Nicholas Ma
Miles Abel
06/03/2019

Project 5

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

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

There are six different data files, each containing data from one hashtag. Table 1 displays the average number of tweets per hour; the average number of followers posting per tweet; and the average number of retweets per tweet for all six hashtags all rounded to one decimal place. Note #gopatriots has the fewest average number of tweets per hour at 41.0 and #superbowl has the largest with 2072.1. The minimum average followers posting per tweet is 1427.3 from #gopatriots and the maximum is 10374.2 from #sb49. Finally, the minimum number of average retweets per tweet is 1.4 from #gopatriots and the maximum is 2.5 from #sb49.

Hashtag	Avg Number of Tweets per Hour	Avg Number of Followers Posting per Tweet	Avg Number of Retweets per Tweet
#gohawks	292.5	2217.9	2.0
#gopatriots	41.0	1427.3	1.4
#nfl	397.0	4662.4	1.5
#patriots	750.9	3280.5	1.8
#sb49	1276.9	10374.2	2.5
#superbowl	2072.1	8815.0	2.4

Table 1: The Average Number of Tweets per Hour, Followers Posting per Tweet, and Retweets per Tweet for each of the Six Hashtags

Question 2: Plot "number of tweets per 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.

Figure 1 displays the number of tweets per hour for #superbowl, and Figure 2 displays the number tweets per hour for #nfl. There is very little tweet activity for #superbowl until the day the of the superbowl, bins 432 through 450, where there is a massive spike in the number of tweets per hour. #superbowl has 249,584 tweets for bin 449 or the 6 PM PST bin. #nfl experiences a higher level of tweets per hour throughout as compared to #supberbowl but there is still a significant increase in the tweets per hour around

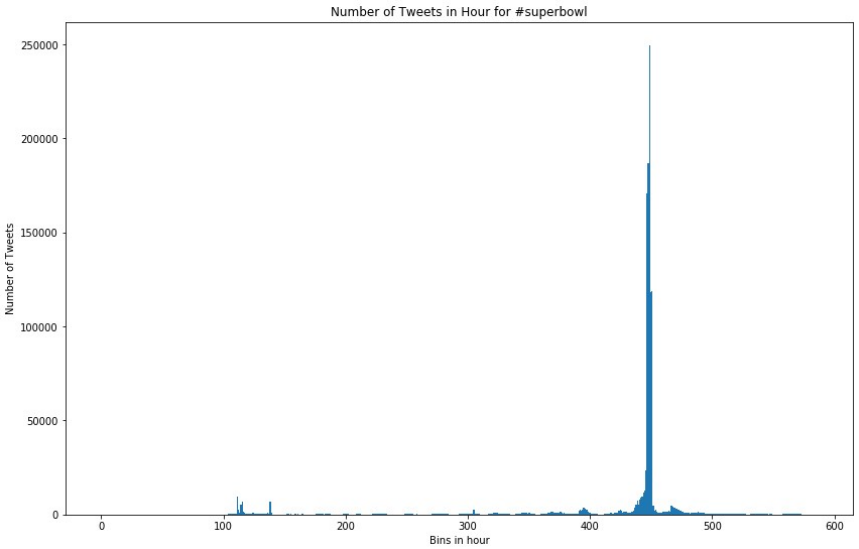


Figure 1: Histogram of the Number of Tweets per Hour for #superbowl

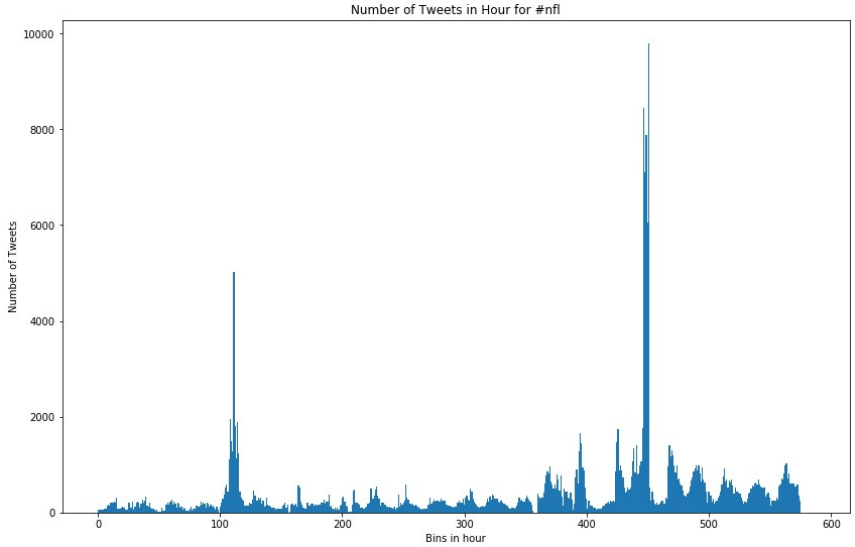


Figure 2: Histogram of the Number of Tweets per Hour for #nfl

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

Table 2 contains the mean squared error (MSE) and R-Squared measure for each of the six hashtags. Additionally, Table 2 lists the t-test and p-value for each of the following features for each hashtag: 1) number of tweets 2) number of retweets 3) number of followers 4) maximum number of followers 5) time of day. For each of the six hashtags, e.g., #superbowl, we train over the entire hashtag dataset. We use the OLS statsmodels library to train the model.

Although the smaller MSE values do not necessarily correlate with the higher R-Squared values for these datasets, there are some patterns that emerge with respect to features and low p-values. For instance, if we use a p-value criteria of less than or equal to 0.3 as identifying a low p-value, then the features Num of Tweets and Num of Retweets have low p-values for all six hashtags. Four of the hashtags have low p-values for Max Num of Followers; three for Num of Followers; and no hashtag has a low p-value for the Time of Day feature. Therefore, the data indicate that the Time of Day feature is insignificant. Furthermore, #superbowl has the highest MSE at 52,483,472.229 but has the second highest R-Squared value of 0.800. #gopatriots has the lowest MSE at 27,588.586 but also a low R-Squared value of 0.629. The data in general seem to indicate that a linear regression model may not be the best for the datasets.

			Num of Tweets		Num of Retweets		Num of Followers		Max Num of Followers		Time of Day	
	MSE	R-Squared	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value
#gohawks	758533.870	0.476	7.829	0.000	-3.140	0.002	-2.427	0.016	0.410	0.682	0.328	0.743
#gopatriots	27588.586	0.629	1.079	0.281	2.550	0.011	-0.504	0.614	-0.108	0.914	0.122	0.903
#nfl	269958.620	0.571	4.201	0.000	-2.593	0.010	4.580	0.000	-3.526	0.000	0.155	0.877
#patriots	5180890.103	0.668	12.937	0.000	-1.178	0.239	-0.417	0.677	1.340	0.181	-0.426	0.670
#sb49	97560.265	0.804	13.019	0.000	-2.041	0.042	0.779	0.437	2.177	0.030	-0.609	0.543
#superbowl	52483472.229	0.800	28.537	0.000	-5.544	0.000	-6.265	0.000	4.889	0.000	-0.470	0.639

Table 2: Mean Squared Error, R-Squared for all 6 Hashtags and t-test and p-value for each Feature

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.

We use 10 features to train an ordinary least squares (OLS) regression model using statsmodels' built in libraries. For each hashtag, we train on the entire set of data. The features are:

- Number of tweets
- Number of retweets
- Number of Followers
- Maximum number of followers
- Number of mentions
- Status count
- Hashtag count
- Number of friends
- Number of favourites
- Time of day

There is a wide amount of variation in the MSE values. #gopatriots has the lowest MSE value at 10153.920, and #superbowl has the largest MSE value at 28907933.712, which is an increase of 284,597.27% from #gopatriots to #superbowl. Refer to Table 3a and Table 3b for the MSE, t-test, and p-values for all six of the hashtags.

For p-values, we seek a low value. For the purposes of this report, we will consider a p-value of less than or equal to 0.25 as low. Three features have a low p-value for all six hashtags: Num of

Tweets, Num of Retweets, and Num of Favourites. An additional four features have a low p-value for five out of the six hashtags: Sum of Followers, Max Followers, Num of Mentions, and Status Count. Finally, one feature clearly adds little positive impact on the model: Time of Day.

The large MSE values indicate that the data may not lend themselves well to linear regression. This may in part be explained by the fact that the data experience highly non-linear tweeting rates. That is the rate of tweets per unit time increase dramatically during the Super Bowl as compared to any other time.

		Num of Tweets		Num of Retweets		Sum of Followers		Max Followers		Num of Mentions	
	MSE	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value
#gohawks	509243.623	-10.109	0.000	-1.261	0.208	-6.605	0.000	3.805	0.000	0.300	0.764
#gopatriots	10153.920	2.396	0.017	-5.489	0.000	3.189	0.002	-3.573	0.000	11.812	0.000
#nfl	182095.887	-10.843	0.000	-3.066	0.002	2.757	0.006	-3.134	0.002	4.642	0.000
#patriots	3557192.833	-2.541	0.011	-3.765	0.000	5.431	0.000	-4.436	0.000	10.294	0.000
#sb49	9471400.808	-13.278	0.000	1.713	0.087	-1.711	0.088	-2.113	0.035	19.658	0.000
#superbowl	28907933.712	-8.788	0.000	-8.998	0.000	0.209	0.835	-0.309	0.758	9.069	0.000

Table 3a: MSE and First Ten Features' T-test and P-value for all Six Hashtags

	Status Count		Hashtag Count		Num of Friends		Num of Favourites		Time of Day	
	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value	t-test	p-value
#gohawks	0.435	0.663	5.468	0.000	0.360	0.719	6.645	0.000	-1.162	0.246
#gopatriots	6.543	0.000	0.505	0.613	1.973	0.049	-18.553	0.000	-0.736	0.462
#nfl	11.700	0.000	12.141	0.000	-5.519	0.000	2.264	0.024	-1.459	0.145
#patriots	6.588	0.000	0.613	0.540	-0.950	0.343	-3.447	0.001	-0.994	0.321
#sb49	11.991	0.000	5.395	0.000	5.491	0.000	-14.599	0.000	-0.936	0.350
#superbowl	1.865	0.063	7.313	0.000	-6.034	0.000	6.082	0.000	-0.679	0.497

Table 3b: Second Ten Features' T-test and P-value for all Six Hashtags

	Num of Tweets	Num of Retweets	Sum of Followers	Max Followers	Num of Mentions	Status Count	Hashtag Count	Num of Friends	Num of Favourites	Time of Day
#gohawks	x	x	x	x			x		x	x
#gopatriots	x	x	x	x	x	x		x	x	
#nfl	x	x	x	x	x	x	x	x	x	x
#patriots	x	x	x	x	x	x			x	
#sb49	x	x	x	x	x	x	x	x	x	
#superbowl	x	x			x	x	x	x	x	

Table 4: Features Having a Significant Positive Effect on the Regression Model

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 samples you have extracted, and analyze it.

Do the regression coefficients agree with the trends in the plots? If not, why?

Figure 3 displays scatter plot of predictant versus values of that feature. The top three features vary between hashtags, see Table 4. In general, we find that the data are not well suited for linear regression modelling. The scatter data are heavily concentrated around the lower levels of tweets per hour for four out of the six plots. For instance, #gohawks and #gopatriots have large clusters of data points close to 0 tweets per hour with a few sporadic data points at higher tweeting rates. However, both #sb49 and #nfl have a more uniform distribution of scatter plot data points and appear more conducive for linear regression modeling.

