Question 10 – Twitter Data Analysis

Describe our task:

The Task we want to do is using sentient analysis to build a machine learning model that can forecast the match (Superbowl XLIX) progress using tweets' sentiments. (Time-Series Correlation between Scores and Tweets) To streamline the data preprocessing process, we opted to concentrate solely on tweets in "#gohawks.txt" and "#gopatriots.txt", and which were shared during the game.

Data Exploration:

As mentioned above, we focus on only the tweets during the match, specifically between:

6:24 pm EST: (datetime.datetime(2015, 2, 1, 18, 24, 6, timezone=<'America/Los_Angeles'))

22:10 pm EST: (datetime.datetime(2015, 2, 2, 22, 10, 5, timezone =<'America/Los_Angeles'))

Which were the start time and the end time of Super Bowl XLIX - Seattle Seahawks vs. New England Patriots - February 1st, 2015.

By reading the project instruction, we initially wanted to analyze which team was leading by looking at the number of mentions and emotions of each team's players, so we plotted the players mentioned in both datasets (Using Spacy Language Detector and Name Entity Recognition), but found that the sample size was limited. Figure 1 shows the number of mentions of each player during the match. We clearly see that Tom Brady and Landon Cohen were absolutely dominating the tweets for each fanbase. We can infer that Tom/Landon might be the most important players in that game. However, due to the limited number of the sample size, we decided to shift our favor to directly use the original tweet data for sentiment analysis.

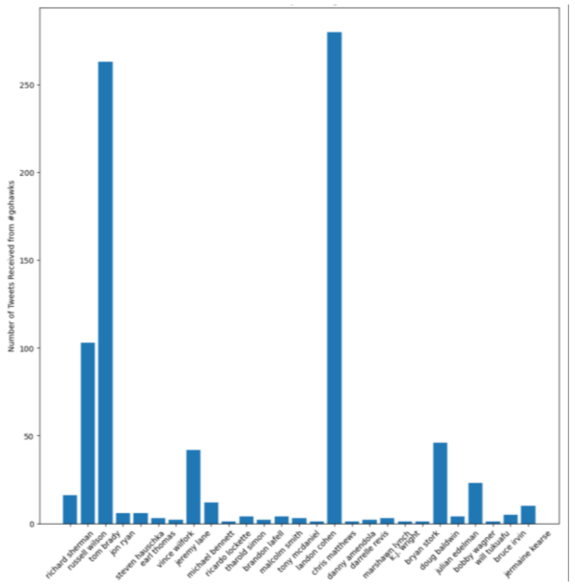
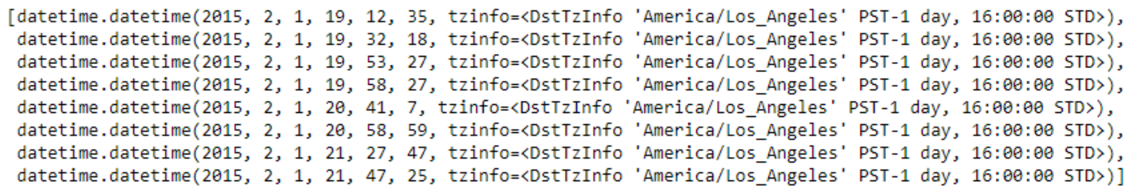


Figure 1. Players mention counts

Feature Engineering – 1. Preprocessing & Naïve Sentiment Analysis:

After loading in the tweets data from "#gohawks" and "#gopatriots", we divide the data by different time range using the timestamp of the scoring events during the game. Specifically, the match score changes in each of the timestamp listed in Figure 2 below. There were 7 scoring during the match so we assigned a total of 8 time ranges and created the score columns for our dataset.



```
[datetime.datetime(2015, 2, 1, 19, 12, 35, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 19, 32, 18, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 19, 53, 27, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 19, 58, 27, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 20, 41, 7, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 20, 58, 59, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 21, 27, 47, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>),
 datetime.datetime(2015, 2, 1, 21, 47, 25, tzinfo=<DstTzInfo 'America/Los_Angeles' PST-1 day, 16:00:00 STD>)]
```

