ML Kaggle Project Writeup

Team Name: 5%

Team Members: Nolan Gray, Kevin Luo, Sheetal Athrey, Siddhant Navali (Kaggle name is "siddhant"), Gauri Jain (Kaggle name is "gaurigjain")

**unless otherwise noted above, our kaggle usernames are the same as our real names*

Public Leaderboard Score: 0.84166

Preprocessing/Data Exploration

The given data came with a series of columns which were either useless or were not provided in the test set. Some columns were all the same value, providing no extra information. Thus, we removed them. The list of removed columns is as follows:

- Favorited
- Truncated
- replyToSN
- replyToSID
- 9th col ID
- Statusstore
- Screenname
- IsRetweet
- Retweeted

We also inspected the class distribution in case there was serious class skew, which might have necessitated special measures (e.g. weights to make the rarer class more important during splits or loss). Luckily, there was no class skew, as the labels were pretty close to 50-50 in the given training set.

Besides the label distribution, we also visualized properties relating to the given columns. For example, we plotted the number of data points that fell within certain ranges for a given feature, in case certain classes tended to occur much more frequently within certain bounds. Such knowledge would be invaluable for better feature creation. For example, we found that tweets that had quotes at the beginning were far more likely to be from Android. By plotting example occurences by class based on if Trump's own twitter handle appeared in the tweet, we also discovered that the appearance of Trump's own twitter handle was much more likely to be Android. Other examples include correlation between the appearance of emoji characters with Android, etc. Such data exploration led to feature engineering on our part, which we discuss in the next section.

Feature Extraction

In the above section, we explained how our data exploration led to the discovery of some good features. In total, we extracted 11 features, ranging from attributes like the appearance of

Trump's own twitter handle to time of tweet to hashtags to the presence of http (links). These features turned out to be good predictors of Android/Iphone when fed into a random forest or Adaboost, as the next section covers. The importance of selecting good features was highlighted by some of the contrasting approaches we tried. For example, we tried just doing bag of words on the text and then feeding it into a LSTM, but such an approach (especially with limited data) turned into an overfitting nightmare as the LSTM was not able to learn the features that we could simply engineer ourselves.

Model

The first model we tried was a random forest. Using sklearn, we implemented a random forest with 1000 trees. This model reached about 89% validation accuracy, and 83% public leaderboard accuracy.

Next, we looked to Adaboost to improve our model. Once again, we used sklearn to implement adaboost with 100 estimators. Our validation accuracy was about the same, but our public leaderboard accuracy bumped up to 84%.

Our next approach was creating an ensemble model. We wanted to combine our random forests and Adaboost models with an XGBoost model. By combining the predictions from each model and doing simple majority voting, we created ensemble predictions. Once again, our validation accuracy was around 89%, but our public leaderboard accuracy was only 83%.

We also tried a very different approach in terms of modelling and feature engineering. We experimented with using a Bag-of-Words representation of the text and then feeding that into an LSTM network. This approach seemed brute force from the beginning, and indeed it resulted in pretty severe overfitting. It reached up to 81% validation accuracy, but resulted in a pretty low 59% score on the public leaderboard. With limited training data, it was unreasonable to expect an LSTM to learn to recognize the very same features that we could manually engineer, so this approach was not expected to be accurate or efficient. However, it was something to try regardless.

Citations

- Random Forest and AdaBoost implementations were from the sci-kit learn library
- LSTM layer implementations were from the keras library