**Screening for bot users in an online auction**

**Problem:**

Botting is an efficient way to take advantage of websites transactions and simple tasks. Those who use bots thus have a potentially unfair advantage over those who are not using them. This situation arises in many different retail scenarios causing things like buyouts, however it is particularly to a healthy environment for an auction or bidding site such as eBay. A bot in an auction can input many more bids in quick succession essentially making it impossible for a human to compete. It is important to be able to identify bot accounts so that they can be banned and/or removed from the auction.

**Client:**

Our client is a bidding website such as eBay, though they may already have bot detection algorithms the focus on user friendliness and community is a key factor for a client. Other auction sites such as the government run municibid and IRS auctions are not interested in building a community as such they are less likely to invest in the best bot detection algorithms.

**Data Wrangling and Analysis:**

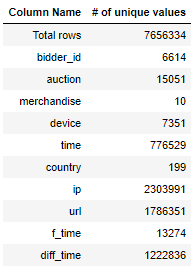
The data used for this project comes from an old kaggle competition which supplies bidding data and a training set of users to build algorithms around and a test set once the algorithms are ready. Data is located at: <https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot/data>

Data wrangling consisted of a few steps to scan the data for faulty data values and determining whether missing data needed to be replaced or removed. First several methods of scanning the data for faulty or missing data will be used. Once any missing or faulty data has been found then depending on severity of the missing data they must be properly removed or kept with the data. The same process will have to be repeated for outliers. This data set was interesting to work with because it consists only of categorical data. The bots consist of 5.2% of the bidders in the data set.



**Figure 1.** *Head of bids data set.*

Sample of the data set to show how data is formatted and what data points look like.



**Figure 2.**  *Value Counts for each feature.*

Number of unique values for each feature used to understand each feature better and the potential usefulness it holds.

Since the data is solely categorical the cleaning steps were designed for each column and was composed of three steps. First, unique values were extracted from the merchandise column. This provides an easy way to see all the values and determine if anything is mispelled, but is not practical for columns with a large amount of unique values, hidden values or random codes such as the auction column or url column. Next, the IP column was iterated through to determine if each value is in the correct form with three periods. Once the IP and merchandise columns were passed over I switched to a full data passover to find data types that did not match the correct value for that column. For instance all the merchandise column of the dataframe should all be of the string type since it represents categorical data such as jewelry, furniture or home goods. This analysis showed that the country column was the only one with faulty data types.

Further analysis revealed that 0.1% of the country data had NaN values and a small sampling of those values showed that the NaN country value is not obviously caused by a bad data collection method. Because the data points where the country is NaN are not poorly collected and represent a small percentage of the total data sample I chose not to remove them from the population. However, if inferential statistic results show us that the country data is potentially important for bot detecting then these values will be removed in the future.

With respect to outliers, because this data was all categorical or labeling data such as phone1, phone2, etc. there were no numerical outliers within the data frame. While there is no categorical data, when grouping the data by bidders we can make scatter plots to view how many times a bidder has done a certain action. These scatters when separated into bots and humans allow us to view outliers in each population. Outliers in this case will be human data points which seem unreasonable. Bots are excluded from this outlier analysis because they can be programmed to bid a humanly unreasonable amount of times. One such “inhuman” outlier are three data points labeled humans which have bid more than 100,000, 150,000 and 250,000 individual times respectively. This seems like maybe one bidder account has been accidentally assigned extra bids, or possibly is incorrectly label as a human. Since we have no reason to believe these data points are bots other than an inhuman amount of individual bids we will remove these data points so they do not skew statistical results for humans.



**Figure 3.** *Histograms of Bid time, Time before the end of a bid, unique auctions and shortest time between bids.*

Histograms are useful for quickly identifying differences in distributions which may indicate whether or not the specified feature is relevant. In these cases it seems the the number of unique bid times and auctions are going to be important features to look at along with the shortest time between bids. The time before a bid ends is less likely to be significant given that the histograms are very close in appearance.