Figure 2. Score Time

After filtering out stopped words, non-English characters, and urls like we typically would for NLP, we generate our target label "winning_team" as a 3 variable categorical column, indicating which the leading team in each time range. Then we proceed to create the sentiment columns using TextBlob library. TextBlob's sentiment analysis function calculates the **"polarity"** and **"subjectivity"** of a piece of text. Polarity is a measure of how positive or negative the text is, with values ranging from -1 (most negative) to +1 (most positive). Subjectivity is a measure of how subjective or objective the text is, with values ranging from 0 (most objective) to 1 (most subjective). Even though the TextBlob is easy to use, it is based on a simple algorithm that may not be accurate enough for more complex or nuanced analysis. Furthermore, TextBlob's sentiment analysis is designed for analyzing individual piece of texts and may not be suitable for analyzing large datasets.

Then, we observe severe data imbalance, meaning there were significantly more tweets when the game was tie or when the Hawks were leading. So, we use *RandomOversampling* to up sample our data to achieve a balanced dataset. See Figure 3 for bar plots.

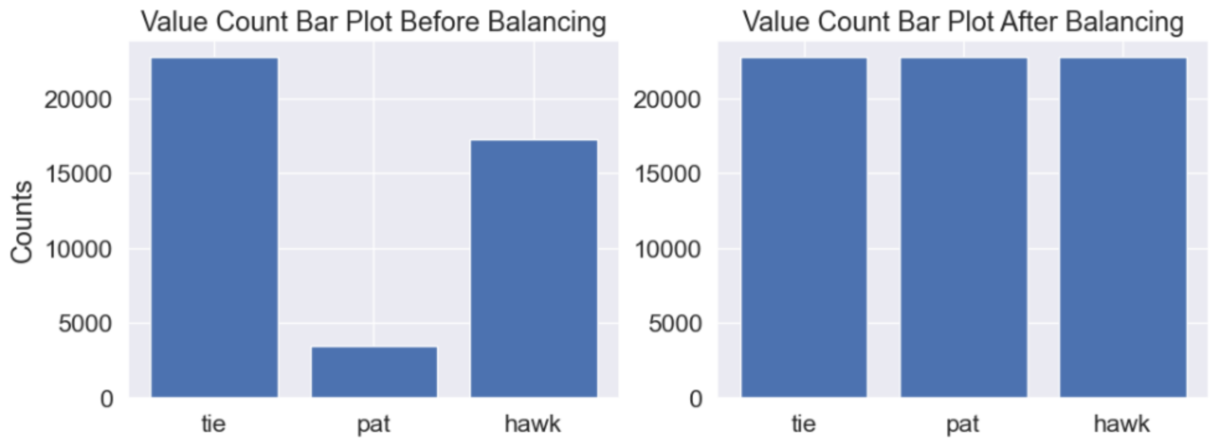


Figure 3. Label Distribution Before & After Oversampling

Baseline Models

After initial feature engineering, our input dataset has four feature columns: "**polarity**", "**subjectivity**", "**length**" indicating the original tweet's length, and "team" indicating which fanbase the tweet came from. And our target label is “**winning_team**” indicating which team is currently leading. So, this is a multi-class classification problem.

We decided to use a **Random-forest classifier** and a **Shallow Neural Network** as our baselines. They provided a good starting point for analyzing the data and identifying important features. The accuracy respects to each are **32%** and **38%,** which are low compares to 33% of random guessing. we also plotted the Multi-Class Confusion Matrix for both Random-forest Classifier and Neural Networks (Figures 4 & 5). We found that the random forest classifier predicts all games to be tie, indicating the model has no convergence whatsoever so we will not proceed forward with it.

