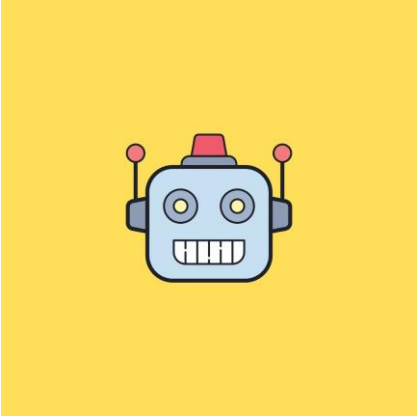
SMU
SINGAPORE MANAGEMENT UNIVERSITY

CS421 Introduction to Machine Learning



Human Or Robot

Topic	**G1T8 Project Report**	
Instructor	**Professor Dai Bing Tian**	
People	**Name**	**Student ID**
	Sze To Ming Hong, Brennan	**01394350**
	Chu Wei Quan	**01355438**
	Ong Guang Qi	**01358545**
	Wu Jiafang	**01364516**
	Loo Yang Kai	**01357240**

1. Problem Definition and Use Cases

Online auction sites face the problem of an increasing number of software-controlled bots performing bids. Bots are able to place bids at high speeds and at different auctions simultaneously, causing real human bidders to often lose out. Becoming increasingly frustrated with their inability to win auctions, human bidders are leaving the auctions sites, causing usage from the sites' core customer base to plummet. Our aim of this project is to identify human bidders from bots which is essentially a Binary Classification problem.

Business Use Case

By identifying bots from human bidders, online auctions sites administrators can then take the appropriate action to remove and ban these bots from their sites. Auction owners can also ban or remove bots from participating in current and future auctions, improving the bidding process and experience of human bidders.

Metrics Explanation

Accuracy in our context describes how well the model is able to accurately make classification predictions between human and bot bidders. We will use accuracy as a measure to indicate to us whether the model is overfitting to the training dataset and hence unable to generalize to unseen data.

In our problem false negatives (FN) are bot bidders that have been incorrectly classified as human bidders. False positives (FP) are the human bidders that have been incorrectly classified as bot bidders. As such, we want to minimize FN as much as possible, hence we prioritise Recall over Precision. As such we also use FBeta, with a beta of 2.0 to give more weight to recall, as one of the performance measures to ensure that we are minimizing FN.

We have also chosen ROC AUC as our performance measure as it is one of the most widely used metrics for evaluation for binary classification problems. Having a higher ROC AUC would indicate a higher probability of detection (TPR), meaning that we are able to detect bots more accurately.

2. Dataset and Pre-processing

Initial features

Of the initial 11 features that were provided, the following are the only features that are useful for analysis:

- Bidder_id: Unique identifier of a bidder (Discrete)
- Auction: Unique identifier of an auction (Discrete)
- Merchandise: Type of merchandise (Categorical)
- Device: Phone model of a visitor (Categorical)
- Time: Time that the bid is made. This feature has been obfuscated (Continuous)
- Country: The country that the IP belongs to (Categorical)
- IP: IP address of a bidder (Categorical)
- URL: URL where the bidder was referred from (Categorical)

Imbalance of the dataset

Based on our initial analysis of the dataset, we found out that the dataset is highly imbalanced, with a ratio of approximately 95% human and 5% bot bidders (Refer to Figure 1 in Appendix). As such, we can expect that models trained on the initial dataset without any pre-processing will have poor predictive performance for the minority class. We can also expect the FBeta

score on the initial model to be low due to this imbalance as well. (Refer to Figure 2 in Appendix)

Clustered data points making it hard for the model to distinguish between human and bots

Plotting the features during EDA also reveals that the human and bot bidders are clustered around each other foremost of the features as shown in Figure 3 in the appendix. As such, we can expect the model to have lots of difficulty trying to distinguish the decision boundary between human and bot bidders.

