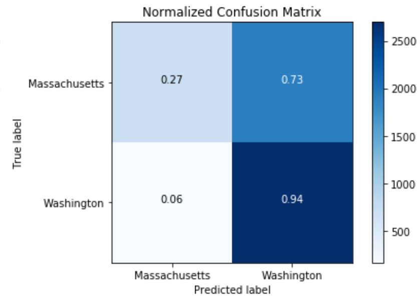



```
-----------------------------
Confusion Matrix without Normalization
[[ 712 1907]
 [ 165 2697]]
Normalized Confusion Matrix
```

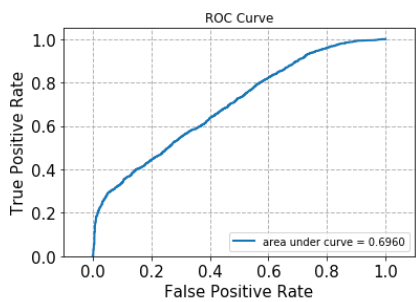
```
[[ 0.27  0.73]
 [ 0.06  0.94]]
Accuracy: 0.621966794381
Recall: 0.942348008386
Precision: 0.585794960904
```

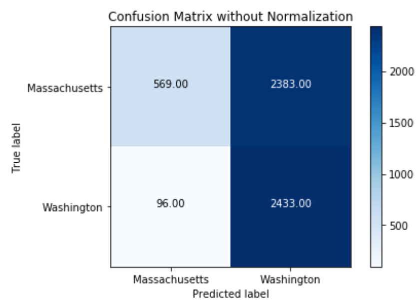


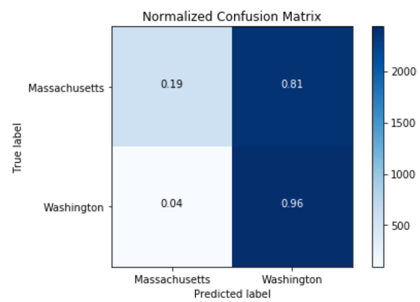




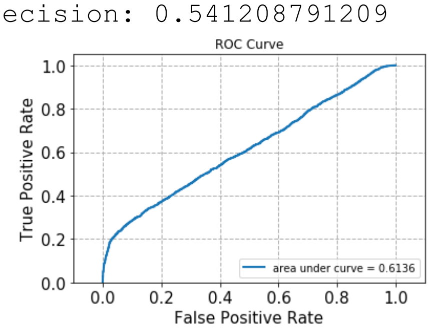

```
------------------------------
Confusion Matrix without Normalization
[[ 569 2383]
 [  96 2433]]
Normalized Confusion Matrix
[[ 0.19  0.81]
 [ 0.04  0.96]]
Accuracy: 0.547710271848
Recall: 0.962040332147
Precision: 0.5051910299
```

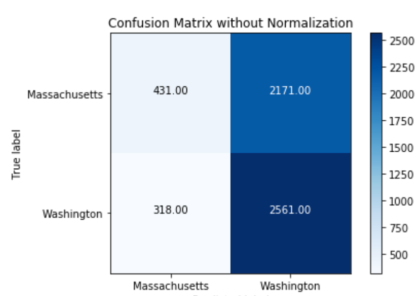


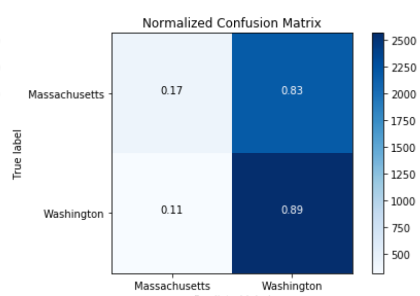




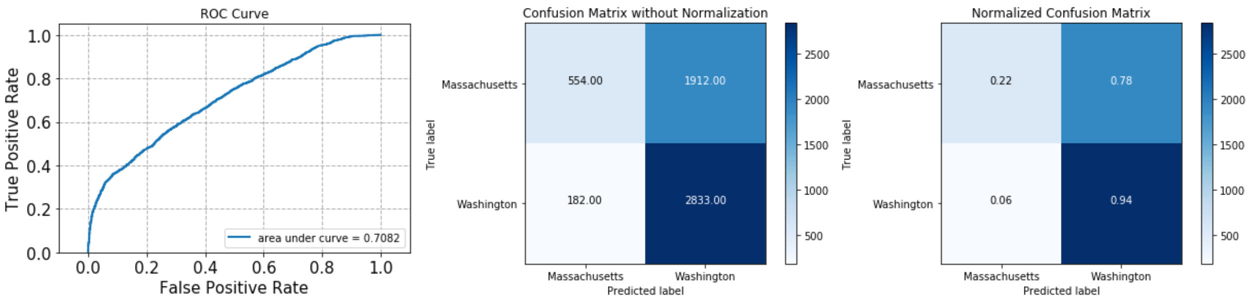

```
------------------------------
Confusion Matrix without Normalization
[[ 431 2171]
 [ 318 2561]]
Normalized Confusion Matrix
[[ 0.17  0.83]
 [ 0.11  0.89]]
Accuracy: 0.545885787265
Recall: 0.889544980896
Precision: 0.541208791209
```



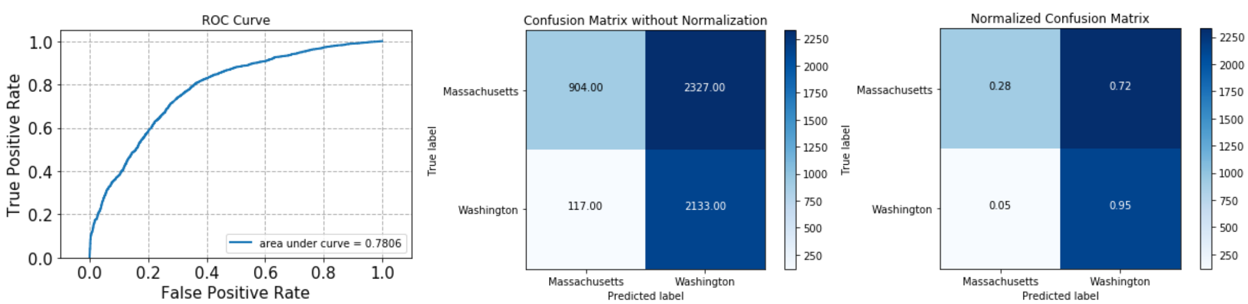




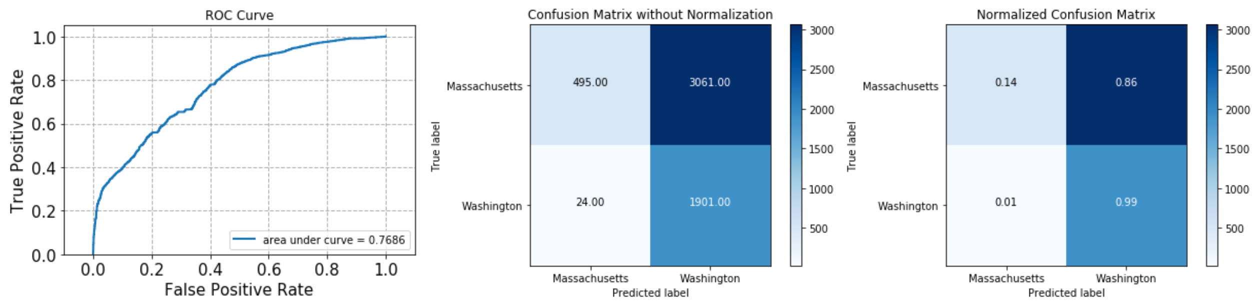
```
------------------------------
Confusion Matrix without Normalization
[[ 554 1912]
 [ 182 2833]]
Normalized Confusion Matrix
[[ 0.22  0.78]
 [ 0.06  0.94]]
Accuracy: 0.617952928298
Recall: 0.939635157546
Precision: 0.597049525817
```




```
------------------------------
Confusion Matrix without Normalization
[[ 904 2327]
 [ 117 2133]]
Normalized Confusion Matrix
[[ 0.28  0.72]
 [ 0.05  0.95]]
Accuracy: 0.554095967889
Recall: 0.948
Precision: 0.478251121076
```



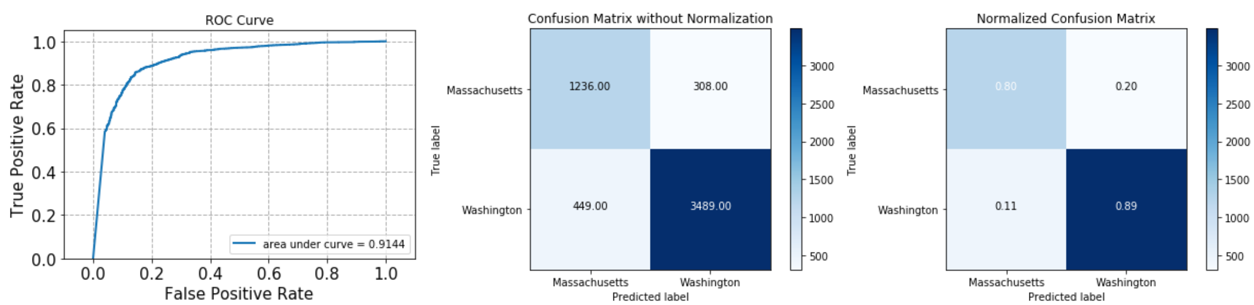

```
------------------------------
Confusion Matrix without Normalization
[[ 495 3061]
 [  24 1901]]
Normalized Confusion Matrix
[[ 0.14  0.86]
 [ 0.01  0.99]]
Accuracy: 0.437146506112
Recall: 0.987532467532
Precision: 0.383111648529
```



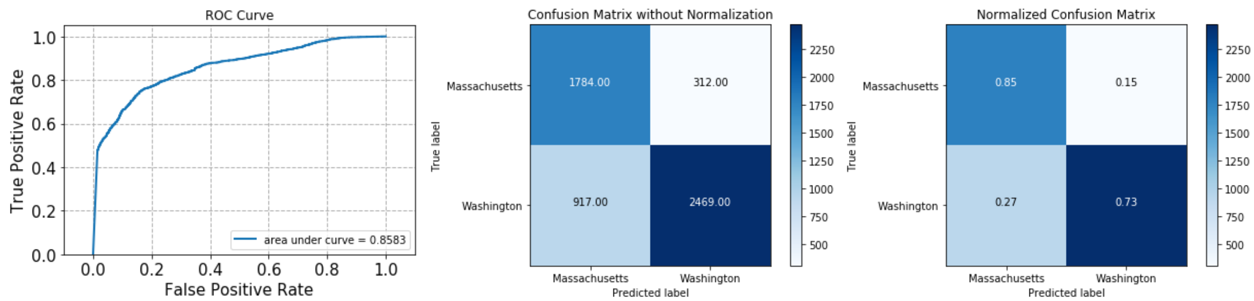
The average accuracy is 0.6318
The average recall is 0.9399
The average precision is 0.6067

GaussianNB:

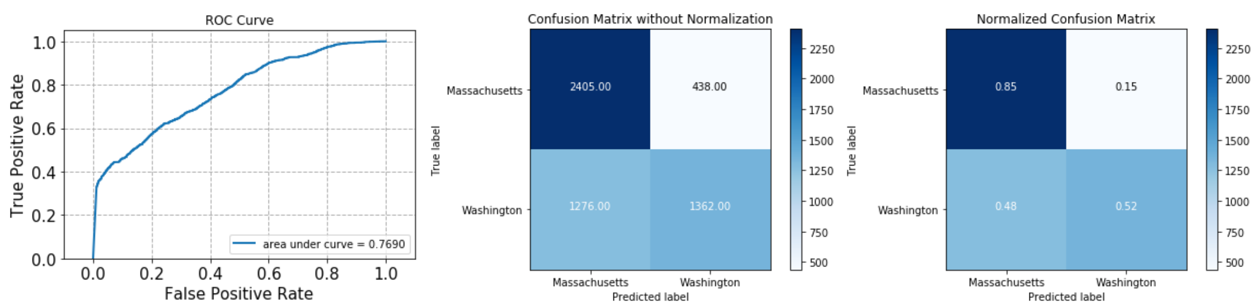
```
Confusion Matrix without Normalization
[[1236  308]
 [ 449 3489]]
Normalized Confusion Matrix
[[ 0.8   0.2 ]
 [ 0.11  0.89]]
Accuracy: 0.861911711054
Recall: 0.885982732351
Precision: 0.918883328944
```



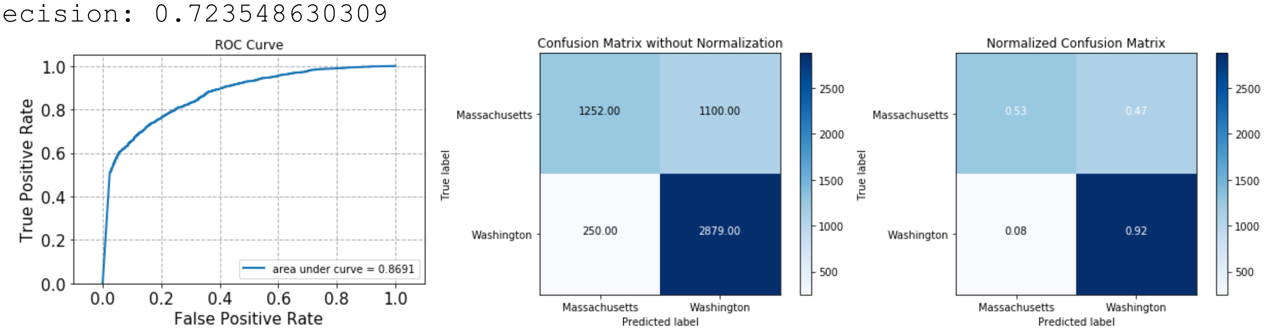
```
Confusion Matrix without Normalization
[[1784  312]
 [ 917 2469]]
Normalized Confusion Matrix
[[ 0.85  0.15]
 [ 0.27  0.73]]
Accuracy: 0.775811747537
Recall: 0.729178972239
Precision: 0.887810140237
```




```
------------------------------
Confusion Matrix without Normalization
[[2405  438]
 [1276 1362]]
Normalized Confusion Matrix
[[ 0.85  0.15]
 [ 0.48  0.52]]
Accuracy: 0.687283342456
Recall: 0.516300227445
Precision: 0.756666666667
```

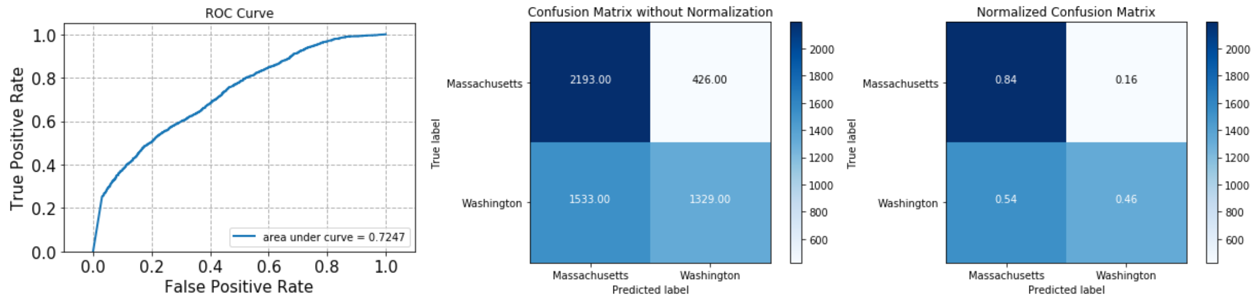



```
------------------------------
Confusion Matrix without Normalization
[[1252 1100]
 [ 250 2879]]
Normalized Confusion Matrix
[[ 0.53  0.47]
 [ 0.08  0.92]]
Accuracy: 0.753694581281
Recall: 0.920102269096
Precision: 0.723548630309
```

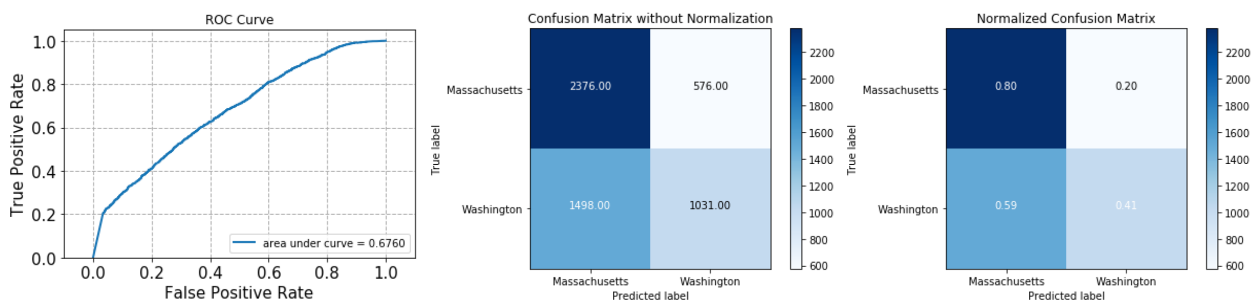



```
------------------------------
Confusion Matrix without Normalization
[[2193  426]
 [1533 1329]]
Normalized Confusion Matrix
```

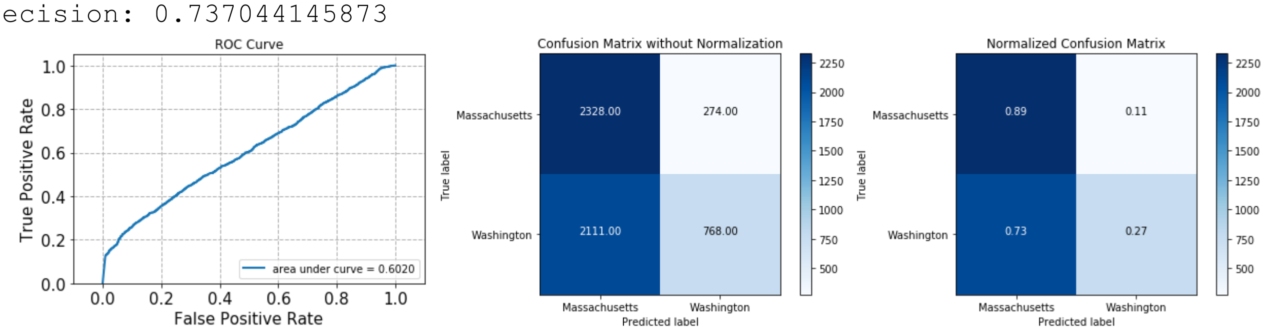
```
[[ 0.84  0.16]
 [ 0.54  0.46]]
Accuracy: 0.64258347017
Recall: 0.464360587002
Precision: 0.757264957265
```



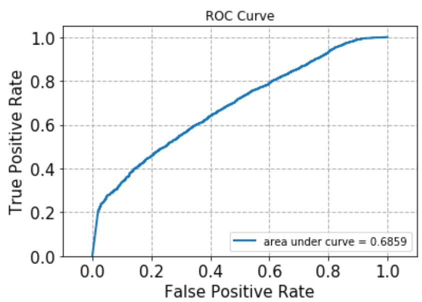

```
-----------------------------
Confusion Matrix without Normalization
[[2376  576]
 [1498 1031]]
Normalized Confusion Matrix
[[ 0.8   0.2 ]
 [ 0.59  0.41]]
Accuracy: 0.621601897464
Recall: 0.407671016212
Precision: 0.64156813939
```

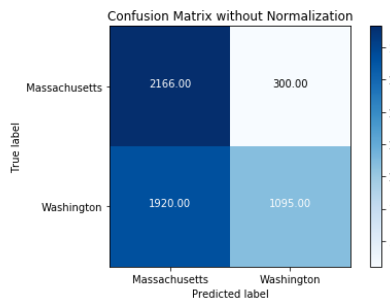


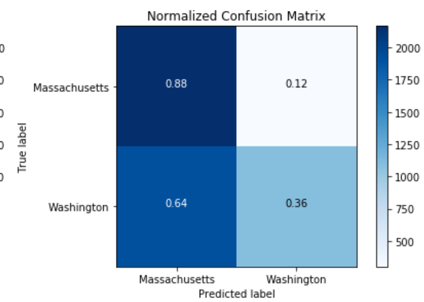

```
-----------------------------
Confusion Matrix without Normalization
[[2328  274]
 [2111  768]]
Normalized Confusion Matrix
[[ 0.89  0.11]
 [ 0.73  0.27]]
Accuracy: 0.564860426929
Recall: 0.266759291421
Precision: 0.737044145873
```



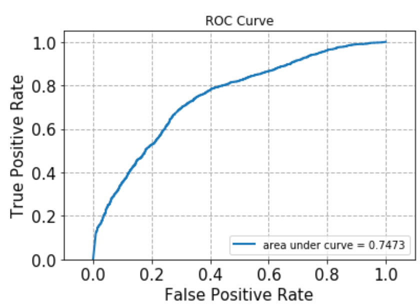
```
------------------------------
Confusion Matrix without Normalization
[[2166  300]
 [1920 1095]]
Normalized Confusion Matrix
[[ 0.88  0.12]
 [ 0.64  0.36]]
Accuracy: 0.594964422551
Recall: 0.363184079602
Precision: 0.784946236559
```

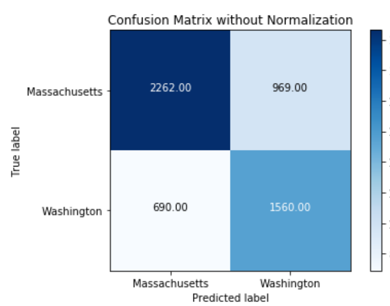


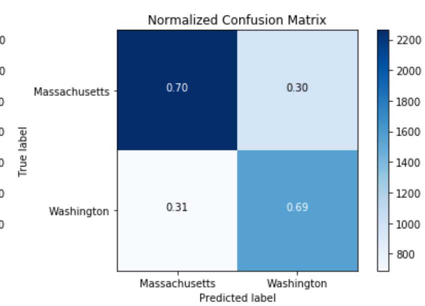




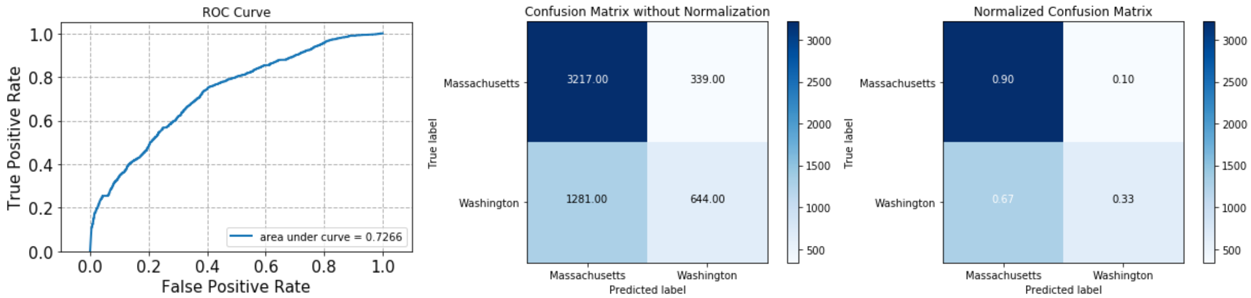

```
------------------------------
Confusion Matrix without Normalization
[[2262  969]
 [ 690 1560]]
Normalized Confusion Matrix
[[ 0.7   0.3 ]
 [ 0.31  0.69]]
Accuracy: 0.697318007663
Recall: 0.693333333333
Precision: 0.61684460261
```







