

Project 5: Application - Twitter Data

Due Monday June 11, 2018 by 11:59 pm

Team Members

Jennifer MacDonald, UID: 604501712

Nguyen Nguyen, UID: 004870721

Sam Yang, UID: 604034791

Introduction

Social network analysis, in its most general form, is the process of investigating and measuring relationships and flows between people, groups, organizations, etc. One useful topic in social network analysis is to predict future popularity of an object or an event. In this project, we investigated and studied Twitter metadata from different users, and formulated an solution to answer “can we predict if it will become more popular, and if so, by how come much?”.

The provided Twitter data were collected from 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 utilized the data from the related hashtags to train a regression model, particularly linear regression, and made predictions. We utilized statsmodel library, a popular statistics library, to perform our main linear regression analysis. However, we utilized scikit-learn library for Random Forest and Gradient Boosting models.

Linear Regression Model

Linear regression is a regression model that uses a linear approach to model the relationship between dependent variables to an independent variable. The usage of a linear regression model is for prediction, forecasting, or error reduction. Furthermore, linear regression model can be used to explain variation in the response variable or applied to determine how common the relationship between the dependent and independent variables. The model is often fitted using the least squared approach. Thus the model can take the form of:

$$y_i = \beta_0 \mathbf{1} + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n,$$

Often, the equation can be expressed as a matrix form.

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

$$X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

Random Forest Ensemble

The goal of an ensemble is to combine predictions of several base estimators in order to improve overall robustness of a single estimator. In random forests, each tree in the ensemble is built from a sample drawn from the training dataset. When the decision tree is split, it is picked by the best split among a random subset of the features. However, it can slightly increase the bias (with respect to the bias of a single non-random tree), however decrease the variance thus yielding an overall better model.

Gradient Boosting Ensemble

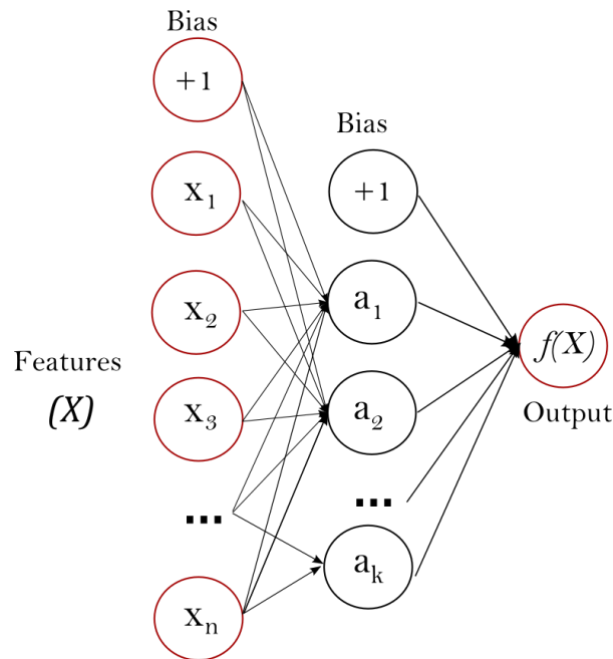
Gradient boosting tries to solve the minimization problem numerically using steepest descent method. The descent direction is the negative gradient of the loss function evaluated at the current model F_{m-1} , that can be calculated as seen below:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^n \nabla_F L(y_i, F_{m-1}(x_i))$$

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)})$$

Neural Network MLP Regressor

MLP Regressor is a supervised learning algorithm that learns by training on a dataset. It can also learn a non-linear function, but differ from logistic regression such that input and output layer can be one or more non-linear layers called hidden layers.



Leftmost layer, also known as input layer, consists of $\{x_i | x_1, x_2, \dots, x_m\}$ representing a set of input features. Each feature in the hidden layer transforms the values from previous layer with weighted value $w_1x_1 + w_2x_2 + \dots + w_mx_m$ followed by a non-linear function $g(\cdot) : R \rightarrow R$ (typically a hyperbolic function).

The advantages of Multi-Layer Perception (MLP) are:

- Ability to learn non-linear models
- Ability to learn models in real-time

Some disadvantages include:

- Hidden layers have a non-convex loss function where it exists more than one local minimum. Different random weight initializations can lead to different validation accuracy.
- Requires extra care of tuning a number of hyperparameters such as number of hidden neurons, layers, iterations.
- Sensitive to feature scaling.

Twitter Data

The provided Twitter raw metadata are divided into popular hashtags: #gohawks, #gopatriots, #nfl, #patriots, #sb49, #superbowl.

Extraction of Twitter Data

Since the Twitter metadata are large files and in JSON format, we had to parse out the data we only need for the analysis. In this project, we write the data into temporary files which are then uploaded back into Pandas dataframe objects.

Part 1: Popularity Prediction

1. A first look at the data

We downloaded the training tweet data set (at <https://ucla.box.com/s/24oxnhsoj6kpxhl6gyvuck25i3s4426d>). The dataset contains the Twitter metadata hashtag information described above.

QUESTION 1: Report the following statistics for each hashtag:

- 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 user posted twice, we count the user and the user's followers twice as well)
- Average number of retweets per tweet

To familiarize ourselves with the Twitter metadata, we first calculated the basic statistics for each hashtag which includes the average number of tweets per hour, the average number of followers of users posting the tweets per tweet, and the average number of retweets per tweet. The results can be seen in Table 1 below.

Table 1: Basic Statistics of Six Hashtags

Statistics	#gohawks	#gopatriots	#nfl	#patriots	#sb49	#superbowl
Avg. # of tweets per hour	292.49	40.95	397.02	750.90	1276.86	2072.12
Avg. # of followers of users posting the tweets per tweet	2217.92	1427.25	4662.38	3280.46	10374.16	8814.97
Avg. # of retweets per tweet	2.01	1.41	1.53	1.79	2.53	2.39

Basing on our observation, #superbowl hashtag has an average number of tweets per hour of 2072 tweets per hour, an average number of followers of users posting the tweets per tweet of 8815, and an average number of retweets per tweet is about 2 retweets per tweet. On the other hand, #nfl hashtag has an average number of tweets of 397 tweets per hour, an average number of followers of users posting the tweets per tweet of 4662, and an average number retweets per tweet of 1 retweets per tweet. From these statistics, we observed that #superbowl hashtag is more active than #nfl, because the date range we observed includes a Super Bowl game.

Interestingly, the average number of tweets per hour for #gohawks is 292 tweets/hour which is about 7x more than the average tweets for #gopatriots hashtag that has only 40 tweets/hour. Likewise, the average number of retweets per tweet for #gohawks is 2x greater than of #gopatriots, and the average number of followers of users posting the tweets per tweet is also 2x greater. When looking at the unique number of users tweeting, there were 72401 unique users who tweeted using the hashtag #gohawks during the time period measured, but only 16558 unique users who tweeted with the hashtag #gopatriots. This data suggests that more fans were on Twitter cheering for the Seahawks than for the Patriots. For many, the Patriots are not a well-liked franchise in the NFL (National Football League), so it makes sense that when fans of other franchises have to pick between the two teams, they would root for the team they dislike the least.

QUESTION 2: Plot “number of tweets in hour” over time for #SuperBowl and #NFL (a histogram with 1-hour bins). The tweets are stored in separate files for different hashtags and files are named as tweet [#hashtag].txt.

Figure 1 through 2 below are the breakdown of tweets per hour for #superbowl and #nfl hashtags. Additionally, we also included Figure 3 through 4 which are the breakdown for #gohawks and #gopatriots in orders to further understand the sentiment aspect of the tweets relating to before the game, during the game, and after the game. The histograms are binned with 1-hour bins between January 14th, 2015, and February 07th, 2015 (24 days timeframe).

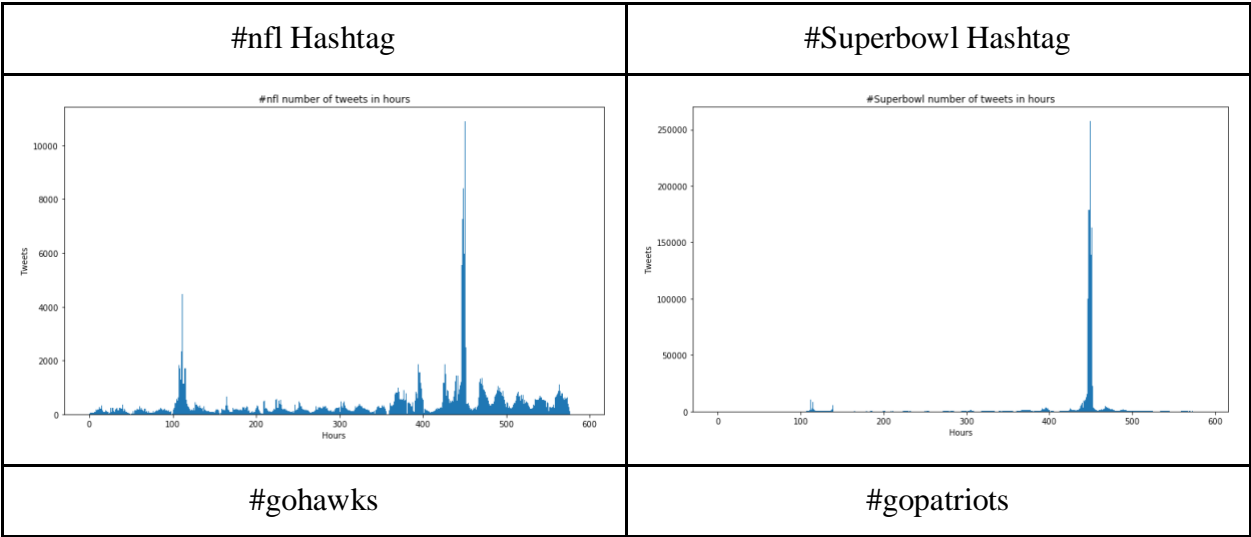
In #nfl and #superbowl histograms below, Figure 1 and Figure 2 respectively, we observed that there are two major peaks; the first peak is smaller than the second peak. Basing on the hour calculation of the histograms, the first major peak (around 100 hour) were the penultimate games for each team for the season, which occurred on January 18th, 2015; while the second peak (around 450 hour) was more dramatic which no doubt is the Super Bowl event which was played on February 1st, 2015.

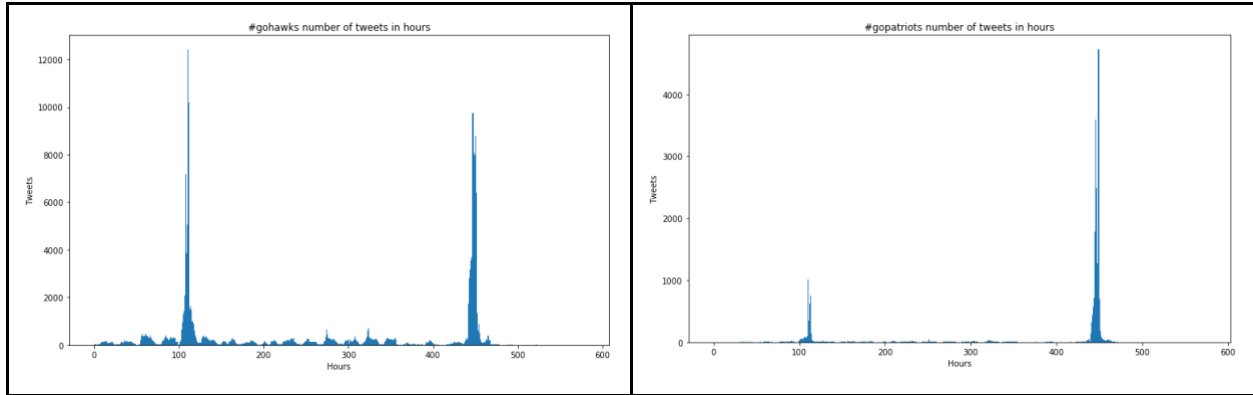
In Figure 3 and Figure 4, #gohawks and #gopatriots histograms respectively, we found that users were using the hashtag for the Seahawks much more than the Patriots for both the penultimate games and the Super Bowl event. This support our sentimental claim earlier that the Seahawks had more Twitter support online in the Super Bowl than the Patriots.

This could also be attributed to the fact that those that live Seattle tend to be younger (the average age for the city is only 35!) and have a number of technology companies in the city and surrounding area, such as Amazon and Microsoft. Those that are younger tend to be more technology-savvy and utilize social media more.

Additionally, we surmise that the spike for the Seahawks during the January 18th game could also be attributed to the fact that they had a much closer game that day than the Patriots did. While the Patriots had a blowout against the Colts 45-7, the Seahawks only beat the Packers in overtime after being tied 22-22 at the end of the fourth quarter. The uncertain outcome of the game could have lead to a spike of support for the Seahawks. Readers can see that the number of tweets for #gohawks is double than the number for tweets for #gopatriots.

Figure 1: Number of Tweets in Hours Plots





2. Linear regression

In this part of the analysis, we aggregated each hashtag data using the following 5 features in order to fit the linear regression model and predict the number of tweets in the next hour by using the following features extracted from tweet data in the previous hour:

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

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 library statsmodels in Python.

In order to predict the number of tweets in the next hour, we manipulated the data by taking the number of tweets in the current hour and shifted the hour ahead and called it 'future tweets'. By doing this, we were able to utilize current tweets and other features to predict future tweets.

The following Table 2 summarizes mean squared error (MSE) and R-squared for each hashtag. R-squared is a statistical measurement of how close the data are to the fitted regression. It ranges from zero to one, with zero indicating that the model does not improve prediction over the mean model, and one indicating perfect prediction. Therefore, improvement in regression model results consequent improves R-squared value. However, one pitfall of R-squared is that it can only increase as predictors are added to the regression model. To aid this, we should look at adjusted R-squared which incorporates the model's degrees of freedom. On the other hand, MSE is

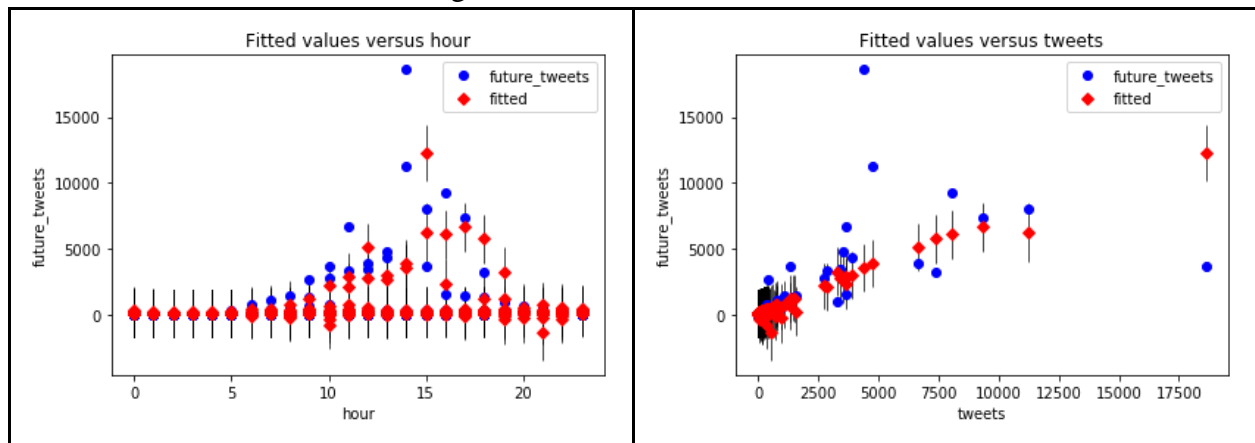
variance of the residuals. It indicates the absolute fit of the model to the data and how close the observed data points are to the model's predicted values, where as R-squared is a relative measure of fit. Lower MSE value indicates better fit, and it is a good measure of how accurately the model predicts the response.