Bias variance Learning Curve

In order to train an accurate model, the model should have low bias and variance. By plotting a learning curve for each model, we found out that the models have relatively low variance but high bias (Refer to Figure 4 in Appendix). This means that with the initial features, the model is able to accurately make predictions around the same area, but predicts slightly far off from the actual outcome.

Feature extraction and engineering hypothesis / intuition

Based on the initial features alone, we know that the models would be unable to distinguish between human and bot bidders. Hence, we will need to generate more features using feature extraction and engineering.

Here are some of the hypothesis and intuition we came up with when generating new features:

- Instant response: Based on real world experience, it is difficult for humans to perform two simultaneous bids at the same time. However, bots are capable of performing such actions, therefore a simultaneous bid would likely identify as a bot bidder. This feature can be mapped counting the number of bids more than 1 at a unique point in time.
- Percentage of auctions above threshold: the goal of bots is to secure the item that a user wants to bid for. As such, we formulated a hypothesis that a bot would perform more bids on auctions that the user wants to win. Percentage of auctions a bidder has participated in where they bidded more than the average number of bids compared to other users.
- First and last bids: we hypothesized that bots would likely have the behaviour of bidding first into an auction as it has been programmed to search the auction site for a particular item and place a bid on it; and the behaviour of bidding last into an auction to secure a bid on that item. This feature is derived from the number of times a unique bidder ID has performed a first bid and last bid given an auction using the time feature.

3. Machine learning model and technique

In this section, we will first start with the techniques employed to help our models achieve better results and move on the individual models and the intuition behind them.

Techniques

a. Resampling

In our project, we employed a method called resampling which combines Synthetic Minority Oversampling Technique (SMOTE) and random undersampling. For this to work, we first import the imbalanced-learn python library. Subsequently we manually create a pipeline to construct a sequence of SMOTE and undersampling techniques to apply to the dataset. The pipeline can choose to first apply an oversampling

technique to the dataset then apply undersampling to the output of the oversampling transformation before giving us the final outcome or vice versa. This helps to bring the minority class distribution to a higher percentage of the majority class and RandomUnderSampling brings the majority class down to a lower percentage, resulting in a more balanced dataset.

b. Stratified K Fold (Refer to Figure 5)
We also employed stratified k fold to split up our dataset (Refer to Figure 5 in Appendix). This technique involves splitting the training dataset into k folds while maintaining the same class ratio throughout the K folds as the ratio in the original dataset. This ensures that the proportion of the human to bot bidders is the same across the original data, training set and test set.

c. Feature Selection (RFE and combined ranking approach)
As for feature Selection, we had 2 approaches. The first approach was Recursive Feature Elimination(RFE). For each model, RFE recursively removes the weakest feature and re-trains the model. This is performed multiple times until the specified number of features is reached. This produces different features for each model, making it an unfair comparison between the models (Refer to Figure 6 in Appendix).

The second and eventual approach we took was the combined ranking approach. We first rank each of the model's features based on the feature importance, then combine them together into a DataFrame while retaining the original feature ranking of each model. We then sum up the ranks of each model to obtain a total rank, where the lowest total indicates the most important feature among the models (Refer to Figure 7 in Appendix). This takes into account the original feature importances of each model rather than arbitrarily selecting features to use.

d. Hyperparameter tuning with RandomSearch and GridSearch
Hyperparameter tuning is an important step as it has a big impact on the model convergence. GridSearch tests for every possible combination of hyperparameters. Given a large search range for each hyperparameter, it takes up much computational time in order to converge to the best hyperparameters. As such, we first perform RandomSearch as it selects random combinations of hyperparameters, obtaining a set of hyperparameters quickly even if there is a large search range. Using those hyperparameters generated from RandomSearch, we then narrow down our search range for GridSearch, tackling the problem of extensive computation, to finally obtain the best set of parameters.

