# Human or Robot

Xiuye Gu, Shuyang Shi
{xiuyegu, bsnsk}@stanford.edu

## Abstract

*In this project, we aim to use machine learning to solve a real-life problem of identifying robot bidders from human bidders. Our pipeline involves feature extraction, feature selection, model selection, and finally classification tests. Experiments reveal different performance achieved by different models, which shed some lights on what kinds of models are suitable for similar problems. Our best performance achieve 0.94 AUROC, which is between the 4th and 5th position on the private leader board. We use cross validation and feature selection during learning to boost performance. To make a better understanding in the experimental results, we do ablative analysis on the selected features. The code is available on Github:* `https://github.com/laoreja/Robot-or-human`.

## 1. Introduction

On online auction sites, bidders participate in auctions by bidding certain objects they want. Due to the existence of software-controlled bidders, i.e. robots, human bidders on the sites are becoming frustrated with their inability to win auctions, and therefore the core customer base of that site can be plummeting. In order to improve customer experience, platforms need to recognize robot bidders and eliminate their bidding from auctions.

Provided with some data about bidding information and labels of bidders, we aim to identify robot bidders according to their special patterns by machine learning techniques. The project mainly includes the following parts: feature extraction and selection, model selection, learning and classification, and ablative and error analysis.

## 2. Problem Statement

The problem is a previous Kaggle competition [1]. Given a bidding database, the goal is to classify human bidders and robot bidders based on their bidding behaviors.

The evaluation criteria is area under the ROC curve (AUROC), which is used in the competition, so that we can compare our results with those on the leader board.

## 3. Dataset

The provided dataset consists of the following two parts.

(1) *Bidder information*, which is basically the IDs and labels (robot or human) of bidders (labels of test bidders are not given). Although some information about payment account and address is also provided, the obfuscation for privacy makes them useless.

(2) *Bidding information*, which is shown in Table 1.

| Fields | Details |
|---|---|
| bid_id | unique id for this bid |
| bidder_id | Unique identifier of a bidder |
| auction | Unique identifier of the auction |
| merchandise | The category of the auction site campaign |
| device | Phone model of a visitor |
| time* | Time that the bid is made |
| IP* | IP address of a bidder |
| country | The country that the IP belongs to |
| URL* | URL where the bidder was referred from |

Table 1. Details on each field in the given bidding information table (fields with * are obfuscated or transformed to protect privacy).

The training set has 1984 labeled human and 103 labeled robots, and the test set has 4700 bidders (4630 of them participate bidding).

## 4. Manufacturing Features

One challenge of this problem is that we have to extract features of each bidder from their bidding information.In the following sub-sections, we probe into the given data, and extract relevant features accordingly.

### 4.1. Data exploration

By basic statistics, we find some distinguish patterns between humans and robots: (1) robots have more bids in auctions, and can be more active, *e.g.*, robots bid more quickly, which is revealed in the time interval between the two consecutive bids made by the same bidder ("tdiff"); and (2) robots win more auctions, which is not surprising because they are designed to win auctions. These differences is shown in Table 3, where each column is the average number of all humans or all robots. Actually, Figure 1 shows the significant difference between the distribution of total bids

| **Dense Features** | |
|---|---|
| **Feature (Dimension)** | **Reasoning** |
| # of bids made (1) | See Sec 4.1 |
| # of auctions participated (1) | Robots participated more auctions, similar distribution difference as Fig 1 |
| mean bids per auction (1) | Similar distribution difference as Fig 1 |
| # of country the bidder went (1) | similar distribution difference as Fig 1 |
| # of device/IP/URL used (3) | similar distribution difference as Fig 1 |
| # of auctions won (1) | Robots are usually designed to win auctions and indeed they are |
| Time difference between consecutive bids made by the same bidder (tdiff) (6) | Robots have much smaller time intervals, we use min, max, mean, median, std, and # zero intervals as features |
| Response time (5) | Robots bid faster other bidders bid, we use min, max, mean, median, std as features |
| bids' price (5) | Bidding patterns/strategy for human and robots are different. The features come from bidding time (a fixed increment of dollar amount for each bid), we use min, max, mean, median, std as features |
| Avg changing IP time (1) | Robots change IP address faster |
| Log entropy of IP/URL used (2) | Entropy(IP) = $\frac{N_{ips}!}{N_{ip1}!...N_{ipn}!}$, robots has higher entropy. Since the entropy can be very large, use logarithm instead. |
| **Sparse Features** | |
| **Feature (Dimension)** | **Reasoning** |
| # of bids made in each small time interval (432) | As is shown in Fig 2, we divide the each of the three bidding periods into 144 small time intervals, and count the number of bids each bidder made in each of these small intervals. Human and robots have different bidding pattern over time. |
| Percentage of bids made in each country (198) | Robots tend to appear in more countries, and they prefer to appear in certain countries (maybe relate to their agent). |
| Merchandise category (one-hot encoding, 10) | Each bidder only bids for one category, and certain categories attract more robots |