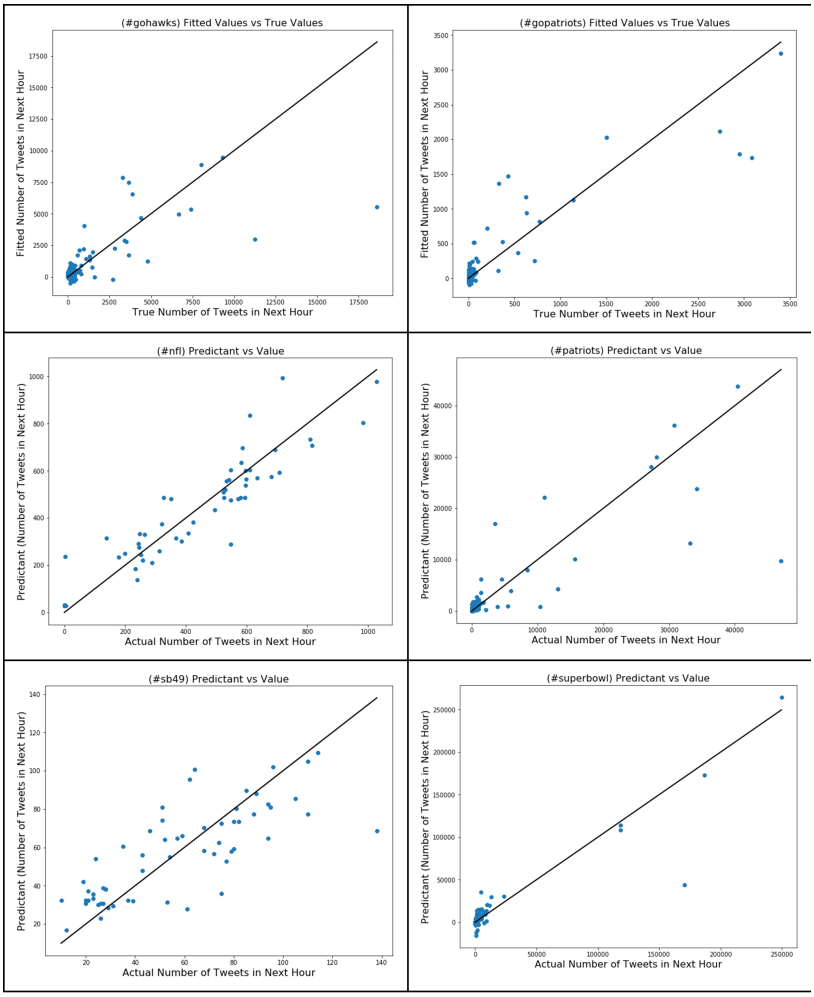


Figure 3: Scatter plots of data and linear Regression Line Prediction for All Six Hashtags

	Num of Tweets	Num of Retweets	Sum of Followers	Max Followers	Num of Mentions	Status Count	Hashtag Count	Num of Friends	Num of Favourites	Time of Day
#gohawks	X		X						X	
#gopatriots					X	X			X	
#nfl	X					X	X			
#patriots			X		X	X				
#sb49	X				X				X	
#superbowl	X	X			X					

Table 4: Top 3 features for each hashtag

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

- Before Feb. 1, 8:00 AM: 1-hour window
- Between Feb. 1, 8:00 AM and 8:00 PM: 5-minute window
- After Feb 1, 8:00 PM: 1-hour window

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

We divide our hashtag training data into three time segments:

- before Feb. 1st at 8:00 AM
- Feb. 1st from 8:00 AM until 8:00 PM

- after Feb. 1st at 8:00 PM.

We train three separate linear regression models using the statsmodels built in OLS library, one for each of the three time windows. The time segments correspond to the data before the Super Bowl; the data during the Super Bowl; and the data after the Super Bowl. Note that the datasets in the before and after timeframes are in one hour time segments and the dataset during the Super Bowl is in 5 minute windows.

MSE			
Hashtag	**Before Feb. 1, 8:00 AM: 1-hour window**	**Feb. 1, 8:00 AM and 8:00 PM: 5-min window**	**After Feb 1, 8:00 PM: 1-hour window**
#gohawks	718241.750	57354.739	178018.054
#gopatriots	1748.428	13159.530	26026.325
#nfl	65919.188	19201.440	53839.349
#patriots	338176.280	622016.186	1186815.931
#sb49	6904.622	1065338.261	4852754.474
#superbowl	515860.323	3532641.909	21496153.117

Table 6: MSE for all six hashtags during the before, during, and after Super Bowl times

R-Squared			
Hashtag	**Before Feb. 1, 8:00 AM: 1-hour window**	**Feb. 1, 8:00 AM and 8:00 PM: 5-min window**	**After Feb 1, 8:00 PM: 1-hour window**
#gohawks	0.303	0.596	0.239
#gopatriots	0.574	0.488	0.295
#nfl	0.512	0.834	0.625
#patriots	0.567	0.732	0.573
#sb49	0.868	0.889	0.618
#superbowl	0.402	0.943	0.717

Table 7: R-Squared for all six hashtags during the before, during, and after Super Bowl times

The MSE and R-Squared results are presented in Table 6 and Table 7, respectively. We analyze the results both between models and within models. We find that #gopatriots during the before time window has the lowest MSE value at 1748.428 of all the hashtag data, and that #superbowl after the Super Bowl has the largest MSE value at 21,496,153.117. The percent increase from the lowest MSE value to the highest is an increase of 1,229,356.119%. Minimizing the MSE is one objective of a regression model. Considering the fact that the vast majority of the data occurs in the during the Super Bowl timeframe, the high variation in regression model MSE values has some basis. See Table 8 for a summary of the MSE and R-Squared statistics.

Considering the R-Squared statistics, we find that the minimum R-Squared value is 0.239 from #gohawks occurring after the Super Bowl, and the maximum R-Squared value is 0.943 from #superbowl occurring during the Super Bowl time window. Having an R-Squared value as close to 1 as possible is the objective for a regression model.
Overall, the before time window has the lowest average of MSE values while the during time window has the largest average of R-Squared values for all six hashtags. The average R-Squared value of 0.538 for the before time window indicates a low correlation between the

features and the predicted output in general for the models. Similarly, the average MSE of 884,952.011 for the during time window is an increase in MSE error of 222.416%. Therefore, no time window improves in both MSE and R-Squared value. This is an indication that linear regression modelling may not be the best model for the Super Bowl data contained in this report.

	MSE			R-Squared		
Measurement	**Before**	**During**	**After**	**Before**	**During**	**After**
Minimum Value	1748.428	13159.530	26026.325	0.303	0.488	0.239
Maximum Value	718241.750	3532641.909	21496153.117	0.868	0.943	0.717
Mean Value	274475.099	884952.011	4632267.875	0.538	0.747	0.511
% Increase from Min to Max Value	40979.290%	26744.743%	82493.887%	186.469%	93.238%	200.000%

Table 8: Summary on the MSE and R-Squared statistics generated by the regression models

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 hour on the aggregated data.

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

We divide the data into three time windows, before, during, and after the Super Bowl, and aggregate the data from all six hashtags for each of the three timeframes, see Figure 4. We train a model for each of the three aggregated time windows, using the best performing features from Question 6 by p-value for each time window. Table 9, Table 10, and Table 11 provide the MSE, R-Squared, and t-test/p-values for each time window. We again use the statsmodels' OLS library for training the model. Comparing the results to the separated hashtag data files in Table 6 and Table 7, we see that the MSEs for the aggregated hashtag data are substantially lower

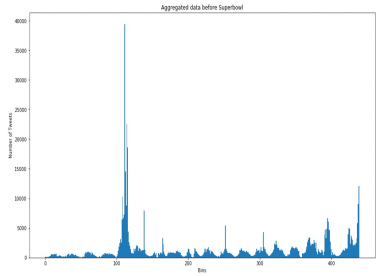
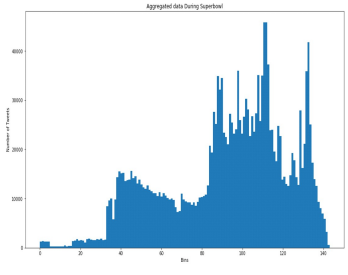
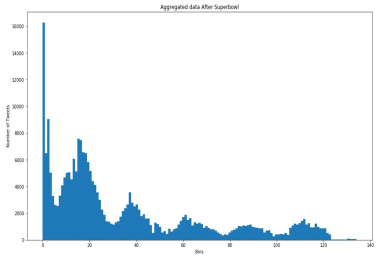
1/14/15 00:00 to 2/1/15 7:59	2/1/15 8:00 to 2/1/15 19:59	2/1/15 8:00 to 2/7/15 11:00
		

Figure 4: Piecewise aggregated number of tweets

	MSE		
Hashtag	**Before Feb. 1, 8:00 AM: 1-hour window**	**Feb. 1, 8:00 AM and 8:00 PM: 5-min window**	**After Feb 1, 8:00 PM: 1-hour window**
Aggregated Data	3202488.029	12530829.424	525027.5637

Table 9: MSE on the aggregated dataset during the before, during, and after Super Bowl time windows

	R-Squared		
Hashtag	**Before Feb. 1, 8:00 AM: 1-hour window**	**Feb. 1, 8:00 AM and 8:00 PM: 5-min window**	**After Feb 1, 8:00 PM: 1-hour window**
Aggregated Data	0.568	0.890	0.837

Table 10: R-Squared on the aggregated dataset during the before, during, and after Super Bowl time windows

for the before and during time windows, and the MSE value is significantly higher for the after time window. We find that the R-Squared values are lower for the before and during time windows, and is slightly higher for the after time window. Thus, we find that aggregating the data has mixed but more negative than positive results in terms of the MSE and R-Squared values. The MSE increases for two out of the three time windows and the R-Squared value decreases for two out of the three time windows. See Table 12 and Table 13 for a comparison summary of the separated and aggregated MSE and R-Squared values.

However, we achieve low p-values for most of our features across all time windows. Only during the after time window are any of the feature p-values not below 0.25.

