Machine Learning Engineer Nanodegree
Capstone Proposal
Giorgos Bouzias
February 18th, 2018

Domain Background

In the broader sense, the project falls into the domain of bot detection by their activities/behaviour in a given platform. Historically, bots have been viewed as a problem for many reasons but mainly DDoSing or spreading malicious software. Another reason that might render bots unwanted, is the fact that they spoil the online experience of a user in for example a game or somewhere like a website where people are supposed to interact with each other.

The first part (the one of DDoSing etc) has seen considerable work through the years like in [1], [2] and a lot more, that aim to deal with this problem by tracking down bandwidth, packet timing and try to take into account traffic behaviour in general. the other problem that refers to corrupting the user experience of an online product has to do with bots using superhuman abilities (by having access to an API, the screen pixels, or they refresh in an optimal manner or something like that) in for example a game or a bidding site which is the proposed case.

Research in that area leverages the users historical data, or other behavioral aspects like for example in cases [3] and [4]. In [3] a single behavioral sequence (as feature) is extracted by social media histories to detect spam bots, and in [4] a CAPTCHA test is proposed in an online game context. In the area there are also companies trying to deal with these kinds of issues in the spectrum they are discussed here [5].

In terms of my engagement with the topic as a capstone project, the main reason is that i found a compelling dataset to work with which i think requires to do things one step further than what i have done so far. That being said, the problem of course falls also into my areas of interest, but my list is really broad given that i am interesting in problems that can be solved by machine learning without any particular preference.

Problem Statement

The proposed problem is quite specific. It is bot detection in a website of auctions. Facebook did run a competition through kaggle on this problem for hiring purposes [6] and it can be described better by discussing the data such a website has. In that sense, websites that do such auctions have some information for each bid such as: the auction or merchandise it came for, the device, ip and country it came from, and the url the user used to make the bid. In addition to the information for every bid, they can match each bid with a person by their corresponding ids (a bid id and a bidder id). The challenge is to use these data to build a classifier that can correctly identify a user, given some features that can be built through that person's history. The data are labeled and if looked by the proposed angle, the problem can be seen as a binary classification problem.

Also, while the logs are very easy to be found/measured from an online platform, that is not the case for labeling each bidder id as bot/no bot. However, the company that issued the dataset has marked some bidders in the train and test sets as bots/fraudulent. Of those users, for some they have clear proof (although they do not specify what is that proof), and for some others they observed excessive behaviour given the system's wide average, but they are not 100% certain that they are bots. Finally in terms of replicability the problem lies in a niche area, that makes sense only for such auction companies; that does not make it very generic but it is definitely replicable in the sense that auction/bidding events occur repeatedly in such a site.

Datasets and Inputs

For this project the 3 proposed datasets are given from the aforementioned kaggle competition that can be found in the data tab of the cited competition [6]. The competition as mentioned is managed by Facebook and the data comes from an unspecified online platform that does auctions, and keeps some data of the users behaviour and bid histories. Also, all the data is categorical except two cells that are numerical (the bid id and the label).

The main dataset (which i will have to enrich with features for training as i explain in the next section) is the bidders dataset for which there is a train and test version. In this dataset there are 6,713 unique bidder ids, linked with a unique payment account, a unique address and their label: 1 for Bot and 0 for human user. It is maybe important to stress here again that the labels are noisy meaning that not all of them are judged with 100% certainty to be bots. Also all these addresses and ids are obfuscated for security reasons.

The other dataset that is included is a table with all the bids which holds 7,656,334 unique bid ids. It goes without saying that for every bidder in the first two sets, there is from 0 to a series of bids that correspond to the user's name, and therefore besides a bid_id, the dataset includes a bidder_id to identify with the first two tables (train and test). In addition to these two columns the bids dataset includes the auction id, the merchandise for which the auction is being held, the device by which the bid was made, the time it happened, the country and ip it came from, and the url the user used.

The datasets are not meant to be used straight away for some casual pre-processing to then train a binary classifier, because the labeled dataset does not have any meaningful features to make classification with. Instead they shall be combined to make features. For example for every bidder corresponds a table with bids from which there can be features made by either counting the user's actions (e.g. number of bids, number of different auctions), or potentially more relevant ones, such as the bidding frequency in a certain auction, the number of different urls/phones/ips/countries changed in a specific time frame etc. Given the size of the data i think i will just be able to load the tables i want in dictionaries, and get the tables i want by looping through the bids dictionary for a certain bidder_id and i will not need to put them in a database.

Solution Statement

The basic idea of the solution is first that i will try to create some features, then do some pre-processing and finally train a binary classifier like a Random Forest or CSV. Essentially the end result would be a csv file with the bidder_id's of the test set bidders, and a corresponding likelihood the model assigns to them being robots - similar to the kaggle's sample submission solution [6]. After that, in order to evaluate the model, the ROC curve can be computed and the corresponding Area Under the Curve can be interpreted. That is also the proposed evaluation method by the competition's issuing authority.