Table 2. Handcrafted features with reasoning (Features are extracted for each bidder).

made by human and that by robots. Most humans tend to have a small number of bids, while most robots have hundreds or thousands of bids in the dataset.

| Category | Human | Robot |
|---|---|---|
| # of bids | 1414 | 4004 |
| # of bids per auction | 6 | 23 |
| # of auctions won | 6 | 18 |
| tdiff median | 0.36 | 0.0026 |
| tdiff mean | 0.63 | 0.0112 |

Table 3. Some differences between robots and humans

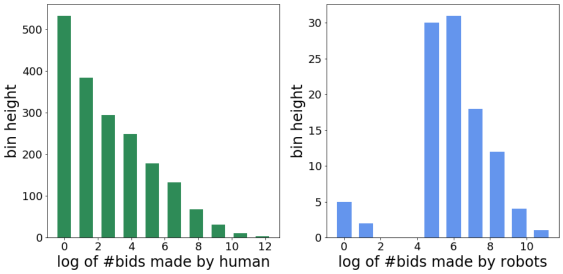

Figure 1. Distribution of number of bids made by human (left) and robots (right). Here we use the logarithm of the number of bids for clarity.

Since the transformed time given has $10^{15}$ magnitude, which will cause numerical overflow problems when applying some models, we need to interpret the time and transform it back to more suitable magnitude. By plotting the number of bids over time (discretizing the whole time span into 100 intervals) in Fig 2, We can observe a periodic pattern: there are three bidding periods and bids do not happen outside these periods; human bidders have several bidding peaks in each period, while the curve of robots is quite smooth within the periods. We transform the obscured time units into 31 time intervals of the same length [2].

## 4.2. Feature extraction

Following similar analysis as Sec 4.1, we extract a list of features as in Table 2, in which we present the features we extract and brief reasoning. Since some kinds of classifiers need features of larger dimension, we divide the features into two groups, where *sparse features* have more zeros in its entries and are prepared for such kinds of classifiers while *dense features* have relatively fewer zeros. These handcrafted features are likely to be useful, but we still need feature selection and more analyses in later sections.

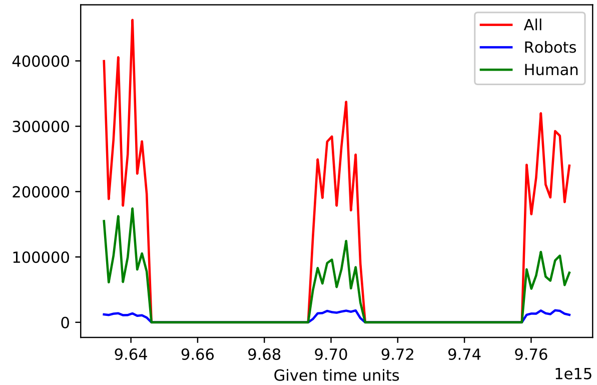

Figure 2. The number of bids over time. (The curve of "all" includes bidders from both the training and test set, while the other two curves only include the training set.)

## 5. Methods

We use different kinds of models on the problem to see how they perform differently. They are listed in Table 4 and elaborations are provided in the following paragraphs.

