ECE219 Large-Scale Data Mining: Models and Algorithms **(Due: Sun, 03/22/20)**

Project 5: Application - Twitter Data

Instructor: Vwani Roychowdhury Zhaoxi Yu(005432230) & Yuhao Yin(104880239)

1 Introduction

A useful practice in social network analysis is to predict future popularity of a subject or event. Twitter, with its public discussion model, is a good platform to perform such analysis. With Twitter's topic structure in mind, the problem can be stated as: knowing current (and previous) tweet activity for a hashtag, can we predict its tweet activity in the future? More specifically, can we predict if it will become more popular and if so by how much? In this project, we will try to formulate and solve an instance of such problems.

The available Twitter data is collected by querying popular hashtags related to the 2015 Super Bowl spanning a period starting from 2 weeks before the game to a week after the game. We will use data from some of the related hashtags to train a regression model and then use the model to make predictions for other hashtags.

2 Popularity Prediction

2.1 A first look at the data

Download the training tweet data. The data consists of 6 text files, each one containing tweet data from one hashtag as indicated in the filenames.

QUESTION 1. Report the following statistics for each hashtag, i.e. each file:

- Average number of tweets per hour.
- Average number of followers of users posting the tweets per tweet. (to make it simple, we average over the number of tweets; if a users posted twice, we count the user and the user's followers twice as well)
- Average number of retweets per tweet.

Remark. Statistics for each hashtag are shown in Table 1.

hashtag	average number of tweets	average number of followers	average number of retweets
`#gohawks`	292.4879	2217.9237	2.0132
`#gopatriots`	40.9547	1427.2526	1.4081
`#nfl`	397.0214	4662.3754	1.5345
`#patriots`	750.8943	3280.4636	1.7853
`#sb49`	1276.8571	10374.1603	2.5271
`#superbowl`	2072.1184	8814.9680	2.3912

Table 1: Statistics for Each Hashtag

QUESTION 2. Plot "number of tweets in hour" over time for `#SuperBowl` and `#NFL` (a bar plot with 1-hour bins). The tweets are stored in separate files for different hashtags and files are named as `tweet_[#hashtag].txt`.

Remark. The plot of number of tweets over time for hashtag `#SuperBowl` and `#NFL` are reported in Figure 1 and Figure 2.

Remark 2. From these two plots we can observe that the number of tweets surges around hour 450 for both hashtags.

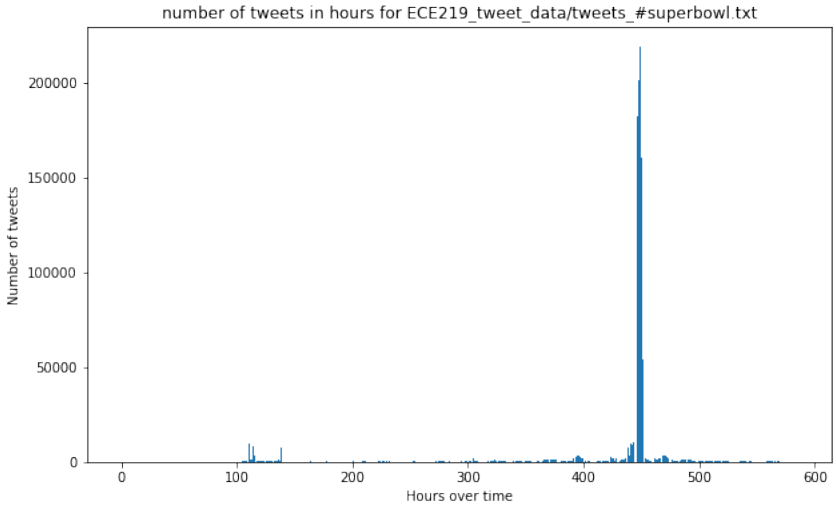


Figure 1: Number of Tweets in Hour for SupoerBowl

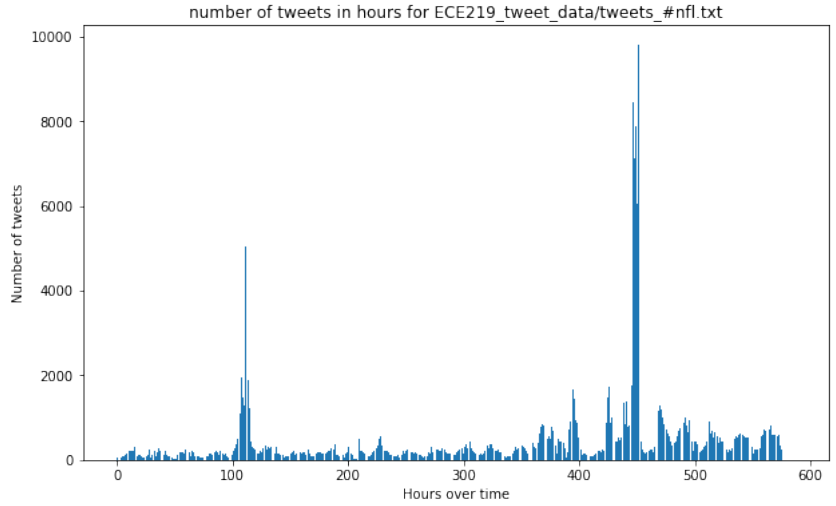


Figure 2: Number of Tweets in Hour for NFL

2.2 Linear regression

Create time windows from the data to extract features. Here, use 1-hour time window (00:00 - 01:00 am, 01:00 - 02:00 am, etc.) and calculate the features in each time window, resulting in `<# of hours>` data points.