Models

Moving on to the models, we created 4 models and 1 stacking model to stack the first 3 classifiers with hyperparameters found through RandomSearch and GridSearch. In this section, we will be going through how the model works and why it is useful and not useful.

a. Classifier Models (Decision Tree, Random Forest and XGboost)
The first model we used was the Decision Tree classifier. We started by initiating our DecisionTreeClassifier with random_state of 42. We then fit our decision tree with the train and test sets that were created previously. We then printed out the accuracy scores. We repeated this for the 10,15,20,25,30,35,40 features datasets. We repeated the whole process with RandomForest and XGBoost.

b. Stacking
We utilised the best estimators for each model, then incorporated them into a StackingClassifier using LogisticRegression as the meta model. The passthrough will

be set to True, allowing the meta model to see and learn from the training data as well together with the predictions made by the best estimators. This helps the metamodel to “learn” from “mistakes” made by each classifier while retaining the original training data.

c. ANN

After creating the models for the supervised classification models earlier. We proceeded to deep learning with our own Artificial neural network. We first create an ANN that has 2 Dense layers with activation function ReLu and 1 output layer with activation function as sigmoid . We then wrapped this model using KerasClassifier and trained the keras classifier. We then used the same sets of evaluation techniques to find the AUC, f1 Score and accuracy of the ANN. We repeated this for 10,15,20,25,30,35,40 features datasets.

Figure 8 in the appendix shows the results from the 10,15,20,25,30,35,40 features experiments. Based on the results, we have decided to go with 30 features as 35 features generally show a huge dip in the performance metrics

4. Justification of the technique that you’ve chosen for section 3

Decision Tree	It is the simplest and minimally effective classification model. It is also relatively easier to interpret and understand as compared to many other models due to the straightforward nature of the model. Furthermore, it requires minimal preparation or data cleaning.
Random Forest	It offers randomised feature selection, and it does not rely on the feature importance given by a single decision tree. By not depending highly on any specific set of features, it helps the random forest model to reduce the variance seen in decision trees, ultimately helping the model to achieve higher accuracy than a decision tree model.
XGBoost	It helps to reduce the prediction error of badly classified observations at each step by trying to reduce the bias of the weak learners. In addition, if the gain from a node is found to be minimal, XGBoost stops constructing the tree to a greater depth. This helps to overcome the challenge of overfitting to a great extent.
ANN	With the complex nature of our dataset due to the many possible scenarios, ANN is used for the project as a neural network may be helpful to find non-direct patterns.
Resampling	SMOTE deals with class imbalance by generating synthetic samples for the minority class, which helps to overcome the overfitting problem posed by random oversampling. In addition, RandomUnderSampling was used in conjunction with SMOTE to reduce the number of examples for the majority class, in an attempt to reduce the bias associated with imbalanced classes of data.
Stratified KFold	StratifiedKFold was used as it equally splits the minority classes for the train and test set for each fold. This results in a better representation of actual accuracy as StratifiedKFold provides a better measurement of accuracy, including taking into account less overfitting of the model.

Feature Selection	Feature selection was also adopted for the project as having many features creates a highly complex model, which tends to overfit to the training data. By eliminating features, it helps to improve interpretability of the model and computational efficiency and removes features that produce noise which may result in a lower accuracy of the model.

5. Can your model provide some explanation to your predictions

Why our model differentiates in terms of predictions		
Model submission	**AUC ROC**	**Hyperparameters**
XGBoost	0.89187	{'colsample_bytree': 0.8, 'gamma': 0.3, 'max_depth': 4, 'min_child_weight': 1, 'subsample': 0.9}
Random Forest	0.87959	{'bootstrap': False, 'max_depth': 90, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 1100}
Stacking Model	0.87064	-
ANN	0.76866	{'momentum': 0.0, 'learn_rate': 0.3, 'epochs': 100, 'batch_size': 10}
Decision Tree	0.75774	{'criterion': 'entropy', 'max_depth': 50, 'max_features': 25, 'min_impurity_decrease': 0.0005, 'min_samples_leaf': 21, 'min_samples_split': 7}

As our dataset originated from the Facebook recruiting competition on Kaggle platform, they had an unseen test dataset where we can test out our models. We submitted our models to see which model produced the best predictions based on unseen test dataset.