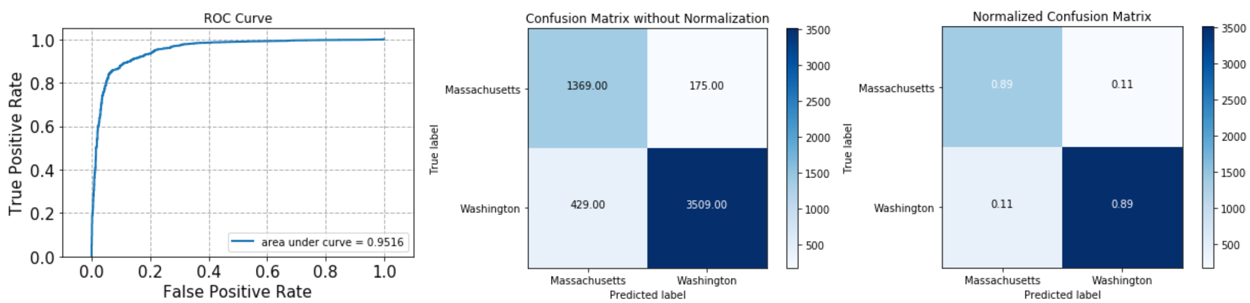

```
------------------------------
Confusion Matrix without Normalization
[[3217  339]
 [1281  644]]
Normalized Confusion Matrix
[[ 0.9   0.1 ]
 [ 0.67  0.33]]
Accuracy: 0.704433497537
Recall: 0.334545454545
Precision: 0.65513733469
```



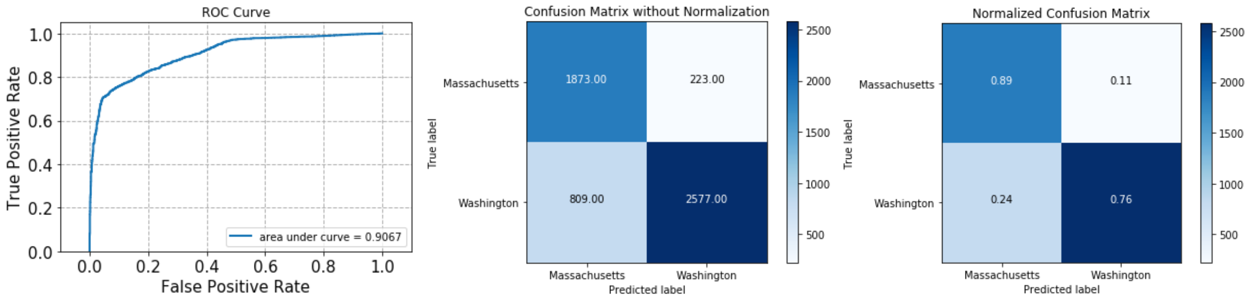

```
------------------------------

The average accuracy is 0.6904
The average recall is 0.5581
The average precision is 0.7480
```

Logistic Regression:

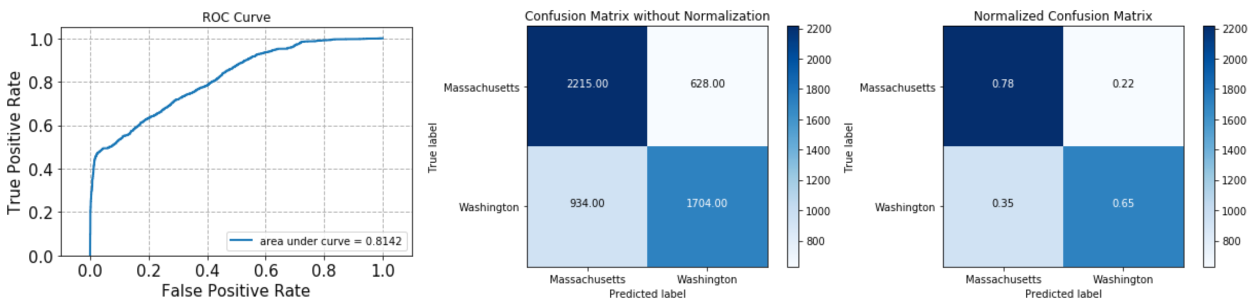
```
Confusion Matrix without Normalization
[[1369  175]
 [ 429 3509]]
Normalized Confusion Matrix
[[ 0.89  0.11]
 [ 0.11  0.89]]
Accuracy: 0.889821233127
Recall: 0.891061452514
Precision: 0.952497285559
```



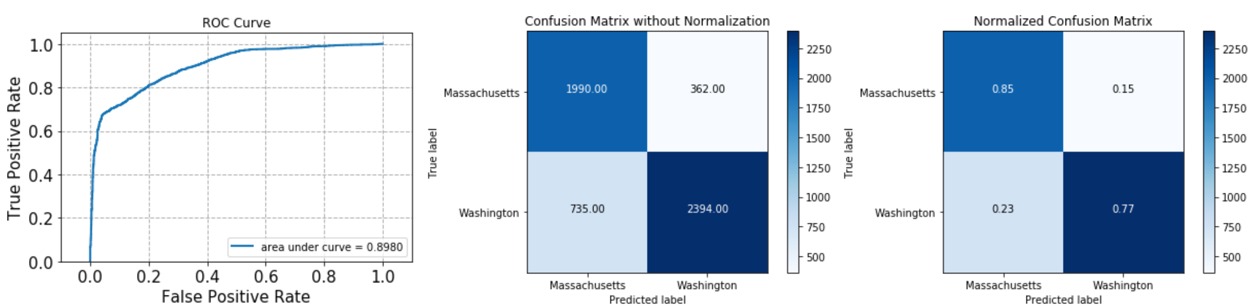

```
------------------------------
Confusion Matrix without Normalization
[[1873  223]
 [ 809 2577]]
Normalized Confusion Matrix
[[ 0.89  0.11]
 [ 0.24  0.76]]
Accuracy: 0.811747537395
Recall: 0.761075014767
Precision: 0.920357142857
```




```
------------------------------
Confusion Matrix without Normalization
[[2215  628]
 [ 934 1704]]
Normalized Confusion Matrix
[[ 0.78  0.22]
 [ 0.35  0.65]]
Accuracy: 0.715015508119
Recall: 0.645943896892
Precision: 0.730703259005
```

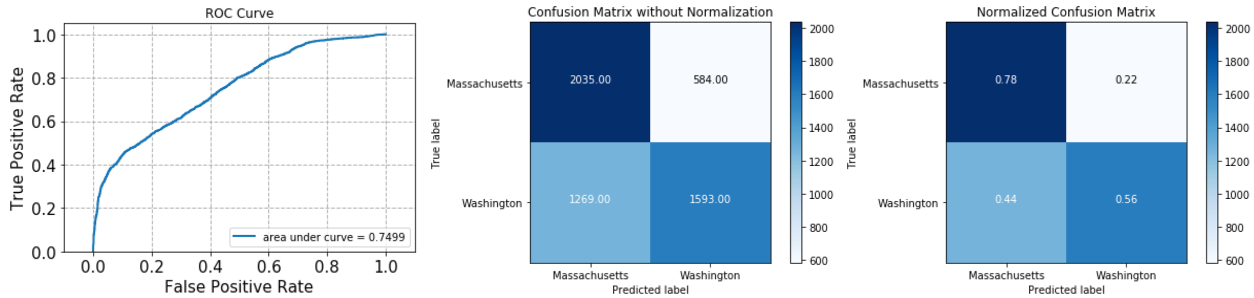



```
------------------------------
Confusion Matrix without Normalization
[[1990  362]
 [ 735 2394]]
Normalized Confusion Matrix
[[ 0.85  0.15]
 [ 0.23  0.77]]
Accuracy: 0.799854041233
Recall: 0.765100671141
Precision: 0.868650217707
```

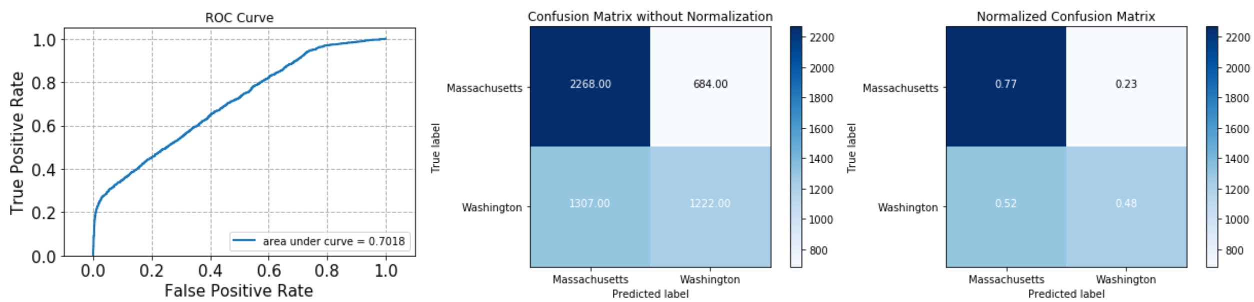



```
------------------------------
Confusion Matrix without Normalization
[[2035  584]
 [1269 1593]]
Normalized Confusion Matrix
```

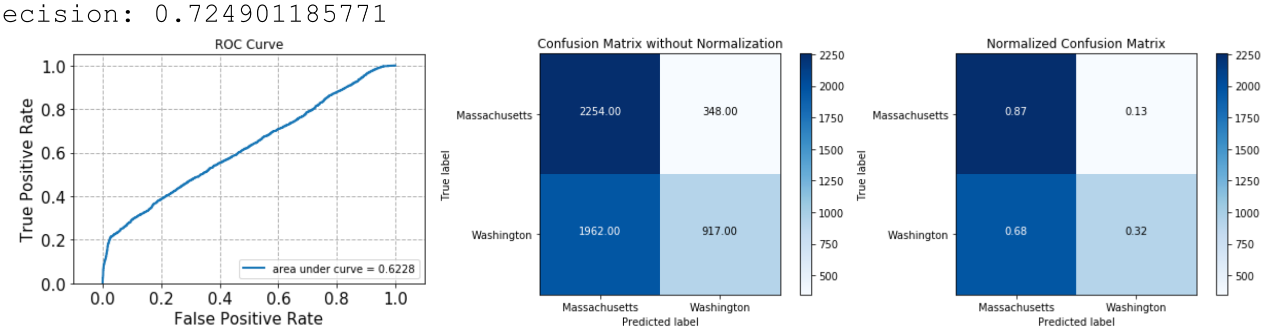
```
[[ 0.78  0.22]
 [ 0.44  0.56]]
Accuracy: 0.661923006751
Recall: 0.556603773585
Precision: 0.731740927882
```