Aggregated Hashtag Dataset												
Time Window	**Num of Tweets**		**Num of Retweets**		**Status Count**		**Hashtag Count**		**Friends**		**Favourites**	
	t-test	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**
Before Feb. 1, 8:00 AM: 1-hour window	8.224	0.000	-3.991	0.000	6.203	0.000	-10.752	0.000	6.190	0.000	8.563	0.000
	Num of Tweets		**Num of Retweets**		**Time of Day**		**Hashtag Count**		**Friends**		**Mentions**	
	t-test	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**
Feb. 1, 8:00 AM and 8:00 PM: 5-min window	5.457	0.000	-3.918	0.000	4.339	0.000	-5.736	0.000	3.759	0.000	4.772	0.000
	Num of Tweets		**Num of Retweets**		**Time of Day**		**Hashtag Count**		**Friends**		**Mentions**	
	t-test	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**	**t-test**	**p-value**
After Feb 1, 8:00 PM: 1-hour window	0.210	0.834	-5.259	0.000	-1.684	0.095	0.113	0.910	1.040	0.300	2.525	0.013

Table 11: T-test and p-values for all five features of aggregated dataset regression model

Maximum MSE Comparison			
Dataset	**Before**	**During**	**After**
Separated	718241.750	3532641.909	21496153.117
Aggregated	3202488.029	12530829.424	525027.564
Aggregated - Separated	2484246.278	8998187.515	-20971125.553

Table 12: Maximum MSE for aggregated hashtag and the maximum separated hashtag MSE

Maximum R-Squared Comparison			
Dataset	**Before**	**During**	**After**
Separated	0.868	0.943	0.717
Aggregated	0.400	0.400	0.839
Aggregated - #superbowl	-0.468	-0.543	0.122

Table 13: Maximum R-Squared value comparison between the aggregated dataset and maximum separated dataset value

Question 8: Use grid search to find the best parameter set for RandomForestRegressor and GradientBoostingRegressor, respectively. Use the following param_grid.

```
{
  'max_depth': [10, 20, 40, 60, 80, 100, 200, None],
  'max_features': ['auto', 'sqrt'],
  'min_samples_leaf': [1, 2, 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.

The data are trained on the aggregate of all the hashtags over the entire time. The best result achieved was with RandomForestRegressor with the following parameters:

RandomForestRegressor(
max_depth=100,
max_features='auto',
min_samples_leaf=4,
min_samples_split=10,
n_estimators=1000,
).

However, the test errors from cross-validation do not look good. This could be caused by the fact that we trained over the entire dataset and the majority of the information occurs during a relatively small period of time compared to the total. That is the timeframe during the Super Bowl is when the vast majority of tweets occur. The mean test score for the negative mean squared error for cross-validation is -202,870,202.4, as seen in Table 13.

split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score
-128716145.6	-256779975.3	-264873121.8	-184347119.4	-180268445.2	-202870202.4	51240513.95

Table 13: The best results for the negative mean squared error for cross-fold validation

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

Compared to OLS trained on the aggregate of the entire hashtag dataset, Gradient Boosting Regressor trained on the entire dataset performs worse than OLS in terms of both MSE and R-Squared values. The MSE increases by 215.026% of the original, which is a substantial degradation. Similarly, the R-Squared value decreases by 12.108% of the OLS value, which is still a significant degradation in correlation. Therefore, the training using Gradient Boosting Regressor does not improve performance on our aggregated data as compared to the OLS model. See Table 14 for more a summary of statistics for both Gradient Boosting Regressor and OLS models.

Gradient Boosting Regressor vs OLS Comparison		
Metric	**Gradient Boosting Regressor**	**OLS**
R-Squared	0.830582272	0.945
MSE	120808184.300	38348647.120

Table 14: Gradient Boosting Regressor vs OLS R-Squared and MSE Comparison

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? Does the best parameter set you find for each period agree with those you found above?

The aggregated hashtag dataset is divided into three time windows again: 1) before 2) during 3) after; and only the best performing features, p-values less than 0.3, are used to train the models. Using GradientBoostingRegressor and performing grid search on each of these three time windows, we find the following best parameter sets.

Before:

Features: 1) num of tweets 2) num of retweets 3) status 4) hashtag 5) friends 6) favourites

Best Parameter Set:

```
GradientBoostingRegressor(
        max_depth=None,
        max_features='sqrt',
        min_samples_leaf=2,
        min_samples_split=10,
        n_estimators=200,
        )
```

MSE: 14880.106
R-Squared: 0.998

split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score
-10851015.620	-1605602.379	-1278513.579	-486818.288	-635473.624	-2976805.908	3963980.691

Table 15: Best negative mean squared error results for cross-validation before the Super Bowl

During:

Features: 1) num of tweets 2) num of retweets 3) time of day 4) hashtag 5) friends 6) mentions

Best Parameter Set:
GradientBoostingRegressor(
max_depth=60,
max_features='sqrt',
min_samples_leaf=1,
min_samples_split=10,
n_estimators=1200,
)

MSE: 9.546×10^{-8}
R-Squared: 1.000

split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score
-13866809.800	-46168073.510	-30071546.070	-19224029.580	-14585151.410	-24893311.330	12205855.810

Table 16: Best negative mean squared error results for cross-validation during the Super Bowl

After:

Features: 1) num of tweets 2) num of retweets 3) time of day 4) hashtag 5) friends 6) mentions

Best Parameter Set:
GradientBoostingRegressor(
max_depth=200,
max_features='sqrt',
min_samples_leaf=1,
min_samples_split=10,
n_estimators=1000,
)

MSE: 9.889×10^{-8}
R-Squared: 1.000

split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score
-14718616.250	-40554145.870	-16522142.880	-24444352.530	-19059071.290	-23077958.830	9399626.750

Table 17: Best negative mean squared error results for cross-validation after the Super Bowl

The cross validation error significantly reduces when using Gradient Boosting Regressor for all three time windows. In the before time window, the reduction is by 99.535%, which is a staggering reduction. However, for both the during and after time windows, the reduction is 15 orders of magnitude for the during time window and 12 orders of magnitude for the after time window, note that both the Gradient Boosting Regressor MSE values are at the 10^{-8} order. Similarly, the R-Squared values reach near perfect correlation between the features and the predicted output. Reference Table 18 and Table 19 for a summary of the MSE and R-Squared findings, respectively.

MSE			
Model	**Before Feb. 1, 8:00 AM: 1-hour window**	**Feb. 1, 8:00 AM and 8:00 PM: 5-min window**	**After Feb 1, 8:00 PM: 1-hour window**
GradientBoostingRegressor	14880.106	0.000	0.000
OLS	3202488.029	12530829.424	525027.5637

Table 18: Comparison of MSE between Gradient Boosting Regressor and OLS for aggregated hashtag data

R-Squared			
Model	**Before Feb. 1, 8:00 AM: 1-hour window**	**Feb. 1, 8:00 AM and 8:00 PM: 5-min window**	**After Feb 1, 8:00 PM: 1-hour window**
GradientBoostingRegressor	0.998	1.000	1.000
OLS	0.568	0.890	0.837

Table 19: Comparison of R-Squared between Gradient Boosting Regressor and OLS for aggregated hashtag data

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.

We trained six different Neural Network MLPRegressor machine learning models on the aggregated data. Table 20 contains a summary of the layers, nodes per layer, and MSE for each architecture. We find that Architecture 2 has the lowest MSE at 721,695,386.760, and Architecture 4 has the highest MSE value at 470,756,000,651.965. Architecture 2 has five layers with one neuron each, whereas Architecture 4 has one layer with thirty neurons. The maximum number of layers tried is seven, and separately the maximum number of neurons per layer tried is 200. The minimum number for both layers and neurons is one.

MLP Regressor	Description	MSE
Architecture 1	(200, 200, 200, 200, 200, 200)	190574663477.360
Architecture 2	(1,1,1,1,1)	721695386.760
Architecture 3	(5,5,5,5,5,5)	81123632254.150
Architecture 4	(30)	470756000651.965
Architecture 5	(200,100,100,50,10,5,1)	721940831.360
Architecture 6	(100,50,25,10,5)	15842375704.530

Table 20: MLPRegressor Neural Network Architectures with MSE Values

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

Architecture 2 was the best performer with an MSE of 721,695,386.760 previously. After StandardScaler, Architecture 2 has an MSE value of 721,942,606.880, which is actually an increase in MSE value. So, although using StandardScaler to scale the data before feeding into the MLPRegressor significantly improves the performance for five architectures, Architecture 2 actually decreases in performance. Table 22 displays the difference between the MSE values from Question 11 and Question 12.

Scaled MLP Regressor	Description	MSE
Architecture 1	(200,200,200,200,200,200)	2335406.520
Architecture 2	(1,1,1,1,1)	721942606.880
Architecture 3	(5,5,5,5,5,5)	150923307.080
Architecture 4	(30)	396656764.390
Architecture 5	(200,100,100,50,10,5,1)	1948651.160
Architecture 6	(100,50,25,10,5)	1736993.860

Table 21: MLPRegressor using StandardScaler to Preprocess aggregated data

MSE Delta (Without StandardScaler - StandardScaler)	
Architecture 1	190572328070.840
Architecture 2	-247220.120
Architecture 3	80972708947.070
Architecture 4	470359343887.575
Architecture 5	719992180.200
Architecture 6	15840638710.670

Table 22: Delta between MSE of MLPRegressor without StandardScaler and with StandardScaler

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

We use grid search to find the best architecture for MLPRegressor for each of the three time windows on scaled data.

Before:

The best architecture is model = MLPRegressor(hidden_layer_sizes=(100,50,25,10,5). That is five hidden layers with 100, 50, 25, 10, and 5 neurons, respectively. For the best architecture, the MSE is 4,195,854.313 and R-Squared is 0.434. From Table 23, the grid search data is displayed, noting the far right column contains the ranking for each neural network model.

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_clf	param_clf__hidden_layer_sizes	param_clf__max_iter	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	99.349933	2.285816	0.016518	0.009939	MLPRegressor(activation='relu', alpha=0.0001, ...	(200, 100, 50, 10)	1000	-5.01E+06	-2.68E+06	-1.28E+06	-1.06E+07	-1.48E+06	-4.21E+06	3.45E+06	4
1	17.257733	0.423199	0.004846	0.004552	MLPRegressor(activation='relu', alpha=0.0001, ...	(40, 10)	1000	-5.25E+06	-3.49E+06	-2.54E+05	-1.44E+07	-1.91E+06	-5.08E+06	4.97E+06	14
2	112.263598	9.37814	0.036471	0.015875	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 100, 100, 100)	1000	-9.52E+06	-1.81E+07	-2.23E+07	-1.08E+07	-9.44E+05	-1.23E+07	7.35E+06	27
3	103.591623	15.130904	0.023152	0.00558	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 100, 100, 50, 20)	1000	-8.37E+06	-2.56E+07	-1.54E+07	-1.07E+07	-7.96E+05	-1.22E+07	8.21E+06	26
4	271.986052	38.612922	0.035065	0.011674	MLPRegressor(activation='relu', alpha=0.0001, ...	(500, 200, 100, 5)	1000	-7.30E+06	-9.97E+06	-2.16E+06	-1.07E+07	-1.40E+06	-6.31E+06	3.86E+06	19