For each hashtag data file, 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 are:

- Number of tweets
- Total number of retweets
- 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, train a separate model.

QUESTION 3. For each of your models, report your model's Mean Squared Error (MSE) and R-squared measure. Also, analyse the significance of each feature using the t-test and p-value. You may use the `OLS` in the libarary `statsmodels` in Python.

Remark. For each of the models, the MSE and R-squared measure are reported in Table 2. From the measure results we can get that linear regression works best for hashtag `#superbowl`.

hashtag	MSE	R-squared
`#gohawks`	717636.4421	0.528
`#gopatriots`	30607.5786	0.606
`#nfl`	274661.9677	0.651
`#patriots`	4581686.3228	0.715
`#sb49`	13170997.9538	0.842
`#superbowl`	34137549.2587	0.870

Table 2: MSE and R-squared Measure for Each Hashtag

Remark 2. For the significance each feature, we list the t-test and p-value for each hashtag from Table 3 to Table 8.
We can concluded that for hashtag `#gohawks`, the important features are `Number of tweets`, `Total number of retweets` and `Sum of the number of followers`.
For hashtag `#gopatriots`, the important feature is `Total number of retweets`.
For hashtag `#nfl`, the important features are `Number of tweets`, `Sum of the number of followers` and `Time of the day`.
For hashtag `#patriots`, the important features are `Number of tweets` and `Total number of retweets`.
For hashtag `#sb49`, the important features are `Number of tweets` and `Maximum number of followers`.
For hashtag `#superbowl`, the important features are `Number of tweets`, `Total number of retweets`, `Sum of the number of followers` and `Maximum number of followers`.

feature	t-test	p-value
`Number of tweets`	9.566	3.2751e-20
`Total number of retweets`	-4.414	1.2138e-05
`Sum of the number of followers`	-3.905	1.0560e-04
`Maximum number of followers`	2.276	2.3187e-02
`Time of the day`	1.900	5.7997e-02

Table 3: T-test and P-value for `#gohawks`

feature	t-test	p-value
`Number of tweets`	-0.495	0.6209
`Total number of retweets`	3.087	0.0021
`Sum of the number of followers`	0.767	0.4432
`Maximum number of followers`	-1.385	0.1667
`Time of the day`	0.782	0.4344

Table 4: T-test and P-value for `#gopatriots`

feature	t-test	p-value
Number of tweets	4.736	2.7437e-06
Total number of retweets	-2.654	8.1840e-03
Sum of the number of followers	4.049	5.8518e-05
Maximum number of followers	-2.971	3.0879e-03
Time of the day	3.833	1.4035e-04

Table 5: T-test and P-value for `#nfl`

feature	t-test	p-value
Number of tweets	15.391	4.4097e-45
Total number of retweets	-4.994	7.8344e-07
Sum of the number of followers	1.581	1.1452e-01
Maximum number of followers	0.756	4.5009e-01
Time of the day	1.275	2.0291e-01

Table 6: T-test and P-value for `#patriots`

feature	t-test	p-value
Number of tweets	13.020	3.6840e-34
Total number of retweets	-2.224	2.6540e-02
Sum of the number of followers	0.912	3.6212e-01
Maximum number of followers	4.439	1.0854e-05
Time of the day	-1.199	2.3095e-01

Table 7: T-test and P-value for `#sb49`

2.3 Feature analysis

QUESTION 4. Design a regression model using any features from 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 fitting MSE and significance of features.

Remark. After reading the papers, we choose another 5 features besides the features in Question 3, which are:

- Sum of ranking scores: Sum of ranking scores of all tweets.
- Sum of passivity: Active users often post or retweet tweets following some hashtags. On the contrary, passive users rarely do so unless the topics are attractive enough. The passivity of a user is defined as the reciprocal of average number of tweets posted by this user per day, which is formed as:

$$Psv(u_i) = \frac{N_d(u_i)}{1.0 + N_t(u_i)} \tag{1}$$

  where $N_d(u_i)$ denotes the number of days since the user account was created, and $N_t(u_i)$ denotes the total number of tweets posted by this user.
- Total number of unique users: The unique number of users who posted tweets containing the hashtag.
- Total number of unique authors: The unique number of authors who posted tweets containing the hashtag. The number of unique users and authors can be used to recognize those hashtags automatically posted by some fake accounts.
- Total number of user mentions: Mention is a directional sharing behavior in Twitter. Messages can be shared to a designated user using @ as the prefix of the user's name. If a user was mentioned in a tweet with a hashtag, he probably took part in the topic, especially when this mention came from his friends.

feature	t-test	p-value
Number of tweets	24.048	3.5874e-89
Total number of retweets	-4.847	1.6126e-06
Sum of the number of followers	-20.110	1.2179e-68
Maximum number of followers	11.127	3.3709e-26
Time of the day	-2.520	1.1993e-02

Table 8: T-test and P-value for #superbowl

Remark. For each of the models, the MSE and R-squared measure are reported in Table 9. Compare the results in Table 2 and Table 9, we can get the result that more features can help us to fit the model better.

hashtag	MSE	R-squared
#gohawks	414339.5843	0.727
#gopatriots	14024.4912	0.819
#nfl	213428.2676	0.729
#patriots	3440833.3536	0.786
#sb49	8694964.9810	0.895
#superbowl	11597363.0835	0.956

Table 9: MSE and R-squared Measure for Each Hashtag

Remark 2. The p-values of features for all hashtags are reported in Table 10. The top features with the smallest p-values vary from hashtag to hashtag. The most common top feature is Total number of user mentions.