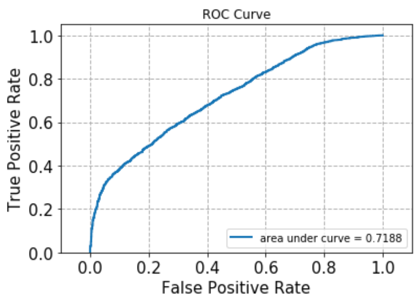

```
------------------------------
Confusion Matrix without Normalization
[[2268  684]
 [1307 1222]]
Normalized Confusion Matrix
[[ 0.77  0.23]
 [ 0.52  0.48]]
Accuracy: 0.636745119504
Recall: 0.483194938711
Precision: 0.641133263379
```

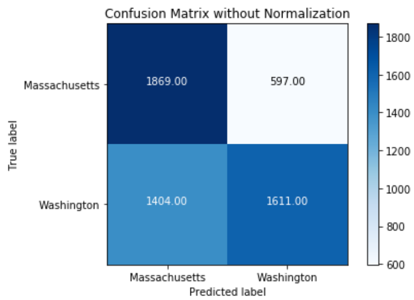


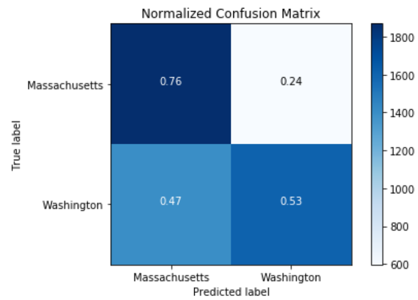

```
------------------------------
Confusion Matrix without Normalization
[[2254  348]
 [1962  917]]
Normalized Confusion Matrix
[[ 0.87  0.13]
 [ 0.68  0.32]]
Accuracy: 0.578544061303
Recall: 0.318513372699
Precision: 0.724901185771
```



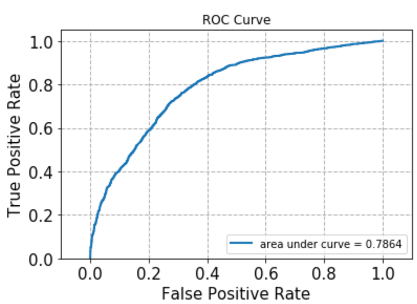
Confusion Matrix without Normalization
[[1869 597]
 [1404 1611]]
Normalized Confusion Matrix
[[0.76 0.24]
 [0.47 0.53]]
Accuracy: 0.634920634921
Recall: 0.534328358209
Precision: 0.729619565217

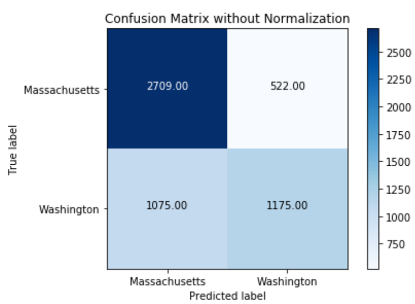


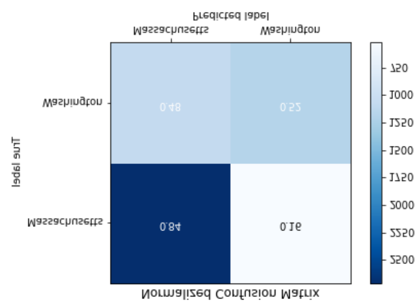




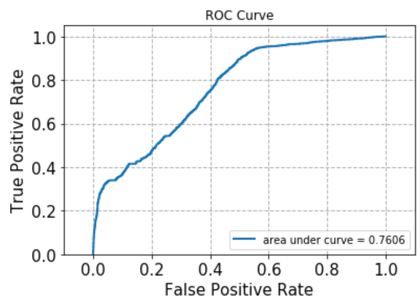
Confusion Matrix without Normalization
[[2709 522]
 [1075 1175]]
Normalized Confusion Matrix
[[0.84 0.16]
 [0.48 0.52]]
Accuracy: 0.708629812078
Recall: 0.522222222222
Precision: 0.692398350029

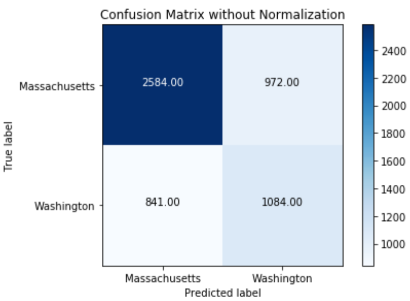


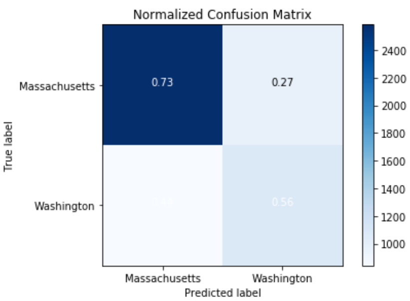




Confusion Matrix without Normalization
[[2584 972]
 [841 1084]]
Normalized Confusion Matrix
[[0.73 0.27]
 [0.44 0.56]]
Accuracy: 0.669220945083
Recall: 0.563116883117
Precision: 0.527237354086



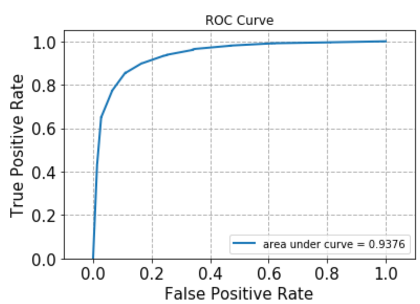


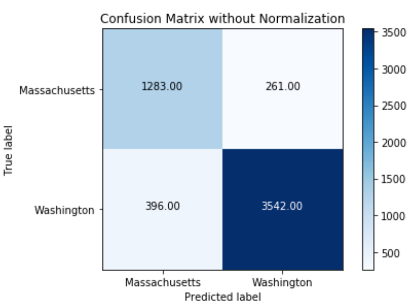


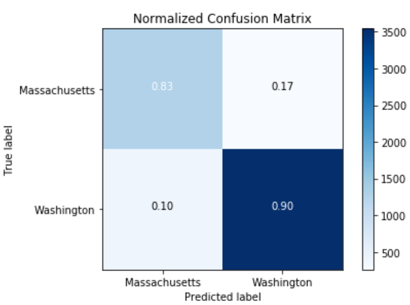

```
------------------------------

The average accuracy is 0.7106
The average recall is 0.6041
The average precision is 0.7519
```

Random Forest Classifier:

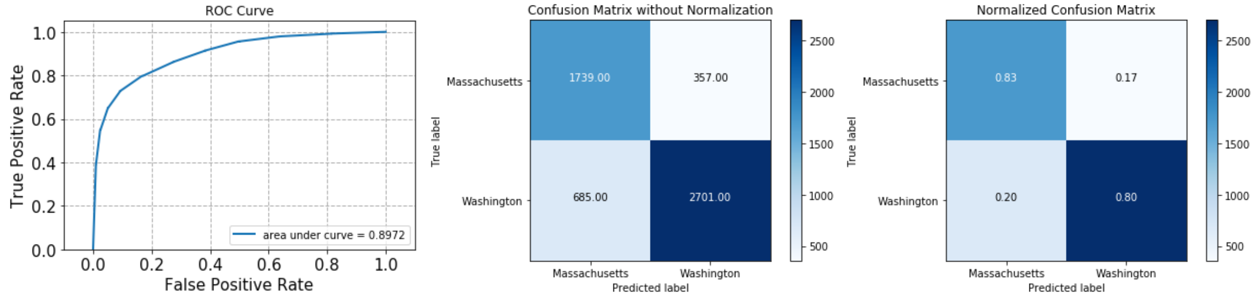
```
Confusion Matrix without Normalization
[[1283  261]
 [ 396 3542]]
Normalized Confusion Matrix
[[ 0.83  0.17]
 [ 0.1   0.9 ]]
Accuracy: 0.880153228749
Recall: 0.899441340782
Precision: 0.931369971075
```



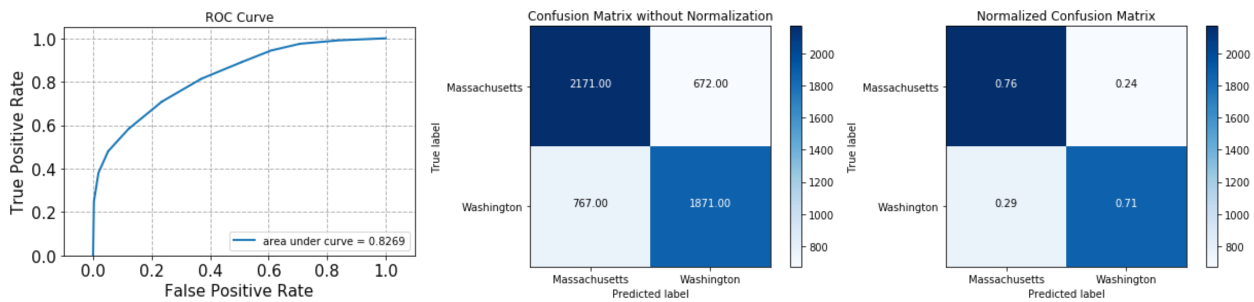




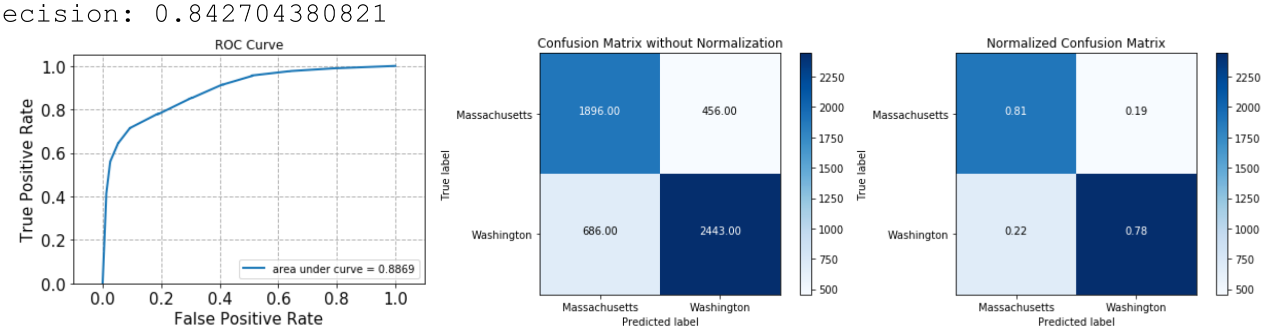

```
------------------------------
Confusion Matrix without Normalization
[[1739  357]
 [ 685 2701]]
Normalized Confusion Matrix
[[ 0.83  0.17]
 [ 0.2   0.8 ]]
Accuracy: 0.809923385626
Recall: 0.797696396929
Precision: 0.883257030739
```



Confusion Matrix without Normalization
[[2171 672]
 [767 1871]]
Normalized Confusion Matrix
[[0.76 0.24]
 [0.29 0.71]]
Accuracy: 0.737456668491
Recall: 0.709249431387
Precision: 0.735745182855

