Q2 Fan Base Prediction

In this part, we predict the regions of the users based on the texts they wrote. In this project, we choose two regions, Washington(WA) and Massachusetts(MA) to test our algorithm. First, we consider all the tweets including #superbowl. And then, we extract the tweets in WA and MA to build our supervised data base. We used the function below to extract the information in the ".txt" file in to a dictionary.

```
twt_sb = open('train/tweets_#superbowl.txt', encoding="utf8")
twt_i      = json.loads(twt_sb.readline())
```

"twt_i" is a dictionanry of all the info in the ith line. The info of location is stored deep inside as shown below. The line with the following location keywords will be sorted into WA or MA.

WA = ['washington'   , 'wash','wa','seattle','kirkland']
MA = ['massachusetts', 'mass','ma','boston' ,'cambridge']

Key	Type	Size	Value
author	dict	9	{'followers':199.0, 'nick':'theun…
citation_date	int	1	1421367835
citation_url	str	1	http://twitter.com/TheUntrainedEy…
firstpost_date	int	1	1419866833
highlight	str	1	I'm so excited the road to #Super…
metrics	dict	6	{'citations':{'data':[...], 'matc…
original_author	dict	8	{'followers':527.0, 'nick':'nwhit…
title	str	1	I'm so excited the road to #Super…
tweet	dict	26	{'coordinates':NoneType, 'source'…
type	str	1	retweet:native
url	str	1	http://twitter.com/NWhite1005/sta…

Key	Type	Size	Value
in_reply_to_user_id_str	NoneType	1	NoneType object of…
lang	str	1	en
place	NoneType	1	NoneType object of…
possibly_sensitive	bool	1	False
retweet_count	int	1	0
retweeted	bool	1	False
source	str	1	<a href="http://tw…
text	str	1	I'm so excited the…
timestamp_ms	str	1	1419866833133
truncated	bool	1	False
user	dict	38	{'profile_text_col…

Key	Type	Size	Value
friends_count	int	1	190
geo_enabled	bool	1	True
id	int	1	2374402878
id_str	str	1	2374402878
is_translator	bool	1	False
lang	str	1	en
listed_count	int	1	6
location	str	1	#Seahawks…
name	str	1	Nick White
notifications	NoneType	1	NoneType …
profile_background_color	str	1	131516

The "twt_i['title']" is our input data, and the locations, "twt_i['tweet']['user']['location']" is our output class, i.e. the "X and Y." We did this to all 1348767 lines, and a set of supervised dataset was generated. The function that operates this is shown below. We marked "WA" as "0" and "MA" as "1."

```
def get_location(all_lines):
    twt_list      = []
    location_list = []
    for i, line in enumerate(all_lines):
        twt_i      = json.loads(line)
        location_i = twt_i['tweet']['user']['location']
        loc = token.tokenize(location_i.lower())
        if  (any(x in loc for x in WA)):
            twt_list.append(twt_i['title'])
            location_list.append(0)
        elif (any(x in loc for x in MA)):
            twt_list.append(twt_i['title'])
            location_list.append(1)
        else:
            pass
    return twt_list,location_list
```

The “X” so far is a list of strings. We want to transform it to a numerical matrix. In order to get a good feature, we have to do some language processing.

We have done lemmatization to remove inflectional endings only and to return the base or dictionary form of a word. After this, we transformed the processed list to a numerical matrix by TFIDF. Finally, to be efficient, we used the SVD to reduce the length of the feature. Once we reach to this step, we are ready to use any binary models to make predictions. The classifiers like SVM has no hyperparameters, so we did not use cross validation at that step.

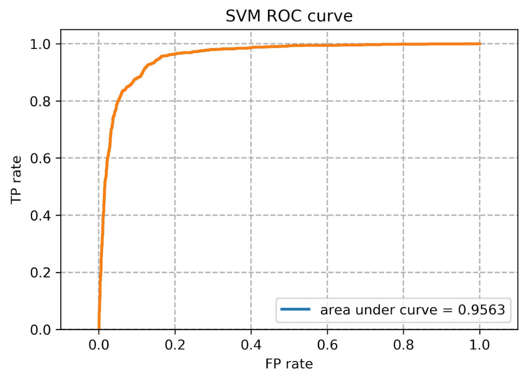
Accuracies, precisions, recalls, confusion matrices and the ROC curves are reported below for each classifier. See next page.