According to our Kaggle submissions, we observed that XGBoost produced the best results of 0.89187 amongst all the other models. Our team was wondering why XGBoost was producing the best results. Thus, we came out with the following research.

Decision Trees

Decision Trees are mainly generated with the algorithm Iterative Dichotomiser 3 (ID3). The algorithm will begin with the original set S (the dataset) as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy or information gain of that attribute and decides on which attribute to split upon.

Thus, since ID3 is highly reliant on the quality of training dataset, it can overfit rather easily. This model would be suitable for a rather small and balanced dataset. However, in our case, we have a rather imbalanced dataset of 95% human vs 5% bots. ID3 grows each branch just deeply enough to perfectly classify the training examples but it can lead to difficulties when the number of training examples is too small to produce a presentative sample of the true target function. ID3 would easily classify information to be biased in favour of the dominant class (Humans).This causes the rate of detection to decrease for AUC ROC.

ANN

According to Anand et al, who explored the effects of class imbalance on the backpropagation algorithm in shallow neural networks. It appears that in class imbalanced datasets, the length of the minority class's gradient component is much smaller than the length of the majority class's gradient component. This means that the majority class is dominating the net gradient that is responsible for updating the model's weights. This results in slightly tailor the model prediction to be overfitted for the majority class (Humans). This also reduces the error of the majority group very quickly during early iterations, but often increases the error of the minority group, causing the network to get stuck in a slow convergence mode. Another possible explanation that ANN underperforms as compared to XGBoost may stem from the fact that there are a total of 6209 parameters as shown in figure 9 which is more than our features, and this may cause the model to not converge. This is coined as a "dying ReLU" problem. Under certain conditions, any neuron with a ReLU activation can be subjected to a (bias) adjustment leading to it never being activated ever again. This can be fixed with a "Leaky ReLU" activation which is outside of the scope of our study.

Random Forest vs XGBoost

According to our Kaggle submission, we quickly realized that our random forest model had relatively close results with our XGBoost model. We wanted to understand why the models had such a slight difference in predicting our classes.

Firstly, we must understand the underlying idea behind the two algorithms for XGBoost and Random Forests. Gradient boosting re-defines boosting as a numerical optimization problem where the objective is to minimize the loss function of the model by adding weak learners using gradient descent. Moreover, it does not modify the sample distribution as weak learners train on the remaining residual errors of a strong learner. This allows the model to give more importance to misclassified observation. The contribution of each weak learner to the final prediction is based on a gradient optimization process to minimize the overall error of the strong learner.

In comparison, Random Forests is a bagging technique that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy. Instead of relying on one tree, it takes predictions from many trees and based on majority votes of prediction, it predicts the final output. This leads to higher accuracy and prevents the problem of overfitting. However, it does not cater specifically for imbalanced datasets.

There are mainly a few reasons why XGBoost is better in this scenario compared to Random Forest.

a. XGBoost better at overcoming issue of overfit
   XGboost straight away prunes the tree with a score called "Similarity score" before entering into the actual modelling purposes. If a gain from a node is found to be minimal, it would just stop constructing the tree to a greater depth which can overcome the challenge of overfitting. Meanwhile, the random forest model might probably overfit the data if the majority of the trees in the forest are provided with similar samples (Humans samples).

b. XGBoost is a good option for unbalanced dataset
   In applications like forgery or fraud, even our bot detection, the classes will almost certainly be imbalanced as most of the authentic users will be huge when compared to fake bot users. In XGBoost, when the model fails to predict the anomaly for the first time, it gives more preferences and weightage to it in the upcoming iterations thereby

increasing its ability to predict the class with low participation; but we cannot assure that random forest will treat the class imbalance with a proper process.

Stacking

The idea of stacking is to learn several different weak learners and combine them by training a meta-model. We picked XGBoost, Random Forests and Decision Tree models to be put into our Stacking Classifier as they were providing the best results that were given. Our meta-model chosen was a Logistic Regressor. We engaged in a StratifiedKFold cross-training approach such that all the observations can be used to train the meta-model. We split the training data in five folds. We then allow weak learners to learn and make predictions for the observation and fit the meta-model on the last fold, using predictions made by the weak learners as input. Figure 10 shows the results of stacking compared to the other classifiers.