Table 2: Mean Squared Error and R-Squared for each Hashtag

Hashtag	Mean Squared Error (MSE)	R-Squared
#gohawks	778598.882	0.476
#gopatriots	36467.260	0.627
#nfl	274541.222	0.570
#patriots	5234485.518	0.668
#sb49	17774930.591	0.804
#superbowl	53026404.701	0.800

From our analysis, #sb49 and #superbowl both have the highest R-squared values. However, as discussed above, high R-squared does not mean that it's the best model. In fact, both #sb49 and #superbowl datasets exhibit largest MSE values which indicates that the variance for this dataset is high, although the high number of datapoints will give a high MSE value even if the error measure of each individual point is minimal. #gopatriots dataset has the lowest MSE value, but a decent R-Squared value compared to other datasets. We can see this visualized in the fitted vs. true values of the hashtags below in Figure 2.

Figure 2: #Gohawks Fitted Plots



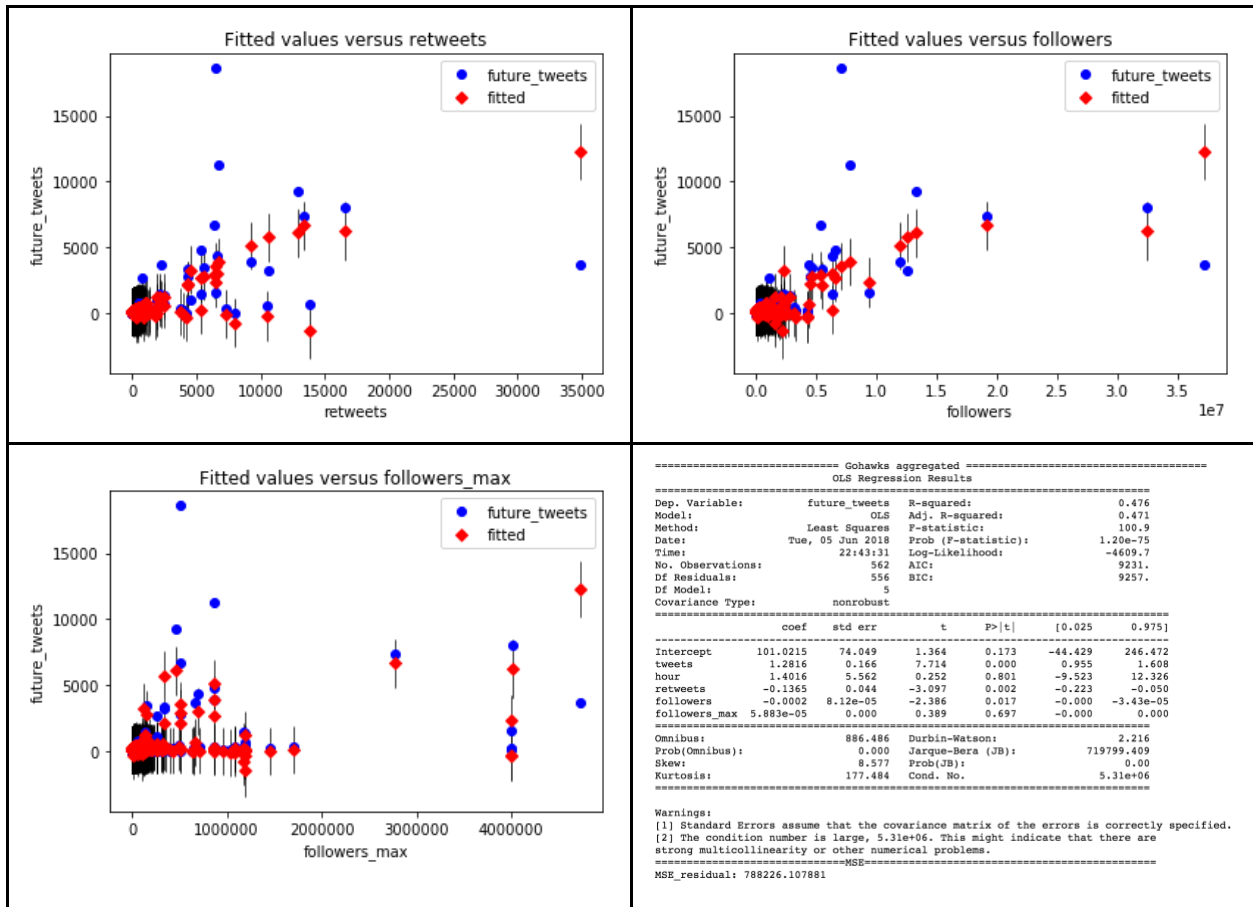
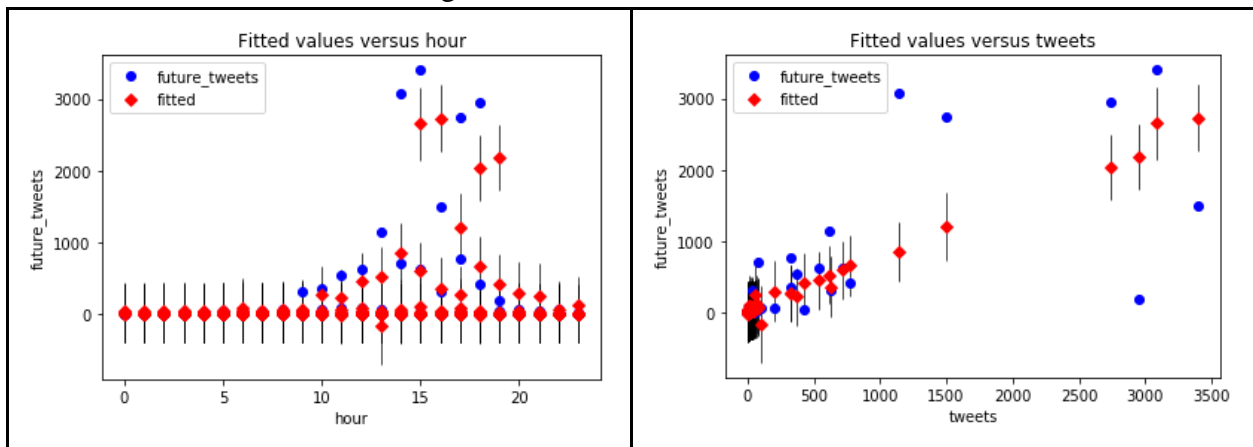


Figure 3: #GoPatriots Fitted Plots



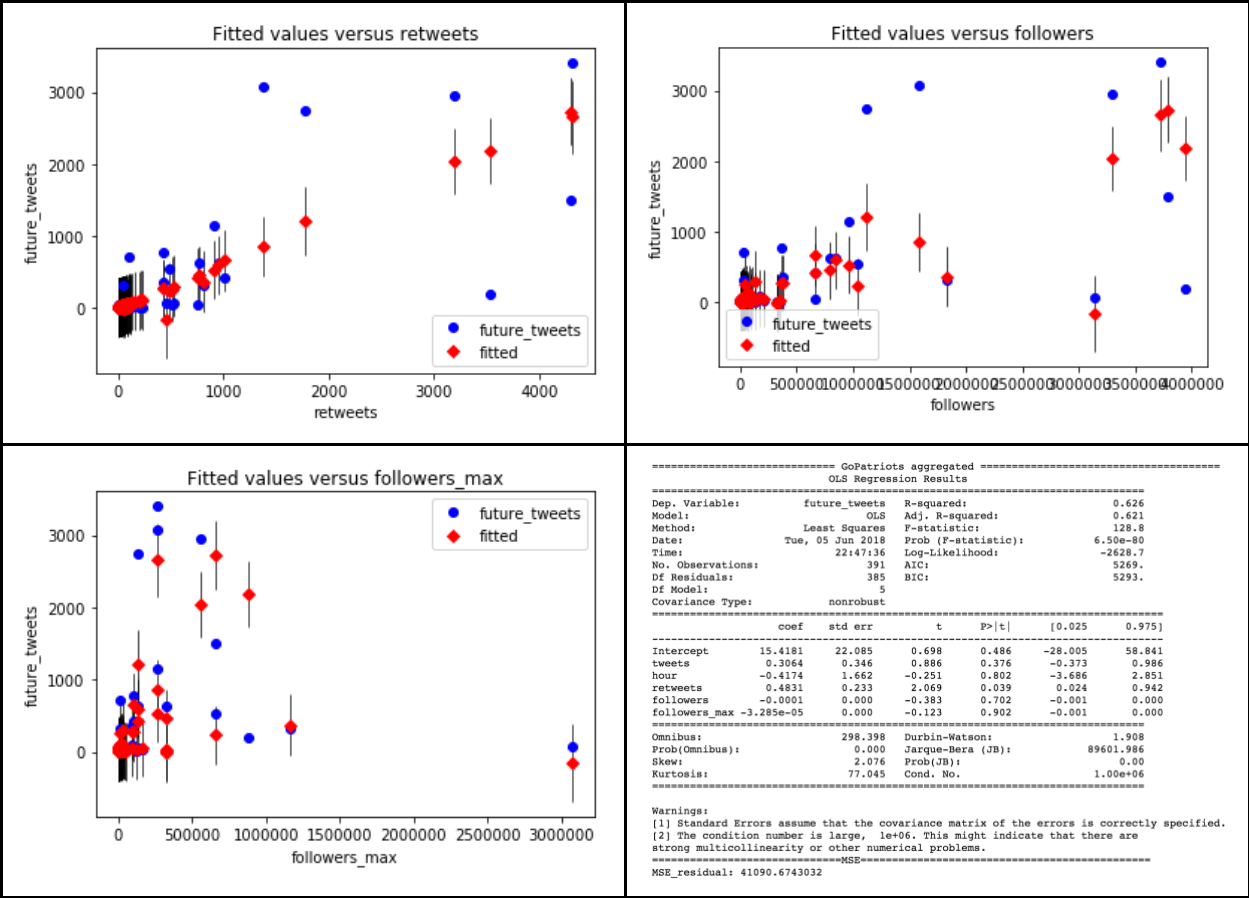
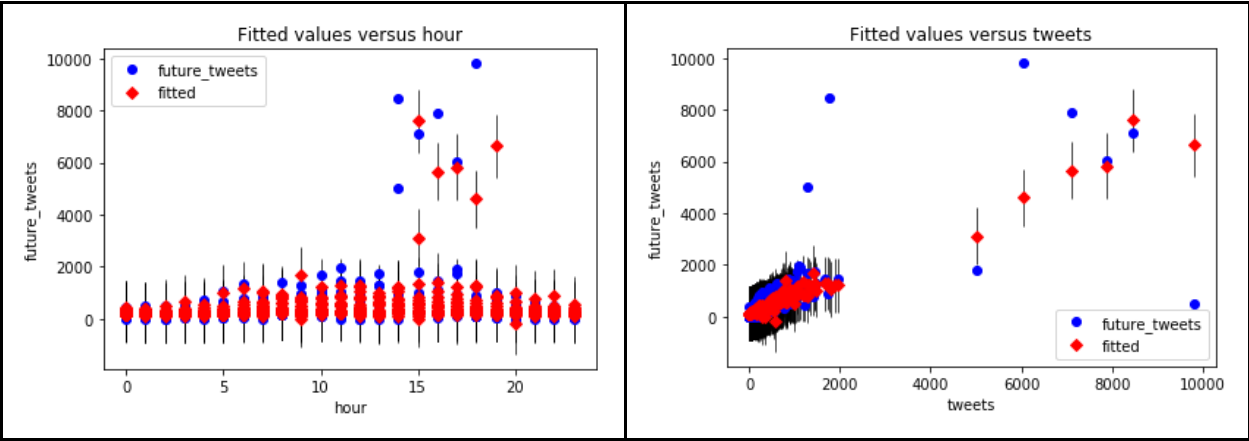


Figure 4: #nfl Fitted Plots



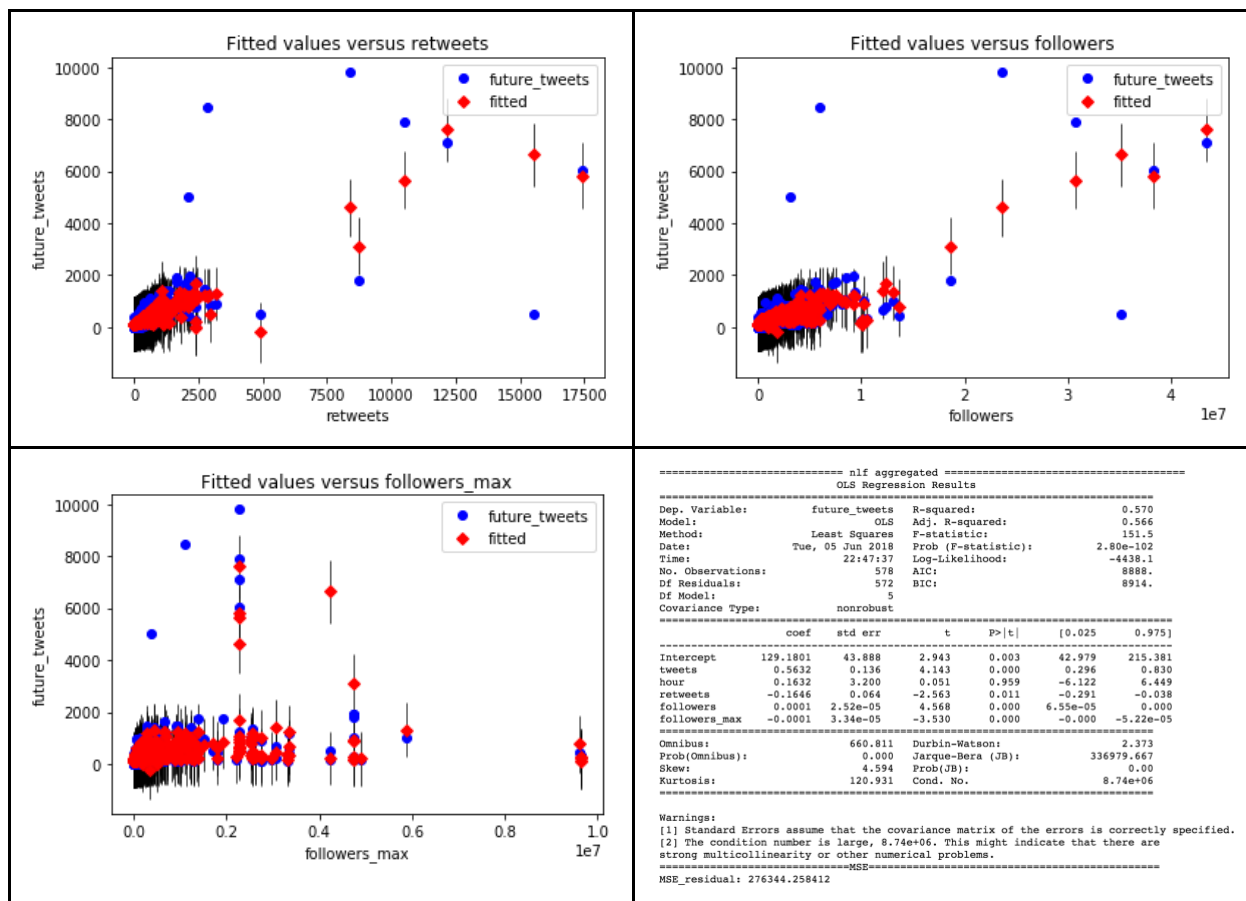
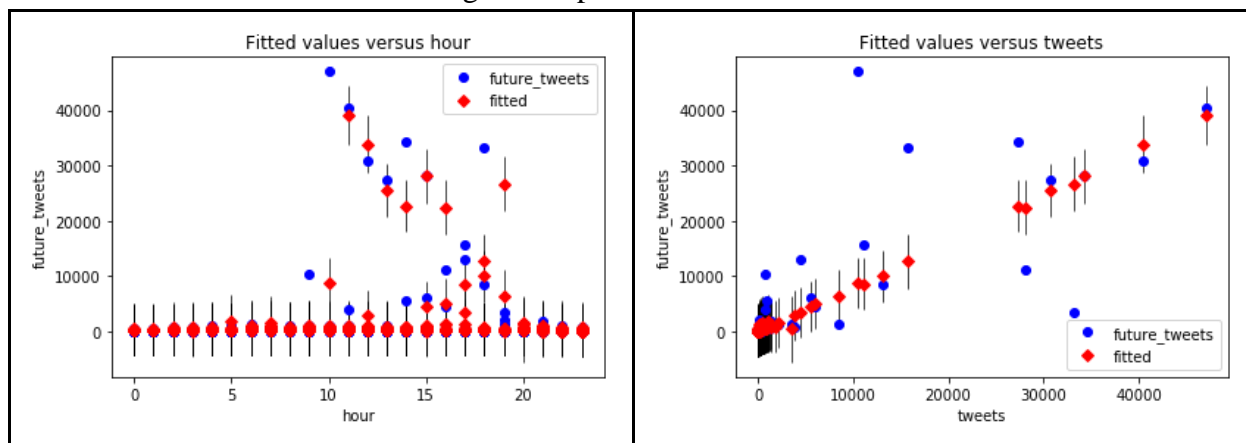