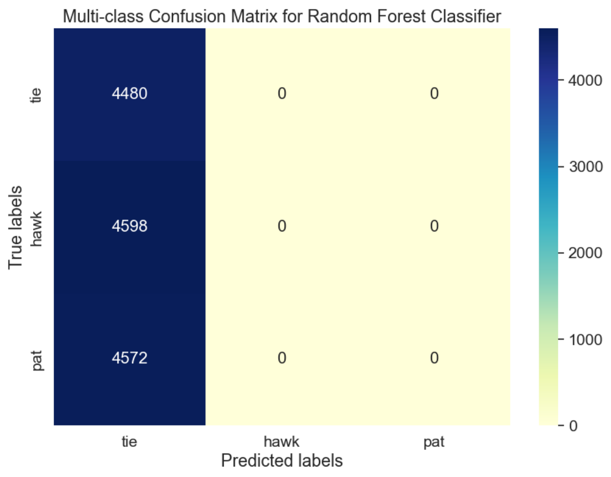


Figure 4. Randomforest Classifier Confusion Matrix

As for the shallow Neural Network, it improves on predicting the Patriots leading but in overall it performs poorly. But the confusion matrix has shown that it has some level of predictability so we will use the shallow Neural Network as our baseline model in the following parts of our project.

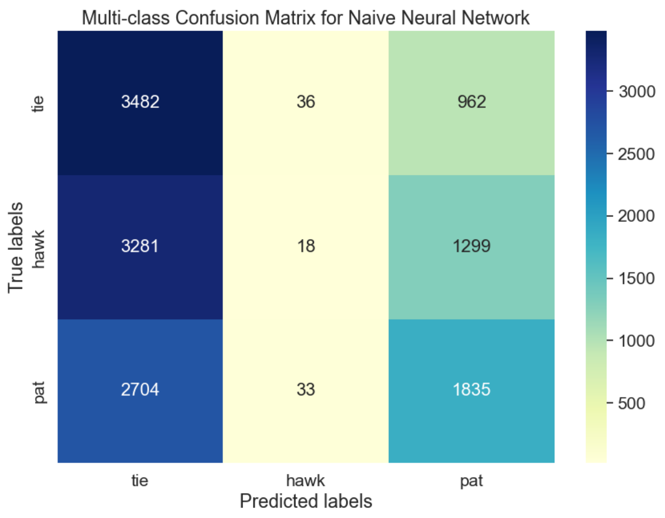


Figure 5. Confusion Matrix for Naive Neural Network

Feature Engineering – 2. Merging Multiple Tweets into a New Tweet:

The intuition behind this is that we can concatenate multiple tweets from each time range into one longer sentence to hopefully get more robust sentiment scores. However, from the model result, the only difference is that the network now is predicting more on Patriots (See Figure 6). So this attempt ended as a failure.

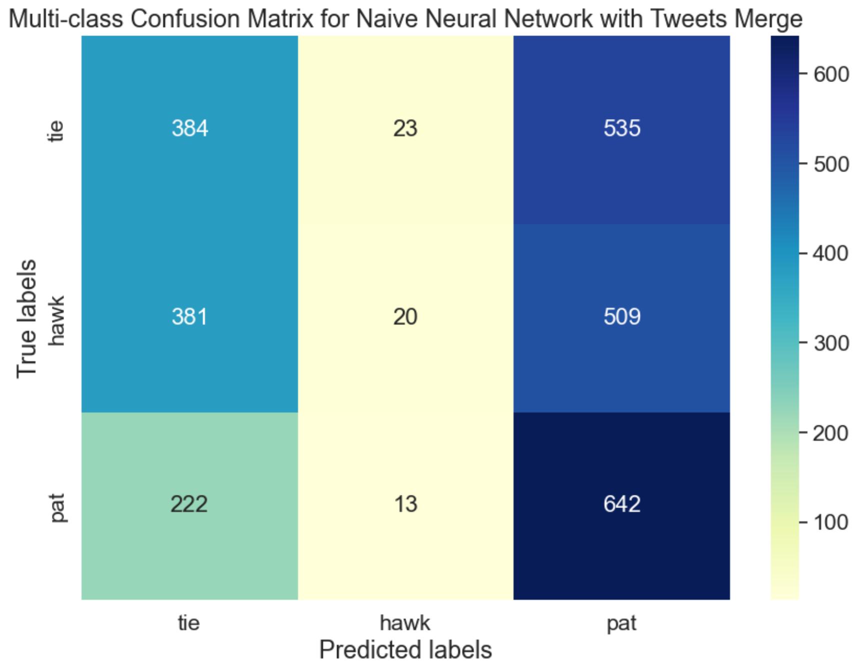


Figure 6. Naive Neural Network with Merged Tweets

Feature Engineering – 3. A Better Sentiment Analysis Model: roBERTa

In the next part, we explore a more advanced pretrained sentiment analysis model from a paper called *"A Robustly Optimized BERT Pretraining Approach"* (roBERTa). Specifically, we choose roBERTa-base as our model since it is trained on tweets data.