Given that mainly the meta-model feeds off the predictions made by the weak learners, we quickly realised that the information being fed into the meta-model is important.

Model diversity is something we did not consider for when we were trying out the stacking classifier. We should have considered what new information each model is bringing to the table. According to 'An analysis of the Impact of Diversity on Stacking Supervised Classifiers', a general analysis of stacking classifiers and its diversity of models indicates that there was more gain of stacking when there is more diversity in the experiments. Stacking classifiers with lower diversification of models had a loss of quality.

In our scenario, we mainly put Decision Trees, Random Forest and XGBoost into the meta-model and all of them are a decision-tree-based machine learning algorithm. We felt that the low diversification of information we provided the meta-model contributed to the loss in quality of predictions. Thus, this decreased the Kaggle score for our Stacking Classifier.

6. Future issues

Engineering the features and running the models takes a long time and increase in data will only contribute to more to the computational load

With the current implementation of our machine learning pipeline, from data pre-processing to modelling, it already takes a significant amount of computational time on 7 million rows of bid data. In future, an increase in the amount of bid data fed into training the model is too computationally expensive, which poses a potential pitfall in future when training the model. As such, identifying the best features to identify bot bidders is extremely important as it helps to significantly shorten the time taken for feature engineering.

Bots behaving more like humans

With rapid development of technology and AI, a potential issue that is likely to arise would be the change in behaviour of bots to be more like humans. It would be harder for the model to identify bot bidders if they are able to closely mimic human bidders' behaviour as the features engineered currently leverages on the fact that bot bidders behave differently from humans. As such, more data related to bots should be collected and analysed to understand bidding trends or specific timings when bot bidders are active. Alternatively, understanding the users' demographics would benefit in preventing bot bidders from creating an account on the bidding site, which leads to lesser bot bidders in the first place.

Classifying multi-class for different profiles of human bidders

According to our initial data analysis, we realized that there are a few human bidders who have over hundreds of thousands of bids (Highest being 500k). These are the outliers but as the auction platform grows, we should consider that there would be different types of human bidders coming into play. Some classes to be considered would be normal retailer auction bidders or huge corporations as bidders. In scenarios where multi-classification is necessary, our random forest model would be a huge contribution on how we can classify these classes as well.

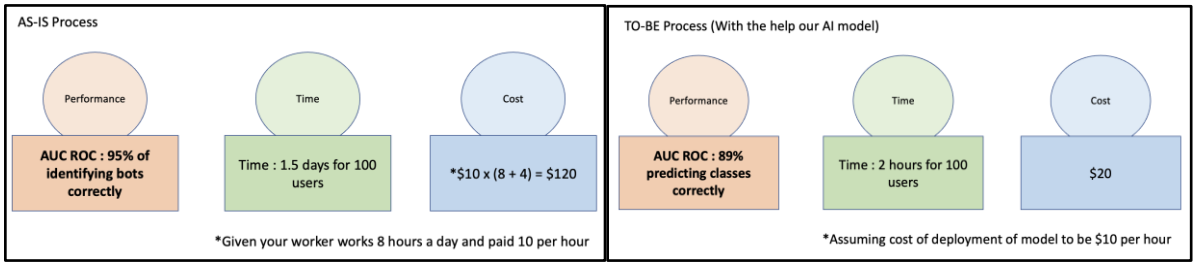
7. Conclusion (why should you use our model)

In conclusion, our model is able to correctly identify and predict 89% of the bots present in the online auction site. Although we had a poor dataset, we were able to already able to correctly identify 89% of the bots present on the platform, hence if we obtain unobfuscated data, we will be able to increase our prediction accuracy.

We highlighted 3 main factors on why the online auction company should use our model below. The 3 factors are shown below.

