Bidder classification

Master Degree in Artificial Intelligence

Course: Machine Learning

Academic Year: 2024 - 2025

Authors:
Dario Simone - VR512966
Fidanza Riccardo - VR516130

UNIVERSITÀ
di VERONA

University of Verona

February, 2025

Abstract

Detecting and classifying bots in online auctions is crucial to ensure fair competition and maintain the integrity of the platform. This study explores two distinct methodologies for bot detection: one, inspired by the state of art, that classifies bidders based on aggregated features and an alternative one, implemented by us, that labels individual bids as bot-generated or human-generated before making a final classification of the bidder. By applying these approaches to a bidding activity data set, we analyze their effectiveness in distinguishing human bidders from automated bots. Our findings highlight the strengths and limitations of each method. The comparison of these methodologies offers valuable guidance for improving bot detection systems in online auction platforms.

Contents

1 Motivation

[3]The proposed project aims to address a classification problem related to the detection of automated bidding activity in online auctions. Successfully detecting these bots enables site owners to flag and remove fraudulent users, ensuring fair auction practices and a level playing field for legitimate bidders. This challenge comes from from a Kaggle competition created by Facebook as part of their hiring process.

This problem is particularly significant, as it allows extensive feature engineering and data preprocessing, both of which are essential for building a robust and generalizable classifier. By extracting meaningful insights from bidding behavior, we can develop models capable of distinguishing between human and automated participants.

2 State of the Art

After reviewing several Kaggle notebooks related to this competition, we observed that the majority of approaches primarily focus on analyzing the bidders themselves, often by extracting relevant features from the bids dataset to characterize each bidder.

2.1 Notebook of competition winner

[2]Achieving an AUC score of 0.95 using a Random Forest model, the winner notebook indentified several useful features for classifying users as bots or humans. Here are the most important:

- The **median time between a user's bid and the previous bid** in the same auction, which provides insight into how often the user places bids in a short period of time.
- The **maximum number of bids in a 20-minute span**, reflecting the intensity and frequency of bidding activity in a short timeframe.
- The **minimum and median times between a user's bid and another user's bid** in the same auction, which can help distinguish between normal bidding behavior and rapid activity.

2.2 Notebook of random forest

Chong and Zhenjie (2021) [1] conducted a study utilizing a Random Forest model for human-robot classification, following a comprehensive feature extraction process. Their approach served as an inspiration for our work, particularly in the way they engineered meaningful features to improve classification performance. We adapted some of their feature extraction techniques and integrated them into both of our methodologies, to enhance the detection of anomalous bidding behavior in our dataset.

3 Objectives

The goal of the project is to present different solutions to the problem by exploring various methodologies and comparing them with the state-of-the-art approaches. The primary evaluation metrics is the **AUC score** (used for ranking in the Kaggle competition), then we consider also **recall**, and **F1-score**. Since this is an anomaly detection problem, recall and F1-score are crucial because they help evaluate how well the model identifies anomalies. High recall ensures that most anomalies are detected, while F1-score balances precision and recall, providing a more

comprehensive performance measure.

The specific objectives of the project are:

- **Work on bidders**, following the state-of-the-art methodologies.
- **Work on bids**: make predictions directly on bids, count them, and then determine the final decision based on the majority rule per bidder. This approach differs from the state-of-the-art methods.
- **Compare the previous two approaches** and identify the best one, to understand if our new proposed model works better.

4 Experimental Process

To achieve our objectives, we followed this steps.

4.1 Analysis of the dataset

The dataset is composed by 2 different *csv* file:

- bids
- train

The *bids.csv* contains all the bids placed by the bidder without labels, the *train.csv* contains all the bidder with the labels (bot or human).

Dataset	**Samples**	**Features**
Bids	7,656,334	9
Bidders	2,013	4

Table 1: Dataset Dimensions

Since the bidder data set is relatively small and the bid data set is much larger, it is probably more efficient to focus on the bidder data set initially. Working with a smaller data set will enhance faster processing and analysis.

4.1.1 Bids dataset

Feature	Description	Type
bid_id	Unique ID for this bid.	String
bidder_id	Unique identifier of a bidder.	String
auction	Unique identifier of an auction.	String
merchandise	The category of the auction site campaign. This indicates that the bidder might come to the site via a search term (e.g., "home goods") but ends up bidding on something like "sporting goods".	Categorical
device	Phone model of a visitor.	String
time	The time the bid is made (transformed for privacy).	Timestamp
country	The country associated with the bidder's IP address.	Categorical
ip	IP address of a bidder (obfuscated for privacy).	String
url	The URL where the bidder was referred from (obfuscated for privacy).	String

Table 2: Bids dataset feature description

4.1.2 Bidder dataset

Feature	Description	Type
payment_account	Payment account associated with the bidder (obfuscated for privacy).	String
address	Mailing address of the bidder (obfuscated for privacy).	String
outcome	Label indicating whether the bidder is a robot (1.0) or a human (0.0).	Binary

Table 3: Bidder dataset features

4.1.3 Feature exploration

Before starting the first test, it is important to explore the two datasets. To achieve this goal, the counting of the unique values for each feature has been implemented. A feature with few unique values could help to find pattern between the samples. Moreover, we ensure that no *Nan values* are present inside the different columns. After this analysis *merchandise*, *device*, *auction* and *country* appear to be the most important.