roBERTa-base is a pre-trained transformer-based neural network model that was introduced in 2019 by researchers at Facebook AI. It is an extension of the renowned Google BERT. roBERTa-base is an extremely powerful and versatile model that has shown excellent performance on a wide range of NLP tasks, such as sentiment analysis, named entity recognition, and text classification. For this specific task, it outputs three sentiment scores (**Negative**, **Neutral**, **Positive**) for each input text, ranging from -1 to 1,

Then, we convert the roBERTa-base sentiment scores into a probability using SoftMax function. The SoftMax function works by exponentiating each of the sentiment scores and dividing them by the sum of the exponentiated scores. This normalizes the scores to probability so that they add up to 1, creating a probability distribution over the three sentiment classes (**Negative**, **Neutral**, **Positive**).

With the same Neural Network, we achieve a 41% testing accuracy with the roBERTa-base sentiment analysis, with a better confusion matrix than previous, indicating the roBERTa-base indeed help with capturing tweet sentiments.

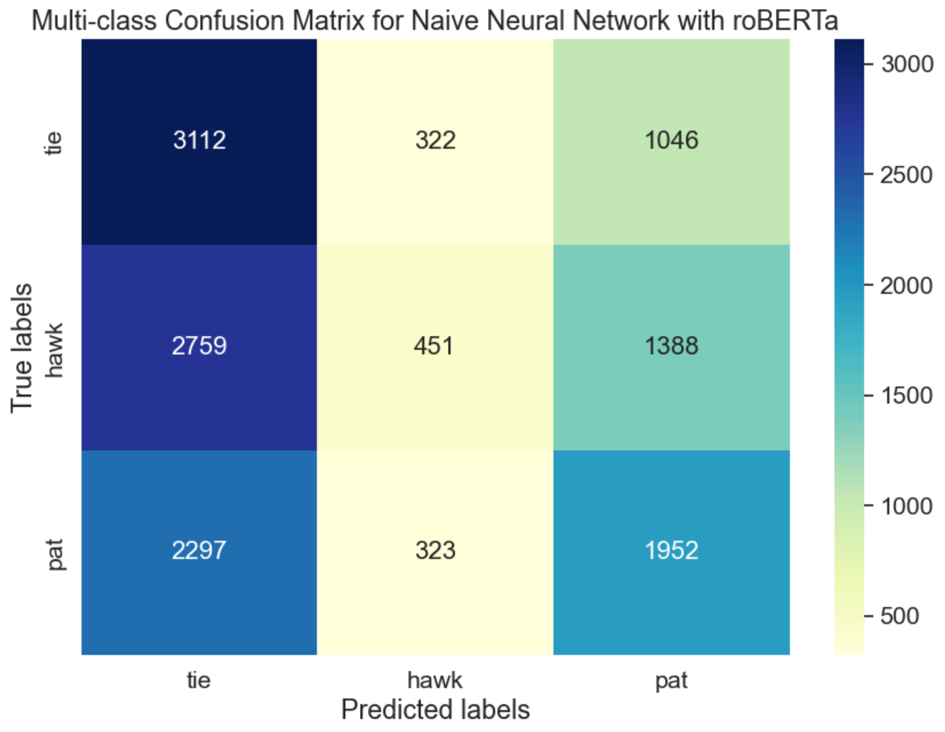


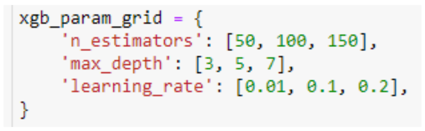
Figure 7. Neural Network + roBERTa Confusion Matrix

Model Evaluation

In this part, we benchmark three different architectures: Deep Neural Network with Dropout and Batch Normalization, Extreme Gradient Boosting (XGBoost) with grid search, and a Support Vector Machine (SVM). Before proceeding to the models, we have also cleared the hashtags and usernames in texts for better sentiment score generation.