5	20.613418	0.861058	0.005813	0.006138	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 10, 10)	1000	-3.92E+06	-3.10E+06	-2.96E+05	-1.45E+07	-1.95E+06	-4.75E+06	5.02E+06	9
6	32.379266	0.730064	0.011132	0.007602	MLPRegressor(activation='relu', alpha=0.0001, ...	(50, 20, 30, 10, 5)	1000	-2.91E+06	-2.73E+06	-7.34E+05	-1.14E+07	-2.07E+06	-3.98E+06	3.81E+06	2
7	139.042462	29.332641	0.039883	0.008735	MLPRegressor(activation='relu', alpha=0.0001, ...	(200, 200, 200, 200, 200, 200)	1000	-6.48E+06	-1.42E+07	-4.39E+06	-1.18E+07	-1.20E+06	-7.62E+06	4.75E+06	22
8	27.671508	0.698215	0.005613	0.006252	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 9, 8, 7, 6, 5)	1000	-4.58E+06	-9.08E+06	-3.74E+05	-1.44E+07	-1.95E+06	-6.09E+06	5.11E+06	17
9	53.804755	3.601039	0.012222	0.001703	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 50, 25, 10, 5)	1000	-2.27E+06	-3.41E+06	-6.70E+05	-1.03E+07	-1.51E+06	-3.63E+06	3.45E+06	1
10	27.836002	2.215373	0.000923	0.000058	MLPRegressor(activation='relu', alpha=0.0001, ...	(50, 20, 10, 10, 10)	1000	-1.02E+07	-2.67E+06	-3.79E+05	-1.18E+07	-1.92E+06	-5.40E+06	4.66E+06	16
11	33.931266	2.24894	0.006825	0.007295	MLPRegressor(activation='relu', alpha=0.0001, ...	(20, 20, 20, 20, 20, 20)	1000	-5.97E+06	-1.68E+07	-2.55E+07	-1.07E+07	-1.34E+06	-1.21E+07	8.43E+06	25
12	32.064837	1.699957	0.014287	0.01164	MLPRegressor(activation='relu', alpha=0.0001, ...	(50, 30, 30)	1000	-4.74E+06	-2.59E+06	-3.76E+05	-1.40E+07	-2.10E+06	-4.77E+06	4.83E+06	10
13	44.135193	2.282837	0.008458	0.004934	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 50, 20)	1000	-4.29E+06	-2.66E+06	-3.72E+05	-1.21E+07	-2.22E+06	-4.33E+06	4.08E+06	5
14	41.015925	1.294982	0.009412	0.003923	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 50, 20)	1000	-4.87E+06	-2.50E+06	-4.55E+05	-1.33E+07	-2.17E+06	-4.66E+06	4.53E+06	8
15	17.351141	1.048848	0.003431	0.005204	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 5, 5)	1000	-4.68E+06	-3.21E+06	-2.85E+05	-1.45E+07	-1.89E+06	-4.92E+06	5.01E+06	12
16	18.077185	1.186279	0.004716	0.004782	MLPRegressor(activation='relu', alpha=0.0001, ...	(60, 10, 1)	1000	-4.47E+06	-3.33E+06	-1.65E+06	-1.44E+07	-1.89E+06	-5.16E+06	4.76E+06	15
17	17.039335	0.41289	0.007434	0.008982	MLPRegressor(activation='relu', alpha=0.0001, ...	(20, 20, 20, 1)	1000	-1.02E+07	-9.85E+06	-1.65E+06	-1.40E+07	-3.54E+06	-7.86E+06	4.56E+06	23
18	26.17436	1.820021	0.011416	0.006091	MLPRegressor(activation='relu', alpha=0.0001, ...	(30, 30, 20, 10, 5, 1)	1000	-4.93E+06	-9.85E+06	-1.65E+06	-2.12E+07	-2.27E+06	-7.99E+06	7.21E+06	24
19	37.630349	1.813597	0.008625	0.006841	MLPRegressor(activation='relu', alpha=0.0001, ...	(80, 20, 30, 10, 10)	1000	-3.80E+06	-3.05E+06	-8.08E+05	-1.06E+07	-2.34E+06	-4.12E+06	3.38E+06	3
20	59.638982	2.970934	0.012195	0.001641	MLPRegressor(activation='relu', alpha=0.0001, ...	(30, 30, 100, 50, 10)	1000	-5.65E+06	-1.25E+07	-4.00E+06	-1.05E+07	-1.22E+06	-6.78E+06	4.16E+06	20
21	30.576369	2.281672	0.011219	0.006174	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 50, 30, 20)	1000	-4.83E+06	-2.28E+06	-1.22E+06	-1.23E+07	-2.15E+06	-4.56E+06	4.04E+06	6
22	22.042027	1.490418	0.001965	0.00215	MLPRegressor(activation='relu', alpha=0.0001, ...	(5, 10, 20, 10, 5, 1)	1000	-1.02E+07	-9.86E+06	-1.65E+06	-1.43E+07	-1.98E+06	-7.61E+06	4.97E+06	21
23	17.291963	0.304068	0.005131	0.004665	MLPRegressor(activation='relu', alpha=0.0001, ...	(20, 20, 10, 10)	1000	-4.88E+06	-3.14E+06	-3.26E+05	-1.42E+07	-1.94E+06	-4.91E+06	4.91E+06	11
24	30.145405	2.297221	0.00288	0.003811	MLPRegressor(activation='relu', alpha=0.0001, ...	(30, 50, 20, 10)	1000	-4.60E+06	-2.35E+06	-3.53E+05	-1.38E+07	-2.09E+06	-4.65E+06	4.79E+06	7
25	14.234297	0.613138	0.003163	0.004843	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 10)	1000	-4.87E+06	-3.38E+06	-2.96E+05	-1.44E+07	-1.95E+06	-4.99E+06	4.95E+06	13
26	13.067226	0.629363	0.003283	0.004934	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 5)	1000	-4.96E+06	-9.85E+06	-3.12E+05	-1.44E+07	-1.87E+06	-6.29E+06	5.21E+06	18

Table 23: Before time window grid search for best MLPRegressor architecture on scaled data

During:

The best architecture is model = MLPRegressor(hidden_layer_sizes=(100,100,100,100). That is four hidden layers with 100 neurons each. For the best architecture, the MSE is 1,111,734.358 and R-Squared is 0.990. The MSE dropped and R-Squared rose significantly from the before to during time window. From Table 24, the grid search data is displayed, noting the far right column contains the ranking for each neural network model.

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_clf	aram_clf__hidden_layer_size	am_clf__max_	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	36.786017	2.117741	0.011657	0.008743	activation='relu',	(200, 100, 50, 10)	1000	-1.36E+07	-8.27E+06	-2.79E+07	-2.17E+07	-1.97E+07	-1.82E+07	6.78E+06	3
1	8.399156	0.659918	0.010725	0.006313	activation='relu',	(40, 10)	1000	-9.98E+07	-1.60E+08	-3.43E+08	-1.41E+08	-1.85E+08	-1.86E+08	8.39E+07	25
2	43.905204	4.923514	0.015194	0.001519	activation='relu',	(100, 100, 100, 100)	1000	-1.09E+07	-1.03E+07	-1.90E+07	-1.97E+07	-1.92E+07	-1.58E+07	4.27E+06	1
3	42.196477	2.384255	0.012326	0.004317	activation='relu',	(100, 100, 100, 50, 20)	1000	-2.68E+07	-9.91E+06	-2.17E+07	-1.79E+07	-1.81E+07	-1.89E+07	5.56E+06	4
4	103.241474	5.898706	0.017016	0.004422	activation='relu',	(500, 200, 100, 5)	1000	-1.16E+07	-1.13E+07	-4.84E+08	-1.81E+07	-1.75E+07	-1.10E+08	1.89E+08	22
5	9.651209	0.570516	0.014928	0.00497	activation='relu',	(10, 10, 10, 10)	1000	-6.27E+07	-1.09E+07	-4.63E+07	-3.18E+07	-2.75E+07	-3.59E+07	1.77E+07	15
6	13.938936	0.750584	0.006732	0.005505	activation='relu',	(50, 20, 30, 10, 5)	1000	-1.99E+08	-8.32E+06	-3.88E+07	-2.63E+08	-2.47E+07	-1.06E+08	1.03E+08	21
7	124.157314	22.59299	0.02506	0.006302	activation='relu',	(200, 200, 200, 200, 200, 200)	1000	-3.21E+07	-1.38E+07	-2.37E+07	-4.17E+07	-1.13E+07	-2.45E+07	1.13E+07	7
8	12.561565	0.302045	0.003307	0.004813	activation='relu',	(10, 9, 8, 7, 6, 5)	1000	-3.23E+07	-9.53E+06	-4.06E+07	-2.83E+07	-3.40E+08	-8.88E+07	1.24E+08	20
9	21.78267	0.525484	0.00092	0.000077	activation='relu',	(100, 50, 25, 10, 5)	1000	-1.41E+07	-8.01E+06	-3.45E+07	-2.25E+07	-1.99E+07	-1.98E+07	8.96E+06	5
10	12.294783	0.636347	0.004664	0.006216	activation='relu',	(50, 20, 10, 10, 10)	1000	-2.22E+07	-8.43E+06	-3.81E+07	-2.40E+07	-3.40E+08	-8.52E+07	1.26E+08	19
11	11.002054	4.445742	0.002428	0.003096	activation='relu',	(20, 20, 20, 20, 20, 20)	1000	-8.51E+07	-8.26E+06	-2.69E+07	-2.11E+07	-1.53E+07	-3.15E+07	2.77E+07	14
12	10.011444	1.188021	0.004774	0.004859	activation='relu',	(50, 30, 30)	1000	-2.95E+07	-9.03E+06	-4.40E+07	-2.74E+07	-2.55E+07	-2.71E+07	1.12E+07	12
13	18.161938	1.714056	0.000887	0.000018	activation='relu',	(100, 50, 20)	1000	-2.91E+07	-8.39E+06	-4.12E+07	-2.62E+07	-2.53E+07	-2.61E+07	1.06E+07	11