First of all, we train several baseline classifiers, which includes Logistic Regression (*LR*), SVM (both the linear version and with RBF kernel), and Decision Tree (*DT*).

Generative models are also tried. We use Gaussian Discriminant Analysis (*GDA*) to see how well the data fit into the assumption of two different Gaussian distributions.

We use several extensions of the decision tree model – Random Forest (*RF*), Gradient Boost Tree (*GBT*), and the AdaBoost of random forest (*RFAda*), which are popular for this kind of classification problems. Usually, RF is faster and GBT have relatively lower error, but they also tend to overfit for the amount of data is limited.

Due to the fact that tree models can be used as feature selection for further models in pipeline, we also experiment two composite models here, *RFLR* and *GBTLR*. Following CTR prediction [4], we use one hot representation for the output of tree models, and feed these feature into a Logistic Regression model for final classification. *RFLR* and *GBTLR* use Random Forest and Gradient Boost Tree as the tree model, respectively.

We test the power of deep learning on this problem (*DNN*) as well. Given the limited data amount and feature dimension, we use a three-layer multilayer perceptron (MLP), with ReLU as the activation function. The size of both hidden layers is 100.

## 6. Experiments

We first train each kind of model using the predefined feature sets, deriving the initial cross validation AUC and the test AUC. We then use the feature selection algorithm to select best feature set for each kind of model. We also do experiments on tuning hyper-parameter via random search.

| **Category** | **Models** |
|---|---|
| Basic Models | Linear Regression<br>SVM (Linear)<br>SVM (RBF)<br>Decision Tree |
| Generative Model | Gaussian Discriminant Analysis |
| Tree-based Models | Random Forest<br>Gradient Boost Tree<br>Adaboost Random Forest |
| Composite Models | RFLR<br>GBTLR |
| Deep Learning | Deep Neural Network |

Table 4. Models for the Problem

The experiments consist of cross-validation evaluation on predefined feature set, feature selection, and hyper-parameter tuning.

### 6.1. Accuracy on Predefined Feature Set

Noticing that different models prefer features of different magnitude of dimensions, we use both the sparse and the dense features (Table 2) to train DNN and linear and RBF SVM (The training of SVMs fail on the lower dimensional dense feature space; it's very likely that the data are not linearly separable in that feature space). And for other models, we use only dense features.

Except DNN (TF), we use K-fold Cross-Validation (*CV*) ($K = 4$) to evaluate the classification accuracy. After retraining the models on the whole training set, we also submit the results to Kaggle to obtain test AUC, for future comparison.

The results are shown in Table 5. The training AUC is also listed to present the generalization ability of each model. We can see from the table that GDA has a poor accuracy, which means the data probably does not fit into its Gaussian distribution assumptions. Linear SVM also has a poor accuracy, for the data may not be linear separable; SVM with RBF kernel and LR has medium accuracies. Besides, Tree-based models have very good CV accuracies (except DT), which means complicated tree models may be a better way to divide all these bidders. Here, test accuracies do not differ largely from CV accuracies, and thus the CV can be an effective indicator of model accuracies. By observing the training accuracies, we can see many models here can be potentially overfitting, except GDA has really limited capability of addressing this problem so that even the training AUC is quite low.

We implement two versions of DNN – one uses the off-the-shelf MLPClassifier class in the Sklearn Library, the other uses TensorFlow for a more flexible implementation, where weighted cross entropy loss is used to deal with the unbalanced data problem and batch normalization is used to prevent overfitting. We use Weight decay in both implementations.

CV AUC for DNN(TF) is not available here, because we tried to follow the convention of deep learning and use holdout validation, but there is always a non-trivial gap between the CV AUC and the Test AUC. One possible reason is that since the dataset is very small and unbalanced, it is impossible to split the provided training set into a training and a dev set and make the distributions of all the train, dev, and test set very similar.