Figure 5: #patriots Fitted Plots



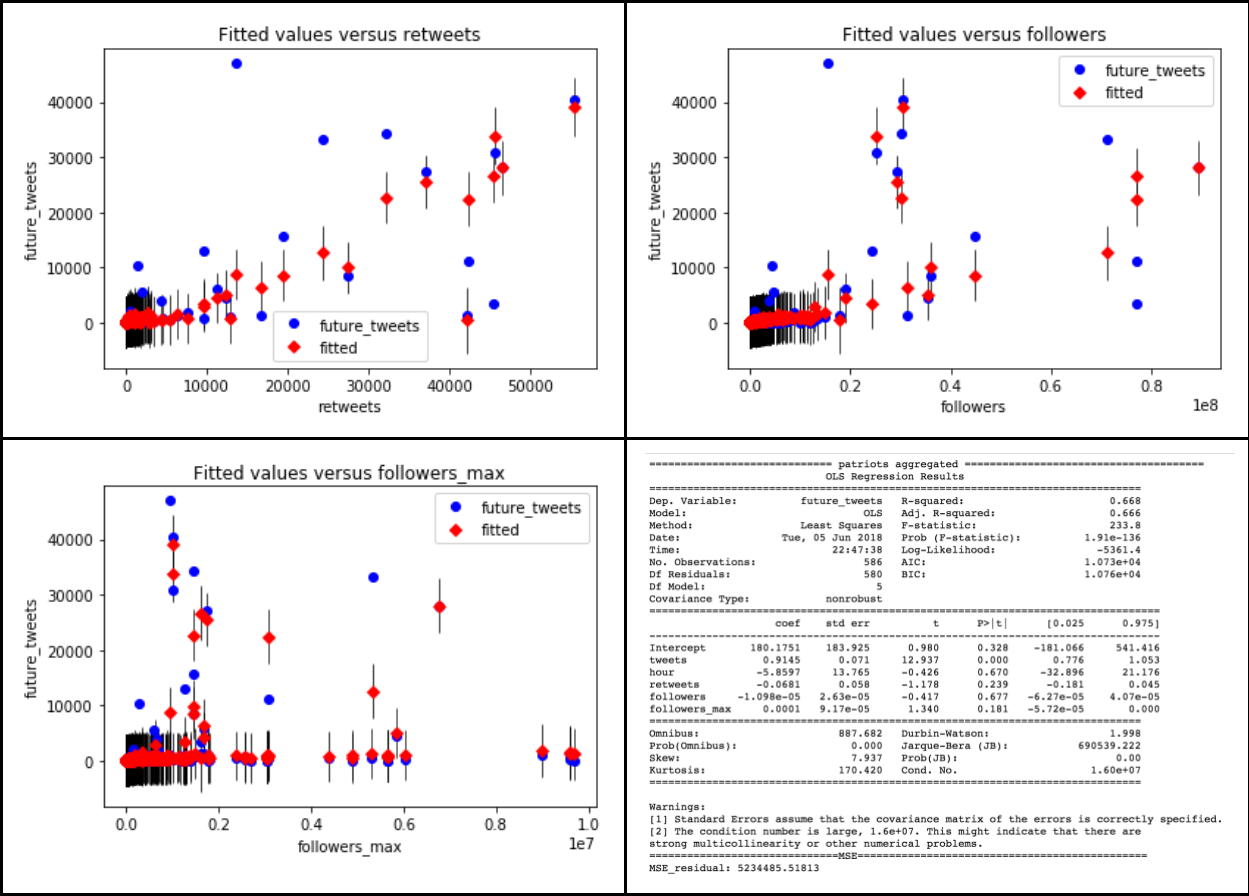
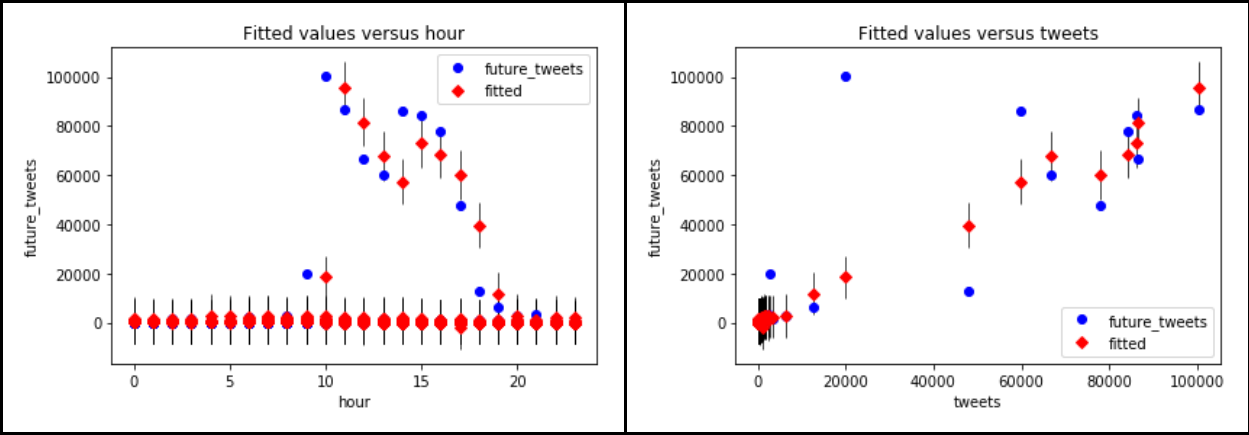


Figure 6: #sb49 Fitted Plots



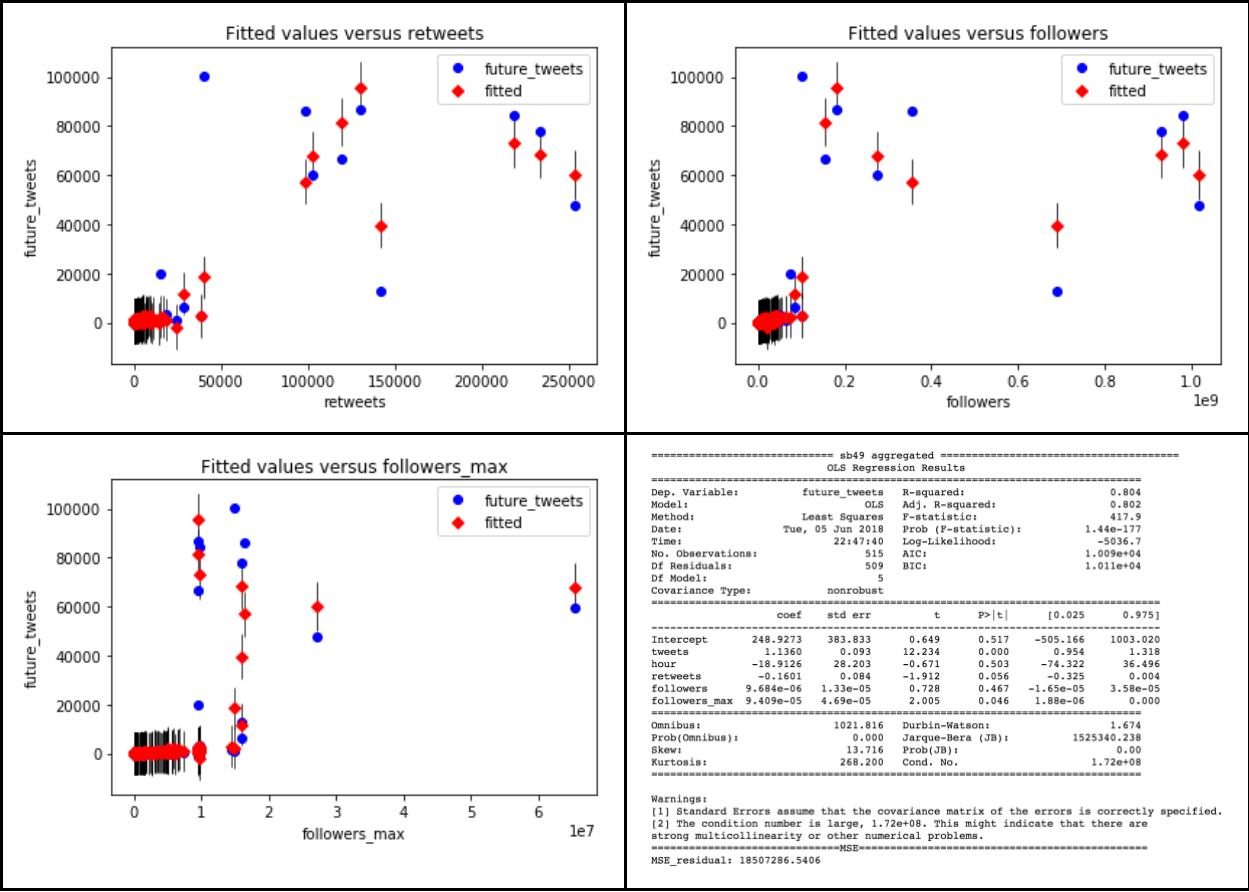
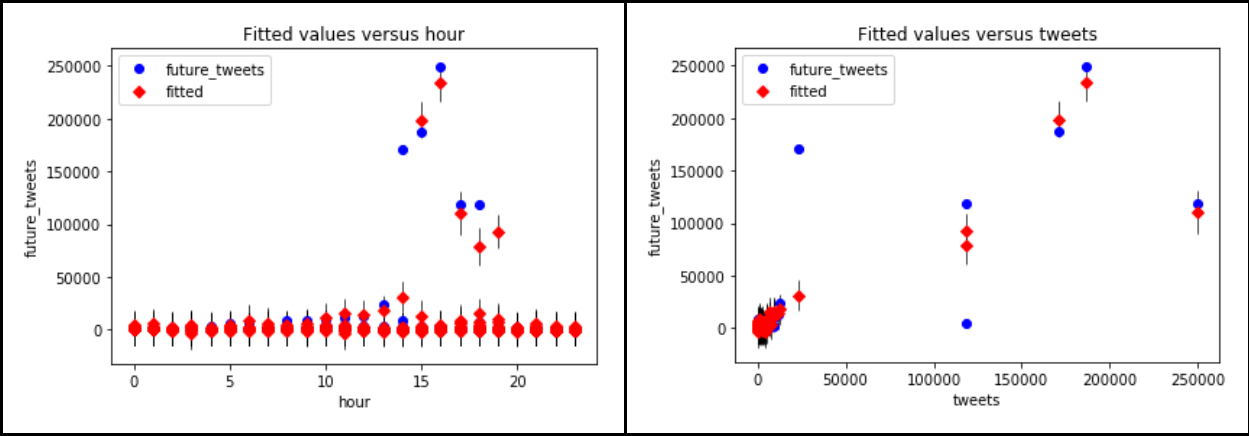
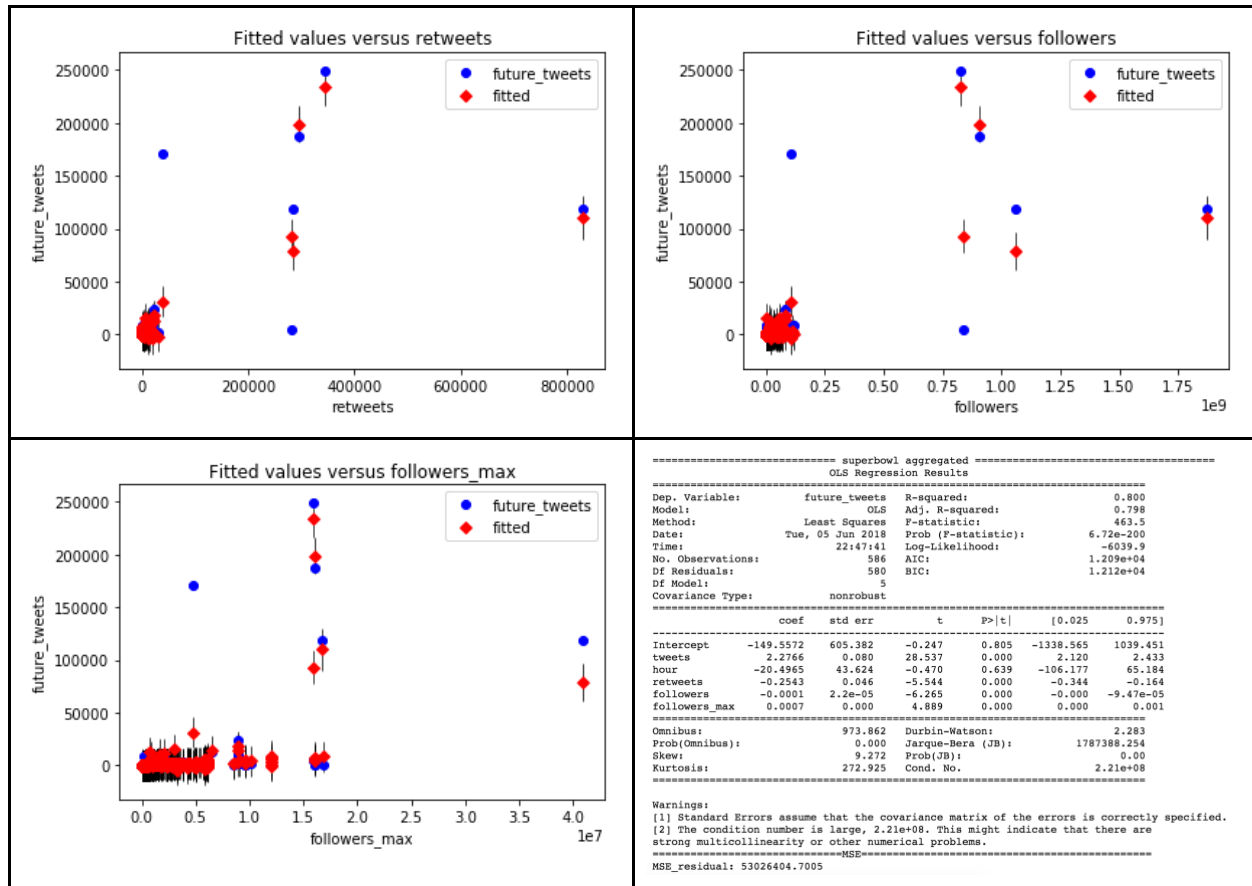


Figure 7: #superbowl Fitted Plots





In order to determine whether a feature is significant or not, we looked into *t-value* and *p-value*. T-value and p-value are linked together. T-test finds evidence of significant difference between the two population means (2- sample t-test) or between the population mean a hypothesized value (1-sample t-test). Basically, the t-test calculates the difference represented of standard error. The larger the t-value is (can be negative or positive), the greater the evidence of rejecting the null hypothesis, which implies that there is significant difference. However, t-value that is closer to 0 implies that there is no significant difference from the average. P-value is the probability of obtaining the t-value or higher. If p-value is very low (typically < 0.05), then it's safely to assume that we can reject the null hypothesis, which there is a statistical difference. Therefore, the larger absolute t-value and the smaller p-value, the greater the evidence against the null hypothesis.