Using histograms we can quickly compare two distributions to determine which ones may be suitable for further exploration and statistics. These four plots show us that different bid times (unique bids) and different auctions may be suitable features and should be explored with statistical tools like the t-test. The bid time before the bid ends may not be useful as the distributions have a very similar shape. The weirdest result is the histogram for the shortest time between bids. This histogram only shows two peaks, one at zero and the other at 500000. While this is alarming it may be the case that the ratios between the two values creates a relevant feature. Below we have shown another useful visualization tool which is the stacked bar chart. Using this we can see the percentage of bidders that were bots that came from a certain country or url.



**Figure 4.** *100 percent stacked bar charts for country and URL*

100 percent stacked bar charts let us see the percentage of bots which come from each country and URL respectively. Both of these data points look as they may be significant with certain countries and urls having a significant percentage of bots coming from them.

Once we have analyzed that data and created visualizations of it to search quickly for possibly relevant categories we apply statistical methods to determine which categories and correlations are significant. The main statistical tests we will focus on in this analysis are t-tests and chi-squared tests. The quantities up for analysis are the unique number of bids per auction, unique urls and ips used, total number of bids, and the average number of bids per auction from a user. By comparing the differences in the bot and human populations between these categories we will be able to identify the most important qualities for distinguishing bots and humans.

First I will discuss the results from the t-test analysis. T-tests were used to analyze the majority of the categories. By splitting the data into bots and humans and then grouping by bidder id populations of each category are created.



**Table 1.** *Dataframes for humans (A) and bots (B)*

The populations of interest are auctions, time (representing total number of unique bids), url and a self-created feature bids per auction. To determine which of these features if any are significant in distinguishing bots vs. humans we applied a t-test to compare the distributions of these features among the bot and human populations. A null hypothesis of a feature being equally distributed was used.

|  |  |  |
| --- | --- | --- |
| Feature | P-value | Significant |
| # of Auction | 4.7e-9 | Yes |
| # of total bids | 0.01 | Yes |
| # of bids / auction | 1.7e-8 | Yes |
| # of unique urls | 0.43 | No |
| Shortest time between bids | 0.0 | Yes |
| Final bid time -> bid end | 0.0 | Yes |

**Table 2.**  *Features and P-values generated from t-tests.*

Of all the features that were tested the only one that is not statistically significant is the # of unique urls that a bidder used. The p-values for the other features are lower than the significance threshold of p = 0.05. Moving forward we will want to keep # of bids, # of auctions and # of bids per auction in mind when creating a model.

Another test that was used to evaluate the data is the Chi-squared, which is used for comparing samples to the population to determine if two features are paired. A null hypothesis that the two observed variables are paired is tested is used for analyzing p-values.

|  |  |  |
| --- | --- | --- |
| Feature | P-value | Significant |
| URL of bid | 0.89 | No |
| IP of bid | 1.00 | No |
| Country of bid | 0.0 | Yes |

**Table 3.** *Features and related p-values from chi-squared tests.*

Based on the results of our test the country that our bidder comes from is a relevant feature to keep in mind but the URL and IP are not so useful.

**Machine Learning:**

Before experimenting with machine learning algorithms its important to split our data into train and test sets. To create the train and test splits the data was grouped by bidder\_id to get unique counts and then split using sklearn’s train\_test\_split keeping 30% of the data for the test data set. However there is an issue with our data because out train set includes 1328 bidders total 75 of which are bots. Because of this we need to implement oversampling/undersampling. Resampling the data like this will create training data sets with equal amounts of bot and human data points. Oversampling takes the smaller groups and resamples them with replacement to create balance while undersampling does the opposite, sizing down the larger set. In addition we used another resampling method called SMOTE which combines the creation of synthetic data points and oversampling for comparison. To compare these samples we will use a logistic regression model which has been tuned with a simple grid search cross validation for the C-value. Our comparison metric will be the area under the curve (AUC) score for a ROC curve. 

**Figure 5.** *ROC curves and AUC scores for a Logistic Regression model with different sampling methods.*

The training sets with bots and humans are very imbalanced in size which we use the above sampling methods to account for. Traditionally oversampling would be the best method but when comparing the ROC and AUC scores undersampling is performing the best and will be used to compare the other models to each other.