14	17.966155	0.281362	0.000829	0.000074	activation='relu',	(100, 50, 20)	1000	-2.39E+07	-8.67E+06	-4.16E+07	-2.57E+07	-2.52E+07	-2.50E+07	1.05E+07	9
15	8.511014	0.572984	0.00485	0.006074	activation='relu',	(10, 10, 5, 5)	1000	-5.12E+07	-1.06E+07	-5.33E+07	-4.08E+07	-3.65E+07	-3.85E+07	1.54E+07	16
16	8.558835	0.365854	0.001901	0.00216	activation='relu',	(60, 10, 1)	1000	-1.99E+08	-2.32E+08	-4.84E+08	-5.08E+07	-3.40E+08	-2.62E+08	1.45E+08	26
17	8.949519	0.513416	0.001076	0.00057	activation='relu',	(20, 20, 20, 1)	1000	-3.16E+07	-9.56E+06	-4.84E+08	-2.84E+07	-3.40E+08	-1.79E+08	1.96E+08	24
18	14.32206	0.460531	0.006205	0.008882	activation='relu',	(30, 30, 20, 10, 5, 1)	1000	-1.83E+07	-7.57E+06	-4.84E+08	-2.63E+08	-2.27E+07	-1.59E+08	1.89E+08	23
19	16.057664	1.10331	0.008889	0.009834	activation='relu',	(80, 20, 30, 10, 10)	1000	-1.97E+07	-8.11E+06	-3.55E+07	-2.25E+07	-2.33E+07	-2.18E+07	8.80E+06	6
20	32.434989	1.617731	0.008673	0.007104	activation='relu',	(30, 30, 100, 50, 10)	1000	-1.31E+07	-8.24E+06	-2.80E+07	-2.13E+07	-1.63E+07	-1.74E+07	6.84E+06	2
21	14.722035	0.920155	0.013693	0.008434	activation='relu',	(10, 50, 30, 20)	1000	-2.33E+07	-7.85E+06	-3.99E+07	-2.55E+07	-2.66E+07	-2.46E+07	1.03E+07	8
22	12.934667	0.223691	0.005235	0.005597	activation='relu',	(5, 10, 20, 10, 5, 1)	1000	-1.99E+08	-2.32E+08	-4.84E+08	-2.63E+08	-3.40E+08	-3.03E+08	1.02E+08	27
23	11.418667	0.722148	0.000812	0.000093	activation='relu',	(20, 20, 10, 10)	1000	-3.23E+07	-8.81E+06	-4.49E+07	-2.91E+07	-2.61E+07	-2.82E+07	1.17E+07	13
24	13.583797	0.724364	0.003703	0.005837	activation='relu',	(30, 50, 20, 10)	1000	-2.86E+07	-8.58E+06	-3.81E+07	-2.49E+07	-2.58E+07	-2.52E+07	9.60E+06	10
25	9.207301	0.194871	0.005239	0.009129	activation='relu',	(10, 10, 10)	1000	-4.41E+07	-1.89E+07	-7.67E+07	-5.87E+07	-7.81E+07	-5.51E+07	2.22E+07	17
26	8.239404	1.039161	0.000777	0.000044	activation='relu',	(10, 10, 5)	1000	-6.66E+07	-5.44E+07	-1.18E+08	-4.66E+07	-6.64E+07	-7.06E+07	2.51E+07	18

Table 24: During time window grid search for best MLPRegressor architecture on scaled data

After:

The best architecture is model = MLPRegressor(hidden_layer_sizes=(40,10). That is two hidden layers with 40 and 10 neurons, respectively. For the best architecture, the MSE is 147,257.379 and R-Squared is 0.954. The MSE again dropped significantly and R-Squared remained close

to 1.0 from the during to after time window. From Table 25, the grid search data is displayed, noting the far right column contains the ranking for each neural network model.

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_clf	param_clf__hidden_layer_sizes	am_clf__max_	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	41.028234	1.864401	0.011842	0.006604	MLPRegressor(activation='relu', alpha=0.0001, ...	(200, 100, 50, 10)	1000	-6.04E+04	-1.13E+06	-5.09E+05	-8.47E+04	-1.42E+07	-3.11E+06	5.44E+06	19
1	9.301832	0.492069	0.007167	0.007426	MLPRegressor(activation='relu', alpha=0.0001, ...	(40, 10)	1000	-9.38E+05	-1.60E+06	-2.28E+06	-1.01E+06	-3.46E+06	-1.85E+06	9.29E+05	1
2	55.110069	1.909309	0.016698	0.005042	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 100, 100, 100)	1000	-6.69E+04	-1.09E+06	-7.07E+05	-1.35E+05	-1.46E+07	-3.24E+06	5.60E+06	20
3	49.281488	2.32629	0.020498	0.008494	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 100, 100, 50, 20)	1000	-1.12E+05	-1.34E+06	-7.04E+05	-1.04E+05	-1.86E+07	-4.07E+06	7.15E+06	23
4	110.164739	6.221022	0.013551	0.000983	MLPRegressor(activation='relu', alpha=0.0001, ...	(500, 200, 100, 5)	1000	-7.73E+04	-1.03E+06	-6.42E+05	-8.57E+04	-1.38E+07	-3.04E+06	5.28E+06	16
5	9.463861	0.75359	0.003694	0.003765	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 10, 10)	1000	-2.75E+05	-1.31E+06	-6.52E+05	-1.54E+05	-8.76E+06	-2.18E+06	3.25E+06	3
6	12.808095	0.14945	0.002789	0.00388	MLPRegressor(activation='relu', alpha=0.0001, ...	(50, 20, 30, 10, 5)	1000	-6.28E+04	-1.55E+06	-5.52E+05	-1.21E+05	-1.10E+07	-2.60E+06	4.16E+06	12
7	101.273269	7.15568	0.026054	0.00245	MLPRegressor(activation='relu', alpha=0.0001, ...	(200, 200, 200, 200, 200, 200)	1000	-1.06E+05	-9.40E+05	-7.53E+05	-2.21E+05	-1.36E+07	-3.05E+06	5.19E+06	17
8	11.987299	0.481652	0.003642	0.005544	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 9, 8, 7, 6, 5)	1000	-2.13E+05	-1.29E+06	-6.35E+05	-4.19E+06	-9.31E+06	-3.08E+06	3.36E+06	18
9	22.096722	0.858467	0.007108	0.005253	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 50, 25, 10, 5)	1000	-5.34E+04	-1.21E+06	-5.47E+05	-1.03E+05	-1.10E+07	-2.51E+06	4.17E+06	10
10	12.655534	0.81909	0.002342	0.001755	MLPRegressor(activation='relu', alpha=0.0001, ...	(50, 20, 10, 10, 10)	1000	-8.09E+04	-1.46E+06	-5.67E+05	-9.13E+04	-1.05E+07	-2.47E+06	3.95E+06	7
11	13.746279	0.75929	0.009369	0.004427	MLPRegressor(activation='relu', alpha=0.0001, ...	(20, 20, 20, 20, 20, 20, 20)	1000	-6.43E+04	-9.54E+05	-6.89E+05	-1.45E+05	-1.08E+07	-2.47E+06	4.10E+06	5
12	10.309957	0.592171	0.005954	0.005113	MLPRegressor(activation='relu', alpha=0.0001, ...	(50, 30, 30)	1000	-7.76E+04	-1.60E+06	-5.90E+05	-1.02E+05	-9.69E+06	-2.36E+06	3.64E+06	4
13	19.307301	0.462949	0.005719	0.006007	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 50, 20)	1000	-6.75E+04	-1.55E+06	-5.48E+05	-7.67E+04	-1.05E+07	-2.49E+06	3.97E+06	8
14	18.678985	0.936113	0.004253	0.006769	MLPRegressor(activation='relu', alpha=0.0001, ...	(100, 50, 20)	1000	-7.25E+04	-1.61E+06	-5.14E+05	-7.62E+04	-1.06E+07	-2.51E+06	4.00E+06	9
15	8.927204	0.763419	0.001081	0.000634	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 5, 5)	1000	-1.08E+06	-6.74E+06	-8.93E+05	-4.19E+06	-1.05E+07	-4.65E+06	3.62E+06	25
16	8.761077	0.469372	0.00704	0.008528	MLPRegressor(activation='relu', alpha=0.0001, ...	(60, 10, 1)	1000	-5.48E+05	-1.17E+06	-7.91E+06	-4.19E+06	-8.29E+06	-4.39E+06	3.24E+06	24
17	8.381796	0.356965	0.005586	0.007642	MLPRegressor(activation='relu', alpha=0.0001, ...	(20, 20, 20, 1)	1000	-2.83E+05	-6.73E+06	-7.90E+06	-1.44E+05	-9.33E+06	-4.84E+06	3.89E+06	26
18	12.529173	0.38592	0.006648	0.011593	MLPRegressor(activation='relu', alpha=0.0001, ...	(30, 30, 20, 10, 5, 1)	1000	-4.24E+06	-6.73E+06	-7.91E+06	-8.83E+04	-1.06E+07	-5.88E+06	3.55E+06	27
19	13.211855	0.598991	0.004282	0.005715	MLPRegressor(activation='relu', alpha=0.0001, ...	(80, 20, 30, 10, 10)	1000	-6.65E+04	-1.46E+06	-5.27E+05	-8.46E+04	-1.11E+07	-2.59E+06	4.22E+06	11
20	24.718213	1.726396	0.010417	0.005538	MLPRegressor(activation='relu', alpha=0.0001, ...	(30, 30, 100, 50, 10)	1000	-9.43E+04	-9.63E+05	-5.81E+05	-9.43E+04	-1.34E+07	-2.94E+06	5.13E+06	15
21	11.057816	1.037902	0.005804	0.006143	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 50, 30, 20)	1000	-7.95E+04	-1.19E+06	-5.26E+05	-7.09E+04	-8.90E+06	-2.10E+06	3.36E+06	2
22	10.140788	0.607013	0.009596	0.00763	MLPRegressor(activation='relu', alpha=0.0001, ...	(5, 10, 20, 10, 5, 1)	1000	-4.24E+06	-9.93E+05	-6.44E+05	-4.20E+06	-8.28E+06	-3.64E+06	2.74E+06	22