Experimental results reveal that to make DNN work, we have to normalize the features with zero-mean and unit-std. Weighted cross entropy loss and batch normalization does not help much on this problem. Noticing that the performance of DNN is worse than RBF SVM and is much worse than RF, DNN does not work on this problem. One possible reason is that the dataset is too small, with less than 100 positive training examples, DNN is unable to learn.

| Models | Training | CV | Test |
|---|---|---|---|
| LR | 0.8732 | 0.8082 | 0.8121 |
| SVM (RBF) | 1.0 | 0.8874 | 0.8217 |
| SVM (Linear) | 0.9981 | 0.6655 | 0.5834 |
| DT | 1.0 | 0.6711 | 0.6050 |
| RF | 1.0000 | **0.9404** | 0.9330 |
| RFAda | 1.0 | 0.9373 | **0.9370** |
| GBT | 1.0 | 0.9180 | 0.9204 |
| GDA | 0.7521 | 0.7468 | 0.7369 |
| RFLR | 0.9937 | 0.9255 | 0.9144 |
| GBTLR | 0.9531 | 0.9125 | 0.8534 |
| DNN(TF) | 0.9655 | NA | 0.8441 |
| DNN(sklearn) | 0.9999 | 0.8384 | 0.8235 |
| DNN(unnormalized) | 0.5 | 0.5 | 0.5025 |

Table 5. Model accuracies (AUC) on different data

## 6.2. Feature Selection

Previously we extracted many relevant features, but in order to have better classification accuracy, we need to perform feature selection for each model so that they can use the features that suit them. For a given model, we start with an empty set of features, and each time select a candidate feature with largest marginal increase of cross-validation AUC.

Figure 3 shows the changes of AUC during the process of feature selection. For simple models like LR, DT, and GDA, their capabilities of adapting to features are quite low and the curves fluctuates a lot. For complicated models like RF, GBT, RFLR, GBTLR, their CV accuracies are more stable among the process of adding features. For these two kinds of models, we pick the feature sets that either maximize the CV accuracy (former kind), or makes the curve stable (latter kind), and evaluate them like before by CV and test accuracies in Table 6. Comparing them with previous results in Table 5, we discover that the models behave better on CV accuracies, and are less likely to be overfitting. As for the test results, simple models like LR, DT, and GDA has an escalation, which matches their curve in feature selection process. Complex tree-based models, however, are not having better accuracies. This is reasonable because (1) they rely on many features to perform the classification; (2) CV results can have minor fluctuations in different runs, and thus maximum CV accuracy for their steady curves does not guarantee better test accuracies.

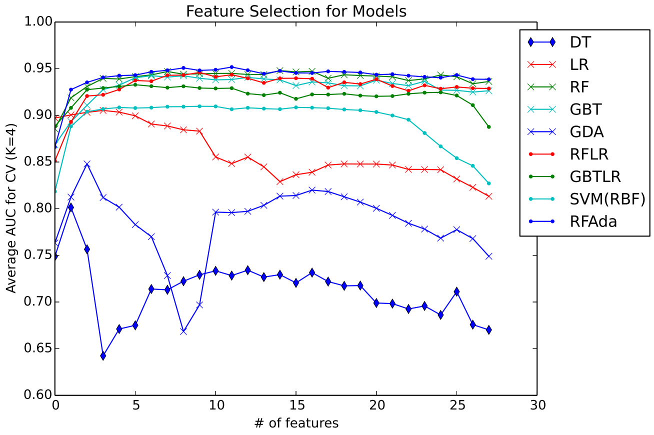

Figure 3. CV AUC against number of features during feature selection

| Models | Training | CV | Test |
|---|---|---|---|
| LR | 0.9054 | 0.9053 | 0.8901 |
| SVM (RBF) | 0.9982 | 0.9097 | 0.8479 |
| DT | 0.8679 | 0.8013 | 0.7934 |
| RF | 0.9997 | 0.9403 | **0.9259** |
| RFAda | 1.0 | **0.9455** | 0.9220 |
| GBT | 1.0 | 0.9341 | 0.9069 |
| GDA | 0.8491 | 0.8480 | 0.8247 |
| RFLR | 0.9888 | 0.9327 | 0.9138 |
| GBTLR | 0.9376 | 0.9121 | 0.8743 |

Table 6. Model Accuracies (AUC) with Selected Features

## 6.3. Hyper-Parameter Tuning

For models like GDA and LR, the model is completely decided by the data. But for models like Random Forest, there exist some hyper-parameters (*e.g.* the number of trees in the forest) that can largely influence the performance of the models. We apply a random search [3] method to automatically search for the best hyper-parameters, according to the AUC of cross-validation results.

