Model Selection

We used scikit-learn library for ML modelling. We tried different Machine learning algorithms and tuned in different features and hyperparameters that gave us the best results. We tried two different modelling approaches (using feature engineering and ML modelling in one (Model 1) and using pipeline based feature processing in the other (Model 2)). However, we noticed that in both the cases, when our training error was low, our test error was high, and therefore concluded that our model had a variance problem. This gave us an inclination for the use of Random Forest classifiers. The description of different models is provided below.

Model 1:

In this model we basically preprocessed the data and identified the important features in the dataset. We then represented these features in our code and applied a grid search for tuning in the best hyperparameters for the algorithms to work in. We achieved an accuracy of 82.5 % in Kaggle submission on a model developed using Random Forest classifier. The details of the implementation are provided below.

Preprocessing Techniques

We load the csv file in a dataframe and extract features out of the dataset given. Loading the dataset to dataframe allows us to easily play around with the data, add/delete columns. Some of the features, in the dataset were not informative hence we removed them from the dataframe.
Just by checking value_counts, it can be inferred that most of the features apart from text were non-informative. The features we removed were were retweeted , replytoSID, replyToSN, favorited, replyToUID, statusSource (column not in the test data), screenName, isRetweet, latitude, longitude, screenName.

Data Exploration

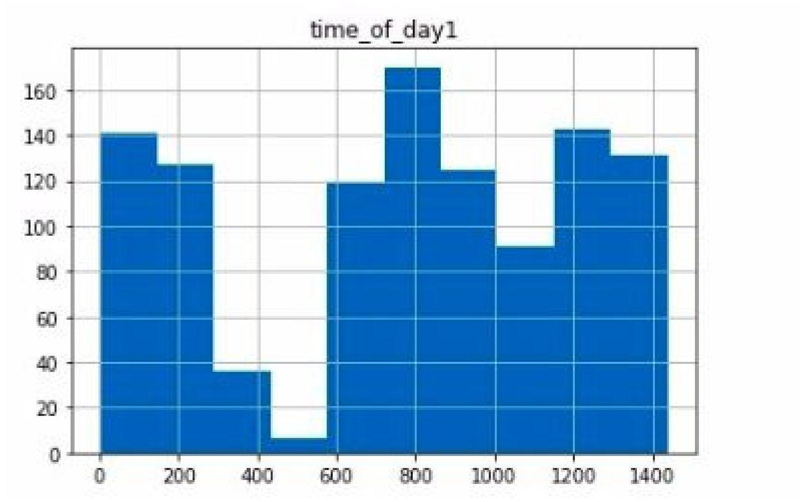
Almost every time trump tweeted about Megyn Kelly (identifiable by the word megyn in text), it was tweeted from Android. Similarly, we found such strong co-relations with the words “CNN”, “Thank you”, “Millions”, etc.

Feature Engineering

1. Day: The boolean feature represents whether or not the tweet was posted during the day. To obtain the feature:

- We used the “created” feature in the original data set. The time was extracted from this feature, and converted to minutes which ranged from 0 to 1440. Example: “23:15” would be represented in minutes by “23*60+15 = 1395”.
- Once we obtained the minute from the start of the day when the tweet was posted, we analyzed the values using a frequency plot. This showed that the tweets mostly occurred in two different times:
    - 500 to 1140 minutes, i.e. 8:30 AM to 7:00 PM; and
    - 0 to 500, and 1140 to 1440 counting as not a day

The plot shows why the thresholds were chosen in such a way.



2. Favorite Count
3. Retweet Count
4. Percentage of Capital Words tweet
5. Percentage of # tags in a tweet
6. Percentage of @ tags in a tweet
7. Whether the text has http link
8. Whether the text contains words like CNN, Thank You, Millions, etc.
9. Total number of words in tweet
10. Sentence Embedding norm (take the norm of sentence embedding)
11. TfIdf vectorizer output vectors

Hyperparameter Search

We did Grid Search for finding hyperparameters for Random Forest classifier and with the best parameters we got an accuracy of 82.5 on the test set. After trying many different parameters for Random Forest classifiers, we did not find the accuracy to increase above 82.5

Model	Hyperparameters	Features	Results (Kaggle Scores)
Random Forest	Number of estimators: 100-1000 Max_depth: 2-7 Impurity function:’entropy’,’gini’ TFIDFVectorizer: True/False	TFIDF Vectorizer for tweet text, Generated features as above	82.5%
Naive Bayes, SVM	Learning rate:1e-2 - 1e-3 Count Vectorizer: True/False TFIDFTransformer: True/False	TFIDF Vectorizer for tweet text, Generated features as above	78.3%
Logistic Regression	Tfidfvectorizer ngram range : [(1,1), (1,2), (1,3)] Alpha : [0.001, 0.01, 0.1, 1, 10, 100] Count Vectorizer: True/False	TFIDF Vectorizer for tweet text, Generated features as above	81.6%

Model 2:

The second model uses the TF-IDF metric using the tfidfVectorizer module from sklearn. A lot of the technical know-how was obtained from this blog: https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a.
We create a pipeline that has the following components: A countvectorizer (configured to remove stop words), a tfidfTransformer and a classification model.
We finally made use of the random forest classifier on the output of the vectorizer which gave an accuracy of 81.66% on Kaggle. Further, using grid search, we obtained an accuracy of 83.33% on Kaggle.

Results:

The following were the accuracy scores with the models tried:

Model 1:

Approach used	**Validation Accuracy**	**Kaggle Leaderboard**
Logistic Regression with features: WordEmbedding (Norm used as feature) + other features defined above	84%	81.6%
Random Forest with (number of estimators = 1000) with features: WordEmbedding (Norm used as feature) + other features defined above	85%	82.5%
Random Forest (number of estimators = 500, max_depth = 4) + other features defined above	79.6%	82.5%
Random Forest (number of estimators = 500, max_depth = 4) + TFIDF vectorizer + other features defined above	90.2%	82.5%
Random Forest (n_estimators=600, criterion='entropy', max_depth=100, min_samples_split=2, max_features='sqrt') + TFIDF vectorizer + other features defined above	**82.5%**	**83.3% (Final Model)**

Model 2:
Approaches used other than the final model:

Approach used	**Training Accuracy**	**Validation Accuracy**	**Kaggle Leaderboard**

Multinomial NB + TFIDF vectorizer + grid search	80%	85%	Not submitted or no improvement
SVM +TFIDF vectorizer + with grid search	86%	85%	Not submitted or no improvement
SVM + count_vectorizer+ removed stop words+ grid search	84.7%	87%	78.33%
Multinomial NB + count_vectorizer+ removed stop words + grid search	79.6%	79.2%	Not submitted or no improvement
Logistic Regression + count_vectorizer+ removed stop words+ grid search	84.9%	86.5%	Not submitted or no improvement
Logistic Regression +count_vectorizer + TFIDF vectorizer + stop words+ grid search	85.5% Training=70% of total data	84% validation = 30% of total data	79.16%
Logistic Regression + count_vectorizer + TFIDF + stop words+ grid search	85.03% 100% of total data	-	80%