23	8.645805	0.485082	0.005516	0.006922	MLPRegressor(activation='relu', alpha=0.0001, ...	(20, 20, 10, 10)	1000	-2.25E+05	-1.36E+06	-6.41E+05	-1.12E+05	-1.03E+07	-2.47E+06	3.87E+06	6
24	11.055163	0.618208	0.003648	0.004432	MLPRegressor(activation='relu', alpha=0.0001, ...	(30, 50, 20, 10)	1000	-7.57E+04	-1.47E+06	-5.21E+05	-8.23E+04	-1.25E+07	-2.86E+06	4.76E+06	13
25	7.20035	0.453627	0.0067	0.011887	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 10)	1000	-5.26E+05	-1.04E+06	-1.33E+06	-2.53E+05	-1.18E+07	-2.92E+06	4.36E+06	14
26	6.353453	0.617936	0.000776	0.000041	MLPRegressor(activation='relu', alpha=0.0001, ...	(10, 10, 5)	1000	-9.86E+05	-1.54E+06	-1.77E+06	-5.32E+05	-1.22E+07	-3.34E+06	4.37E+06	21

Table 25: After time window grid search for best MLPRegressor architecture on scaled data

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.

We predict the number of tweets in the next time window for each of the nine test files. Note that all of the time intervals for test files for P1 and P3 are approximately 6 hours long, and all of the time intervals for test files for P2 are approximately 30 minutes long. Table 26 provides a summary of the results of these predictions for all nine of the files. The results vary from a low of 77.930 tweets for samp1_p3 to 16,640.929 tweets for samp0_p2. The training bin configurations used for each of the three time windows are described below. We also include the actual timespan of tweets for each file; when the first tweet actually occurs in each file; and what the first time bin corresponding to the first tweet below.

Before:

We train on the entirety of the before aggregated data divided into 6 hour segments, i.e., 6 one hour windows. For instance, we begin with Bin1 that contains the data for all of the feature data for the first six hours. The model then predicts the next hour after this first 6 hour segment, i.e., the 7^{th} hour. This process is repeated for the next six hour segment, shown below as Bin2. That is X_train consists of the six hours of data and y_train is the prediction of the next hour.

Bin1: 1^{st} hour [number of tweets, number of retweets, status count, hashtag count, friends count, favourites counts] … 6^{th} hour [number of tweets, number of retweets, status count, hashtag count, friends count, favourites counts] Y_train predict on 7^{th} hour

Bin2: 7th hour [number of tweets, number of retweets, status count, hashtag count, friends count, favourites counts] … 12th hour [number of tweets, number of retweets, status count, hashtag count, friends count, favourites counts] Y_train predict on 13th hour

During:

We train on the entirety of the before aggregated data divided into 30 minute segments, i.e., 6 five minute windows. For instance, we begin with Bin1 that contains the data for all of the feature data for the first 30 minutes. The model then predicts the next 5 minutes after this 30 minute segment, i.e., the 7th 5-minute segment. This process is repeated for the next 30 minute time interval, shown below in Bin2. That is X_train consists of the 30 minutes of data and y_train is the prediction of the next 5 minutes.

Bin1: 1st 5 minute interval [number of tweets, number of retweets, status count, hashtag count, friends count, favourites counts] … 6th 5 minute interval [number of tweets, number of retweets, status count, hashtag count, friends count, favourites counts]

Bin2: 7th 5 minute interval [number of tweets, number of retweets, time of day, hashtag count, friends count, mentions] … 12th 5 minute interval [number of tweets, number of retweets, time of day, hashtag count, friends count, mentions]

After:

We train on the entirety of the before aggregated data divided into 6 hour segments, i.e., 6 one hour windows. For instance, we begin with Bin1 that contains the data for all of the feature data for the first six hours. The model then predicts the next hour after this first 6 hour segment, i.e., the 7th hour. This process is repeated for the next six hour segment, shown below as Bin2. That is X_train consists of the six hours of data and y_train is the prediction of the next hour.

Bin1: 1st hour [number of tweets, number of retweets, time of day, hashtag count, friends count, mentions] … 6th hour [number of tweets, number of retweets, time of day, hashtag count, friends count, mentions]

Bin2: 7th hour [number of tweets, number of retweets, time of day, hashtag count, friends count, mentions] … 12th hour [number of tweets, number of retweets, time of day, hashtag count, friends count, mentions]

Testing file	Model	Trained Period	Predicted Number of Tweets
samp0_p1	(100, 50, 25, 10, 5)	Before Superbowl	449.1989724
samp1_p1	(100, 50, 25, 10, 5)	Before Superbowl	1258.087905
samp2_p1	(100, 50, 25, 10, 5)	Before Superbowl	385.286605
samp0_p2	(100, 100, 100, 100)	During Superbowl	16640.92899
samp1_p2	(100, 100, 100, 100)	During Superbowl	4019.114301
samp2_p2	(100, 100, 100, 100)	During Superbowl	463.3779892
samp0_p3	(40,10)	After Superbowl	88.60053958
samp1_p3	(40,10)	After Superbowl	77.92960851
samp2_p3	(40,10)	After Superbowl	309.5709351

Table 26: Predicted number of tweets for the next time window for each of the nine test files

Question 15:

- 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.
- 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.

We use the state or the state abbreviation as the primary means to identify the location of a tweeter. Only a few cities names are included as standalone location identifiers. For Massachusetts, '.* MA .*', '.* Mass.*', and '.* Ma .*' are chosen. It is important to note that there are spaces between the 'A' and 'a' of the state abbreviation and the '.*' but not between the 's' and the '.*' of 'Mass'. The reason is that 'Ma' could be the beginning of a city name, such as Manchester. The same coding scheme is used for Washington state as well, excluding Washington, D.C. will be addressed later. Because a city name is often found in multiple states and even different countries throughout the world, only a few city names are used. Already any city that references the state of either Massachusetts or Washington will be captured. Thus, only the following cities or regional references were used: Seattle, Tacoma, Spokane, Kirkland, Olympia, Boston, Worcester, Cambridge, and New England.

We train three binary classifiers to predict the location of the tweet author - either Washington or Massachusetts. The three are: 1) linear SVC 2) logistic regression 3) Naïve Bayes GuassianNB classifier. We split the data from #superbowl 90/10 for training and testing, respectively, to be able to generate the accuracy, precision, and recall scores.

Comparing the three classifiers, we find logistic regression has the highest accuracy at 77.242 of the three; Naïve Bayes GuassianNB classifier has the highest precision score at 0.692. And, logistic regression also has the highest recall score at 0.790.