	#gohawks	#gopatriots	#nfl	#patriots	#sb49	#superbowl
Number of tweets	3.267e-28	3.027e-05	1.251e-03	1.780e-15	5.406e-02	1.026e-73
Total number of retweets	2.226e-02	2.736e-02	4.723e-01	2.849e-02	4.036e-01	2.603e-14
Sum of the number of followers	9.530e-08	5.524e-03	3.056e-01	1.652e-22	1.499e-11	5.520e-01
Maximum number of followers	1.557e-01	1.721e-04	3.022e-01	1.324e-13	1.968e-03	7.423e-01
Time of the day	5.212e-01	1.084e-01	1.480e-01	1.511e-01	1.342e-01	2.300e-03
Sum of ranking scores	1.949e-29	4.684e-07	5.169e-02	4.724e-13	1.144e-01	5.496e-72
Sum of passivity	3.860e-04	3.470e-23	2.629e-03	4.096e-01	2.403e-10	2.591e-20
Total number of unique users	8.462e-20	2.535e-05	1.676e-06	1.327e-02	1.121e-01	1.363e-65
Total number of unique authors	5.445e-21	1.640e-05	5.408e-07	2.239e-02	1.047e-01	3.547e-69
Total number of user mentions	1.357e-32	1.271e-48	4.587e-13	3.781e-26	8.106e-21	4.825e-65

Table 10: P-value of Each Feature for Each Hashtag

QUESTION 5. For each of the top 3 features (i.e. with the smallest p-values) in your measurements, draw a scatter plot of predictant (number of tweets for next hour) versus value of that feature, using all the samples you have extracted, and analyze it.
Do the regression coefficients agree with the trends in the plots? If not, why?

Remark. From the scatter plots in Table 11 we can observe that a relatively linear relationship between the top features and the predictant. Thus these features have high correlation with the number of tweets in next hour.

Remark 2. Not all the regression coefficients agree with the trends in the plots. This is reasonable because we did not use the mono feature. Thus, the regression coefficient of each feature is the result of inter-correlation with other features, which may result in a negative regression coefficient even though the slope of scatter plot is positive.

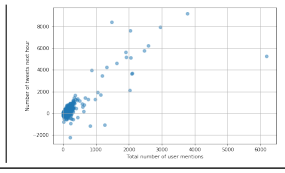
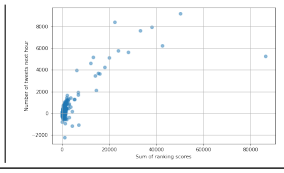
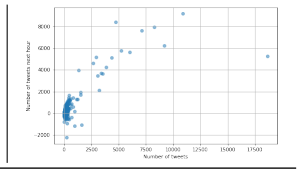
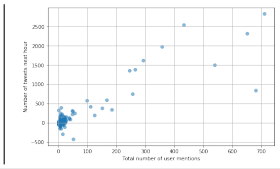
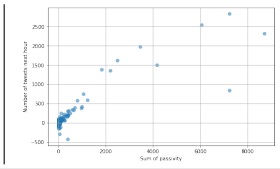
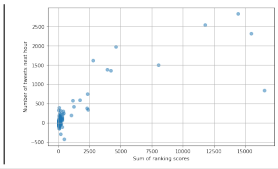
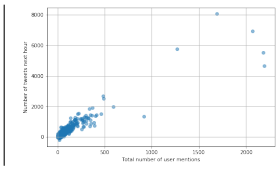
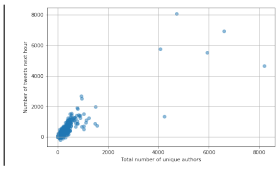
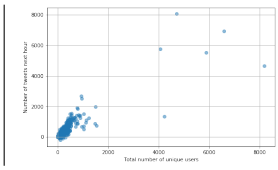
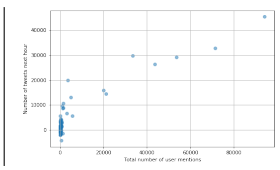
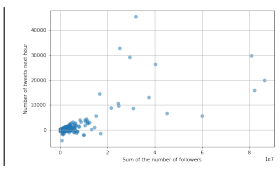
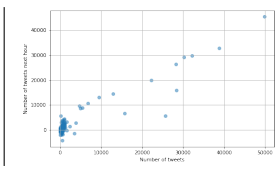
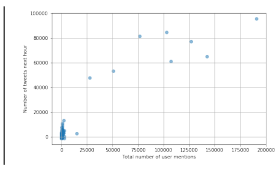
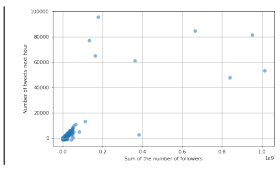
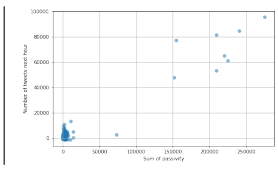
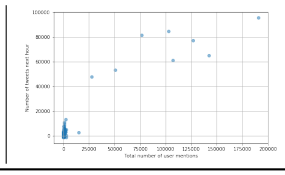
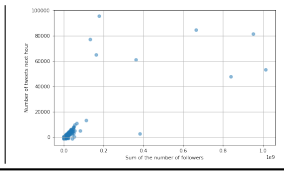
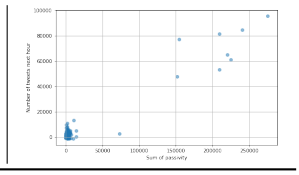
#gohawks			
#gopatriots			
#nfl			
#patriots			
#sb49			
#superbowl			