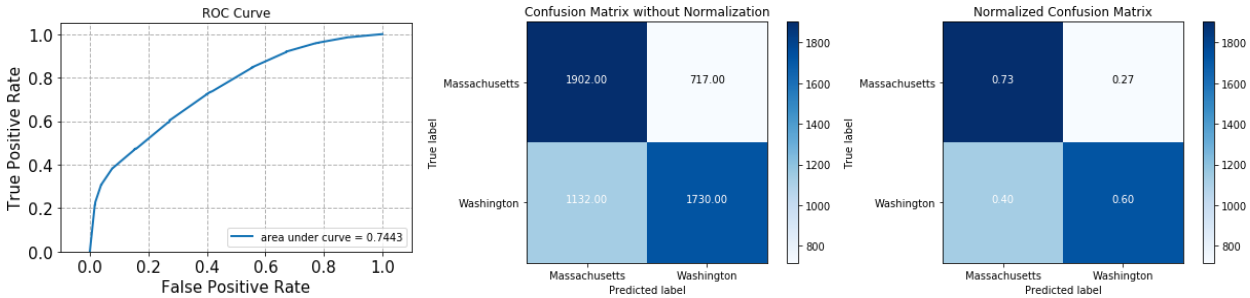


Confusion Matrix without Normalization
[[1896 456]
 [686 2443]]
Normalized Confusion Matrix
[[0.81 0.19]
 [0.22 0.78]]
Accuracy: 0.791643860609
Recall: 0.780760626398
Precision: 0.842704380821

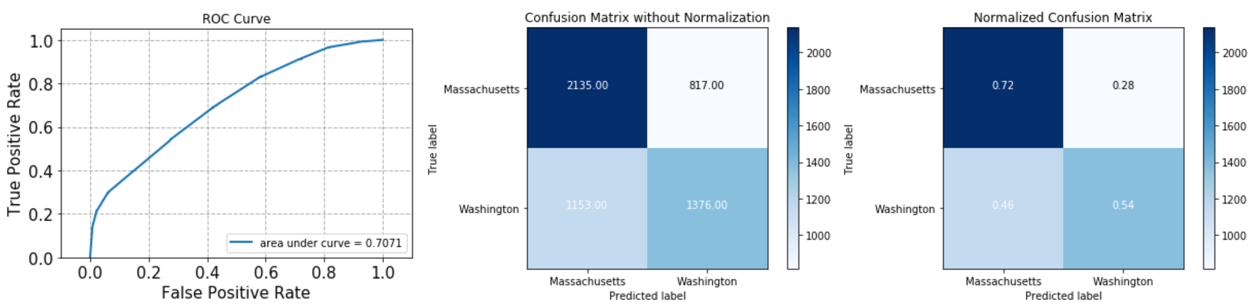


Confusion Matrix without Normalization
[[1902 717]
 [1132 1730]]
Normalized Confusion Matrix

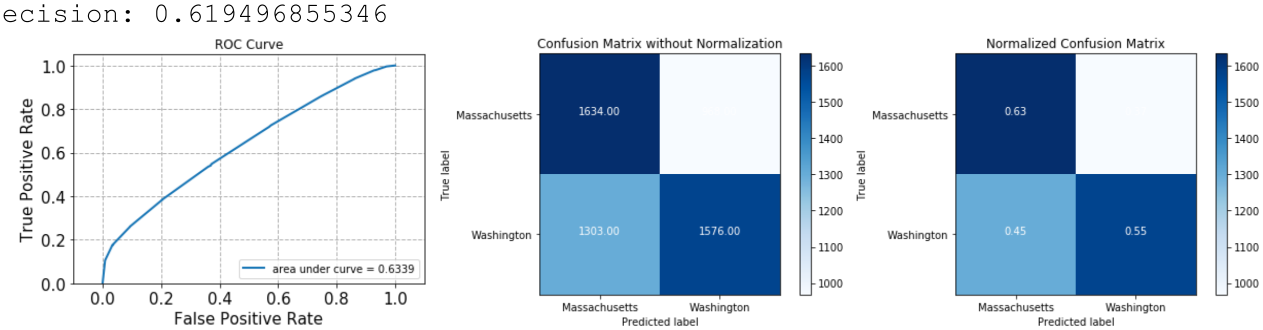
```
[[ 0.73  0.27]
 [ 0.4   0.6 ]]
Accuracy: 0.662652800584
Recall: 0.604472396925
Precision: 0.706988148754
```



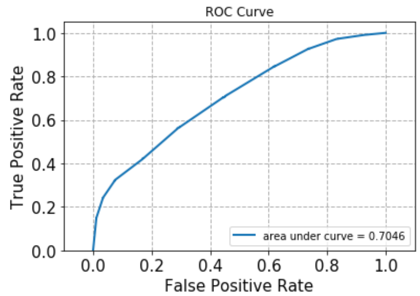

```
------------------------------
Confusion Matrix without Normalization
[[2135  817]
 [1153 1376]]
Normalized Confusion Matrix
[[ 0.72  0.28]
 [ 0.46  0.54]]
Accuracy: 0.640576537128
Recall: 0.544088572558
Precision: 0.627450980392
```

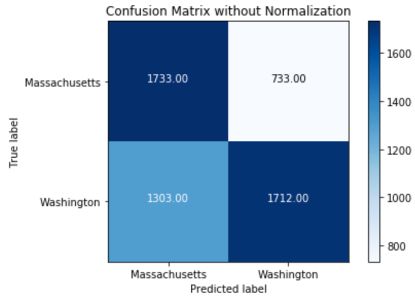


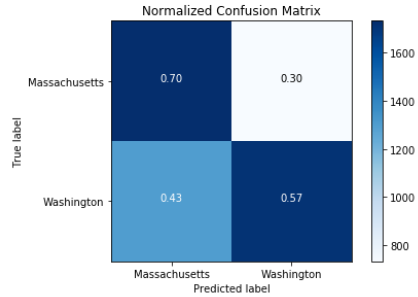

```
------------------------------
Confusion Matrix without Normalization
[[1634  968]
 [1303 1576]]
Normalized Confusion Matrix
[[ 0.63  0.37]
 [ 0.45  0.55]]
Accuracy: 0.585659551177
Recall: 0.547412295936
Precision: 0.619496855346
```



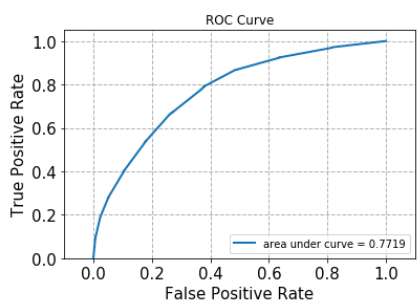
```
------------------------------
Confusion Matrix without Normalization
[[1733  733]
 [1303 1712]]
Normalized Confusion Matrix
[[ 0.7   0.3 ]
 [ 0.43  0.57]]
Accuracy: 0.62853493888
Recall: 0.567827529022
Precision: 0.700204498978
```

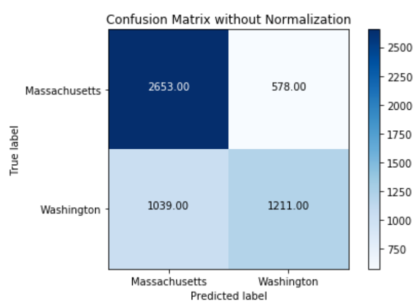


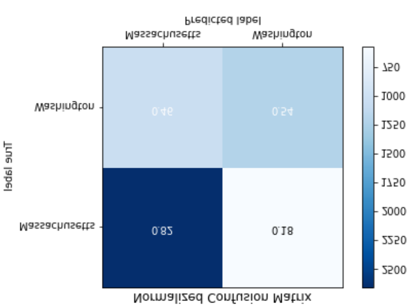




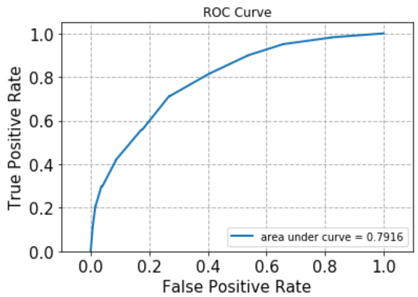

```
------------------------------
Confusion Matrix without Normalization
[[2653  578]
 [1039 1211]]
Normalized Confusion Matrix
[[ 0.82  0.18]
 [ 0.46  0.54]]
Accuracy: 0.704980842912
Recall: 0.538222222222
Precision: 0.676914477362
```

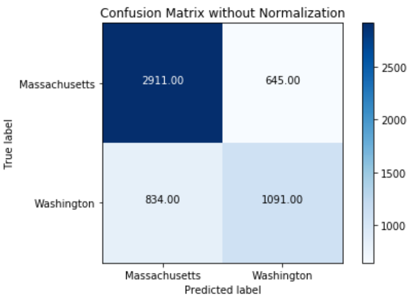


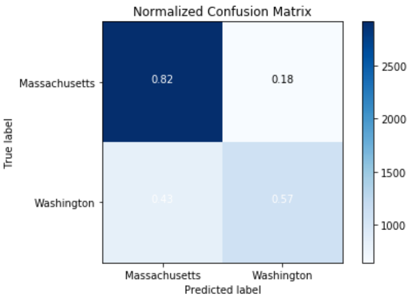




```
------------------------------
Confusion Matrix without Normalization
[[2911  645]
 [ 834 1091]]
Normalized Confusion Matrix
[[ 0.82  0.18]
 [ 0.43  0.57]]
Accuracy: 0.730158730159
Recall: 0.566753246753
Precision: 0.628456221198
```








```
------------------------------

The average accuracy is 0.7172
The average recall is 0.6556
The average precision is 0.7353
```

Methods / Features	LinearSVC	MultinomialNB	GaussianNB	LogisticRegression	RandomForest Classifier
Accuracy	0.6427	0.6318	0.6904	0.7106	0.7172
Recall	0.8625	0.9399	0.5581	0.6041	0.6556
Precision	0.6614	0.6067	0.7480	0.7519	0.7353

Discussion:
We have applied 5 classifiers listed above to predict the location of the author of a tweet. For each classifier, we applied 10-fold to split the data into trainset and testset in order to eliminate the occasionality and computed the average accuracy, recall and precision in the end. From the result, it could be included that none of the classifiers perform quite well. And the Random Forest Classifier seems to be comparatively the best in those 5 classifiers whose accuracy achieves 0.7172.

Part 3: Sentiment analysis and language distribution of the fan

Requirement: The dataset in hands is rich as there is a lot of metadata to each tweet. Be creative and propose a new problem (something interesting that can be inferred from this dataset) other than the previous parts. You can look into the literature of Twitter data analysis to get some ideas. Implement your idea and show that it works. As a suggestion, you might provide some analysis based on changes of tweet sentiments for fans of the opponent teams participating in the match. You get full credit for bringing in novelty and full or partial implementation of your new ideas.