- **Performance:** technical performance such as model accuracy in identifying bots
- **Time:** time taken to identify bots
- **Cost:** how much it costs to identify bots

The three main criteria to evaluate your model in a business



Although our AI model might be less accurate than humans, it can take much less time to inspect the same number of users which means 18 times faster than manual inspection. Moreover, it saves $100 per 100 cases. Hence, we believe that our AI model brings great benefit to the auction company operating as it helps to save costs and time.

8. References

ID3 - https://medium.com/@pralhad2481/chapter-3-decision-tree-learning-part-2-issues-in-decision-tree-learning-babdfdf15ec3

ANN imbalance classification impact - https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0192-5

XGBoost better at overcoming overfit - https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Stacking - https://www.scitepress.org/Papers/2017/62912/62912.pdf

9. Appendix

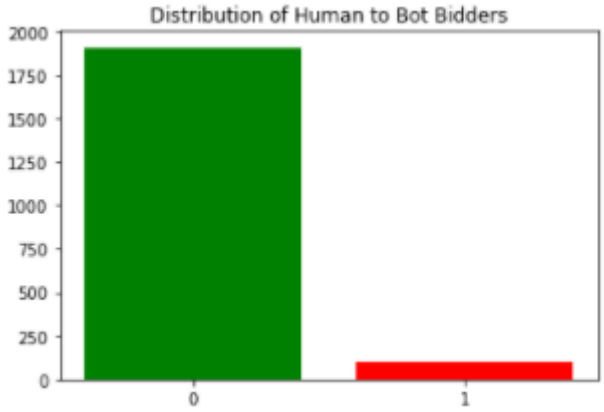


Figure 1. Imbalance of the dataset between human and bot users

Initial Results of the base models with 8 features

Model	DT	RF	XGB	ANN
Accuracy	0.9354	0.9603	0.9619	0.8626
ROC AUC	0.6109	0.8841	0.8507	0.3264
Fbeta	0.3261	0.2174	0.4032	0.1437

Figure 2. Results with initial features for each model

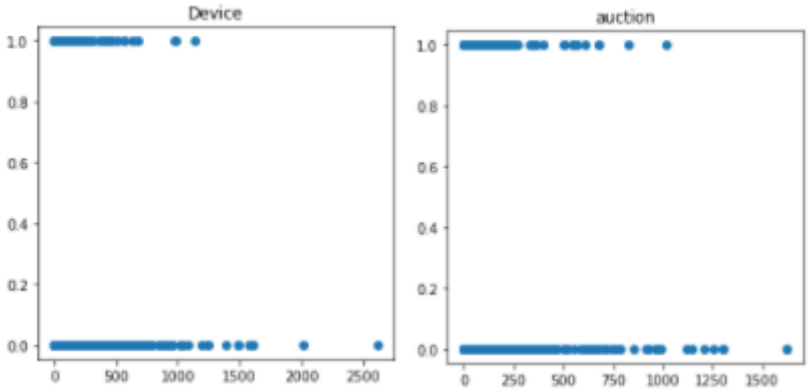


Figure 3 Clustered data points make it hard for the model to distinguish between human and bot bidders

