Project Mcnulty Summary

Project Design

My objective for this project was to practice applying various classification algorithms. I chose my dataset based on a few further criteria. For one, I wanted to focus on binary classification as I thought it would be a good inbetween step in building up to later multi class classification problems. This way I could get in some practice before diving off into methods such as one vs. rest. I was also looking for a relatively “clean” data set that would allow me to perform EDA and move on to modeling without dwelling too long on issues such as missing values and inconsistent data types.

Tools

My initial plan was to store the data files in S3 on AWS and run the classification models on an EC2 instance. However, upon exploring the data a bit more, I realized that if I wanted to focus on unique bidder bidding statistics the data files became small enough to be used within Jupyter. Ultimately, I performed all my data cleaning, exploratory data analysis, visualization, and final modeling in a few different Jupyter notebook.

Data

I used data from a 2015 Kaggle competition sponsored by Facebook. The challenge is to classify bidders on a bidding site into real users and bots. One must do so from a history of 7.5 million bids performed by roughly 2000 users. The timestamp of each bid has been deliberately obfuscated, which made engineering time related features difficult. I wanted to include bid frequency per bidder in an auction and the number of bids a bidder made at the absolute minimum time delta as I thought those would be quite different for bots who can perform many more operation much more frequently. Looking into time provided in the original data file, I noticed that the bids were from three distinct time periods. However, I was unable to devise a way to transform the quadrillion sized numbers provided into something I could use more easily. Therefore, I did not include time related features in my final data frame.

Another challenge I had to combat is the dramatic class imbalance in the data. Only about 5% of the total users were marked as non-humans, which meant that if I did not adjust either the data or the algorithms most (if not all) of the bots would simply be misclassified.

A quick fix for this was to undersample the 0th class, meaning I randomly picked the same number of classified humans as there are classified bots to build my models on. While this *a* solution, I do not believe this is the most elegant or efficient one. I would like to go back and try two different approaches. One, is to oversample the positive class and another is to generate new. The second one is to generates "synthetic" minority observations to boost the number of bots.

Original Dataset

Algorithm

I have performed the following on my final dataset:

- SVM
- SVM using RBF Kernel
- KNN, using 5 neighbours

The RBF Kernel gave me the highest accuracy score of .878. I was surprised by how high it was as I though not having time would have a more severe impact on my accuracy.