Table 11: Scatter Plots of Predictant vs Value of Top 3 Features for Each Hashtag

2.4 Piece-wise linear regression

Since we know the Super Bowl's date and time, we can create different regression models for different periods of time. First, when the hashtags haven't become very active; second, their active period; and third, after they pass their high-activity time.

QUESTION 6. We define three time periods and their corresponding window length as follows:

1. Before Feb. 1, 8:00 a.m.: 1-hour window
2. Between Feb. 1, 8:00 a.m. and 8:00 p.m.: 5-minute window
3. After Feb. 1, 8:00 p.m.: 1-hour window

For each hashtag, train 3 regression models, one for each of these time periods (the times are all in PST). Report the MSE and R-squared score for each case.

Remark. The MSE and R-squared scores of three time periods for each hashtag are listed in Table 12. Compare the results in Table 9 and Table 12, we can conclude that the piece-wise linear regression works better than linear regression during and after the active period for most hashtags.

Hashtag	Time Period 1		Time Period 2		Time Period 3	
	MSE	R-squared	MSE	R-squared	MSE	R-squared
#gohawks	352857.9560	0.676	70828.0786	0.765	1336.9915	0.892
#gopatriots	1157.8918	0.730	12979.8407	0.670	17.7720	0.912
#nfl	63690.5813	0.698	16445.3582	0.926	16377.1812	0.948
#patriots	316359.2049	0.639	655033.0345	0.893	8303.0991	0.929
#sb49	5553.7907	0.909	1125342.8731	0.963	21858.4152	0.956
#superbowl	478668.7082	0.547	4888804.8501	0.953	80450.6889	0.932

Table 12: MSE and R-squared Measure for Each Hashtag in Three Time Periods

QUESTION 7. 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 time window on the aggregated data. Perform the same evaluations on your combined model and compare with models you trained for individual hashtags.

Remark. The MSE and R-squared scores of three time periods for aggregated data are listed in Table 13. Compare the results in Table 12 and Table 13, we can conclude that the aggregated data has no big contribution to improve the performance of piece-wise linear regression. That might because the top features for different hashtags are also different. Thus, the aggregated data works worse than individual hashtags.

Time Period 1		Time Period 2		Time Period 3	
MSE	R-squared	MSE	R-squared	MSE	R-squared
4186900.7223	0.558	14483087.0756	0.952	281104.9010	0.959

Table 13: MSE and R-squared Measure for Aggregated Data in Three Time Periods

2.5 Nonlinear regressions

2.5.1 Ensemble methods

In this part, we use `RandomForestRegressor` and `GradientBoostingRegressor` from `sklearn` as two examples of ensemble regressors. Still use the aggregated data in this part.

QUESTION 8. Use grid search to find the best parameter set for `RandomForestRegressor` and `GradientBoostingRegressor` respectively. Use the following `param_grid`

```
{
    'max_depth': [10, 30, 50, 70, 100, 200, None],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [1, 2,3, 4],
    'min_samples_split': [2, 5, 10],
    'n_estimators': [200, 400, 600, 800, 1000,
                     1200, 1400, 1600, 1800, 2000]
}
```

Set `cv = KFold(5, shuffle=True), scoring='neg_mean_squared_error'` for the grid search. Analyze the result of the grid search. Do the test errors from cross-validation look good? If not, please explain the reason.

Remark. For `RandomForestRegressor`, we report the top 5 hyper-parameter settings of the grid search result in Table 14,

Remark 2. Similarly, we also report the top 5 hyper-parameter settings for `GradientBoostingRegressor` in Table 15.

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-2.028581e+08	10	sqrt	2	2	800
-2.034092e+08	10	sqrt	2	2	600
-2.041853e+08	10	sqrt	2	5	800
-2.042903e+08	70	sqrt	2	5	800
-2.042903e+08	200	sqrt	2	5	800

Table 14: Grid Search Result for RandomForestRegressor on Aggregated Data

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-2.698456e+08	10	sqrt	3	10	200
-2.721131e+08	30	sqrt	3	10	200
-2.730286e+08	100	sqrt	3	10	200
-2.730286e+08	50	sqrt	3	10	200
-2.730286e+08	70	sqrt	3	10	200

Table 15: Grid Search Result for GradientBoostingRegressor on Aggregated Data

Remark 3. The testing errors from cross-validation are enormously large for two ensemble methods reported above. Possible reasons may include that blindly aggregating tweet features from all hashtags will not improve the overall model performances, since underlying relationships between target and predictors might vary from one hashtag to another and this inference is validated from the top 3 features in respective OLS models on the data of each hashtag in Question 5.