Benchmark Model

It is easy to point out to a benchmark model for this case at least in terms of results. The competition that took place in kaggle 3 years before this submission, has 985 results from which around 45% of all the solutions scored above 90% in terms of Area Under the ROC curve evaluation. However, these approaches are not shared so i do not know how each algorithm performed as to have a more accurate benchmark for the one i will be using (RF, SVM etc.). Given the complexity of the task and the results of presumably more experienced data scientists, i would say that achieving above 90% would be satisfying.

Evaluation Metrics

The benchmark model's performance that is mentioned is already measured in Area Under the Receiver Operating Characteristic curve terms. That is also the proposed method for the solution model. The reasons for that - besides that it is convenient to compare with the benchmark results - is that it is quite robust compared to other metrics, and that by itself it offers a very nice probabilistic interpretation that can be useful in the sense that it describes the performance of the model beyond the narrow lines of a single operating point. For example if the auction company was to say that it is ok to have one false positive for one false negative - which is probably adjusted by how big of a damage the bots are causing to the consumer experience - by looking at the curve they can make this adjustment.

The key assumption that defines the interpretation of the AUROC curve is that there is a classifier that outputs a score $s(x)$, where x are the features, and a function $f(s(x), t)$, that given some threshold t it accepts elements of a number of classes (e.g. bot/no bot) and it takes a final decision about the instance of a class [7].

In terms of other possible metrics like accuracy, that is probably not a good metric for the given problem, given that both the training/testing samples are heavily skewed with human samples. On the other hand, maybe F1 score, precision and recall are also suitable alternatives but all things considered, given the problem statement, the overall context, the intended solution and the benchmark model interpreting the AUROC curve seems more compelling.

Project Design

The summary of the workflow is pretty simple and it consists of three steps: (1) create features for the train and test sets, (2) do some pre-processing of the features if necessary, and (3) build a classification model that can assign a likelihood of being a robot to a given bidder given that there are features created for him/her with the provided functionality. I plan to do the programming part in python 3 using the jupyter notebook through the anaconda distribution along with the well known data manipulation libraries for python like numpy, pandas, matplotlib.pyplot etc.

For the first step the plan as has been discussed before, is to create a bunch of features that describe the data by combining the information found in the bidder and bid datasets. It is obvious given what has already been said that for each bidder there is a table of bids that can be extracted from the bids dataset that can further be used to create features. There are some features that i am thinking of from the get go. These are either counts of each variable (for example in how many auction the user has participated in, or for how many types of merchandise), or a little more complicated features that take the time factor into account, which could be the mean bidding frequency in auctions, or the number of different urls/phones/ips/countries a user changes in a specific time frame etc.

The mean bidding frequency means isolating the table of the bids for each auction, then get the mean time interval (or some other descriptive statistic of the intervals distribution) and then the mean of this feature over all the auctions the user has participated in. In addition to that, for the numbers of different characteristics in a narrow time frame i am planning to use a sliding window based on the times of each bid and find for example how many countries on average are the bids coming from in a narrow time frame of for example 10 bids. However those things are not set in stone as i might change them if i see that they are not helping in any way, or if by being implemented a bit differently they improve the results, or if i get a new idea about making them along the way.

Given that this step is finished, the data will already be in some good numerical form because they would have been just created for that purpose. In that regard i think that the only adjustments i will have to make are to drop some columns from the bidders dataset (the payment_account and address as they are unique ids that are not useful for training). Also after looking at the distribution or correlations i will probably transform or drop some more columns of features that either have skewed distributions or that they are very much correlated with others. In addition to that i will normalize each column to improve the performance of the classifier in a manner that i can fit the transformation - MinMax - to the test data.

The final step would be to train the classifier. For that i will most likely use a Random Forest or Logistic Regression and perform a grid search to optimize it. I think i will avoid using an SVM because to my experience so far with it, it takes too long to train in particularly with large datasets. Also i will use the RF to get an intuition about the features and evaluate my prior assumptions about how useful they are. After that i am planning to use the predict_proba() method to get the likelihood for every bidder in the test set to be a robot. The final result will consist of a csv with ids and the scores, along with a replicable AUC graph that evaluates the process and can be compared with the benchmark.

References

[1] Strayer, W. Timothy, et al. "Botnet detection based on network behavior." *Botnet Detection.* Springer, Boston, MA, 2008. 1-24.

[2] Cooke, Evan, Farnam Jahanian, and Danny McPherson. "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets." *SRUTI* 5 (2005): 6-6.

[3] Cresci, Stefano, et al. "DNA-inspired online behavioral modeling and its application to spambot detection." *IEEE Intelligent Systems* 31.5 (2016): 58-64.

[4] Golle, Philippe, and Nicolas Ducheneaut. "Preventing bots from playing online games." *Computers in Entertainment (CIE)* 3.3 (2005): 3-3.

[5] https://datadome.co/how-to-detect-malicious-bots/

[6] https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot

[7] https://stats.stackexchange.com/questions/180638/how-to-derive-the-probabilistic-interpretation-of-the-auc