In our analysis, we observed that *tweets* and *number of reweets* features are statistically significant in #gohawks dataset since the absolute t-values are high, and the p-value is lower than 0.05. Interestingly, we did not observe any feature that was statistically significant in #gopatritots dataset; although the *total number of tweets* feature seems to have low value, we did not consider

it to have any significant difference. The probable main driving reason here is that #gohawks dataset has more data points (total number of unique users in this case), while #gopatriots has a less. The model is most likely overfitting in #gohawks dataset. Further investigation is likely needed to understand the behavior of #gopatriots model.

Number of tweets, sum of followers, and max number of followers features are significant in #nfl dataset. Likewise, *number of tweets, sum of retweets, sum of followers and max number of followers* features are significant in #superbowl. *Number of tweets, sum of retweets, and max number of followers* features are significant in #sb49 dataset. Interestingly, *number of tweets* feature is the only feature that is significant in #patriots dataset. *Time of the day* feature consistently is not significant through all the 6 datasets. Therefore, we can conclude that time of the day is not a significant feature. Though from our observation, we could not see much of a pattern for the other features. One implication is that the models could be overfitting for some datasets, but further investigation is needed.

Table 3: Significance of Each Feature Using t-test and p-value for Each Hashtag

#gohawks		
Feature	t-statistic	p-value
Number of Tweets	7.767	0.000
Time of the Day	0.254	0.799
Total Number of Retweets	-3.113	0.002
Sum of the Number of Followers of the Users Posting the Hashtag	-2.407	0.016
Maximum Number of Followers of the Users Posting the Hashtag	0.403	0.687
#gopatriots		
Feature	t-statistic	p-value
Number of Tweets	0.937	0.349
Time of the Day	-0.239	0.812

Total Number of Retweets	2.223	0.027
Sum of the Number of Followers of the Users Posting the Hashtag	-0.424	0.672
Maximum Number of Followers of the Users Posting the Hashtag	-0.113	0.910
#nfl		
Feature	t-statistic	p-value
Number of Tweets	4.173	0.000
Time of the Day	0.093	0.926
Total Number of Retweets	-2.578	0.010
Sum of the Number of Followers of the Users Posting the Hashtag	4.573	0.000
Maximum Number of Followers of the Users Posting the Hashtag	-3.527	0.000
#patriots		
Feature	t-statistic	p-value
Number of Tweets	12.937	0.000
Time of the Day	-0.426	0.670
Total Number of Retweets	-1.178	0.239
Sum of the Number of Followers of the Users Posting the Hashtag	-0.417	0.677
Maximum Number of Followers of the Users Posting the Hashtag	1.340	0.181

#sb49		
Feature	t-statistic	p-value
Number of Tweets	12.485	0.000
Time of the Day	-0.666	0.506
Total Number of Retweets	-1.953	0.051
Sum of the Number of Followers of the Users Posting the Hashtag	0.743	0.458
Maximum Number of Followers of the Users Posting the Hashtag	2.056	0.040
#superbowl		
Feature	t-statistic	p-value
Number of Tweets	28.537	0.000
Time of the Day	-0.470	0.639
Total Number of Retweets	-5.544	0.000
Sum of the Number of Followers of the Users Posting the Hashtag	-6.265	0.000
Maximum Number of Followers of the Users Posting the Hashtag	4.889	0.000

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.

We then used the same linear regression model (the OLS in the Python statsmodel library) but added additional features for analysis. In addition to the hour, number of tweets, retweets,

followers, and the max number of followers of the users posting the tweets, we also tracked the number of urls posted in a tweet, the number of mentions in a tweet, and the number of hashtags in a tweet that occurred during every hour measured. We chose these additional features to measure because they were easy to count after aggregating and seemed like they would serve well in a linear model. Our reasoning is that since we expected as the number of tweets each hour increases then the aggregate count of these features would increase as well. We didn't want to remove any of the five original features since each one seemed like a reasonably good predictant for the number of tweets that occurred in the next hour. We used these features to determine which were the most significant in determining the number of tweets in the following hour.

Table 4: Mean Squared Error and Significance of Features for each Hashtag

Hashtag	Mean Squared Error (MSE)	Features and their p-value Significance
#gohawks	639814.157	{ 'followers': 1.015e-06, 'hashtags': 0.002, 'urls': 0.014, 'mentions': 0.032, 'followers_max': 0.077, 'tweets': 0.260, 'retweets': 0.412, 'hour': 0.803 }
#gopatriots	23558.694	{ 'mentions': 1.987e-13, 'retweets': 1.346e-05, 'urls': 0.001, 'followers_max': 0.334, 'followers': 0.486, 'tweets': 0.539, 'hashtags': 0.722, 'hour': 0.843 }
#nfl	205175.326	{ 'hashtags': 1.387e-31, 'tweets': 1.469e-25, 'urls': 3.266e-21, 'mentions': 0.018, 'retweets': 0.140, 'followers_max': 0.2121, 'hour': 0.238, 'followers': 0.311 }

#patriots	3830667.075	{'followers': 5.650e-25, 'followers_max': 1.195e-08, 'mentions': 8.288e-07, 'tweets': 5.559e-06, 'urls': 7.273e-05, 'hashtags': 0.000, 'retweets': 0.092, 'hour': 0.663}
#sb49	15885209.463	{'mentions': 1.541e-11, 'followers': 1.794e-10, 'urls': 2.191e-08, 'tweets': 3.418e-08, 'hashtags': 2.607e-05, 'followers_max': 0.010, 'retweets': 0.019, 'hour': 0.318}
#superbowl	33009634.383	{'mentions': 1.869e-43, 'retweets': 3.946e-34, 'tweets': 2.286e-08, 'hashtags': 1.286e-06, 'urls': 0.000, 'followers': 0.253, 'hour': 0.467, 'followers_max': 0.506}

For the #gohawks hashtag, followers was the only significant feature with a p-value under 0.001. For the #gopatriots hashtag, the total number of mentions and number of retweets were significant; for the #nfl hashtag, the number of hashtags, the number of tweets, and the number of urls were significant; for the #patriots the number of hashtags, the number of followers, the number of max followers, the number of mentions, the number of tweets, and the number of urls were significant; for the #sb49 hashtag, the number of mentions, the number of followers, the number of urls, the number of tweets, and the number of hashtags were significant; and for the #superbowl hashtag, the number of mentions, the number of retweets, the number of hashtags, and the number of urls were significant.

There does not appear to be a consistent feature that best predicts the number of tweets in the following hour. This is most likely due to the fact that that since the data is not linear, using a linear design made it hard to fit line to the data.

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?

Here are the top 3 features (in this case, the features with the smallest p-values) for each hashtag:

#gohawks: 'followers': 1.015e-06, 'hashtags': 0.002, 'urls': 0.014

#gopatriots: 'mentions': 1.987e-13, 'retweets': 1.346e-05, 'urls': 0.001

#nfl: 'hashtags': 1.387e-31, 'tweets': 1.469e-25, 'urls': 3.266e-21

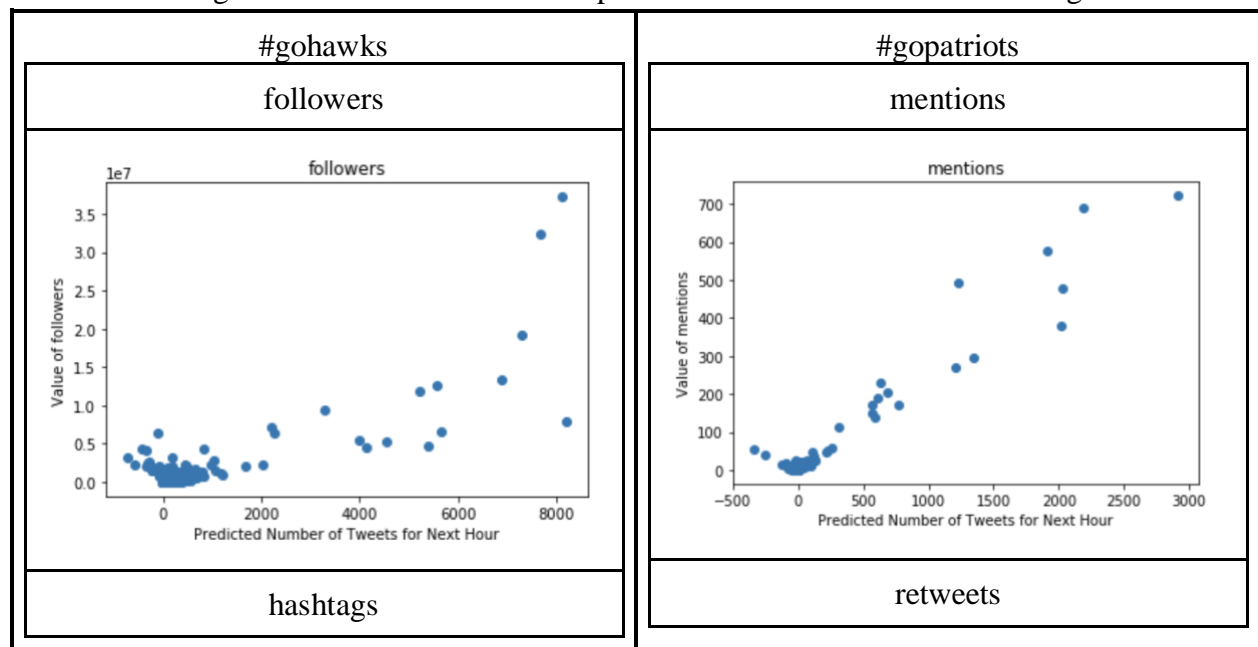
#patriots: 'followers': 5.650e-25, 'followers_max': 1.195e-08, 'mentions': 8.288e-07

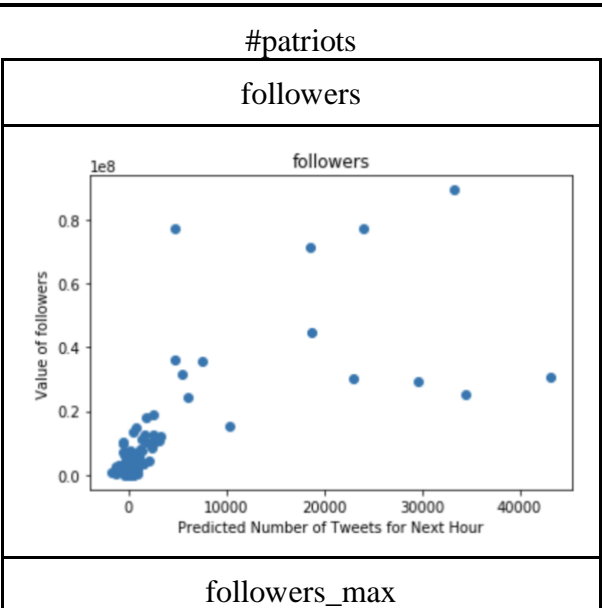
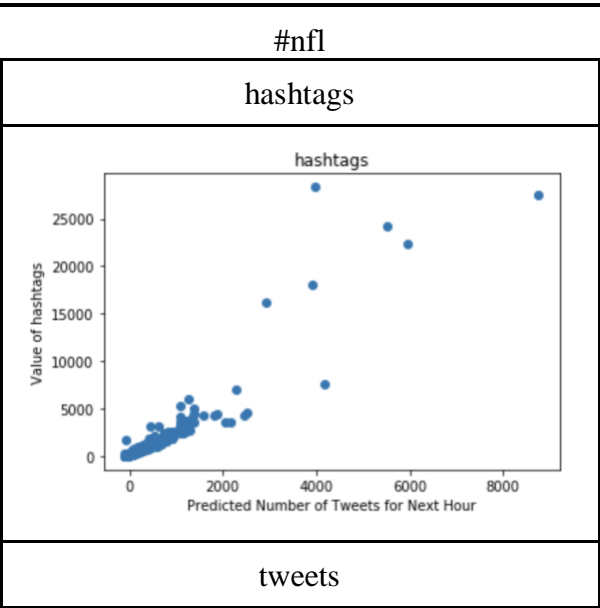
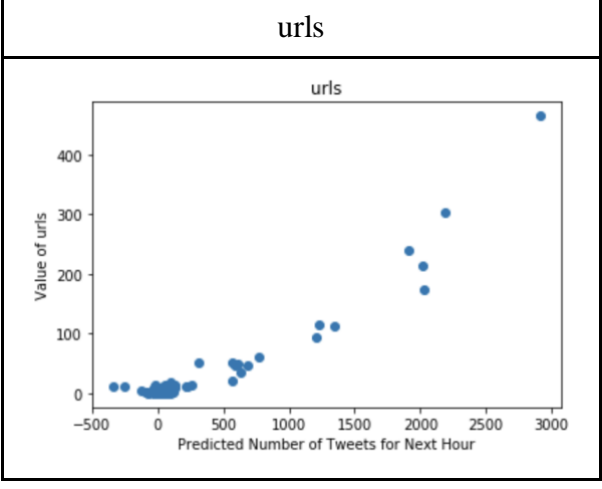
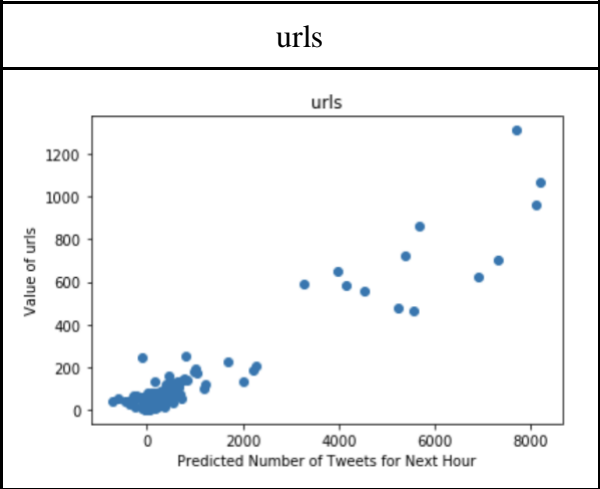
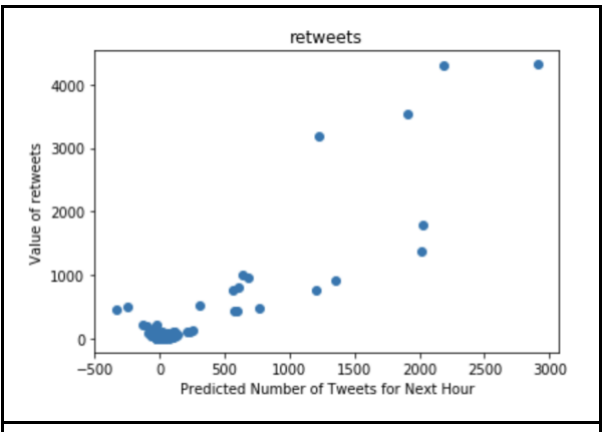
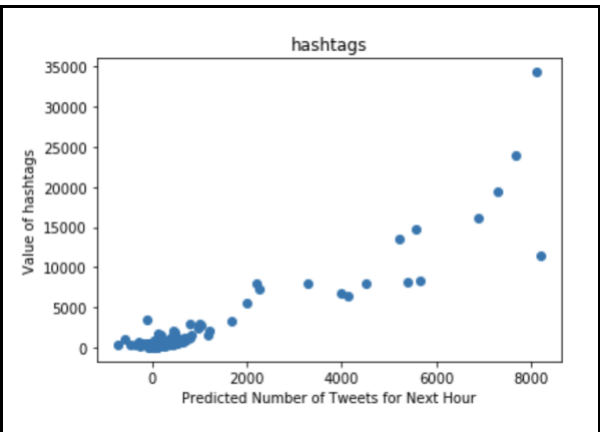
#sb49: 'mentions': 1.541e-11, 'followers': 1.794e-10, 'urls': 2.191e-08

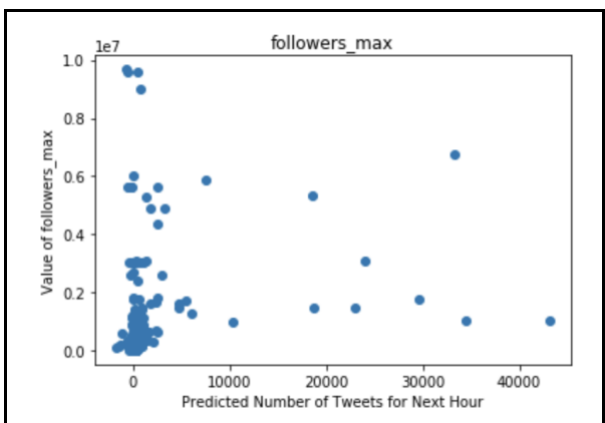
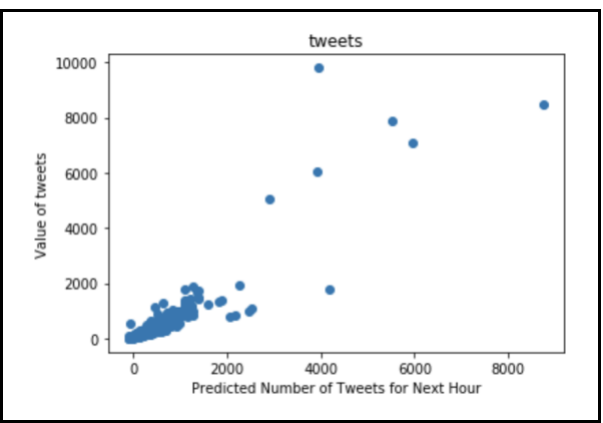
#superbowl: 'mentions': 1.869e-43, 'retweets': 3.946e-34, 'tweets': 2.286e-08

For each feature, we drew a scatter plot with the number of tweets during the next hour on the x-axis and the value of the feature being measured on the y-axis, seen in Figure 8.