Consequently, non-linear ensemble methods won't be able to decrease the testing errors, since there isn't really a relation for them to capture and complex models will easily overfit the training data even though we fine-tune many hyper-parameters within.

QUESTION 9. Compare the best estimator you found in the grid search with OLS on the entire dataset.

Remark. Comparisons among non-linear ensemble methods (after fine-tuning) and linear regression model is summarized in Table 16.

Prediction Models	Test Score
Linear Regression	-2.028581e+08
Random Forest	-2.698456e+08
Gradient Boosting	-3.973045e+08

Table 16: Comparisons Among Different Models on Aggregated Data

We conclude that both random forest and gradient boosting ensemble methods perform slightly better than linear regression model in terms of testing MSE, however all the testing MSE is unreasonably large mainly due to the complexity of aggregated tweets data. In order to make any of the learning algorithms to work, it's worth trying splitting the aggregated data into smaller datasets based on the time period, since the number of tweets can vary a lot before, at or even after the Super Bowl.

QUESTION 10. For each time period described in Question 6, perform the same grid search above for `GradientBoostingRegressor` (with corresponding time window length). Does the cross-validation test error change? Are the best parameter set you find in each period agree with those you found above?

Remark. For the first time period, which is before Feb. 1, 8:00 a.m., we report the top 5 hyper-parameter settings for `GradientBoostingRegressor` in Table 17 using the same grid search above.

Remark 2. For the second time period, which is between Feb. 1, 8:00 a.m. and 8:00 p.m., we also report the top 5 hyper-parameter settings for `GradientBoostingRegressor` in Table 18 using the same grid search above.

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-650067.208973	10	sqrt	3	2	2000
-650067.208973	10	sqrt	3	5	2000
-650067.235187	10	sqrt	3	2	1800
-650067.235187	10	sqrt	3	5	1800
-650067.442207	10	sqrt	3	5	1600

Table 17: Grid Search Result for GradientBoostingRegressor on the First Time Period

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-1.345699e+07	100	sqrt	1	10	200
-1.345699e+07	50	sqrt	1	10	200
-1.345699e+07	70	sqrt	1	10	200
-1.345699e+07	None	sqrt	1	10	200
-1.345699e+07	200	sqrt	1	10	200

Table 18: Grid Search Result for GradientBoostingRegressor on the Second Time Period

Remark 3. For the last time period, which is after Feb. 1, 8:00 p.m., we also report the top 5 hyper-parameter settings for `GradientBoostingRegressor` in Table 19 using the same grid search above.

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-122158.054498	10	sqrt	4	10	200
-122176.199398	10	sqrt	4	10	1800
-122176.199398	10	sqrt	4	10	2000
-122176.199510	10	sqrt	4	10	1600
-122176.222168	10	sqrt	4	10	1400

Table 19: Grid Search Result for GradientBoostingRegressor on the Third Time Period

Remark 4. From these grid search results, we figure out that the test error from cross-validation drops significantly for models on all three time periods compared to the entire aggregated dataset, indicating that the task of prediction number of tweets can be better accomplished if we separate the data based on different time periods. The best parameter set we find in each period differs greatly from those we found above.

2.5.2 Neural Network

QUESTION 11. Now try to regress the aggregated data with `MLPRegressor`. Try different architectures (i.e. the structure of the network) by adjusting hidden layer sizes. You should try at least 5 architectures with various numbers of layers and layer sizes. Report the architectures you tried, as well as its MSE of fitting the entire aggregated data.

Remark. In order to find the optimal architecture for the neural network, we perform the grid search on a two-layer perceptron, where the number of units in both hidden layers can vary from 1 to 100. Results of top 5 structures together with their corresponding test scores are summarized in Table 20.

Remark 2. As expected, due to the complexity of this Super Bowl dataset over time, fitting a neural network on the entire data leads to an enormous testing MSE even after fine-tuning the network architecture. Hence, further data manipulation is required in order to better predict the number of tweets for the next 1-hour window.

QUESTION 12. Use `StandardScaler` to scale the features before feeding it to `MLPRegressor` (with the best architecture you got above). Does its performance increase?

Remark. Adopting the best network architecture from above, which is a two-layer perceptron, with the first layer containing 11 units and second layer 15 units, we scale the features before feeding it to

mean_test_score	hidden_layer_sizes
-2.136831e+08	(11,15)
-2.234267e+08	(33,16)
-2.421104e+08	(83,49)
-2.433288e+08	(50,17)
-2.531110e+08	(31,19)

Table 20: Grid Search Result for Neural Network on Aggregated Data

the neural network. The testing MSE from 5-fold cross-validation is **1.651838e+08**, which is smaller compared to that without feature standardization.

Consequently, it is safe to conclude that feature scaling indeed helps improve the model performance to some extent, since it will balance out the influence of unequal variance among features and stabilize the learning process, e.g. gradient descent.

QUESTION 13. Using grid search, find the best architecture (for scaled data) for each period (with corresponding window length) described in Question 6.