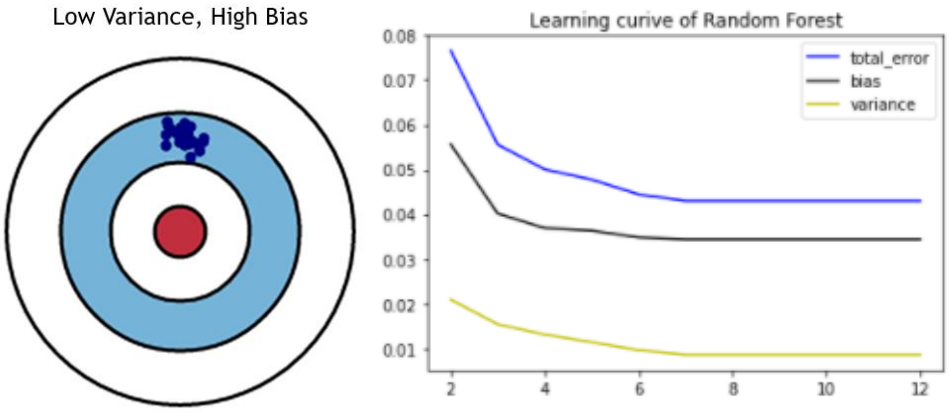


Figure 4. Learning curve plotting bias, variance and total error

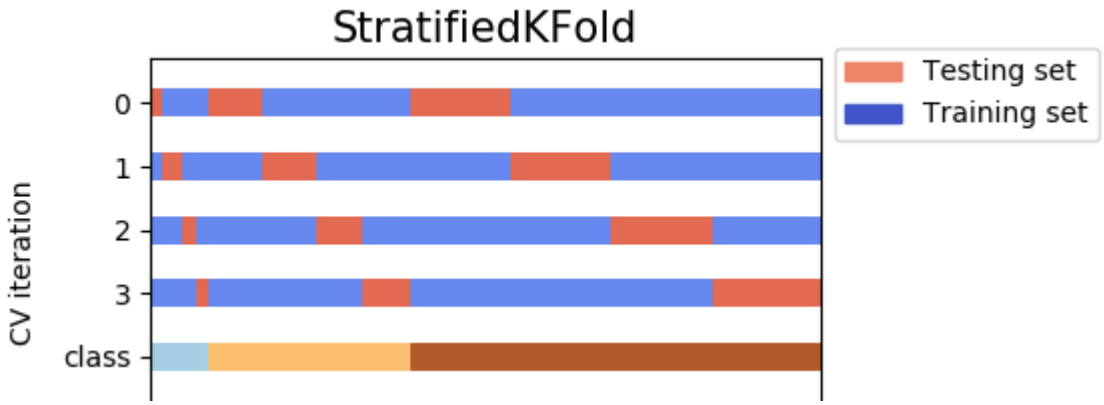


Figure 5. How StratifiedKFold splits the training and testing dataset

	XGBoost features	selected
2	time	True
3	country	True
4	ip	True
5	url	True
6	num_bids	True
7	num_first_bids	True
8	num_last_bids	True
9	time_to_bid	True
10	inst_resp	True
15	computers	True
18	jewelry	True
22	num_bids_per_auction	True
23	num_bids_per_device	True
24	num_bids_per_country	True
25	num_bids_per_ip	True
33	mean_devices_per_auction	True
37	mean_ip_per_auction	True
41	mean_url_per_auction	True
46	no_of_auction_exceeds_threshold	True
51	on_url_that_has_a_bot_mean	True

	Random Forest features	selected
2	time	True
4	ip	True
6	num_bids	True
9	time_to_bid	True
10	inst_resp	True
11	perc_inst_resp	True
22	num_bids_per_auction	True
23	num_bids_per_device	True
24	num_bids_per_country	True
25	num_bids_per_ip	True
26	on_ip_that_has_a_bot_mean	True
27	ip_entropy	True
28	url_entropy	True
32	std_country_per_auction	True
33	mean_devices_per_auction	True
37	mean_ip_per_auction	True
41	mean_url_per_auction	True
46	no_of_auction_exceeds_threshold	True
47	percentage_of_auctions_above_threshold	True
51	on_url_that_has_a_bot_mean	True