SVM soft

Accuracy = 0.904449565679844

	precision	recall	f1-score	support
WA	0.92	0.95	0.93	3992
MA	0.87	0.79	0.83	1649
avg / total	0.90	0.90	0.90	5641

confusion matrix
[[3800 192]
 [347 1302]]



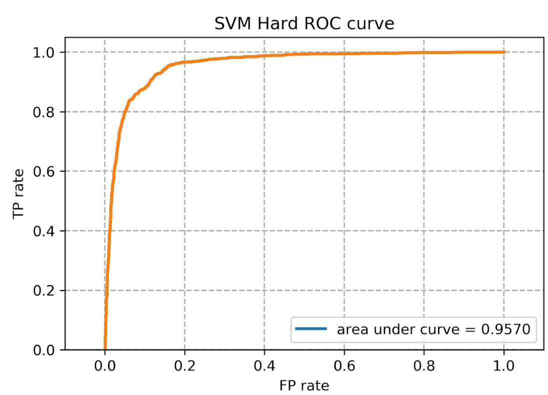
The result was quite good at first glance. In order to have better result, we can consider penalty on weights to prevent overfitting.

SVM hard with Penalty parameter = 20

Accuracy = 0.90356319801453 64

	precision	recall	f1-score	support
WA	0.91	0.95	0.93	3992
MA	0.87	0.78	0.83	1649
avg / total	0.90	0.90	0.90	5641

confusion matrix
[[3805 187]
 [357 1292]]



The accuracy went lower, but the AUC increased. That means this was a better model.

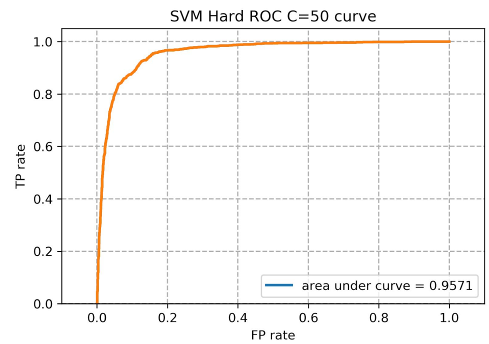
SVM hard with Penalty parameter = 50

Accuracy = 0.904449565679844

	precision	recall	f1-score	support
WA	0.92	0.95	0.93	3992
MA	0.87	0.79	0.83	1649
avg / total	0.90	0.90	0.90	5641

confusion matrix

[[3801 191]
[348 1301]]



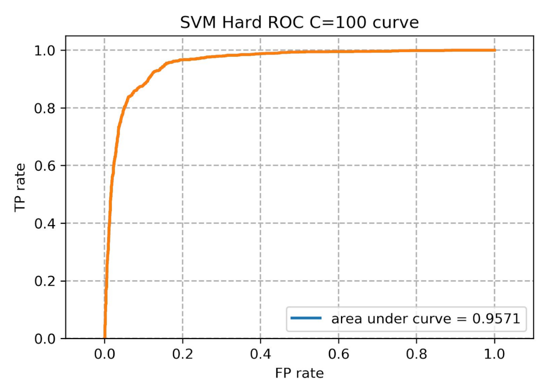
SVM hard with Penalty parameter = 100

Accuracy = 0.9046268392129055

	precision	recall	f1-score	support
WA	0.92	0.95	0.93	3992
MA	0.87	0.79	0.83	1649
avg / total	0.90	0.90	0.90	5641

confusion matrix

[[3801 191]
[347 1302]]



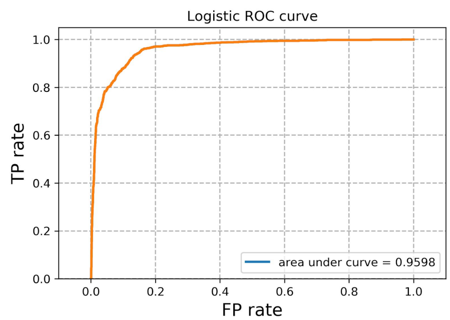
The accuracies and area under curve are simular. They are both good models.

Logistic

Accuracy = 0.9053359333451516

	precision	recall	f1-score	support
WA	0.92	0.95	0.93	3992
MA	0.87	0.79	0.83	1649
avg / total	0.90	0.91	0.90	5641

confusion matrix
[[3798 194]
[340 1309]]



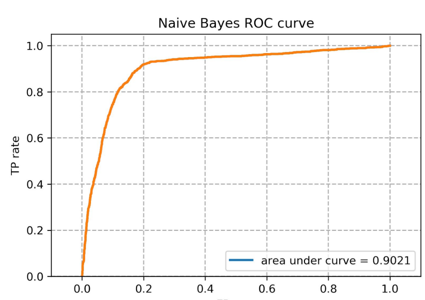
Again, the accuracy was simular, and the area under curve slight increased. It is good model. Something even better is that, it takes a lot of time to train the SVM. However, the Logistic Regression can be trained so fast.

Naive Bayes

Accuracy = 0.8431129232405602

	precision	recall	f1-score	support
WA	0.86	0.93	0.89	3992
MA	0.78	0.64	0.71	1649
avg / total	0.84	0.84	0.84	5641

confusion matrix
[[3698 294]
[591 1058]]



The Naive Bayes was not as good as the others. Both accuracy and the area under curve.