Remark. For the first time period, which is before Feb. 1, 8:00 a.m., we report the top 5 hyper-parameter settings for a two-layer perceptron, where the number of units in both hidden layers can vary from 1 to 100, in Table 21.

mean_test_score	hidden_layer_sizes
-471580.467828	(58, 7)
-479726.078631	(85, 39)
-480257.477739	(15, 96)
-480838.856568	(74, 4)
-481419.161602	(15, 68)

Table 21: Grid Search Result for Neural Network on the First Time Period

Remark 2. For the second time period, which is between Feb. 1, 8:00 a.m. and 8:00 p.m., we also report the top 5 hyper-parameter settings for `MLPRegressor` in Table 22 using the same grid search above.

mean_test_score	hidden_layer_sizes
-1.001959e+07	(68, 90)
-1.013312e+07	(71, 81)
-1.015468e+07	(100, 48)
-1.017348e+07	(47, 97)
-1.017736e+07	(52, 95)

Table 22: Grid Search Result for Neural Network on the Second Time Period

Remark 3. For the last time period, which is after Feb. 1, 8:00 p.m., we also report the top 5 hyper-parameter settings for `MLPRegressor` in Table 23 using the same grid search above.

Remark 4. Comparing both gradient boosting and neural network, we find out that the fine-tuned model performances are very similar for all three different time periods. Moreover, all their testing errors drop dramatically, especially for the first and third time periods, compared to that trained on the entire dataset.

2.6 Using 6x window to predict

Download the test data. Each file in the test data contains a hashtag's tweets from a 6xwindow-length time range. Fit a model on the aggregate of the training data for all hashtags, and predict the number of tweets in the next hour for each test file. The file names consist of sample number followed by the period number the data is from.

mean_test_score	hidden_layer_sizes
-163388.524527	(4, 56)
-169701.272932	(4, 64)
-169870.640562	(4, 60)
-170015.454647	(10, 27)
-170408.290910	(4, 54)

Table 23: Grid Search Result for Neural Network on the Third Time Period

For example, a file named `sample0_period1.txt` contains tweets in a sample 6-hour window that lies in the 1st time period described in Question 6, while a file named `sample0_period2.txt` contains tweets in a sample 30-min window that lies in the 2nd time period. One can be creative here, and use the data from all previous 6-hour windows for making more accurate predictions (as opposed to using features from the previous hour only).

QUESTION 14. Report the model you use. For each test file, provide your predictions on the number of tweets in the next time window. **Note:** Test data should not be used as a source for training. You are not bounded to only linear models. You can find your best model through cross validation of your training data.

Remark. Based on the grid search results in preceding questions, we figure out that both ensemble methods (specifically gradient boosting) and neural network share very similar performances on predicting the number of tweets based on three different time periods, and both non-linear models dominate the linear regression model. Therefore, when using 6x window to predict the number of tweets, we will mainly focus on non-linear models, **gradient boosting regressor** in particular, and perform grid search through cross-validation to find the best architecture of that.

Remark 2. Since here we are using 6x window for prediction, the definition of input features will be slightly different from before, where any summary statistic (e.g. maximum or summation) will be based on the 6x time window. For example, the first feature, which is the total number of tweets, will be the total number of tweets in preceding 6x time window, instead of just one. Also, the fourth feature, the maximum number followers, will be the maximum number of followers among users posting in preceding 6x time window. All the other features can be manipulated in a similar fashion on both training and testing data.

Remark 3. The grid search result for the first time period is summarized in Table 24, where we report the top 5 architectures based on the testing score (negative MSE). Argument `param_grid` is identical to that originally defined in Question 8.

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-441855.607608	10	auto	1	10	200
-441877.200609	10	auto	1	10	400
-441878.578758	10	auto	1	10	600
-441878.662452	10	auto	1	10	2000
-441878.662452	10	auto	1	10	800

Table 24: Grid Search Result for GradientBoostingRegressor on the First Time Period (6x Window)

Testing data that belong to the first time period are `sample0_period1.txt`, `sample1_period1.txt` and `sample2_period1.txt`. Predictions on the number of tweets in the next 1-hour window based on the fine-tuned gradient boosting model are **253.98764894, 954.18771702, and 160.28270653** respectively.

Remark 4. The grid search result for the second time period is summarized in Table 25, where we report the top 5 architectures based on the testing score (negative MSE).
Testing data that belong to the second time period are `sample0_period2.txt`, `sample1_period2.txt` and `sample2_period2.txt`. Predictions on the number of tweets in the next 1-hour window based on the fine-tuned gradient boosting model are **2701.04838335, 2372.05084914, and 167.88304972** respectively.

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-1.465649e+07	50	auto	3	5	1600
-1.465649e+07	200	auto	3	2	1600
-1.465649e+07	70	auto	3	5	1600
-1.465649e+07	100	auto	3	2	1600
-1.465649e+07	None	auto	3	5	1600

Table 25: Grid Search Result for GradientBoostingRegressor on the Second Time Period (6x Window)

The first two predictions look a little bit unreasonable, since the testing error of the second time period is significantly larger than the other two.