Figure 6. RFE features comparison between XGBoost and Random Forest

	dt_rank	Features	dt_weights	rf_rank	rf_weights	xgb_rank	xgb_weights	final_rank
0	1	num_bids_per_ip	0.066281	7	0.036461	9	0.020130	17
1	17	num_bids_per_auction	0.007070	1	0.093065	0	0.213447	18
2	0	num_bids_per_country	0.555570	5	0.045099	22	0.012542	27
3	7	mean_ip_per_auction	0.026810	18	0.015114	5	0.029687	30
4	9	perc_inst_resp	0.022071	3	0.060903	18	0.014533	30
5	3	time	0.039731	6	0.041678	25	0.011852	34
6	5	mean_url_per_auction	0.030475	13	0.020574	20	0.013554	38
7	39	inst_resp	0.000000	2	0.068570	1	0.106258	42
8	37	num_bids	0.000000	4	0.057389	3	0.064537	44
9	2	time_to_bid	0.048772	10	0.027101	33	0.008664	45
10	21	num_last_bids	0.005206	17	0.015239	7	0.024257	45
11	31	percentage_of_auctions_above_threshold	0.000000	9	0.034575	6	0.025204	46
12	18	device	0.006969	20	0.014425	8	0.020187	46
13	20	on_url_that_has_a_bot_mean	0.005985	11	0.025198	15	0.015770	46
14	49	num_bids_per_device	0.000000	0	0.101344	2	0.089635	51
15	14	ip_entropy	0.008892	12	0.023732	26	0.011802	52
16	11	ip	0.017186	14	0.020087	29	0.011108	54
17	12	max_country_per_auction	0.012854	30	0.011132	14	0.016390	56
18	24	max_ip_per_auction	0.000000	16	0.016822	16	0.015237	56
19	6	mean_devices_per_auction	0.028039	19	0.014431	32	0.009916	57
20	4	url_entropy	0.033945	15	0.020053	40	0.003345	59

Figure 7. Combined ranking between Decision Tree, Random Forest and XGBoost models

Figure 8. Results of experiments using 10 to 40 features for XGBoost and Random Forest

Layer (type)	Output Shape	Param #
dense_54 (Dense)	(None, 64)	1984
dense_55 (Dense)	(None, 64)	4160
dense_56 (Dense)	(None, 1)	65

Total params: 6,209
Trainable params: 6,209
Non-trainable params: 0

Figure 9. Parameters of ANN

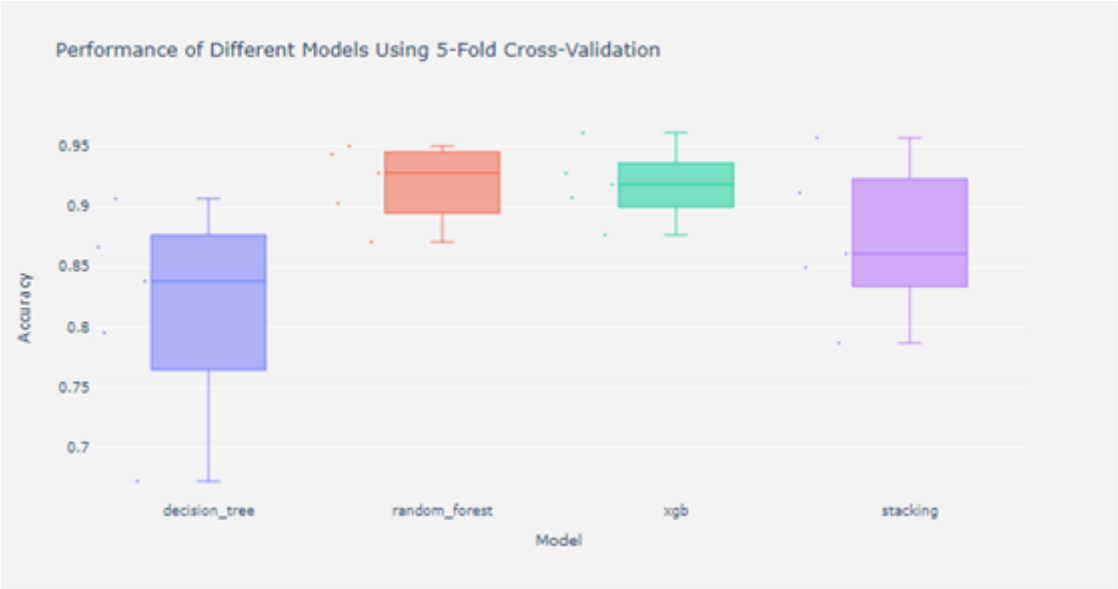


Figure 10. Results of stacking classifier against other classifiers