Deep Neural Network: In this architecture, we use eight fully connected layers each with batch normalization and dropouts. We choose “elu” as activation functions, and uses SoftMax score as output activation. The loss function is defined as “Categorical Cross entropy” and the “Adam” optimizer with momentum is used during compilation.

XGBoost: For this architecture, the classification objective is set to be “multi:softmax” and we use grid search with 5-fold cross validation to select the optimal hyperparameters (See Figure 8 for hyperparameter choices). We then choose the best hyperparameters using the mean accuracy across folds in each setting. The best setting is found to be “n_estimator”: 150, “max_depth”: 7, “learning_rate”: 0.2.



```
xgb_param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
}
```

Figure 8. XGBoost Hyperparameters

SVM: For SVM, the fitting time is too long due to high dimension of our dataset so we did not perform grid search. The model is the SVC from sklearn library. We use "linear" kernel and "ovr" decision function with no maximum iteration limit.

Below is a table listing accuracy across different classifiers and sentiment analysis methods, we can see that our accuracy has a quite a significant jump from the 0.37 from Naïve Neural Network with TextBlob to the best of 0.57 with XGBoost and roBERTa. In overall, roBERTa is more reliable than TextBlob in sentiment analysis. However, whether removing hashtag from original text does not seem to be significant in improving performance. This may be due to the fact that roBERTa is trained on tweets data so that it already has some internal mechanics in dealing with hashtags.

	Naïve Neural Network	Deep Neural Network	XGBoost	SVM
TextBloB	0.37	0.37	0.47	0.36
roBERTa	0.39	0.41	0.56	0.39
roBERTa + Remove Hashtag	0.40	0.41	0.57	0.40

Best model: XGBoost with roBERTa on hashtag removed data (See Figure 9 for confusion matrix)

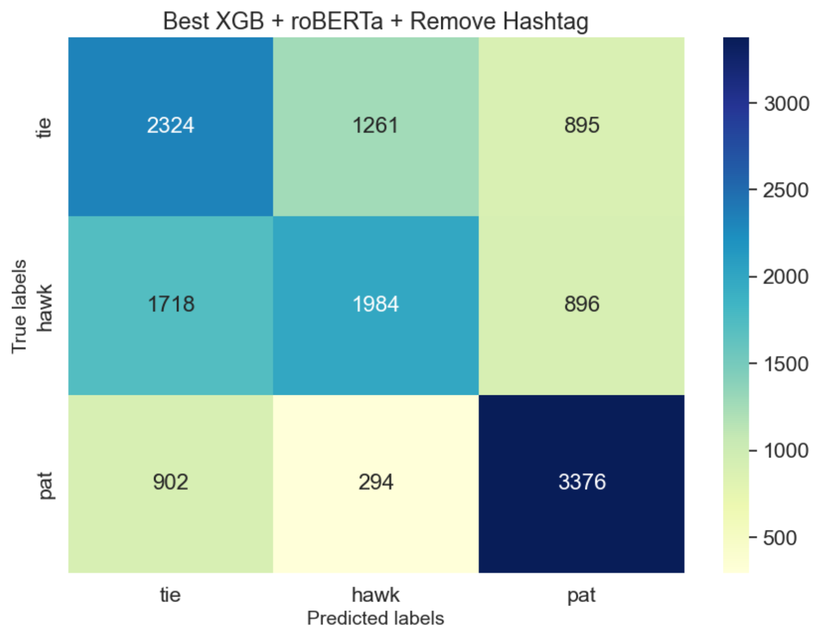


Figure 9. Best Model's Confusion Matrix

Reference Links:

TextBlob Spacy:

https://www.geeksforgeeks.org/python-named-entity-recognition-ner-using-spacy/

ESPL game data:

https://www.espn.com/nfl/playbyplay/_/gameId/400749027

RoBERTa: A Robustly Optimized BERT Pretraining Approach:

https://arxiv.org/abs/1907.11692