Remark 5. The grid search result for the third time period is summarized in Table 26, where we report the top 5 architectures based on the testing score (negative MSE).

test_score	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
-191244.340279	None	auto	4	5	2000
-191244.340279	70	auto	4	2	2000
-191244.340279	100	auto	4	5	2000
-191244.340279	30	auto	4	2	2000
-191244.340279	50	auto	4	2	2000

Table 26: Grid Search Result for GradientBoostingRegressor on the Third Time Period (6x Window)

Testing data that belong to the last time period are `sample0_period3.txt`, `sample1_period3.txt` and `sample2_period3.txt`. Predictions for the number of tweets in the next 1-hour window based on the fine-tuned gradient boosting model are **232.22204549, 249.63951101, and 112.07556901** respectively.

3 Fan Base Prediction

The textual content of a tweet can reveal some information about the author. For instance, users tweeting on a topic may have opposing views about it. In particular, tweets posted by fans of different teams during a sport game describe similar events in different terms and sentiments. Recognizing that supporting a sport team has a lot to do with the user location, we try to use the textual content of the tweet posted by a user to predict their location. In order to make the problem more specific, let us consider all the tweets including `#superbowl`, posted by the users whose specified location is either in the state of Washington (not D.C.!) or Massachusetts.

QUESTION 15.

1. Explain the method you use to determine whether the location is in Washington, Massachusetts or neither. Only use the tweets whose authors belong to either Washington or Massachusetts for the next part.
2. 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 learned in project 1). Try different classification algorithms (at least 3). For each, plot ROC curve, report confusion matrix, and calculate accuracy, recall and precision.

Remark. To find out the tweets whose authors belong to Washington, we go to the location field of each tweet (`json_object['tweet']['user']['location']`) and decide whether the location contains any of the following substrings: {`'Washington', 'Seattle', 'WA'`}. If it is true, then we extract the tweet content together with the location for later textual data classification analysis.

Similarly, to find out the tweets whose authors belong to Massachusetts, the corresponding substrings list is {`'Massachusetts', 'Boston', 'MA'`}.

Remark 2. Before training, we first clean the textual data by lemmatizing each word token in the tweet content. After that, we calculate the tf-idf metrics of the textual content using `TfidfVectorizer`

from `sklearn.feature_extraction.text` package. Last but not least, we perform dimension reduction through singular value decomposition on the tf-idf metrics as our final input features.

Remark 3. Since the task here is to predict the location of the author (Washington or Massachusetts) purely based on the textual content of the tweet, we are dealing with a binary classification problem. Three classification learning algorithms considered here are logistic regression, random forest classifier, and gradient boosting classifier, and we will report their respective model performance on the left-out testing data.

First off, we perform a grid search to find out the best hyper-parameter settings for logistic regression, random forest and gradient boosting respectively. For logistic regression, we are grid-searching both penalty category and regularization strength. While for both random forest and gradient boosting, due to the large volume of this twitter dataset, we will limit our search scope and mainly focus on one argument `max_depth`. The top 5 hyper-parameter settings through cross-validation are reported in Table 27 to Table 29.

mean_test_score	param_C	param_penalty
0.724697	100	l2
0.724651	10	l2
0.724651	1000	l2
0.723890	1	l2
0.719582	0.1	l2

Table 27: Grid Search Result for Logistic Regression on Location Prediction

mean_test_score	max_depth
0.724766	30
0.722623	70
0.722623	100
0.722623	200
0.722623	None

Table 28: Grid Search Result for Random Forest on Location Prediction

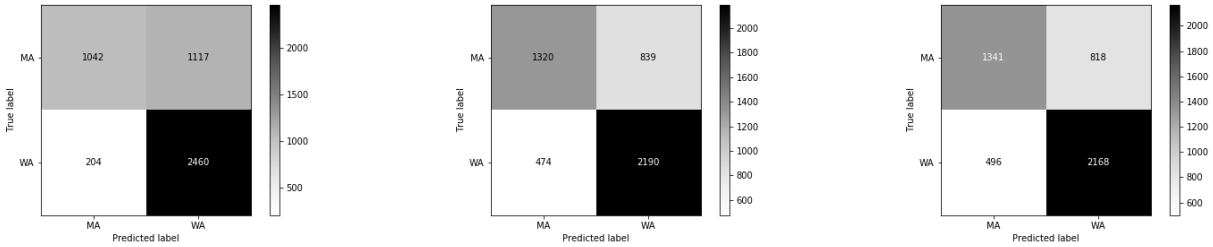
mean_test_score	max_depth
0.721978	10
0.698132	30
0.687141	70
0.686980	100
0.686980	200

Table 29: Grid Search Result for Gradient Boosting on Location Prediction

From there, we further predict the location of author for the left-out testing textual data, and calculate accuracy, recall, precision, F-1 score, as well as plotting the confusion matrices for three different classification algorithms as follows.

	Logistic Regression	Random Forest	Gradient Boosting
Accuracy	0.7261	0.7278	0.7278
Recall	0.9234	0.8221	0.8221
Precision	0.6877	0.7230	0.7230
F-1 Score	0.7883	0.7694	0.7694

Table 30: Classification Measures for Three Different Classification Algorithms



Logistic Regression Random Forest Gradient Boosting

Figure 3: Confusion Matrices for Three Different Classification Algorithms

Last but not least, we also report the ROC curves for logistic regression, random forest classifier and gradient boosting classifier in one figure for better comparison.