Figure 8: Scatter Plots of the Top Three Predictants For Each Hashtag

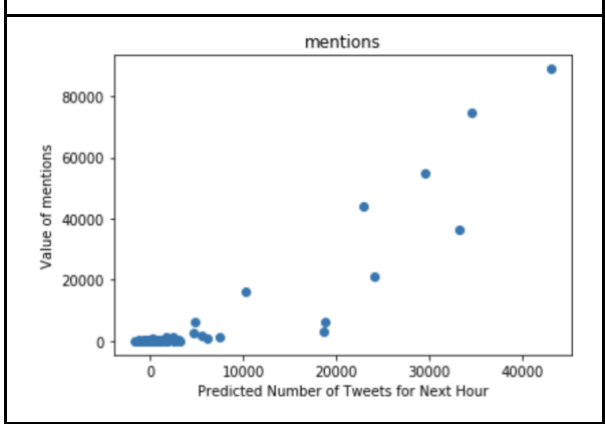
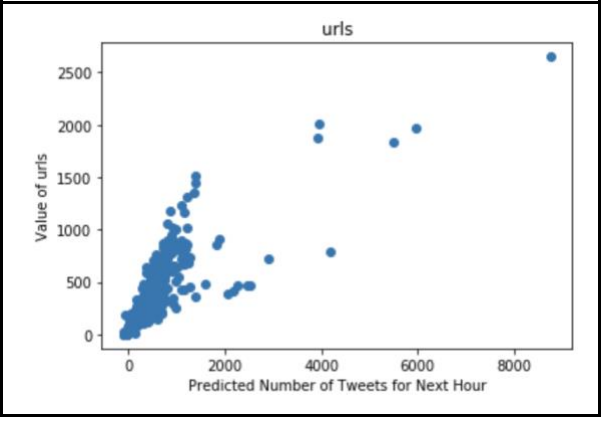






urls

mentions

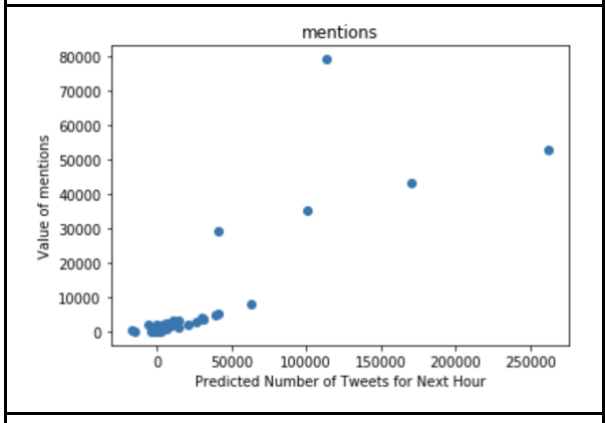
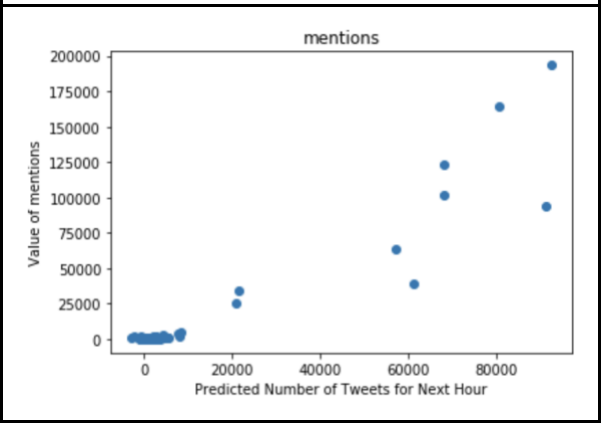


#sb49

#superbowl

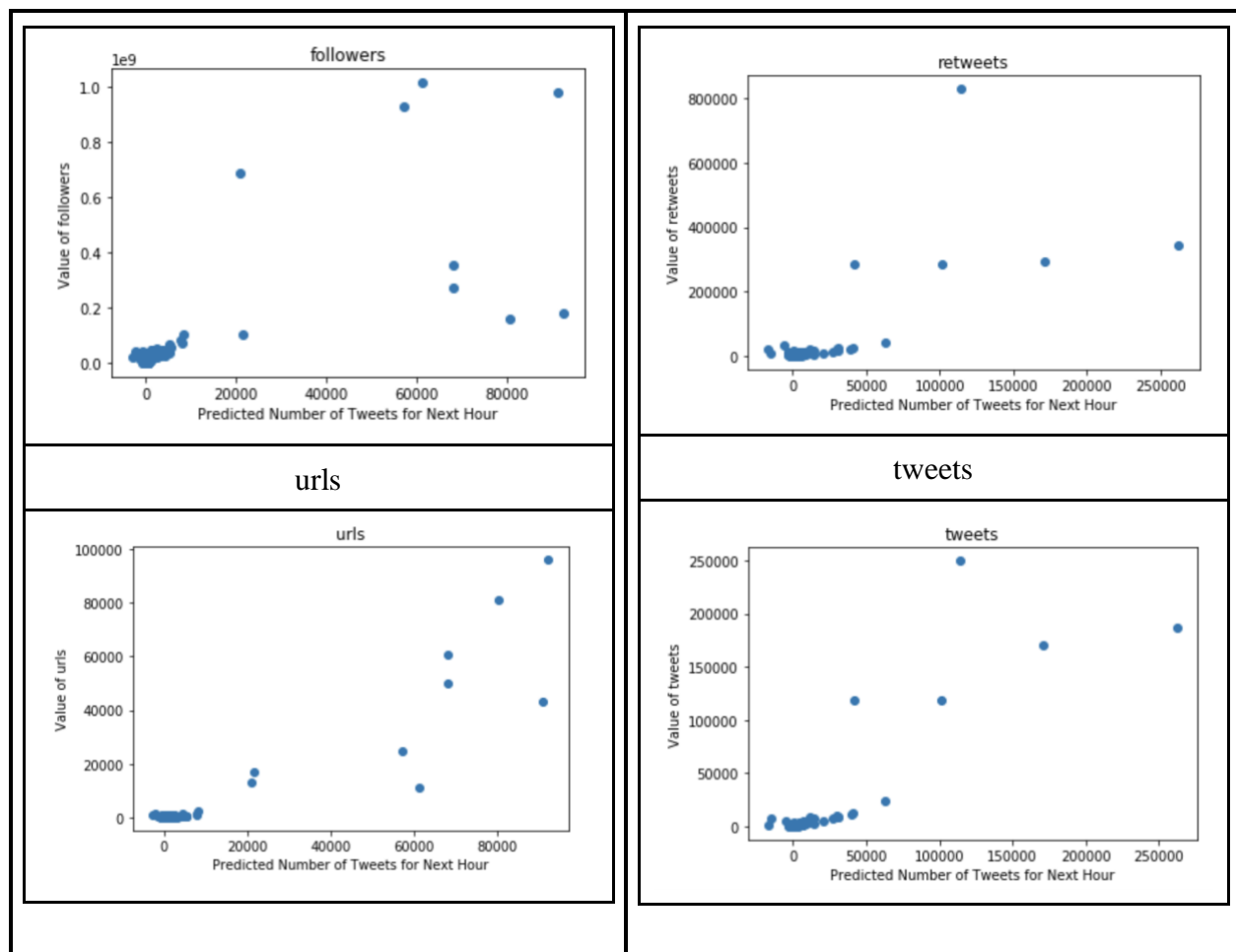
mentions

mentions



followers

retweets



Overall, all the scatter plots have a general trend upwards: as the predicted number of tweets for the next hour rises, the predictant also rises. This makes sense since we would expect if there are more tweets in the next hour, in the previous hour before would have lesser but close amount of tweets and a similar ratio of retweets, mentions, hashtags, and so on.

While the #gohawks and #gopatriots scatterplots have very similar shapes for all predictants, there seems to be a larger number of datapoints for #gohawks than for #gopatriots, and more of it seems to be concentrated towards the bottom where there's a fewer number of predicted tweets for the next hour. There's very little similarity in the top three features between the two hashtags, with only the feature url in common, and ranked third at that.

Since #gopatriots and #patriots are similar hashtags, we would guess that there would be a lot of similarity between the two. While #patriots mentions tended to look like the the plots of #gopatriots, both in terms of shape and sparsity, followers and followers_max had less

discenable shapes. Mentions was a top three feature for all of them, although it was first for #gopatriots and only third for #patriots. This could be because #gopatriots tends to imply a positive relationship with the Patriots team, but #patriots does not necessarily imply that the tweeter is rooting for that team.

#nfl has most of the data points concentrated at the bottom and getting sparser as the predictant increases. The hashtag measurement does look like it has the clearest shape out of the three features, and is ranked as the most significant. #sb49 has a comparatively very sparse amount of data points for all three features' scatter plots. Although the first ranked feature has only a few points, the shape looks quite linear. For #superbowl, for all three features the data points seem to be placed in a very similar fashion and shape and sparsity, even though the scale differs.

The regression coefficients for each of the eight features measured for each hashtag can be seen in Table 5.

Table 5: Regression Coefficients for Each of the Eight Features for Each Hashtag

Hashtag	Regression Coefficient (in ascending order)
#gohawks	Hour: -1.2398 Mentions: -1.0964 Tweets: -0.7887 Retweets: -0.0349 Followers: -0.00004 Followers_max: 0.0003 Hashtags: 1.1789 Urls: 3.9397
#gopatriots	Retweets: -1.0118 Hour: -0.2568 Hashtags: -0.1944 Followers_max: -0.0003 Followers: 0.0002 Tweets: 0.7729 Urls: 4.0730 Mentions: 5.1816
#patriots	Retweets: -1.0118 Hour: -0.2568 Hashtags: -0.1944 Followers_max: -0.0003 Followers: 0.0002

	Tweets: 0.7729 Urls: 4.0730 Mentions: 5.1816
#nfl	Tweets: -3.3482 Hour: -3.3229 Retweets: -0.0894 Followers_max: -3.934e-05 Followers: 2.531e-05 Hashtags: 1.1192 Urls: 1.2313 Mentions: 1.5180
#sb49	Urls: -7.0726 Tweets: -4.4153 Retweets: -0.0837 Followers_max: -0.0006 Followers: 0.0005 Hashtags: 1.4114 Mentions: 4.6991 Hour: 5.1634
#superbowl	Hour: -24.6846 Urls: -6.9439 Tweets: -5.9352 Followers_max: -0.0001 Followers: 0.0001 Retweets: 0.2383 Hashtags: 1.4921 Mentions: 5.1760

The regression coefficients don't seem to always agree with the trends in the plots. Since the regression coefficients represent the rate of change of one variable as a function of changes in the other. So if one variable generally increases as the other increases, the regression coefficient would have a positive value. Since all the scatter plots have a general positive trend, we would expect the regression coefficients to be positive as well. However, this is not the case. On closer examination, this can happen when there are multiple predictor variables. Because of intercorrelations between independent variables, the coefficient may be negative even though there is a positive slope associated with that variable.

4. Piece-wise linear regression

Due to the nature of a big event, the tweets referencing said event tend to increase in the days, weeks, or even months before, peak in the duration, and decline as time passes afterwards. Since there are records of when the Super Bowl began and ended, we can create a different regression model for each period of activity: before the Super Bowl, the peak hours during game day, and the time after those peak hours.

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.

Table 6 below is a summary of MSE and R-squared values for each case. In this part of the analysis, we looked at piece-wise linear regression by splitting each dataset into different timelines: before February 1st, 8:00 AM with 1-hour window, between February 1st, 8:00 AM and 8:00 PM with 5 minute window, and after February 1st, 8:00 PM with 1-hour window. Each timeline is trained against a linear regression model. Table 6 below is a summary of MSE and R-Squared values of each timeline for each hashtag.

The values overall seem to be consistent with the average of all three timelines against the models that we have trained for individual hashtags in Question 3 above. Interestingly, we observed that the MSE and R-squared values are significantly better in *After Feb. 1, 8:00pm* timeline for all hashtags. One possible explanation is that once the game is done, not many users are tweeting about the Super Bowl game. Thus, because of low volume of tweets, the model seemed to overfit in this timeline.

Table 6: MSE and R-squared for Every Timeline of Individual Hashtags

Hashtags	<i>Before Feb. 1, 8:00 a.m</i>		<i>Between Feb. 1, 8:00 a.m. and 8:00 p.m.</i>		<i>After Feb. 1, 8:00 p.m.</i>	
	MSE	R-squared	MSE	R-squared	MSE	R-squared
#gohawks	728194.29	0.303	74898.45	0.494	2218.20	0.832
#gopatriots	2236.32	0.570	14976.29	0.459	189.14	0.647
#nfl	66832.62	0.512	21929.60	0.818	18058.01	0.772

#patriots	342862.32	0.567	698545.64	0.711	11650.80	0.876
#sb49	8233.85	0.864	1372480	0.863	76934.32	0.800
#superbowl	523008.50	0.402	7158490.5	0.890	117813.99	0.842

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 combined the data from all six hashtags, and aggregated the data so that the total number of each category was calculated for each hour/five minute time frame for each period. For each period, a model was calculated and evaluated. The MSE and R-squared values are shown below in Table 7.

Table 7: Predictions of each Timeframe using Aggregate the data of all Hashtags

<i>Before Feb. 1, 8:00 a.m</i>		<i>Between Feb. 1, 8:00 a.m. and 8:00 p.m.</i>		<i>After Feb. 1, 8:00 p.m.</i>	
MSE	R-squared	MSE	R-squared	MSE	R-squared
4487889.57	0.410	18128477.77	0.848	467316.23	0.861

As expected, the MSE values for all three categories is much higher than for the individual hashtags, but follows a similar trend where the MSE and R-squared values are significantly better in *After Feb. 1, 8:00pm* timeline than the other two periods. The R-squared values also follow the same trend as the individual hashtags, with the *Before Feb. 1, 8:00 a.m* period having the lowest R-squared value and the *After Feb. 1, 8:00 p.m.* being the highest. The overall R-squared and MSE values seem to be within range with Question 3.

Table 8 below is our prediction for the next hour for each timeline. We predicted that the next hour after the *Before Feb. 1, 8:00am* period has about 9474 tweets, the next 5 minutes after the *between Feb. 1, 8:00am and 8:00pm* period has about 2760 tweets, and the next hour after the *After Feb. 1, 8:00pm* period has about 161 tweets. To check out accuracy, we compared these values to the actual time frame we're predicting right after that period. It turns out that the next hour after the *Before Feb. 1, 8:00am* period has 8213 tweets and the next 5 minutes after the *between Feb. 1, 8:00am and 8:00pm* period has 650 tweets. The the next hour after the *After*

Feb. 1, 8:00pm period wasn't available, but we took the number of tweets in the hour before that, which has 52 tweets as a good estimate of a "true" value.