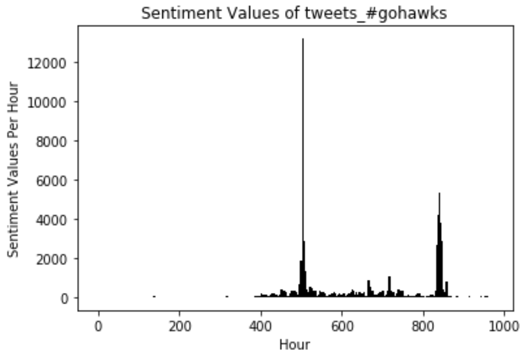
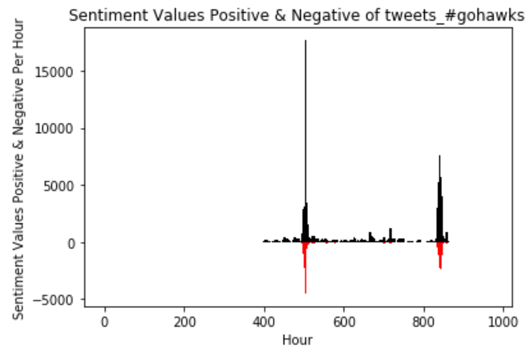
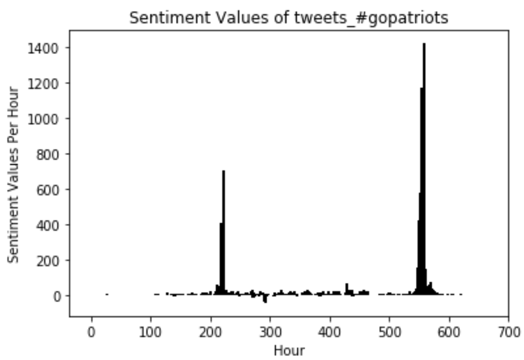
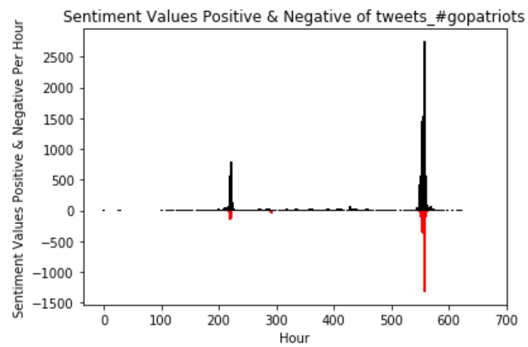
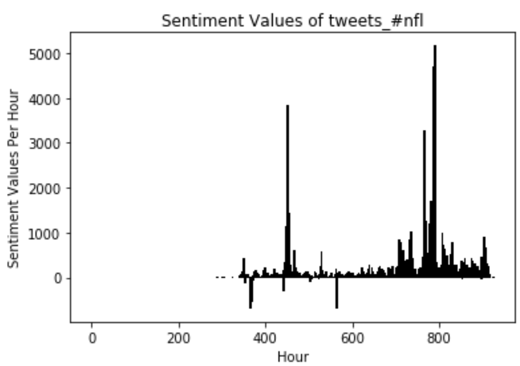
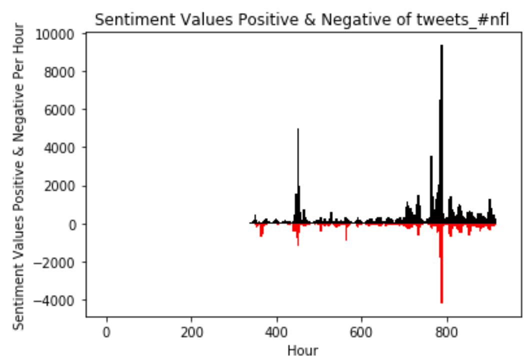
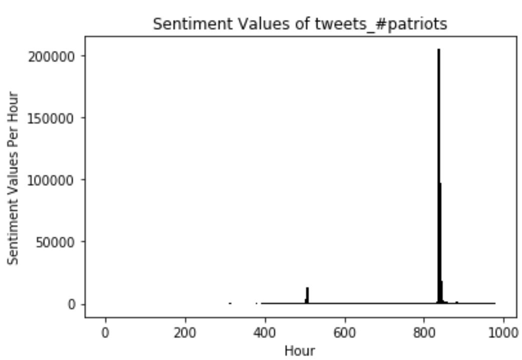
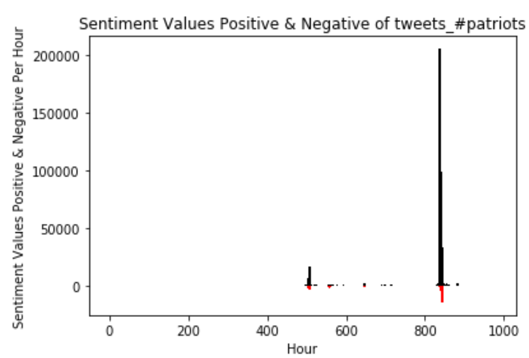
Idea:

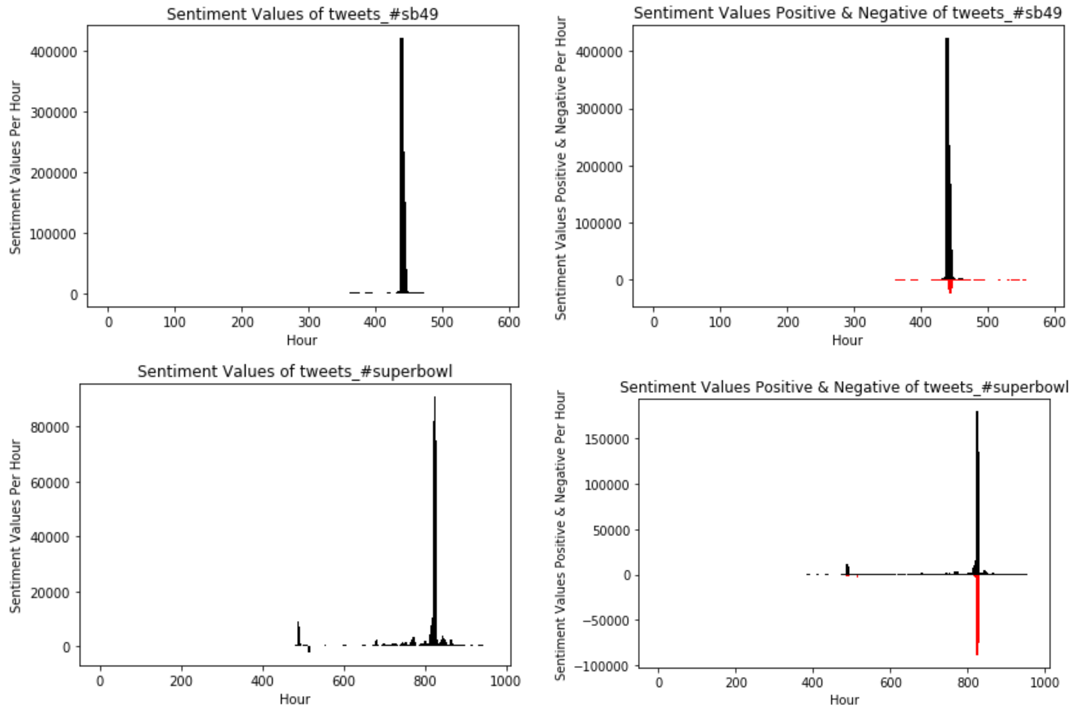
In this final project, we want to explore something new. We choose sentiment analysis of this tweet data as our study goal with some mature analysis tool on internet.

Results:

Use AFINN wordlist to calculate the sentiment values:

The first thing we do is searching the background knowledge of sentiment analysis. I've found a word list named AFINN containing the sentimental score of every words ranging from . We utilize AFINN to calculate the sentimental values (the summation of all the scores in certain tweet)of every tweet. Then we sum all the sentimental values in every hour for each file and plot them in the histogram. Also, we divide the sentimental values into positive scores and negative scores and we plot the same score vs time histogram for every file.

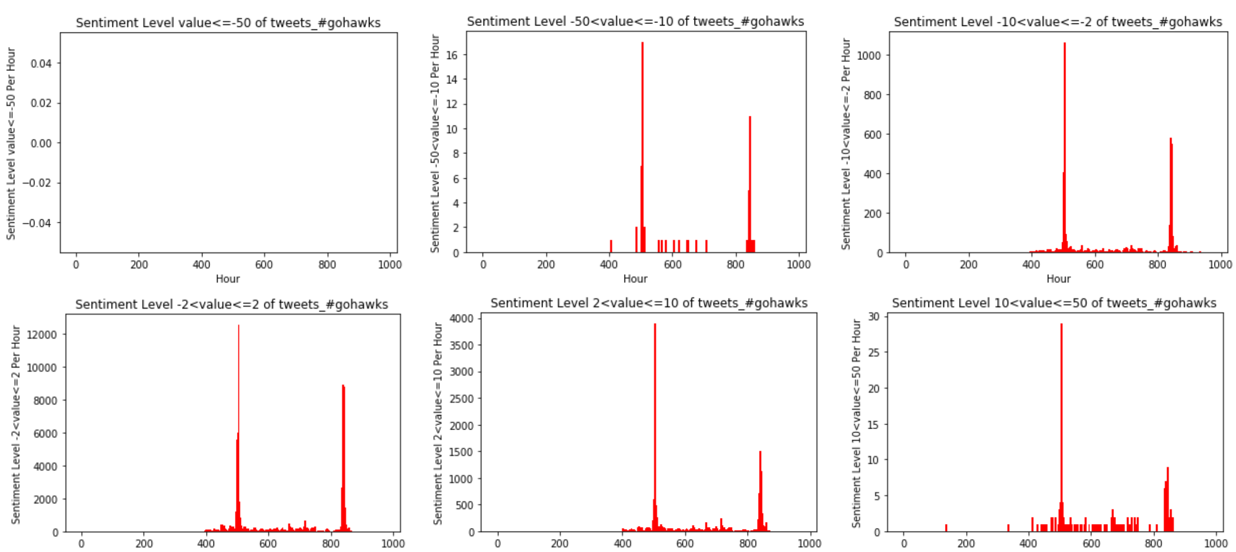

Sentiment Values of tweets_#gohawks
Sentiment Values Per Hour
Hour

Sentiment Values Positive & Negative of tweets_#gohawks
Sentiment Values Positive & Negative Per Hour
Hour

Sentiment Values of tweets_#gopatriots
Sentiment Values Per Hour
Hour

Sentiment Values Positive & Negative of tweets_#gopatriots
Sentiment Values Positive & Negative Per Hour
Hour

Sentiment Values of tweets_#nfl
Sentiment Values Per Hour
Hour

Sentiment Values Positive & Negative of tweets_#nfl
Sentiment Values Positive & Negative Per Hour
Hour

Sentiment Values of tweets_#patriots
Sentiment Values Per Hour
Hour

Sentiment Values Positive & Negative of tweets_#patriots
Sentiment Values Positive & Negative Per Hour
Hour

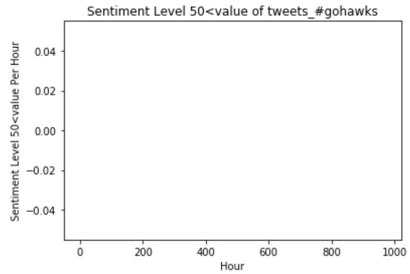


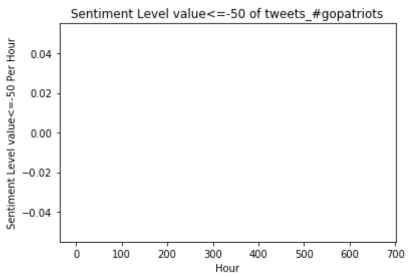
From the results, we can find that, no matter the positive sentiment or negative sentiment they suddenly reach their local maximums, which are lots of slender peaks according to the figures. What's more, the local maximums of the positive tweets usually come with the negative ones, but the former outnumber the latte most of time. And for different hashtag, the peaks are not identical. We made a conclusion from those results that some important events might happen at certain time or the hashtag was used by some celebrities at that time.

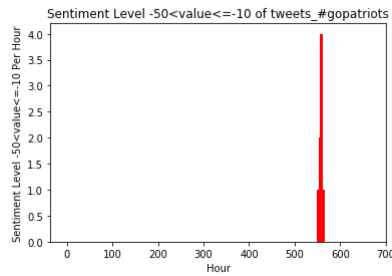
Use AFINN wordlist for the sentiment level:

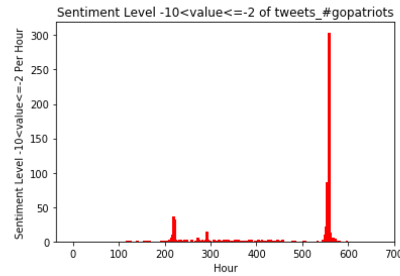
We then divide the sentimental values of each tweets into seven sentiment levels. Then we count the showing times for each level in every hour and plot them into the histogram as usual. Below are the results.