Linear SVC:

Accuracy: 76.171
Precision score: 0.669
Recall score: 0.782
F1 score: 0.721

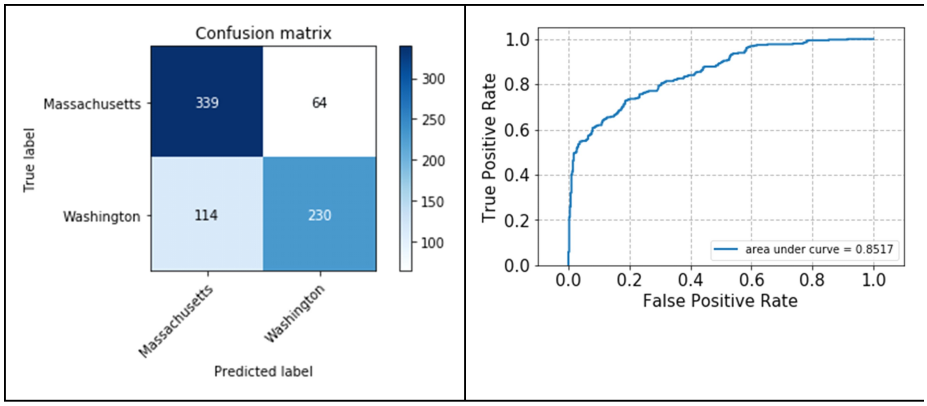


Figure 5: Linear SVC confusion matrix and ROC curve for state identification

Logistic Regression:

Accuracy: 77.242
Precision score: 0.689
Recall score: 0.790
F1 score: 0.736

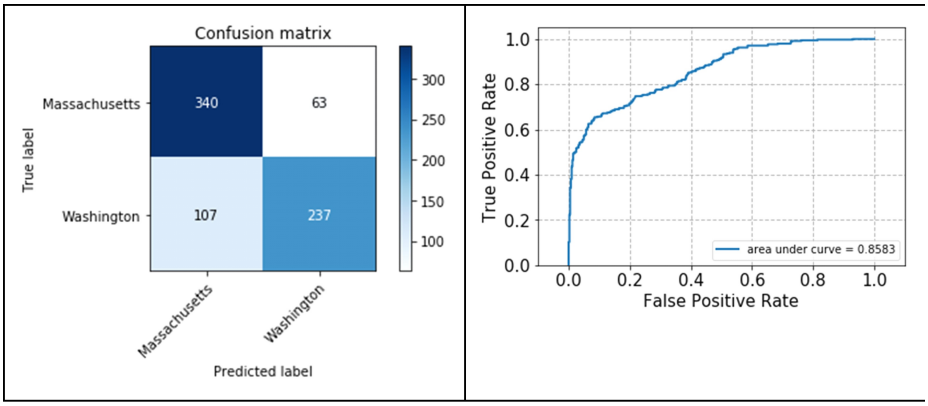


Figure 6: Logistic regression confusion matrix and ROC curve for state identification

Naïve Bayes GuassianNB Classifier

Accuracy: 73.896

Precision score: 0.692
Recall score: 0.728
F1 score: 0.709

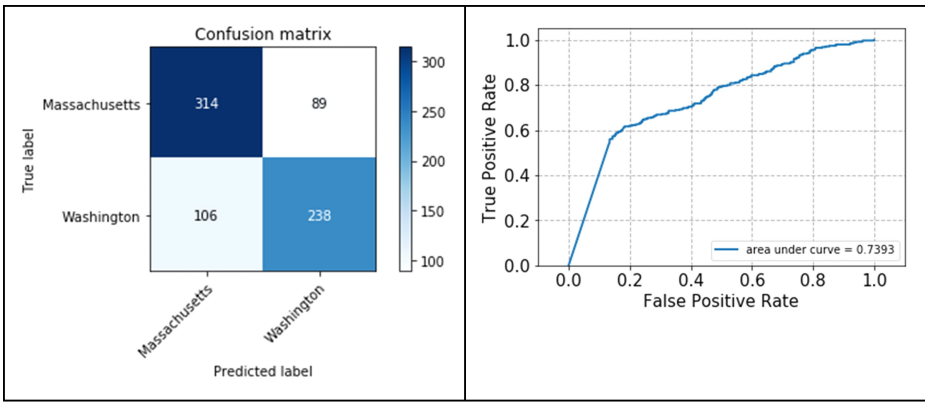


Figure 7: Naïve Bayes GuassianNB classifier confusion matrix and ROC curve for state identification

Question 16: The dataset in hand 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 you 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 you new ideas.

Of the many reasons why people watch the Super Bowl, one is for the excitement on the field and another is to watch and cast judgements on the halftime advertisements. We explore the use of the sentiments of individual tweets as a means to perform data analytics on these two topics. We identify five high importance individuals and six advertisements to investigate. The five high importance people are Patriots quarterback Tom Brady; Seahawks quarterback Russell Wilson; Patriots coach Bill Belichick, Seahawks coach Pete Carroll; and Seahawks running back Marshawn Lynch. And, the six advertisements we analyze are for Budweiser, Fiat, Microsoft, Dodge, Toyota, and Doritos.

We analyze the #superbowl dataset from 8:00 AM until 8:00 PM on the day of the Super Bowl for each of the five high importance people separately and use all six advertising companies to represent advertisement sentiment collectively. We use bins of 5 minute duration for our analysis. First, we preprocess the data by removing hashtags, punctuation, links, stopwords, and digits. Then, we use tweepy and the class TwitterClient to generate sentiment scores for every tweet, where 1 is positive and -1 is negative. For each high importance person, we search for the following terms and aggregate the total number of positive tweets, negative tweets, and the total number of tweets pertaining to each person.

Brady=[‘brady’, ‘tom’,’tom brady’]
Wilson=[‘wilson’,’russell’, ‘russ’]
Bill=[‘bill’,’belichick’,’hoody’]
Carroll=[‘pet’,’carroll’]
Lynch=[‘marshawn’,’lynch’,’beast mode’]

In addition, we also collect the scores for positive and negative tweets from all of the #superbowl data from 8:00 AM until 8:00 PM on Feb. 1 (Super Bowl Sunday). Figure 8 and Figure 9 display positive and negative sentiments (as well as total tweets for Figure 8) on the entire #superbowl dataset. Looking at Figure 8, we notice a substantial rise in tweets that begins around bin 80. This corresponds to about 2:30 PM (PST) of Feb. 1; the Super Bowl began at 3:30 PM (PST), so the rise in tweeting correlates with the beginning of the major pregame show that starts an hour before kickoff. Furthermore, we see from Figure 8 that there are more positive tweets, blue line, than negative tweets, red line, but there are far more neutral tweets than positive and negative combined. Additionally, we see that the peaks in both positive and negative tweets tend to follow the general trend of peaks for total tweets.

In the Figure 9, we normalize the positive and negative tweets by dividing by the total number of tweets per 5 minute interval on each 5 minute interval. We do this to see if there was a shift in emotional sentiment overall at one point in time as compared to the rest. Although the positive sentiment trends remain above negative sentiment and fairly consistent throughout, the negative sentiment does spike just above bin 100 or 4:20 PM (PST). This may correspond to New England fans reacting negatively to the Seahawk’s touchdown with 4:11 left in the 2nd quarter.

Figure 10 represents the sentiments for Tom Brady, the Patriots’ quarterback. We see positive sentiment clearly grows as the game progresses with major spikes towards the end. The latter spikes may correlate to big plays that lead ultimately to the Patriots winning the Super Bowl. Plays such as the Julian Edelman TD catch from Brady with 2:02 left in the 4th quarter after a drive where Brady completed 10 plays for 64 yards.

Interestingly, there is a clear drop in the total number of positive or negative tweets for all of the people during halftime, except for Pete Carroll. Carroll experiences a massive increase in positive tweets in the bins around bin 110. See Figure 13. Russell Wilson’s sentiments were more positive than negative until the very end, which could be due to the Malcolm Butler interception of Russell Wilson’s pass to ensure the Seahawk’s defeat, see Figure 11. Belichick interestingly has the graph where there are clearly more negative sentiments than positive ones. This may be due to the animosity and general dislike many football fans have for the Patriots’ coach; this dislike could be seen as partially founded based on past scandals and partially from jealously because of the success Belichick has had leading his team. Marshawn Lynch, a Seahawks running back, has three major positive spikes, which probably correspond to his big plays early in the games, such as his TD with 4:11 left in the 2nd quarter. This TD probably corresponds to the big spike just above bin 100 in Figure 14.

Finally, the advertisements are in Figure 15. There is little going on except right around halftime when the advertisements are being aired. Clearly, the sentiment is overwhelming positive at least for the six companies listed in the code snippet showing what keywords we searched for.

ads=[‘budweiser’, ‘fiat’, ‘microsoft’, ‘dodge’,’toyota’, ‘doritos’]

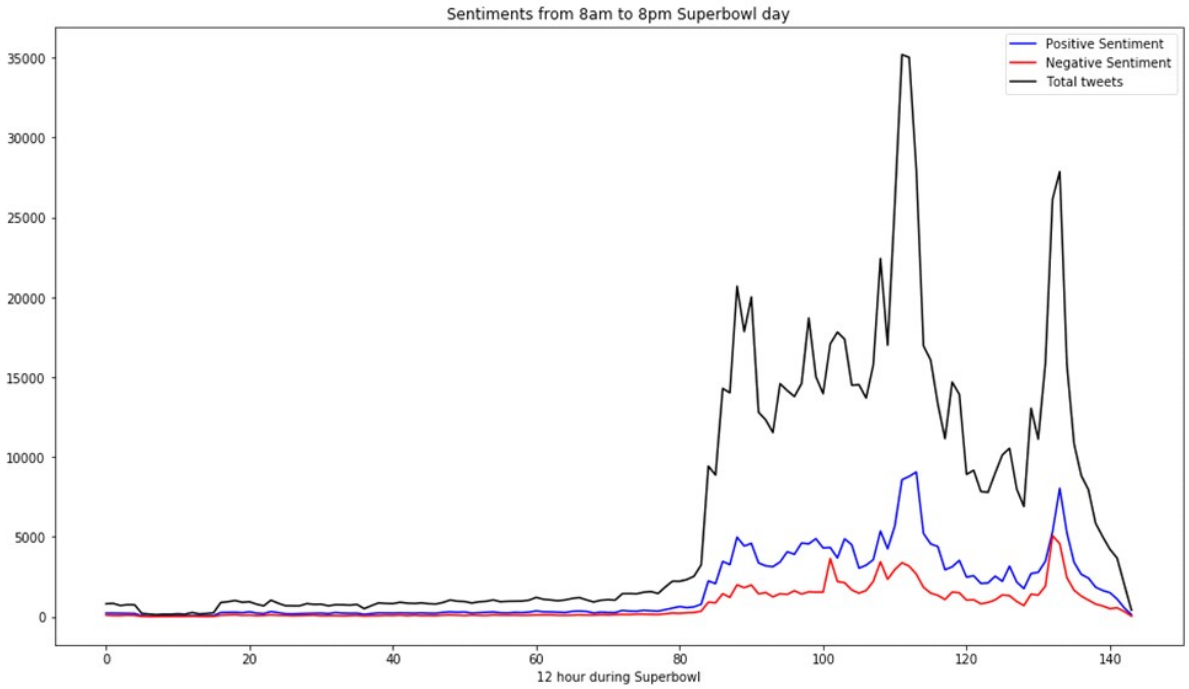


Figure 8: Sentiments from #superbowl during the Super Bowl time window

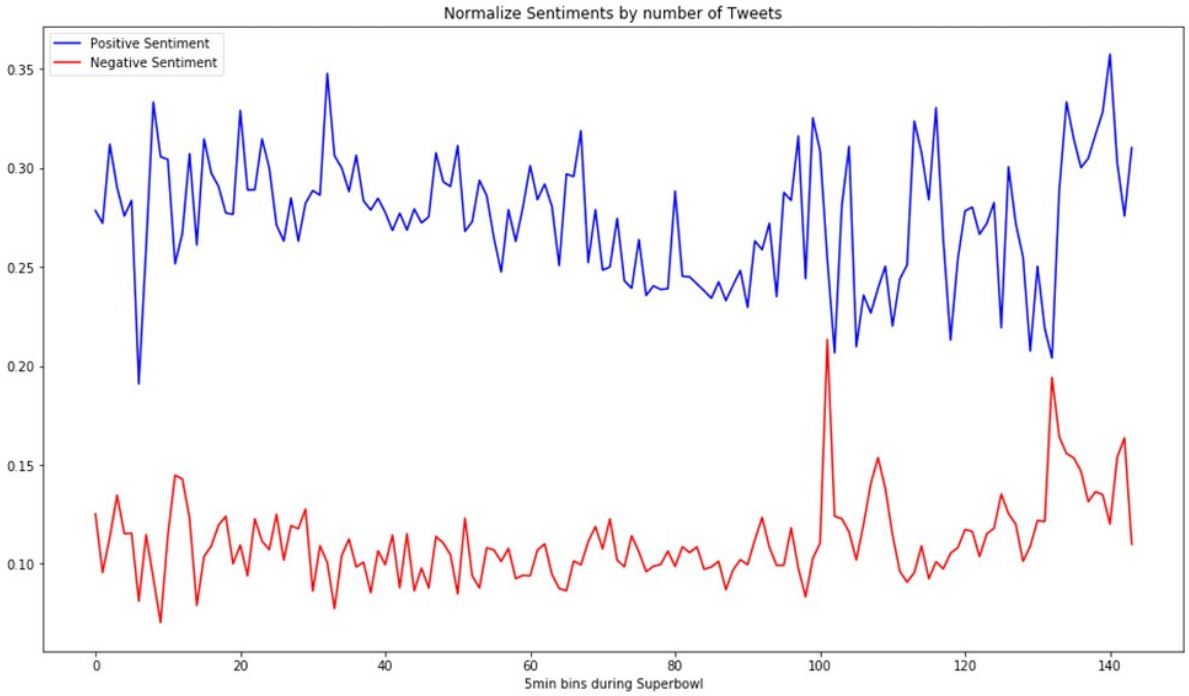


Figure 9: Normalized sentiments from #superbowl during the Super Bowl time window

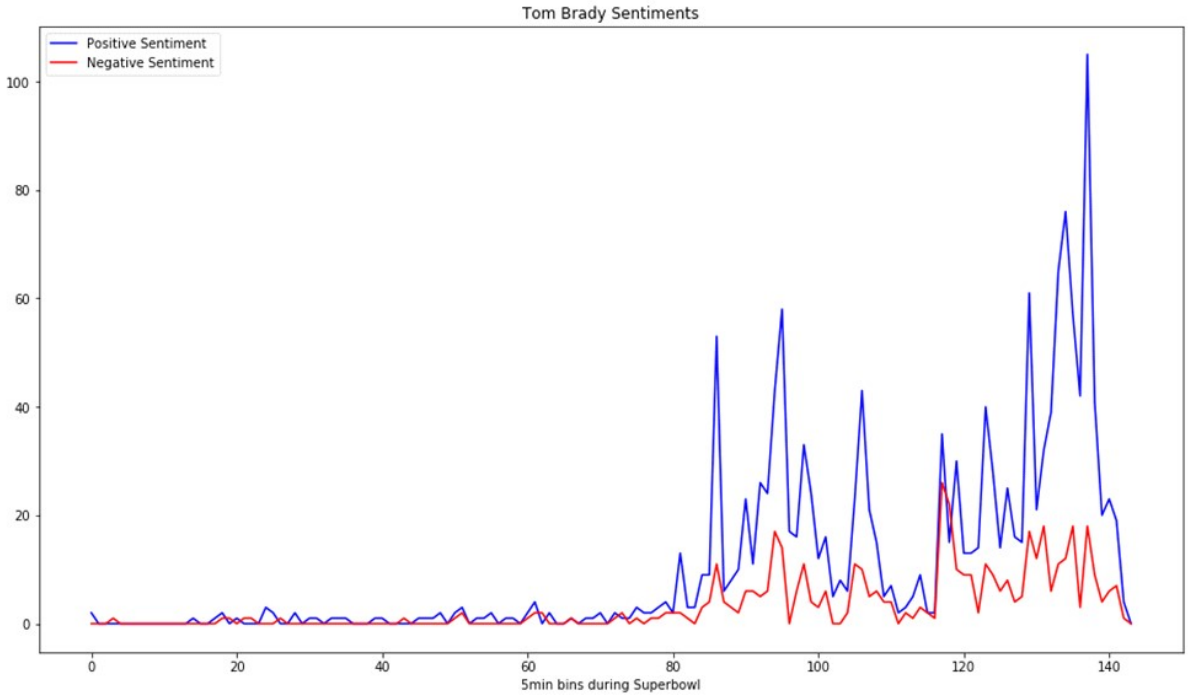


Figure 10: Tom Brady sentiments from #superbowl during the Super Bowl time window

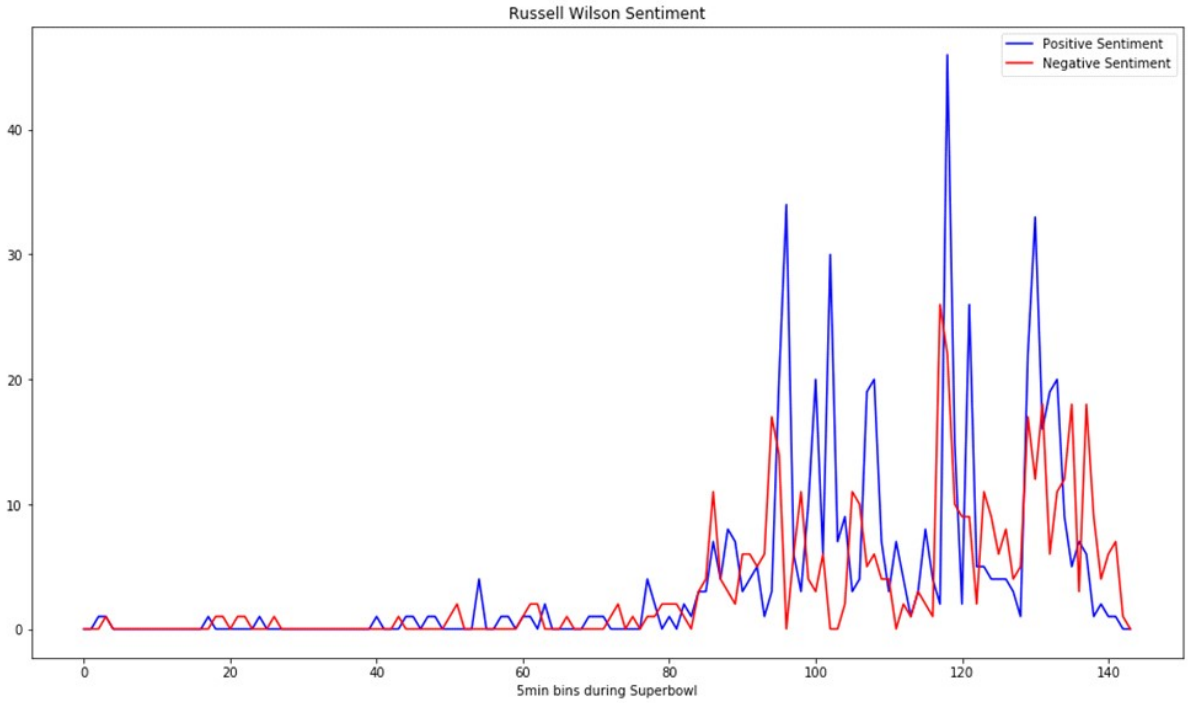


Figure 11: Russell Wilson sentiments from #superbowl during the Super Bowl time window

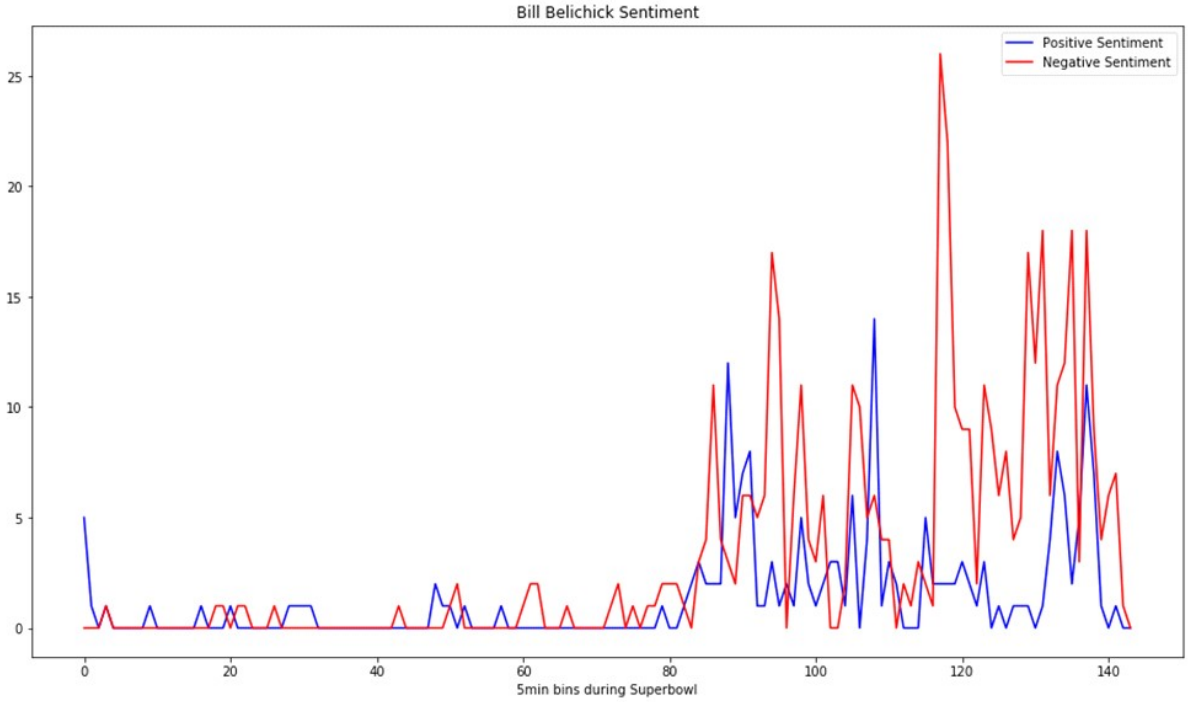


Figure 12: Bill Belichick sentiments from #superbowl during the Super Bowl time window

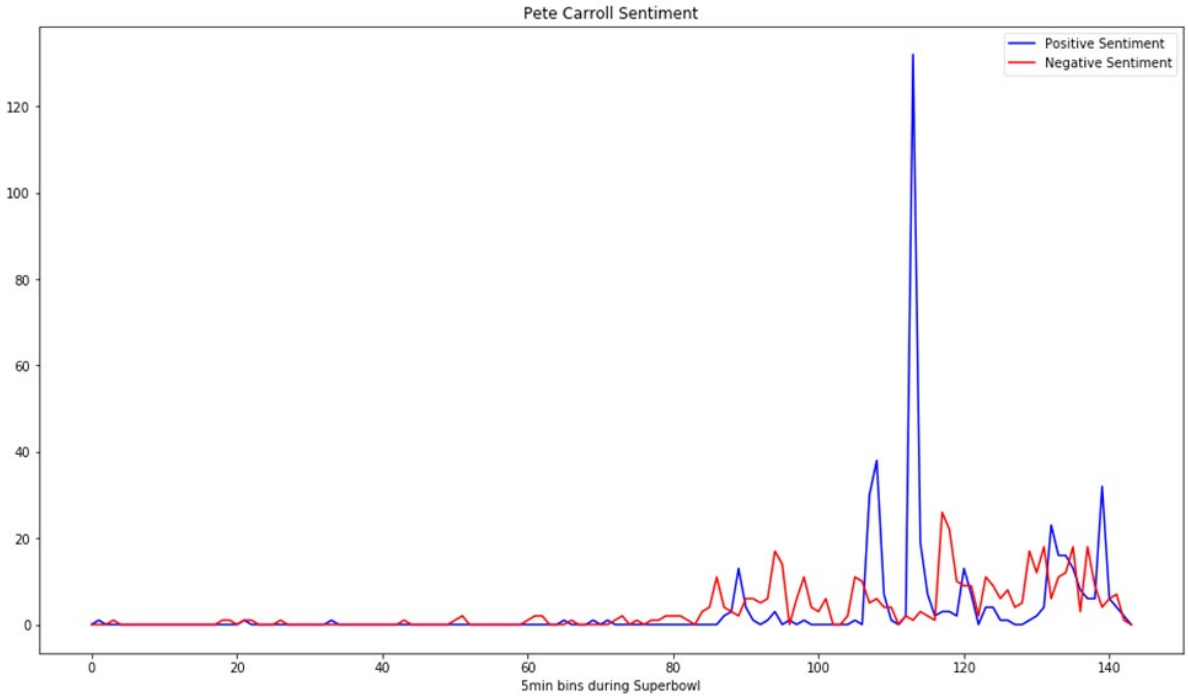


Figure 13: Pete Carroll sentiments from #superbowl during the Super Bowl time window

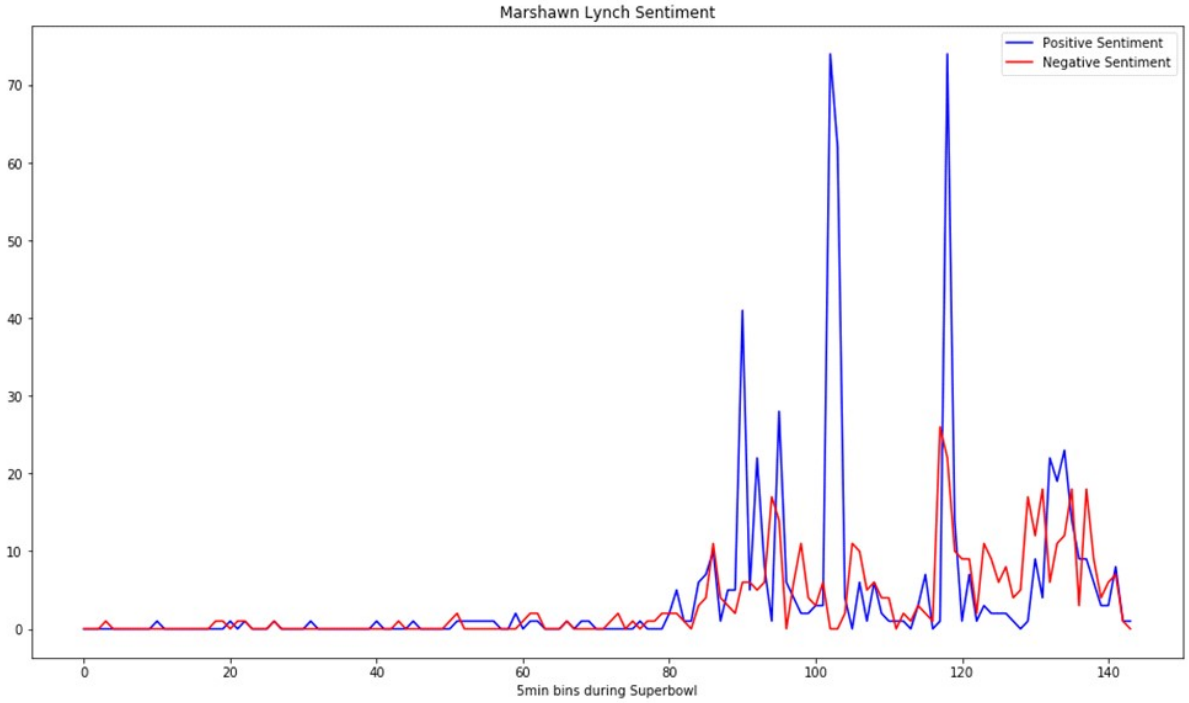


Figure 14: Marshawn Lynch sentiments from #superbowl during the Super Bowl time window

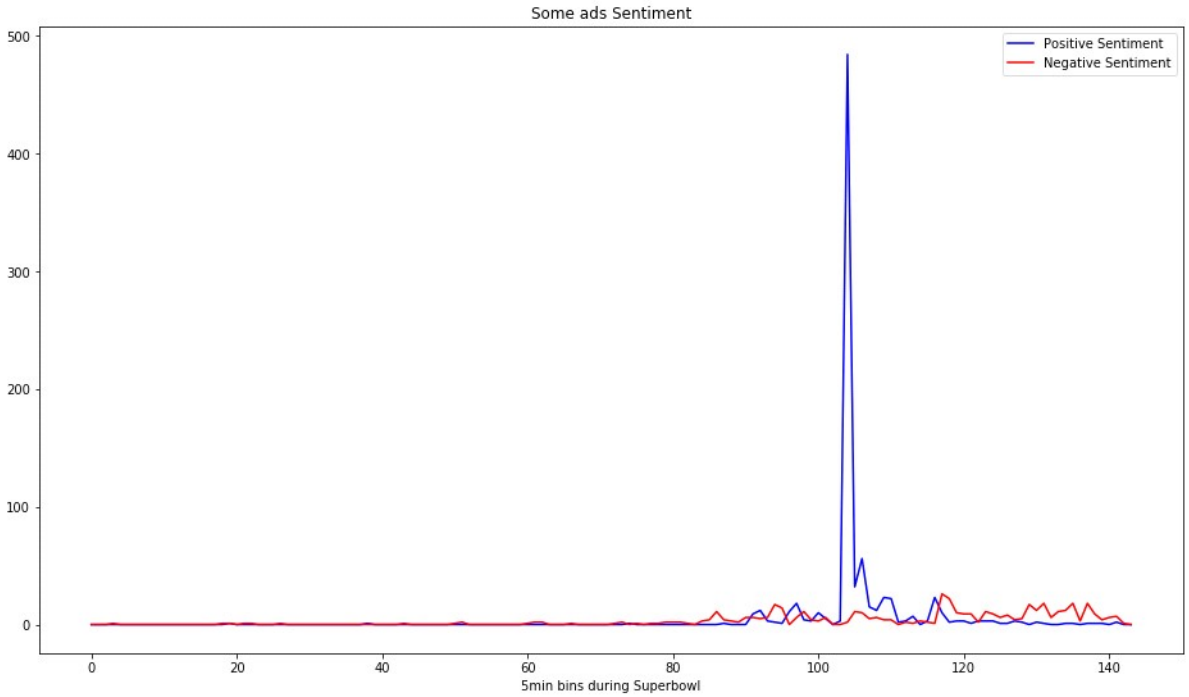


Figure 15: Advertisement sentiments from #superbowl during the Super Bowl time window