Table 8: Predictions and Actual* Values of each Timeframe for Combined Aggregates

TimeFrames	Unit	Prediction	Actual
<i>Before Feb. 1, 8:00 a.m</i>	<i>Next 1 hour</i>	9474.33	8213
<i>Between Feb. 1, 8:00 a.m. and 8:00 p.m.</i>	<i>Next 5 minutes</i>	2759.66	650
<i>After Feb. 1, 8:00 p.m.</i>	<i>Next 1 hour</i>	161.44	(closest) 52

**Note: Since the data past the third period does not exist, were using the last hour of aggregate data to estimate the "actual" data in the hour afterwards*

It appears that our model is for the most part reliable, but tends to over-predict the number of tweets in the next hour. This could be because since were passing in values from the previous hour, it throws the accuracy of the model off. However, attempts to fix this would be circular, since if we wanted to predict the value of another variable, we would have to know the exact values of the other variables, and so on. The predicted values seem reasonably close in the *Before Feb. 1, 8:00 a.m* and *After Feb. 1, 8:00 p.m.* periods, but it's not very accurate with respect to the *Between Feb. 1, 8:00 a.m. and 8:00 p.m.* We hypothesize that the model for this period is not as accurate because there is a lot of variance in the number of tweets every five minutes, leading to a model that is not as accurate as the other two.

5. Nonlinear regressions

Ensemble Methods

In this part of the project, we turn to nonlinear regression, ensemble methods in particular. The formulas for the two estimators that we're using, `RandomForestRegressor` and `GradientBoostingRegressor`, can be seen in detail at the beginning of this report.

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.

In this part of the analysis, we performed grid search to find the optimum parameter set for RandomForestRegressor and GradientBoostingRegressor. The parameter set includes: max_depth, max_features, min_samples_leaf, min_samples_split, and n_estimators. The cross-validation is set to 5-folds, and the scoring is set to mean squared error (negated). Note that mean squared error (negated) is the opposite of mean squared error. Thus, larger value (less negative) is better than smaller value (more negative). In Tables 9 and 10 below, we observed that the absolute MSE scores are within the -470k to -565k range, which are smaller than OLS MSE scores above. This shows that random forest and gradient boosting regressors have better performance than OLS model, so we would say that the test errors (the MSE score on the validation fold) look pretty good. One explanation is that our aggregated tweets data do not behave linearly since the users do not tweet and retweet at a constant rate (per hour). If users tweets and retweets consistently throughout the superbowl event, then our OLS model could have performed better. As pointed out earlier, we did notice multiple spikes especially before and during gametime. Therefore, our non-linear models have a better edge in performance.

Table 9: Optimized Parameters for Random Forest Regressor

Hashtags	Optimized Parameters for Random Forest Regressor					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	100	sqrt	1	2	200	-471817.29
#gopatriots	40	auto	4	2	400	-38790.07
#nfl	80	sqrt	2	10	600	-251681.08
#patriots	20	auto	1	2	400	-5328e+06

#sb49	10	auto	4	5	200	-2.76e+07
#superbowl	10	auto	4	2	200	-1.22e+08

Table 10: Optimized Parameters for Gradient Boosting Regressor

Hashtags	Optimized Parameters for Gradient Boosting Regressor					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	200	sqrt	2	2	800	-472547.69
#gopatriots	60	sqrt	4	5	1400	-16727.13
#nfl	10	sqrt	2	10	400	-279380.50
#patriots	60	auto	4	2	200	-4.42e+06
#sb49	10	sqrt	1	2	1000	-2.66e+07
#superbowl	10	auto	1	5	800	-1.16e+08

In Table 11, we observed that R-squared is surprisingly high for both our random forest and gradient boosting models. It seems that our non-linear models are almost perfect basing on our observed R-squared scores, however, they are based on the assumption that the model is fitting linearly. Since we are looking at non-linear models, using R-squared is not the correct measurement to compare thus we would have to rely on a different measurement.

Table 11: R-Squared Scores for Random Forest and Gradient Boosting for each Hashtag.

HashTag	Random Forest R-Squared	Gradient Boosting R-Squared
#gohawks	0.9416	0.9886
#gopatriots	0.7758	0.9999
#nfl	0.7072	0.9999
#patriots	0.9538	0.9930
#sb49	0.8693	0.9999

#superbowl	0.7488	1.0
------------	--------	-----

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

Table 12 and Table 13 below are the optimized parameters for Random Forest Regressor and Gradient Boosting Regressor respectively using the entire dataset. We observed that the negated MSE score seem to perform worse than OLS counterpart.

Table 12: Optimized Parameters for Random Forest Regressor Entire Dataset

Hashtags	Optimized Parameters for Random Forest Regressor					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
Entire dataset	80	sqrt	1	5	200	-1.98e+08

Table 13: Optimized Parameters for Gradient Boosting Regressor Entire Dataset

Hashtags	Optimized Parameters for Gradient Boosting Regressor					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
Entire dataset	None	sqrt	2	5	1600	-3.16e+08

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?

Table 14 through Table 19 below summaries the optimized parameters for Gradient Boosting and Random Forest Regressor for each timeline of each hashtag. We observed that the MSE scores are much lower than OLS model thus added to our confidence that non-linear models are

better models than linear regression model. Therefore, the cross-validation test error did change in this scenario. Furthermore, random forest and gradient boosting regressors are better at distinguishing outliers which can significantly affect the outcome. In OLS model, it cannot distinguish outliers thus they could throw off overall R-Squared scores. These parameters seem to be quite close to those found in Question 9, so we would say that the best parameters that we found agree with our previous findings.

Although R-Squared for these models are nearly perfect in Table 20, they do not provide good performance indicator in non-linear models. In fact, R-Squared value should only be used for linear regression model.

Table 14: Optimized Parameters for Gradient Boosting *Before Feb. 1, 8:00 a.m*

Hashtags	Optimized Parameters for Gradient Boosting Regressor <i>Before Feb. 1, 8:00 a.m</i>					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	200	sqrt	4	2	200	-669703.80
#gopatriots	80	auto	4	2	800	-1887.97
#nfl	40	auto	1	10	400	-61197.02
#patriots	20	sqrt	4	2	200	-428439.87
#sb49	80	sqrt	1	2	600	-13511.81
#superbowl	200	auto	2	2	200	-522936.47

Table 15: Optimized Parameters for Gradient Boosting *Between Feb. 1, 8:00 a.m. and 8:00 p.m.*

Hashtags	Optimized Parameters for Gradient Boosting Regressor <i>Between Feb. 1, 8:00 a.m. and 8:00 p.m.</i>					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	200	sqrt	4	10	200	-81484.55
#gopatriots	None	auto	4	2	200	-12208.11

#nfl	100	auto	2	2	200	-25590.90
#patriots	60	sqrt	4	2	200	-783687.39
#sb49	40	sqrt	2	2	200	-1.152e+06
#superbowl	60	sqrt	2	10	200	-8.678e+06

Table 16: Optimized Parameters for Gradient Boosting *After Feb. 1, 8:00 p.m.*

Hashtags	Optimized Parameters for Gradient Boosting Regressor <i>After Feb. 1, 8:00 p.m.</i>					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	20	auto	4	2	200	-3298.67
#gopatriots	60	sqrt	1	2	200	-20.728
#nfl	20	auto	2	2	400	-19334.23
#patriots	40	sqrt	1	5	400	-16509.82
#sb49	None	auto	4	5	200	-38456.43
#superbowl	20	auto	2	2	200	-114878.29

Table 17: Optimized Parameters for Random Forest *Before Feb. 1, 8:00 a.m*

Hashtags	Optimized Parameters for Random Forest Regressor <i>Before Feb. 1, 8:00 a.m</i>					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	60	auto	4	10	600	-581925.12
#gopatriots	40	auto	4	10	200	-2330.69
#nfl	None	auto	4	10	400	-59611.48

#patriots	60	sqrt	4	5	200	-452414.57
#sb49	20	sqrt	4	5	600	-15304.64
#superbowl	200	sqrt	4	5	200	-519189.67

Table 18: Optimized Parameters for Random Forest *Between Feb. 1, 8:00 a.m. and 8:00 p.m.*

Hashtags	Optimized Parameters for Random Forest Regressor <i>Between Feb. 1, 8:00 a.m. and 8:00 p.m.</i>					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	40	sqrt	4	10	200	-76476.01
#gopatriots	60	auto	4	10	200	-12127.71
#nfl	None	auto	2	10	200	-28935.16
#patriots	20	sqrt	4	10	200	-779955.97
#sb49	60	sqrt	1	2	200	-1.085e+06
#superbowl	80	auto	1	10	200	-8.059e+06

Table 19: Optimized Parameters for Random Forest *After Feb. 1, 8:00 p.m.*

Hashtags	Optimized Parameters for Random Forest Regressor <i>After Feb. 1, 8:00 p.m.</i>					
	Max depth	Max features	Min samples leaf	Min samples split	N estimators	Negated MSE scores
#gohawks	100	sqrt	4	5	200	-3346.87
#gopatriots	10	sqrt	1	2	200	-34.729
#nfl	60	auto	2	2	200	-20237.71
#patriots	10	auto	2	2	200	-12995.05
#sb49	None	auto	2	2	400	-43060.47
#superbowl	None	sqrt	1	2	200	-140128.48

Table 20: Splitted Timeframe for Each Hashtag Non-Linear Regressors

Hashtag	Before Feb. 1, 8:00 a.m		Between Feb. 1, 8:00 a.m. and 8:00 p.m.		After Feb. 1, 8:00 p.m.	
	Random Forest R-Squared	Gradient Boosting R-Squared	Random Forest R-Squared	Gradient Boosting R-Squared	Random Forest R-Squared	Gradient Boosting R-Squared
#gohawks	0.5613	0.9921	0.9994	0.6689	0.7754	0.9896
#gopatriots	0.6472	0.9998	0.9995	0.7424	0.9541	0.9993
#nfl	0.7104	0.9999	0.8934	0.9999	0.9432	0.9999
#patriots	0.8937	0.9997	0.7983	0.9994	0.9433	0.9999
#sb49	0.9999	0.9467	0.9824	0.9999	0.9323	0.9999
#superbowl	0.59957	0.9999	0.9339	0.9999	0.9746	0.9999

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.

We took our aggregated dataset and modified our “future tweets” column by replacing any NA values with a valid value from the row before it to allow every single row in the dataset to be analyzed.

We then regressed the aggregated data with sklearn’s MLPRegressor estimator, using various values for the hidden layer size. We fit the model using the model created from the aggregated data and then used that model to predict the tweets in the next hour. The architectures tried, as well as their MSE values from fitting the entire aggregated data, is seen below in Table 21.

Table 21: MLPRegressor Architectures Used and Corresponding MSE Values

Architecture (hidden layer size) used	MSE Value
---------------------------------------	-----------

(100,)	9669648158464.504
(50,)	382002807209.09424
(10,)	658261600687.1708
(5,)	1257401693.7368064
(100, 100,)	2664946855309.2593
(100, 50,)	100900355488.10873
(100, 10,)	2023792349424.1196
(100, 5,)	339996133034.48926
(50, 100,)	4319230048446.4873
(50, 50,)	28568943065942.688
(50, 10,)	1853252968054.0874
(50, 5,)	9960074294446.94
(10, 100,)	20124689454516.723
(10, 50,)	24759158585890.094
(10, 10,)	6214726821.2744465
(10, 5,)	757656423.7559854
(5, 100,)	5098557324050.35
(5, 50,)	1845923859.1372094
(5, 10,)	53793142105.37786
(5, 5,)	817725877.7263925

The MSE values obtained using MLPRegressor are significantly bigger than when using a linear regression model.

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

We used StandardScaler and scaled the data before using MLPRegressor with our best architecture, (10, 5,). The MSE value actually increases using the Standard Scaler, seen below. Scaling is not always a good method for dealing with frequencies since it can often make the variables used indistinguishable from one another after scaling. This is especially prevalent in larger datasets, like the ones we're using for this project. Instead, preserving the relative frequency of terms can lead to better performance.

MSE value: 812863675.9628963

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

Table 22 below is the optimized parameter using MLP Regressor against the full dataset (combined all hashtags into one). We observed that the optimized hidden layer sizes parameter is 17, and the negated MSE score is at $-5.97e+10$, which is much higher than OLS, Random Forest, and Gradient Boosting models. Furthermore, we observed that the hidden layer sizes parameter changes each run. As mentioned above, MLP Regressor can lead to different validation accuracy based on different random weight initializations, and its hidden layers have a non-convex loss function where there exists more than one local minimum. Since we're looking only at hidden layers sizes hyperparameter, we should explore other hyperparameters such as tuning number of hidden neurons, layers, and iterations which can help better with the performance.

Table 22: Optimized Parameter using MLP Regressor

Time Frame	Hidden Layer Sizes	Negated MSE
<i>Before Feb. 1, 8:00 a.m.: 1-hour window</i>	17	$-5.97e+10$
<i>Between Feb. 1, 8:00 a.m. and 8:00 p.m.: 5-minute window</i>	25	$-6.157e+11$
<i>After Feb. 1, 8:00 p.m.: 1-hour window</i>	17	$-8.44e+10$

6. Using 6x window to predict

For the last section in Part 1, we used a different set of data for analysis. In this dataset, each file contains a hashtag's tweets, but with a 6x-length time frame. Each file contains either a sample 6-hour period or a 30-minute period that lies within the before/during/after active phases used before.

We fitted a model on the aggregate of the training data for all hashtags and predicted the number of tweets in the next hour for each text file.

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.

For this problem, we download the additional data set and turned it into a dataframe with the same values used as before:

- 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

We also made sure that we were analyzing the data using Pacific Standard Time, and aggregated the data for each file so that the total number during an hour or 5 minutes appeared for each column. Unique to this problem, we also dropped any rows that contained NA values. For each dataset, we combined the data so that it also contained the data in that period that occurred before it, so as to give our models more data to train on.

To find out which model we wanted for this problem, for each of the four models we had previously used (linear regression, MLP regressor, random forest regressor, and gradient boosting regressor) we created a model and fit it to our data and target future tweets. Then we used cross validation and took the aggregate scores to see which one was the best to use with this data. The results are seen below in Table 23.

Table 23: Aggregated Cross-Validation Values for Different Models

LinearRegression	MLPRegressor	RandomForestRegressor	GradientBoostingRegressor
-60610.43	-10771092292.80	-379.56	-327.87

As you can see, the GradientBoostingRegressor performed the best with the data, so that is what we used to predict the number of tweets in the next time period. We took the file created from each aggregated data frame and used the gradient boosting regressor models we had trained to predict the number of tweets in the next hour. The results can be seen below in Table 24.

Table 24: Predictions on the Number of Tweets in the Next Time Windows

Time Window	Predicted Tweets in Next Time Window
sample0_period1	120.47
sample1_period1	845.99
sample2_period1	56.88
sample0_period2	1123.02
sample1_period2	873.08
sample2_period2	0
sample0_period3	88.55
sample1_period3	30.87
sample2_period3	35.01

As you can see above, the number of tweets in the next time window appear to be reasonable given the numbers of tweets in previous time periods. The calculations for the middle period seem to be the most skewed, but this could be because the datasets for the middle periods were given in five minute intervals, but the problem instructed to find the next hour.

Part 2: Fan Base Prediction

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.*

To determine if the location was in Washington, Massachusetts, or neither, we isolated the location field from the data and checked if any of the words within the location field were in the

set {"Massachusetts", "Boston", "MA"}. If so, then the user would be flagged to be from Massachusetts. If not, we also checked if any of the words within the location field were in the set {"Washington", "Seattle", "WA"}. If so, then the user would be flagged to be from Washington. If neither of these conditions were satisfied, the user would be flagged as “neither.”