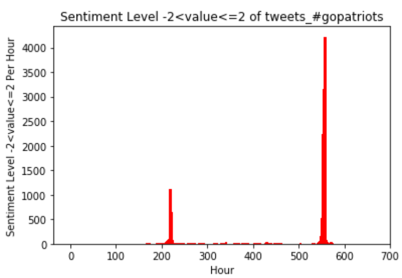


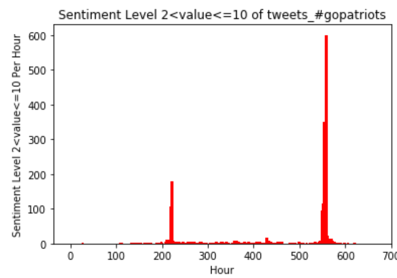

Sentiment Level 50<value of tweets_#gohawks
Sentiment Level 50<value Per Hour
Hour

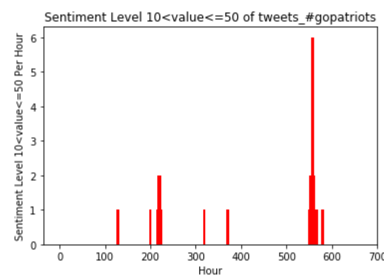

Sentiment Level value<=-50 of tweets_#gopatriots
Sentiment Level value<=-50 Per Hour
Hour

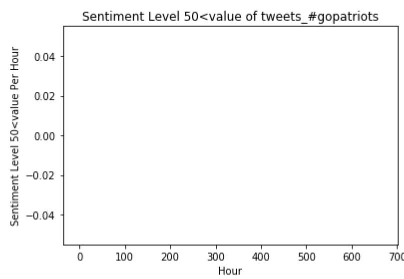

Sentiment Level -50<value<=-10 of tweets_#gopatriots
Sentiment Level -50<value<=-10 Per Hour
Hour

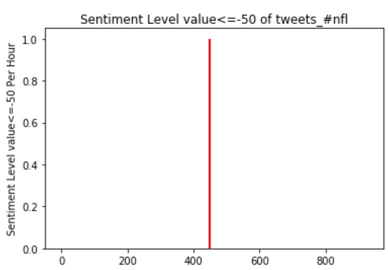

Sentiment Level -10<value<=-2 of tweets_#gopatriots
Sentiment Level -10<value<=-2 Per Hour
Hour

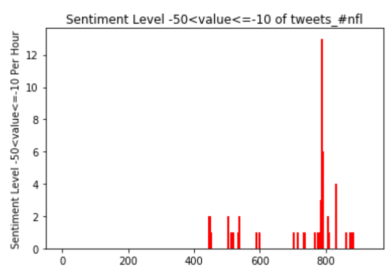

Sentiment Level -2<value<=2 of tweets_#gopatriots
Sentiment Level -2<value<=2 Per Hour
Hour

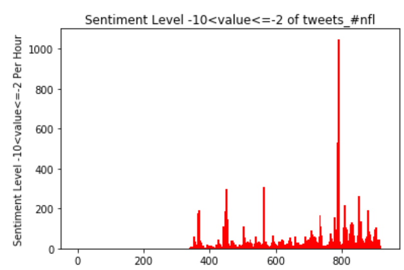

Sentiment Level 2<value<=10 of tweets_#gopatriots
Sentiment Level 2<value<=10 Per Hour
Hour

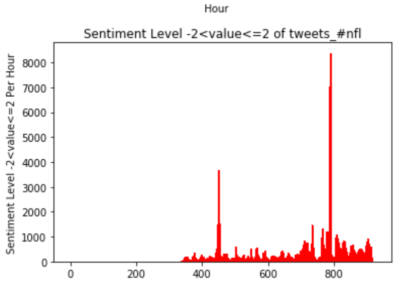

Sentiment Level 10<value<=50 of tweets_#gopatriots
Sentiment Level 10<value<=50 Per Hour
Hour

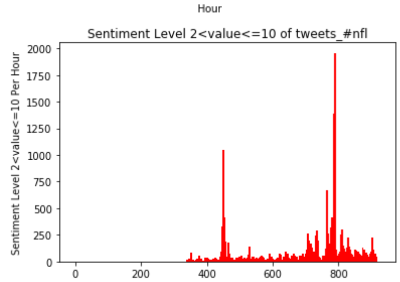

Sentiment Level 50<value of tweets_#gopatriots
Sentiment Level 50<value Per Hour
Hour

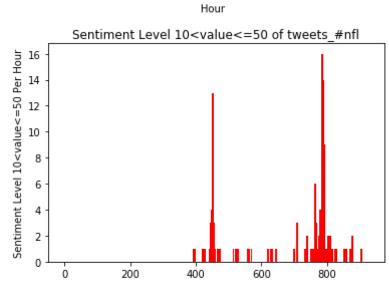

Sentiment Level value<=-50 of tweets_#nfl
Sentiment Level value<=-50 Per Hour
Hour

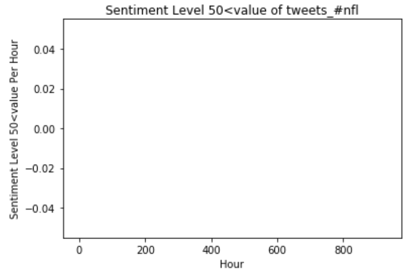

Sentiment Level -50<value<=-10 of tweets_#nfl
Sentiment Level -50<value<=-10 Per Hour
Hour

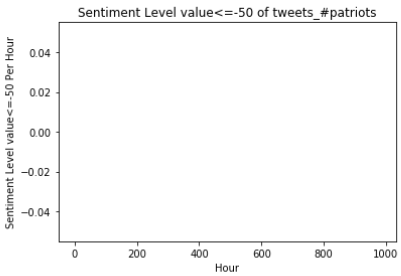

Sentiment Level -10<value<=-2 of tweets_#nfl
Sentiment Level -10<value<=-2 Per Hour
Hour

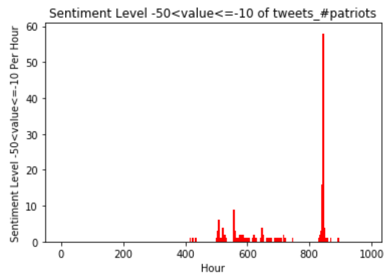

Sentiment Level -2<value<=2 of tweets_#nfl
Sentiment Level -2<value<=2 Per Hour
Hour

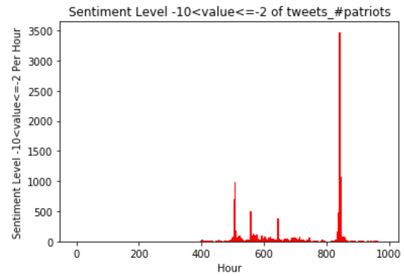

Sentiment Level 2<value<=10 of tweets_#nfl
Sentiment Level 2<value<=10 Per Hour
Hour

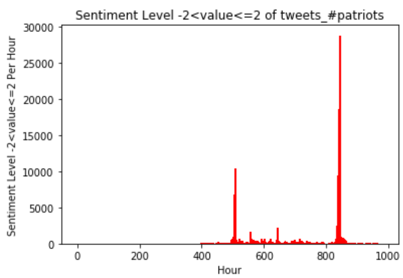

Sentiment Level 10<value<=50 of tweets_#nfl
Sentiment Level 10<value<=50 Per Hour
Hour

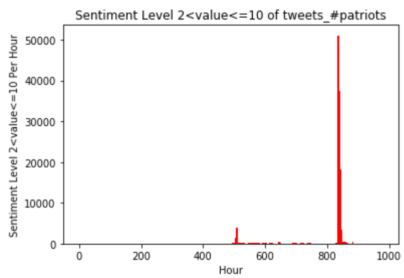

Sentiment Level 50<value of tweets_#nfl
Sentiment Level 50<value Per Hour
Hour

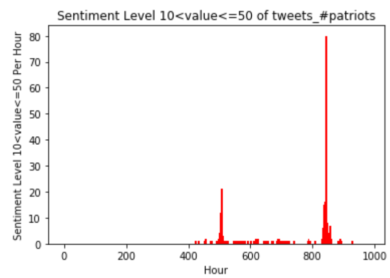

Sentiment Level value<=-50 of tweets_#patriots
Sentiment Level value<=-50 Per Hour
Hour

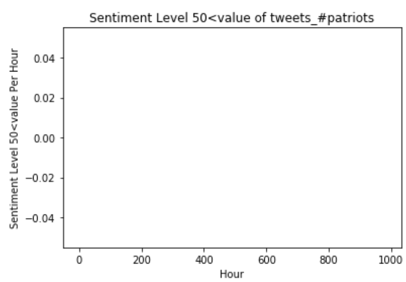

Sentiment Level -50<value<=-10 of tweets_#patriots
Sentiment Level -50<value<=-10 Per Hour
Hour

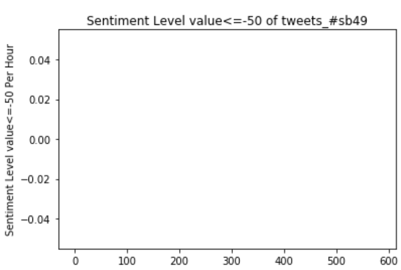

Sentiment Level -10<value<=-2 of tweets_#patriots
Sentiment Level -10<value<=-2 Per Hour
Hour

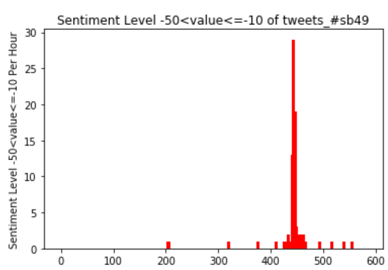

Sentiment Level -2<value<=2 of tweets_#patriots
Sentiment Level -2<value<=2 Per Hour
Hour

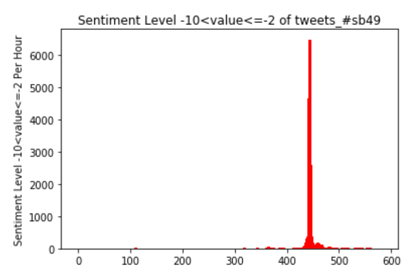

Sentiment Level 2<value<=10 of tweets_#patriots
Sentiment Level 2<value<=10 Per Hour
Hour

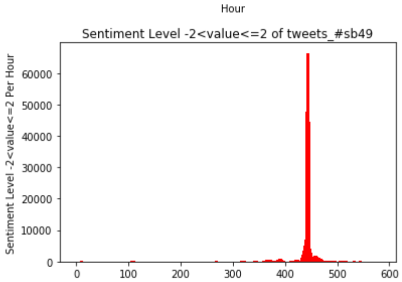

Sentiment Level 10<value<=50 of tweets_#patriots
Sentiment Level 10<value<=50 Per Hour
Hour

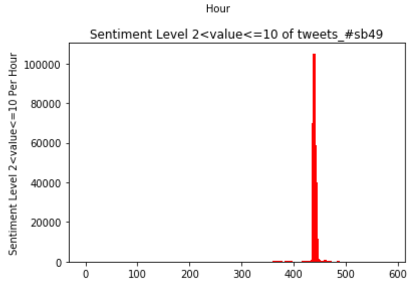

Sentiment Level 50<value of tweets_#patriots
Sentiment Level 50<value Per Hour
Hour