Having examined the ROC curves, we can see that the best results are coming from the undersampled data. This is likely because in the over sampled data we are resampling too much from a small amount of biddre samples to bring it up to the size of the human bidders which is almost 20 times larger.

Once the most effective training data has been obtained we evaluate several other machine learning models namely Random Forest, K-Nearest Neighbors and Support Vector Machines. We will be comparing these using the undersampled training set and AUC score and ROC score similar to before.



**Figure 6.** *Comparison of ROC and AUC scores between a few machine learning models.*

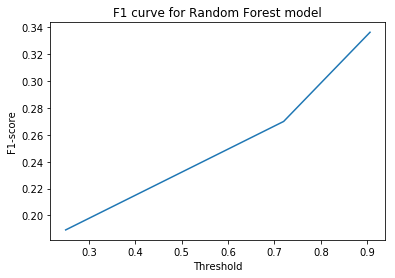
By comparing K-nearest neighbors, Random Forest and support vector classifiers using the AUC score Random Forest seems to perform the best.

When using the support vectors for classification we show the model with the oversampled data set rather than the under sampled one because it performed better based on the AUC score. Using the AUC score as a comparison metric we find that the Random Forest classifier should be the best. However if we use another common comparison tool, the f1-score we find that the Logistic Regression model (shown below). A comparison of their classification reports will be used to discuss which model would be better suited for this approach and how it may change depending on the needs of the client. We expect that the f1-scores would be higher but by using thresholding on probabilities an f1 curve was created and shows that the classification report already found the optimal f1-score. (figure 7)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data | Precision | Recall | F1-Score | AUC Score |
| LogReg Bot | 0.22 | 0.86 | 0.35 | 0.88 |
| Forest Bot | 0.19 | 0.89 | 0.31 | 0.91 |

**Table 4.** *Comparison of precision and recall between the logistic regression and Random Forest Models.*

Logisitic regression has a higher bot precision than the random forest model, but forest wins in bot recall. This is important in considering what kind of task the model will be applied to and what outcomes are more desirable between high bot detection and reduced false detections resulting in bannings of human bidders.



**Figure 7.** *F1-scores plotted vs. threshold*

By plotting f1-scores vs. threshold we determine the best f1 score and at what threshold we achieve that score.

These classification reports give us precision and recall which are very important for determining whether the model is reasonable to use. It is important for the client that bot precision is maximized because if a lot of humans are being banned falsely as bots then users may not want to use the platform. For this criteria the logistic regression performs better with a bot precision of 0.22. Bot recall will tell us the percentage of total bots that were detected. It is important to note that the recall of bots with the random forest is better which means that while it is more successful at total bot detection it also detects more humans as bots. It will be important to discuss with a possible client at which solution is the most important. Detection of the most bot users or sacrificing bot detection capabilities for less false bots. A potential solution to this is to do a more in depth analysis on the accounts chosen as bots by the model or rescreening while adding a “previously detected as a bot” feature which may help distinguish bots and humans further assuming that banning bots immediately is not a major concern for the client.

**Conclusions:**

After analyzing the data visually, picking relevant features with t-tests and testing machine learning models, the best performing model are the logistic regression and random forest classifiers. However these models excel at different aspects of the problem, because of this the client may value them differently. I would recommend that the client use the logistic regression model over the random forest model. This is because the bots make up a very small percent of the population. The logistic regression has a higher bot precisions score which means that while it may not detect as many bots, it preserves more human users. Since the client is having an issue with users being unhappy with losing out bids to bot users it is tempting to maximize bot detection. However, if too many humans are declared bots the clients may end up having a different problem that is harder to fix. Another way to correct for this issue may be to implement a rolling detection system to add a feature for the number of times a user was determined to be a bot in previous detections. This would increase our certainty that a user is a bot over time. The down side of this method would be that it slows the banning of bots. If the client is pressed for time then it may be more effective to do a quick pass to ban many bots using the random forest model and then switch to the rolling detection system described before.

**Project Location:** <https://github.com/cvgardner/Springboard/tree/master/Capstone%201>