In the remainder of the analysis, we only looked at tweets from users labeled as “from Massachusetts” or “from Washington”. The raw number of tweets from users from each state is as follows:

Table 25: Number of Tweets from WA

Number of Tweets from WA	Number of Tweets from MA
32441	21195

When we showed some example tweets from each state, however, we saw a good amount of duplicate tweets (tweets with the same text, perhaps directed at another user). We felt that this would lead to inflated prediction accuracy during classification, since the test set would have identical examples in the training set. We thus opted to remove any tweets that were not unique (after cleaning the tweets, which will be described in the next section).

Table 26: Number of Unique Clean Tweets from WA

Number of Unique Clean Tweets from WA	Number of Unique Clean Tweets from MA
25502	20186

With these cleaned dataset, we have not only removed unwanted biases -- we have also obtained a more balanced dataset. The best a “dumb classifier” could do by guessing only the majority class is 0.56. At the bare minimum, we are hoping to obtain an accuracy higher than this.

Question 15 (continued)

- 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 learnt in project 1). Try different classification algorithms (at least 3). For each, plot ROC curve, report confusion matrix, and calculate accuracy, recall and precision.*

Pre-processing Data

The first step to creating the binary classifier is to make sure that the data is clean. Here is an example of the raw data:

Table 27: Original Tweet vs. Cleaned Tweet

Original Tweet	Cleaned Tweet
Our @ButchStearns talked #Patriots w/3X #SuperBowl champ Matt Light Full interview @ 10 http://t.co/6s9XqUUcyj #FOX25 http://t.co/iTAFnwQc2G	talked patriots w3x superbowl champ matt light full interview 10 fox25
@DougBaldwinJr just this #mediocre #SuperBowlChampion wide receiver #GoHawks #RePete http://t.co/HUwZhG6JwQ's video http://t.co/wE4PP5YaHd	mediocre superbowlchampion wide receiver gohawks repete video

We can see that in the original tweets, there are a lot of features that would be bad for binary classification. URLs likely will be unnecessary features, and capitalization or punctuation would cause “#repete” and “repete” to be different features, even though they should be aggregated to one.

By removing URLs, non-alphanumeric characters, and making everything lowercase, we think the features will be better suited for classifying these tweets into the two locations.

Generating Feature Vectors

To generate the feature vectors for binary classification, we first split the dataset into an 80/20 train/test split. We then took the training set and vectorized it using the tf-idf approach, with a minimum document frequency of 15. The resulting vectorized training set had a shape of (36550, 2067), and the test set had a shape of (9138, 2067).

Logistic Regression Classifier

With our tf-idf vectorized dataset, we trained a logistic regression classifier. There are two parameters we tuned for using 5-fold cross validation: inverse-regularization strength, and loss function (L1 vs L2). We first show the results of our hyperparameter tuning:

Table 28: Logistic Regression Penalty

Penalty	C = 0.001	C = 0.01	C = 0.1	C = 1	C = 10	C = 100	C = 1000
---------	-----------	----------	---------	-------	--------	---------	----------

L2	0.562052	0.693653	0.726320	0.725034	0.714884	0.712804	0.712558
L1	0.559152	0.660848	0.712093	0.728345	0.714856	0.712914	0.712503

From our cross-validation results, we see that an inverse-regularization strength value of 1 and L1 regularization yields the best results. Using these parameters, we train a logistic regression classifier and run metrics on its attempt to predict locations of tweets in the test set.

Table 29: Logistic Regression Results

Logistic Regression Classification Results (Test Set)			
<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F-1</i>
0.721930400525	0.705470530772	0.855478775913	0.773266708307

	Predicted label: Massachusetts	Predicted label: Washington
True label: Massachusetts	2264	1809
True label: Washington	732	4333

We will show the ROC curve below in a joint plot to compare the different classifiers.

Random Forest Classifier

The next classifier we chose to investigate was the random forest classifier. In parameter tuning, we looked for the minimum max_depth value such that any higher max_depth values would lead to negligible increases (or substantial decreases) in accuracy. The following chart shows the result of this parameter optimization:

Table 30: Max_depth Parameter Values

	Max_depth Values						
	5	10	15	20	30	40	50
CV Acc	0.610643	0.678194	0.704487	0.710780	0.720903	0.719425	0.722845

With an optimal max_value of 30 as determined by 5-fold cross validation, we now report the test results using this parameter:

Table 31: Random Forest Classification Results

Random Forest Classification Results (Test Set)			
<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F-1</i>
0.712300284526	0.671500985638	0.941559723593	0.783923728117

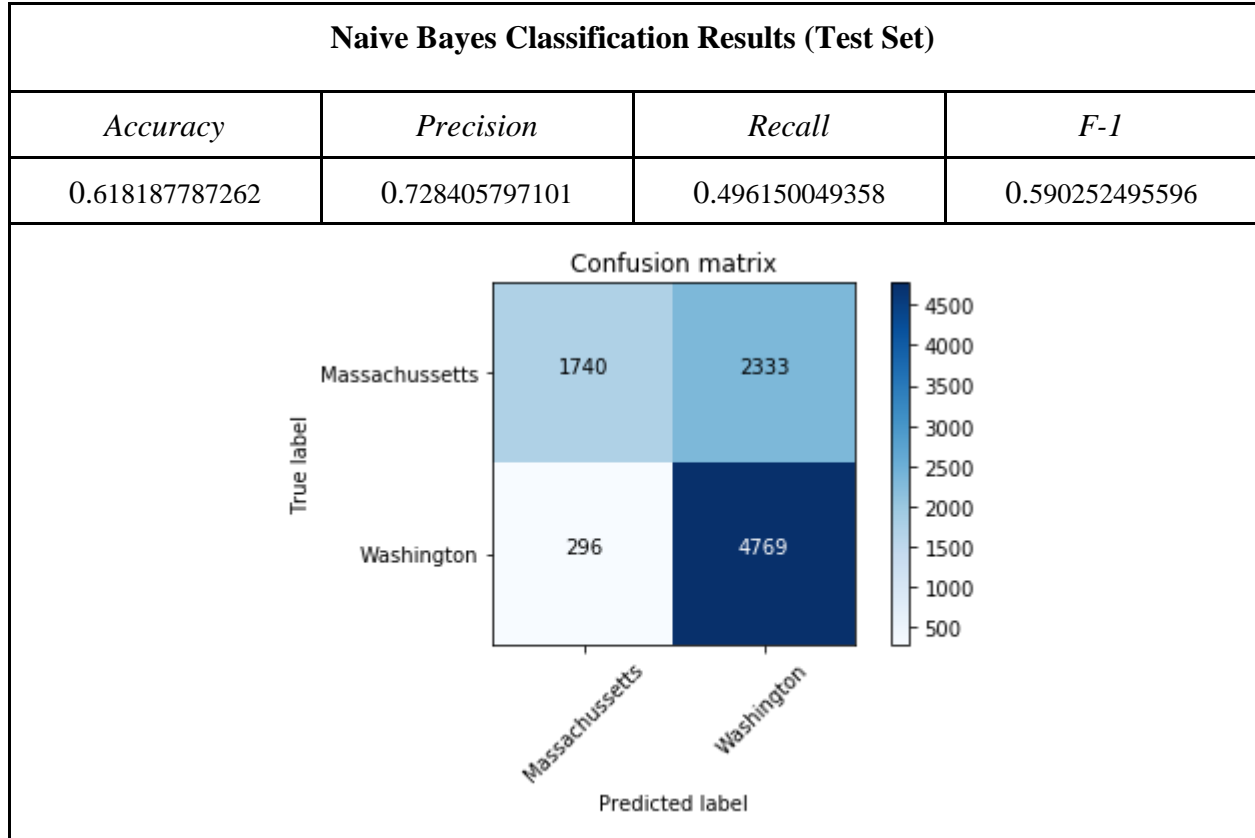
Confusion matrix

True \ Predicted	Massachussetts	Washington
Massachussetts	1740	2333
Washington	296	4769

We will show the ROC curve below in a joint plot to compare the different classifiers.

For our 3rd classifier, we use a Naive Bayes classifier to classify the test results.

Table 32: Naive Bayes Classification Results



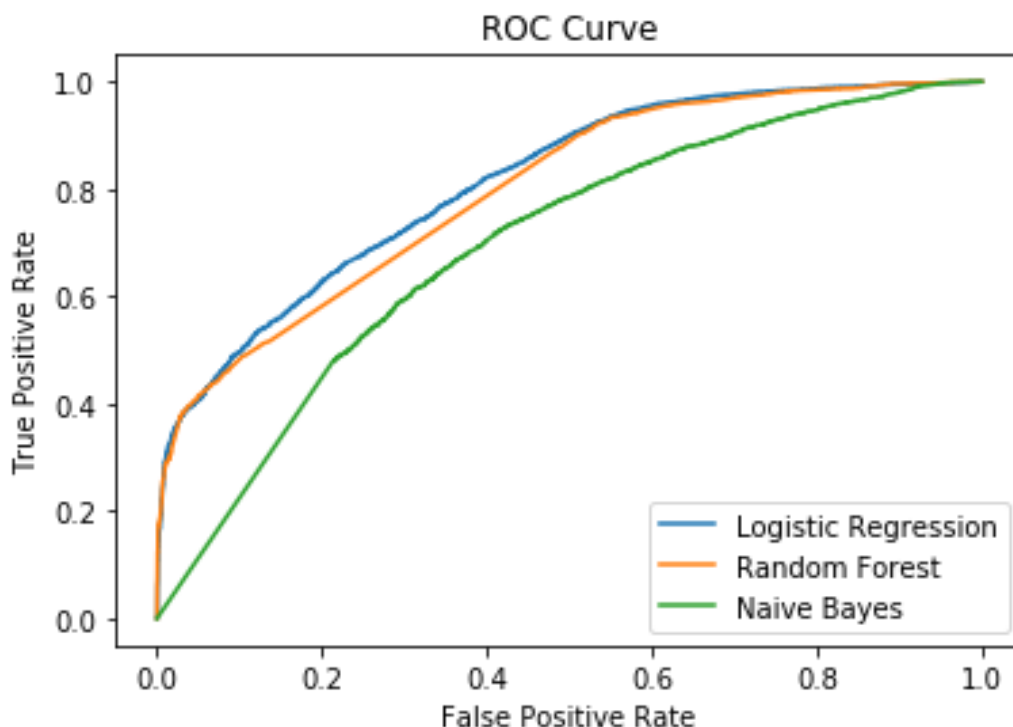
To compare the 3 classifiers, we aggregate the results in the following chart:

Table 33: Aggregated Classification Results

	Classification Results (Test Set)			
	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F-1</i>
Log. Regression	0.721930400525	0.705470530772	0.855478775913	0.773266708307
Random Forest	0.712300284526	0.671500985638	0.941559723593	0.783923728117
Naive Bayes	0.618187787262	0.728405797101	0.496150049358	0.590252495596

Since there isn't one classifier that consistently has the highest metrics in all categories, we plot an ROC curve to compare the classifiers:

Figure 9: ROC Curve



From the ROC Curve, we can see that all 3 classifiers perform better than random guessing. We can also see that for this dataset, the logistic regression classifier with L1 regularization and an inverse-regularization strength parameter of 1 has the best performance.

On a qualitative scale, there are several reason to expect why a classifier wouldn't have extremely high performance metrics. The data in itself is quite messy, as there are many tweets that simply do not provide any context of where the user is located. Take for example, the tweet: [“Neurological center to represent NFL at #SuperBowl <http://t.co/Hs9IA tqJIu> #TBI #concussion #sportsinjury #braininjury”](https://twitter.com/NeurologicalCenter/status/1091111111).

As a human, there is no way to say whether or not this tweet is from Washington or Massachussetts, unless you perhaps know ahead of time that the neurological center is located in one of the two states.

That being said, there are also many reasons to explain why our classifiers end up doing better than chance. The presence of “gopats” or “gohawks” would reasonably indicate what kind of fan

the user is, which increases the probability that they live in their team's local state. One nice aspect of the logistic regression classifier is it allows us to see which terms are most weighted when making a prediction. We looked at the weights with the highest magnitudes so that we could get a better understanding of what terms the classifier relied on the most to generate its prediction:

Table 34: Predicted Massachusetts and Washington

	Predict Massachusetts	Predict Washington
Top Terms By Weight	patriots, weight = -6.43917065456 gopats, weight = -6.1232834672 gopatriots, weight = -5.92751958644 7news, weight = -5.70939448314 boston, weight = -5.46642246906 wbz, weight = -5.4412693323 patriotsnation, weight = -5.42566400697 patsnation, weight = -4.79224094029 wcvb, weight = -4.31701551609 pats, weight = -4.29733638712	gohawks, weight = 13.8004922101 hawks, weight = 6.78421098619 12thman, weight = 6.33881715786 seahawks, weight = 5.67398402486 12s, weight = 5.10488783856 goseahawks, weight = 4.76809924668 repete, weight = 4.05050032111 dc, weight = 3.90776976408 q13fox, weight = 3.6601940557 twelfie, weight = 3.3920750668

From this analysis, we obtain a better understanding of how the logistic regression classifier thinks terms like “patriots” or “gopats” will increase the probability that a tweet is from Massachusetts, and how terms like “gohawks” or “repete” will increase the probability that a tweet is from Washington.

Part 3: Design 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 bringing in novelty and full or partial implementation of your new ideas.

Introduction

To dive deeper into the content of the textual data, we decided to perform a sentiment analysis study comparing how fans of each team felt as time went on throughout Super Bowl game day (02/01/2015). Our goal was to evaluate if sentiment analysis can provide sense for how users are feeling in response to real-world events. If so, we anticipate that the changes in sentiment should happen around the same time at which noteworthy real-world events occur (i.e. a touchdown, winning the game, etc).

Data Selection

In order to identify how a tweet's sentiment corresponds with real world events, it is important to know who is posting the tweet. In the case of the Super Bowl, we thought it would be informative if we created two categories of users: Patriots fans and Seahawks fans. Given our dataset with 6 hashtags, we figured the best way to perform the sentiment analysis study was to isolate tweets with the hashtags #gopatriots and #gohawks. Users who include #gopatriots are likely Patriots fans, and the same logic follows for Seahawks fans.

After loading in the #gopatriots and #goseahawks tweets and removing any repeats (to prevent overcounting any spammy tweets), we have the following total tweet counts:

Table 35: Number of #gopatriots and #gohawks Tweets

Number of #gopatriots tweets	Number of #gohawks tweets
16905	121119

We noticed that the number of tweets is not quite balanced among the two hashtags. We considered including all the tweets with the hashtag #patriots to bolster the number of tweets coming from assumed Patriots fans, but in the end decided not to do this since we did not have #seahawks to go along with it. It might be possible that users who are posting #gopatriots tend to remain more positive than users who just post #patriots, which could introduce unwanted biases on our sentiment analysis. In an effort to keep this analysis as “apples-to-apples” as possible, we decided to stick with #gopatriots and #gohawks.

Tweet Pre-Processing

Before throwing the tweets into a sentiment analyzer, we cleaned up the text by removing non-alphanumeric characters, stopwords, URLs, usernames, hashtags, and making everything

lowercase. The sentiment analyzer we found (to be described below) uses a Naive Bayes classifier with a 1-gram bag-of-words method, so jumbling up the words from stopword removal is not a major issue. To get a feel for how the pre-processing changes the tweets, here are a couple of examples:

Table 36: Original Tweet vs Cleaned Tweet

Original Tweet	Cleaned Tweet
LeGarrette Blount does the Ray Lewis Dance #ThrowbackThursday https://t.co/F5FX5KVmdX Hope to see it at least 3 times tomorrow. #GoPatriots	legarrette blount ray lewis dance hope see least 3 times tomorrow
"Oh no big deal, just NFC West Champs and the number 1 seed again" #Seahawks #GoHawks #12s http://t.co/7NigOjTPz2	oh big deal nfc west champs number 1 seed

Sentiment Analysis

In order to create a sentiment analyzer, we need to have a dataset that contains tweets and their associated sentiments (positive, negative, neutral). Because we do not have the sentiments for each of these tweets, and do not have the time to manually label all of the data so that we could train an analyzer, we instead opted to use an open-source sentiment analysis tool known as TextBlob.