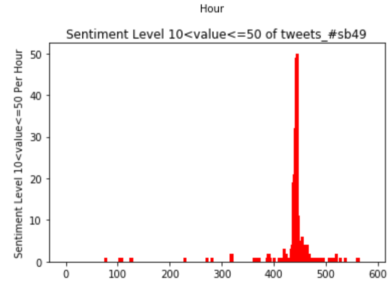

Sentiment Level value<=-50 of tweets_#sb49
Sentiment Level value<=-50 Per Hour
Hour

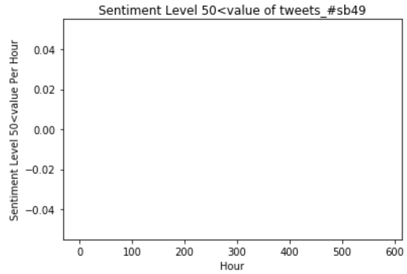

Sentiment Level -50<value<=-10 of tweets_#sb49
Sentiment Level -50<value<=-10 Per Hour
Hour

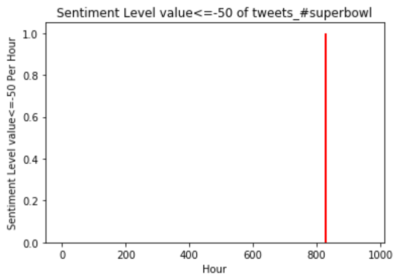

Sentiment Level -10<value<=-2 of tweets_#sb49
Sentiment Level -10<value<=-2 Per Hour
Hour

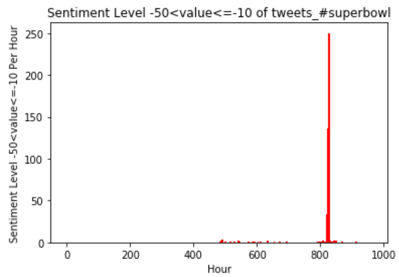

Sentiment Level -2<value<=2 of tweets_#sb49
Sentiment Level -2<value<=2 Per Hour
Hour

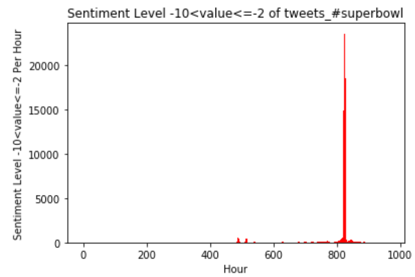

Sentiment Level 2<value<=10 of tweets_#sb49
Sentiment Level 2<value<=10 Per Hour
Hour

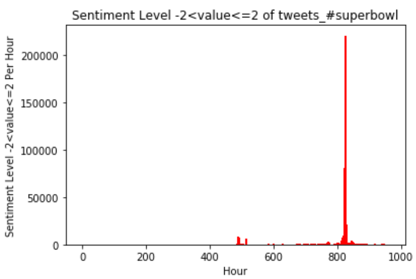

Sentiment Level 10<value<=50 of tweets_#sb49
Sentiment Level 10<value<=50 Per Hour
Hour

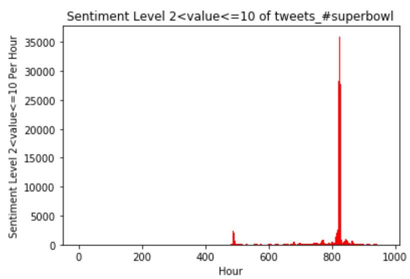


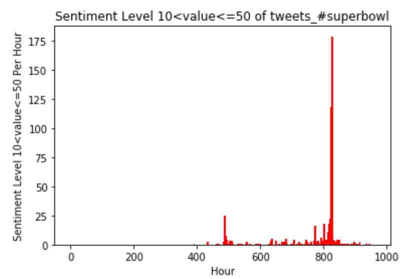


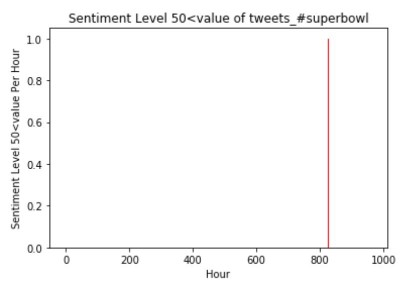






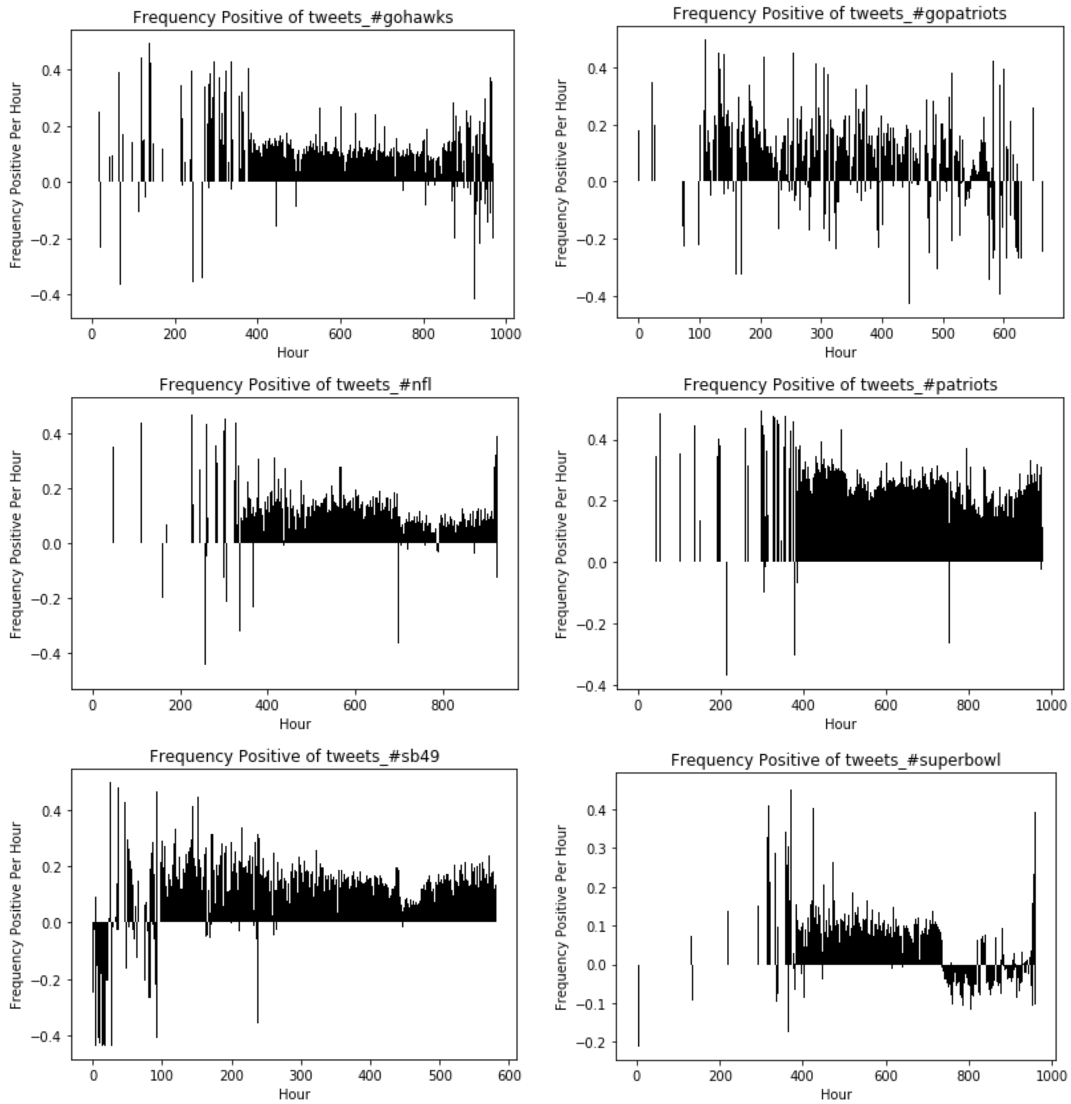






Use TextBlob to calculate the sentiment values:

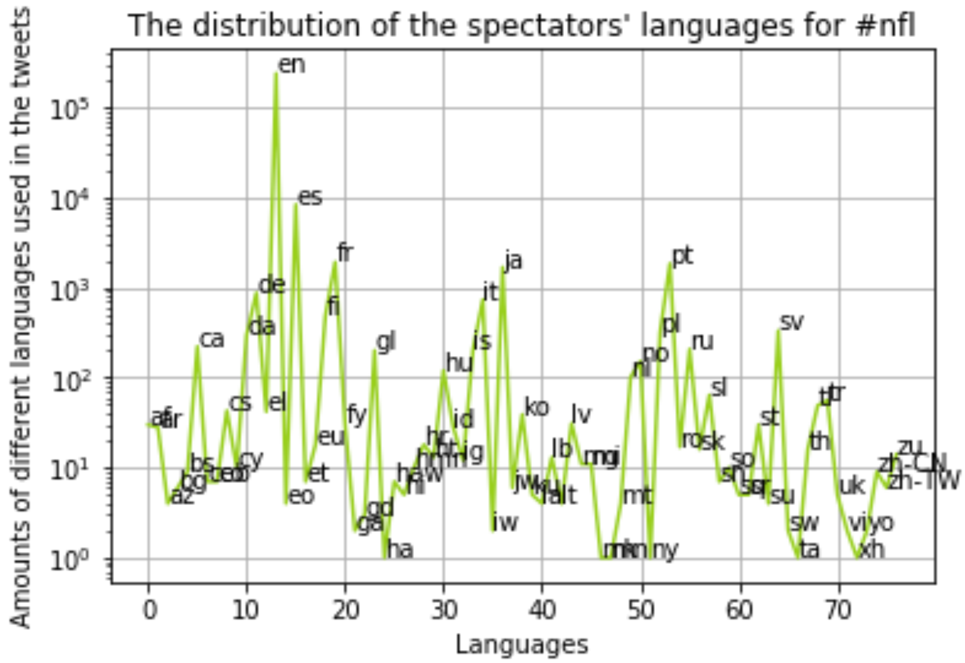
Another thing we have got is a library usually used in text processing written in Python named Textblob. With this library, we can simply perform many natural language processing tasks, such as part-of-speech tagging, nominal component extraction, sentiment analysis, text translation, and so on. In the official documentation, we learned more about how to use it. The function we need is textblob.sentiments module which contains two sentiment analysis implementations, PatternAnalyzer based on the pattern library and NaiveBayesAnalyzer, an NLTK classifier trained on a movie reviews corpus. In the project, we use the latter one to perform the sentiment analysis. The sentiment property returns a named tuple of the form Sentiment(classification, p_pos, p_neg). The p_pos and p_neg are the probability of being positive and being negative respectively.



In this analysis method, we consider the summation of probabilities of positive sentiment in a period as the showing frequency of the positive tweets. We can obviously obtain the observations that mainly there are positive tweets with low frequency, but the peaks, namely the local maximums both for positive and negative sentiment in tweets will show at some certain time. Although random the peaks seem to be, the inner cause may be the events in the football field.

Use TextBlob to detect the national distribution of the tweeters:

Another interesting part of this library is the language detection, which can detect the language of a certain text using the Google Translate API. We are excited about making use of this function to obtain the language distribution of the tweeters.



We use the language detection to tell what language every tweet is and count the number of every languages. The most languages showing in the tweets are English, Spanish, French, Japanese, Portuguese, Italian, German, which, to some extent, implies that the popularity in different countries.