For instance, for RF, we need to decide at least three hyper-parameters as follows. (1) *n_estimaters* – the number of estimators (trees) in the forest; more estimators can potentially have more complicated classification criteria for the task. (2) *max_depth* – the maximum depth of trees in the forest; the deeper the trees are, the better the trees fit training data, but trees too deep can easily overfit. (3) *max_features* – the size of the random subsets of features to consider when splitting a node.

Adopting the random search method, we can see the Cross-Validation AUC changes in Figure 4. Although the random search can have unstable result each round (the curve with dots fluctuates largely), after a few (∼10) round we can have a good result (the curve with crosses becomes steady soon). The best hyper-parameters for the run in Figure 4 are: *n_estimators*=305, *max_depth*=7, *max_features*=4. Actually, if we run this random search several times, we can obtain very close best AUC, though the hyper-parameters can have slight difference.

Similarly, for other models that have hyper-parameters, random searches are also applied for finding the best hyper-parameters.

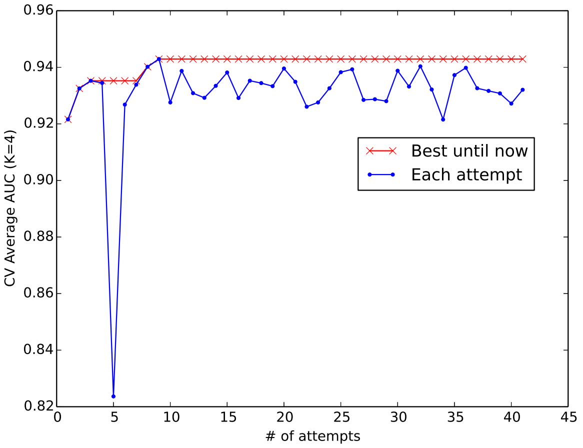

Figure 4. AUC changes in the process of hyper parameter selection

## 7. Analyses

Since the dataset is a Kaggle competition, we do not have access to the test labels. And as mentioned before, the training set is small and unbalanced and cannot be split into train/dev/test subsets while preserving the data distribution, so we are unable to find or split a test set with reasonable data distribution to do analyses like confusion matrix or ROC/Precision-Recall curves. However, we did some ablative analysis on features.

### 7.1. Ablative Analysis

After we selected the best feature set for each model, we do ablative analysis on the selected features to see which feature makes the biggest contribution. Due to the limit of space, we only present the results on one of the models with best performance, RF.

The results are shown in Table 7. We can observe that the contribution of each feature is very similar; even with only one feature, the CV AUC still achieves 0.85. The larger gaps in the last four features do not necessarily mean that they are the most important, since the low dimension of features can affect the model's performance. The similarity among the importance of features may relate to the internal property of random forest, where subsets of features are randomly chosen to train decision trees, and then form a forest.

| Feature | CV AUC |
|---|---|
| Full feature sets | 0.9416 |
| median of tdiff | 0.9448 |
| # of device | 0.9347 |
| min of price | 0.9300 |
| std of price | 0.9315 |
| min of response | 0.9232 |
| log entropy of ip | 0.9250 |
| # of country | 0.9269 |
| # of bids | 0.9255 |
| min of tdiff | 0.9178 |
| Avg changing IP time | 0.9119 |
| # of auction | 0.8988 |
| log entropy of url | 0.8897 |
| mean of tdiff | 0.8523 |

Table 7. Ablative analysis on RF, removing features one by one.

## 8. Conclusion

In this real-life problem, different machine learning models have very different results. By feature extraction, feature selection, model selection, and ablative analysis, we understand that tree-based models have good accuracies in such problems, while models like GDA make poor options, and we explored the way to find the best possible method in such problems. This experience can be well-generalized in other machine learning problems.

## References

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

[2] `http://small-yellow-duck.github.io/auction.html`.

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[4] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.