From looking through the source code, TextBlob predicts the sentiment of a list of words by using a Naive Bayes classifier that was trained on a dataset of movie reviews. While this introduces all sorts of unwanted biases (movie reviews are quite different from Super Bowl tweets), we found that the tool often performed in a reasonable way when we tested it on the #gopatriots and #gohawks tweets.

Here are some examples of the tweets with a sentiment label.

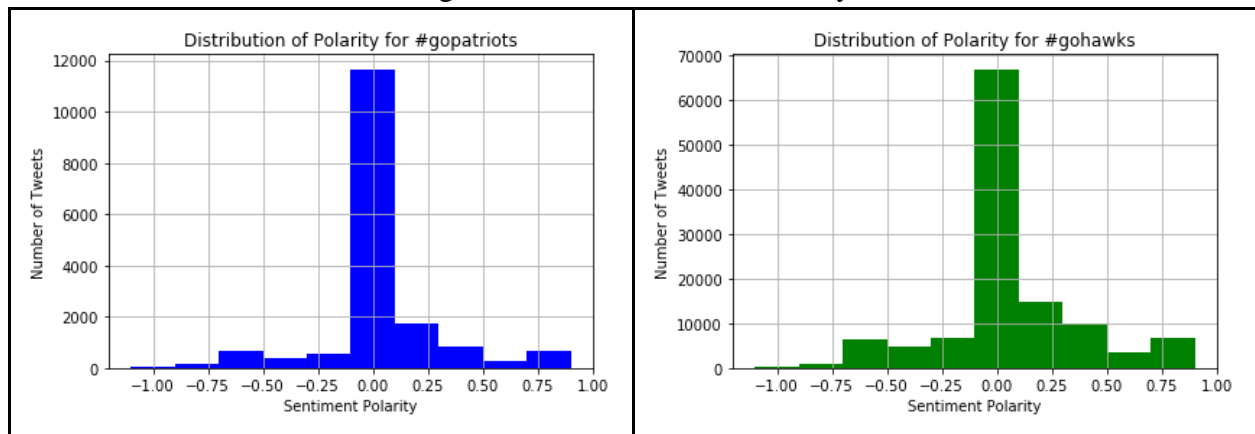
Table 37: Tweets with High Positive and Negative Scores

Tweets with High Positive Score (+1.00)	Tweets with High Negative Score (-1.00)
We looking awesome tonight in every way #GoPatriots	Omg! Worst minute ever! #GoPatriots

That was freaking awesome #GoPatriots	@briannakortuem just as pathetic as packer fans are! #GoPatriots
The greatest pic of the night #RichardSherman 😊 #sucka #GoPatriots #SuperBowIChamps 🏈 http://t.co/6TcAWHnDjA	Praying for these players safety though. This is terrible weather. #GoPatriots

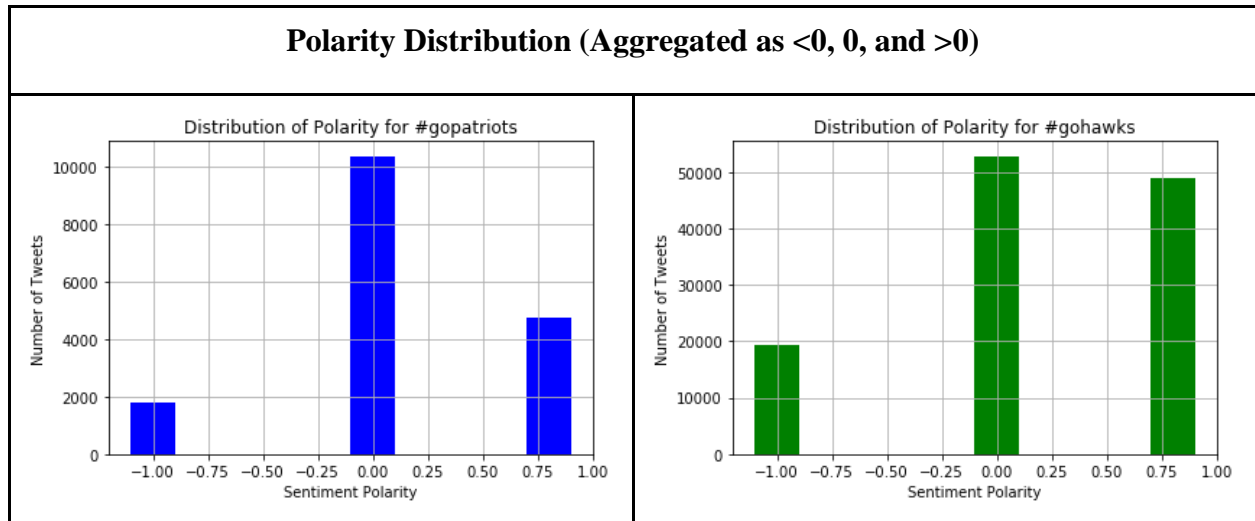
When the tweets are passed into the TextBlob sentiment analysis tool, the output contains a polarity score between -1.00 (most negative) and 1.00 (most positive). The table shows reasonable results -- tweets containing common positive words like “greatest” or “awesome” end up with a score of 1.00, whereas tweets containing words like “worst” or “pathetic” end up with a score of -1.00. The dataset also contains scores between -1.00 and 1.00. Here is a visualization of the distribution of sentiment polarities within the dataset:

Figure 10: Distribution of Polarity



We notice that a majority of the tweets in both the #gopatriots and #gohawks camps contain neutral sentiments. Due to the relative sparsity of tweets in each polarity bin, it is difficult to gain an understanding of how the behaviors of the two groups differ. To get a better idea of this, we aggregate the data in bins of polarities less than zero, zero, and greater than zero:

Figure 11: Distribution of Polarity Aggregated



From here we can see that Seahawks fans tend to be more polarized than Patriots fans, particularly towards the positive region. The ratio of positive tweets to neutral tweets is higher for the Seahawks, and the same holds true for the ratio of negative tweets to neutral tweets. In both cases, it seems that tweets are generally more positive than they are negative.

This leads us to generate possible questions for further research. Are Seahawks fans more passionate about football by nature compared to Patriots fans? Or maybe because they won in 2014 they have more excitement for victory (and more animosity towards the opponent).









Sentiment Analysis on Game Day

The 2015 Super Bowl took place on Feb 1st from 3:30 PM PST to about 8:00 PM PST. In this part of the analysis, we aggregate tweets by the hour between 3PM - 8PM and analyze the changes over time with regards to sentiment. When looking at the sentiment, it is also important to look at some characteristics of the game itself to provide a sanity check for whether or not the sentiment makes sense.

The following picture shows a game recap (provided by www.ESPN.com):

Figure 12: Scoring Summary

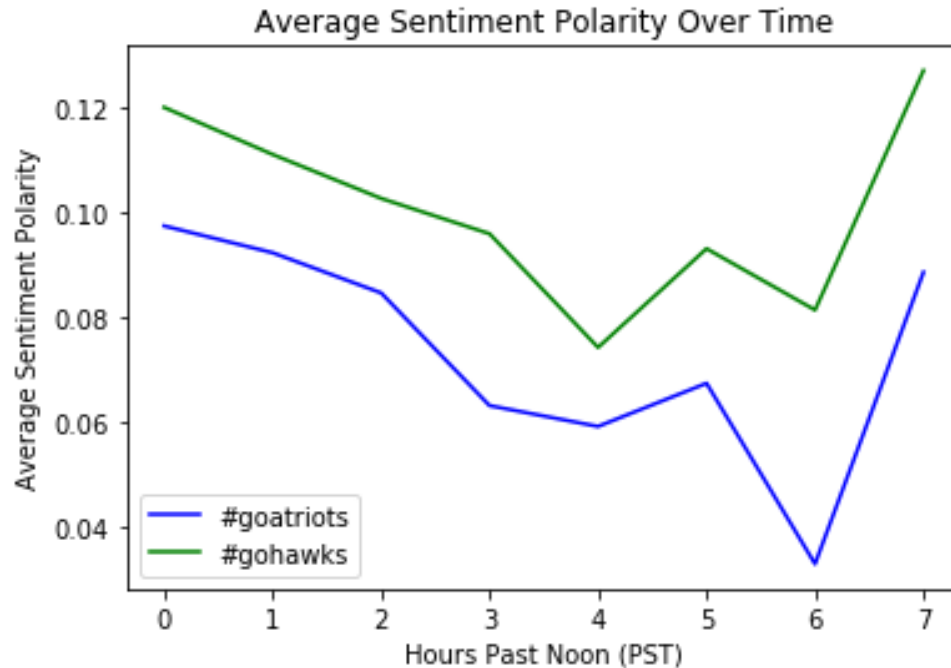
Scoring Summary

SECOND QUARTER			NE	SEA
	TD 9:47	Brandon LaFell 11 Yd pass from Tom Brady (Stephen Gostkowski Kick) <i>9 plays, 65 yards, 4:10</i>	7	0
	TD 2:16	Marshawn Lynch 3 Yd Run (Steven Hauschka Kick) <i>8 plays, 70 yards, 4:51</i>	7	7
	TD 0:31	Rob Gronkowski 22 Yd pass from Tom Brady (Stephen Gostkowski Kick) <i>8 plays, 80 yards, 1:45</i>	14	7
	TD 0:02	Chris Matthews 11 Yd pass from Russell Wilson (Steven Hauschka Kick) <i>5 plays, 80 yards, 0:29</i>	14	14
THIRD QUARTER			NE	SEA
	FG 11:09	Steven Hauschka 27 Yd Field Goal <i>7 plays, 72 yards, 3:51</i>	14	17
	TD 4:54	Doug Baldwin 3 Yd pass from Russell Wilson (Steven Hauschka Kick) <i>6 plays, 50 yards, 3:13</i>	14	24
FOURTH QUARTER			NE	SEA
	TD 7:55	Danny Amendola 4 Yd pass from Tom Brady (Stephen Gostkowski Kick) <i>9 plays, 68 yards, 4:15</i>	21	24
	TD 2:02	Julian Edelman 3 Yd pass from Tom Brady (Stephen Gostkowski Kick) <i>10 plays, 64 yards, 4:50</i>	28	24

We can see that the 1st quarter is score-free, and in the second quarter the two teams play quite evenly. The 3rd quarter completely belongs to the Seahawks, and the 4th quarter shows an exciting comeback for the Patriots.

The first metric we looked at was average sentiment polarity per hour throughout the game. Here is the result:

Figure 13: Average Sentiment Polarity Over Time



From this figure, we see that the #gohawks tweeters have a more positive sentiment throughout the entire game (we had also noticed that throughout the whole month they had a higher percentage of tweets that were positive). We also note that the trends are shared between the two user groups. Every decrease for the #gohawks tweeters is accompanied with a decrease with the #gopatriots tweeters. From this figure, we do not see a tendency for opposing teams to have opposite sentiment polarity trends (which is what one might initially expect in tweets regarding a zero-sum game).

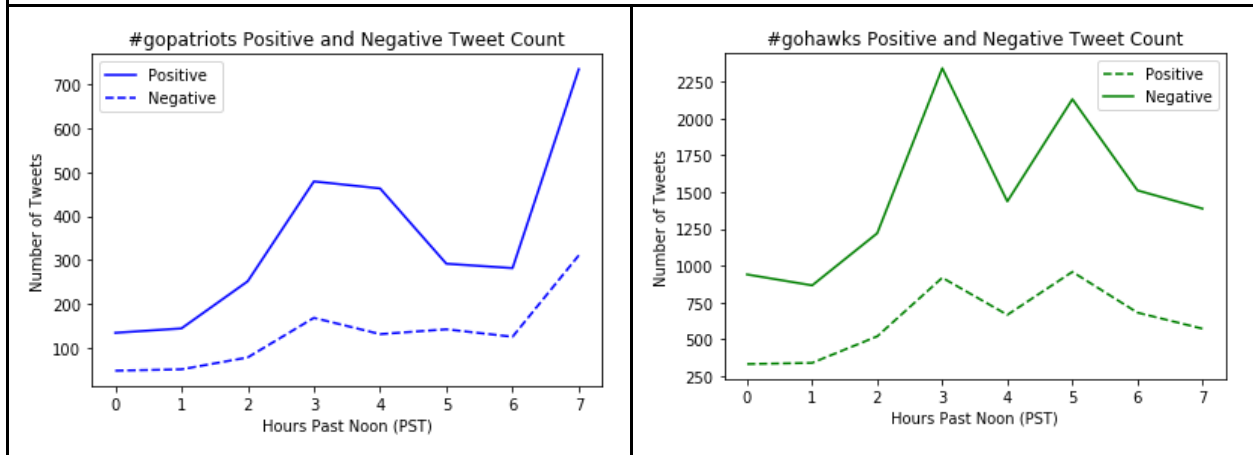
However, some interesting observations can be gleaned from this analysis. Of note is the sudden increase between where the x-axis = 6 (tweets between 6pm-7pm) and when x-axis = 7 (tweets between 7pm-8pm). From 7pm-8pm, the Patriots were coming back and there was a definite possibility they might win. It makes sense that users from both sides are more galvanized to cheer for their respective teams (so hashtags like #gopatriots and #gohawks might reasonably be tweeted).

One might wonder then, if it's not necessarily the average sentiment polarity trend of the two hashtags that would differ between fans of the two opposing teams, but perhaps the volume of positive / negative tweets in response to certain events.

Here we plot the number of positive and negative tweets over time for each team:

Figure 14: Number of Positive and Negative Tweets for Both HashTags

Number of Positive and Negative Tweets For Both Hashtags



One commonality we can see is that the positive tweets consistently outweigh the negative tweets, for both teams and for every hour (12PM-8PM). This may suggest that the #gopatriots and #gohawks teams are typically biased towards a positive sentiment, which makes sense because tweets with this hashtag are generally about users cheering for their team.

However, as opposed to sentiment polarity, the changes in positive and negative tweet count over time differ between fans of the two teams. Leading up to kickoff (3:30PM), the trends stay fairly consistent. The number of positive and negative tweets gradually climb. From 4-5 PM, the number of positive tweets drops compared to tweets from 3-4PM. This effect is much more drastic for #gohawks tweets, which may suggest that the uneventful 1st half of the game was more demoralizing for them than it was to the users tweeting #gopatriots.

In the hour of 5-6 PM, the #gopatriots positive tweet count declined while the #gohawks positive tweet count climbed. This is likely due to the fact that 5-6PM occupies the 3rd quarter of the game, where the Seahawks took the lead. In the next hour (6PM to 7PM), the number of positive tweets declines for both teams, although much more extremely for #gohawks users. Finally in the last hour of the game, #gohawks decreases further while #gopatriots increases considerably (considering they scored twice in the last quarter and thus won the game).

Possible explanations for why negative tweet trends seem to match positive tweet trends are that users who say #go<team> might also be bashing on the other team. So even as the Patriots won, we still saw an increase in the number of negative tweets for #gopatriots.

Concluding Remarks

All of these “correlations” between real-world events and positive tweet counts are qualitative and speculative. One important note is that the number of #gopatriots tweets is considerably lower than that of #gohawks tweets, so it’s not exactly easy to statistically compare the trends over team. The purpose of this introductory analysis is to provide hypotheses for future questions to be answered in a more qualitative way. For example, now we are inspired to ask if these same trends hold for tweets about other Super Bowls and other football games in general. Do these same ideas translate to other sports? Can these sentiments tell us anything about how fans of different teams might differ in how strongly they admire their team? In the application space, it might help to sell apparel or ads to a team that is more polarized. Perhaps you may want to run ads on users who have a more polarized sentiment since they are more likely to want to spend money to represent their passion.

This analysis shows that there is a rich space for sentiment analysis in tweet data. It suggests perhaps that behavioral and maybe even psychological insights can be gleaned from cleaning and processing tweets and running them through a sentiment analyzer derived from machine learning techniques. If we were to dive deeper into this rabbit hole, we might try to find a sentiment analyzer trained on a corpus closer to our target dataset, and also explore other hashtags to reduce the bias of any particular tag.