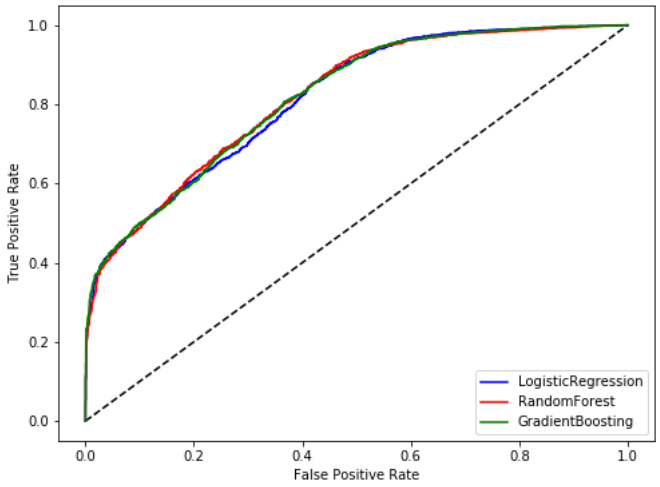


Figure 4: ROC Curves for Three Different Classification Algorithms

Remark. In summary, logistic regression performs worst in terms of AUC score, while random forest and gradient boosting share similar prediction performance.

4 Define Your Own Project

QUESTION 16. 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 briniging in novelty and full or partial implementation of your new ideas.

Remark. In this part, we would like to perform sentiment analysis on fans of the opponent teams especially when the Super Bowl event was active, to have an overall understanding about how their emotion changes as the game going on.

Remark 2. Since we have no idea about which team the author of a tweet supports, we simply take any tweet with the hashtag `#gohawks` as tweets posted by Seattle Seahawks' fans, and take any tweet with hashtag `#gopatriots` as tweets posted by New England Patriots' fans. Then, we predict the sentiment polarity given the textual content of each tweet using a well-trained predictive model called `TextBlob`. We further aggregate the sentiment polarity using a 1-hour window and calculate the average polarity within each window for tweets posted from Feb. 1, 8:00 a.m. to Feb. 1, 8:00 p.m.. The result is shown in Figure 5.

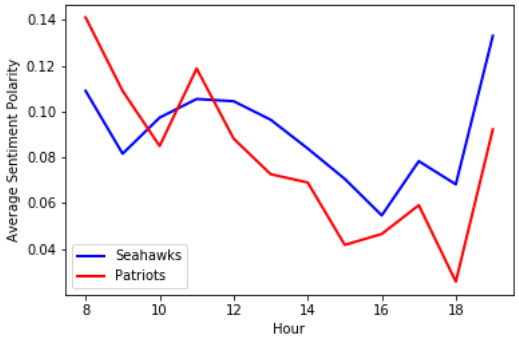
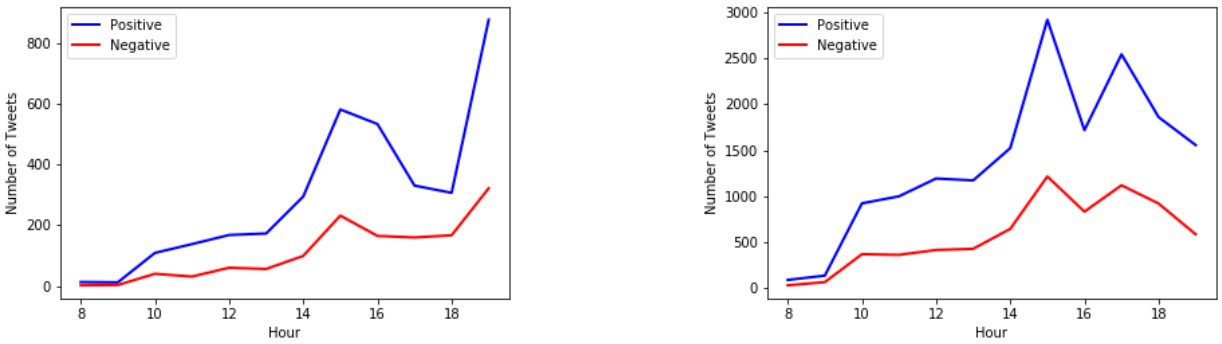


Figure 5: Average Sentiment Polarity Over Time For Fans of Opponent Teams

From this sentiment analysis, we figure out that sentiment polarity of fans of New England Patriots dropped significantly around 6 p.m. compared to fans of Seattle Seahawks. According to the game record in history, we would ascribe this phenomenon to the fact that team New England Patriots lost the third quarter to Seattle Seahawks with 0-10, and their fans must be frustrated at that point consequently. Moreover, polarity of fans from both teams increases dramatically at the end of the game, since scores were very close when the game was approaching to the end, and The Patriots rallied to take a 28-24 lead at the last moment.

Remark 3. We define a tweet to be positive if its corresponding sentiment polarity is positive, and vice versa. Based on this, we can plot the changes of number of positive and negative tweets for fans from both teams in Figure 6 to perform another sentiment analysis.



`#gopatriots` `#gohawks`

Figure 6: Number of Positive and Negative Tweets Over Time For Fans of Opponent Teams