4.2 First test on bids dataset and overfitting

After identifying the most important features, we need to convert all string-based features into numerical representations to ensure proper training and validation of the dataset (*encoding*). Once encoded, we can run a *Decision Tree* model on the bids dataset to analyze how these features influence the classification. The results are shown in the figure 1.
The performance is outstanding, but why? We realized that our model learns too much about each individual bidder. In fact, on the test set, the performance drops drastically. This indicates a case of overfitting, where the model memorizes specific patterns rather than generalizing to new, unseen data. It is critical to carefully select the features used during the training phase to create a classifier that can distinguish human bidders from bots based on the bids. We saw that including the bidder id in the model carries a significant risk of overfitting and excluding it

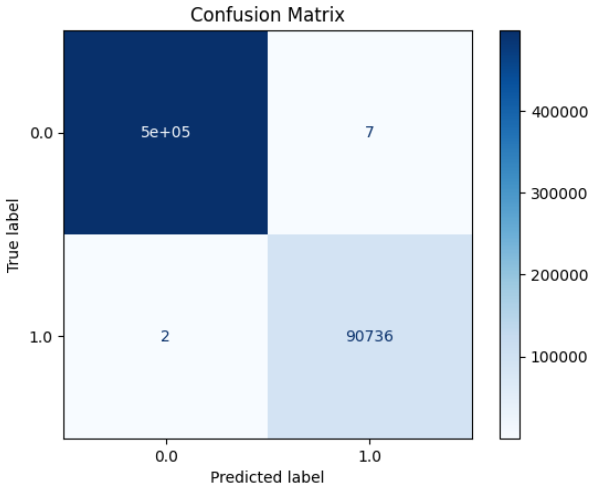

Figure 1: Confusion Matrix on validation set

leads to a drop in performance, highlighting the need to derive additional meaningful features from the dataset by applying effective **feature extraction**.

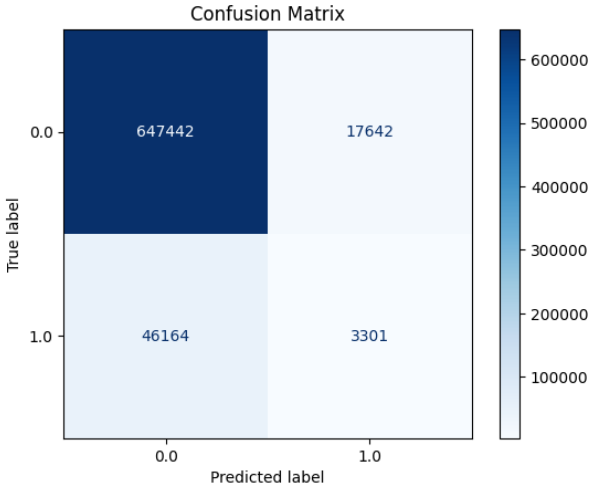

Figure 2: Confusion Matrix on test set

4.3 Working directly on bidder

4.3.1 Feature extraction

The extracted features include several types of information:

- **Basic characteristics**: total number of bids placed (`N_bids`), number of auctions participated in (`N_auctions`), number of product categories in which bids were placed (`N_merchandise`), and number of devices used (`N_device`).
- **Geographical and technical distribution**: number of countries from which the bidder participated in(`N_country`), number of IP addresses used (`N_ip`), number of unique URLs visited (`N_url`), with an entropy analysis of URLs per auction (`url_entropy_per_auction`).
- **Ratio metrics**: average number of bids per URL (`bids/url`), per IP (`bids/ip`), per auction (`bids/auction`), per country (`bids/country`), and per device (`bids/device`).

- **Temporal patterns**: number of bids placed in short concurrent time windows (`count_concurrent_bids`) and statistics on time differences between bids, including mean (`mean_difference`), standard deviation (`std_difference`), minimum and maximum values (`min_difference`, `max_difference`), percentiles (`25_difference`, `median_difference`, `75_difference`), and interquartile range (`iqrDiff`).
- **Auction performance**: number of auctions won (`N_win`) and percentage of wins relative to participated auctions (`percent_Win`).
- **Entropy analysis**: entropy related to devices used (`device_entropy`), IPs (`ip_entropy`), auctions (`auction_entropy`), time intervals (`time_entropy`), countries (`country_entropy`), and URLs (`url_entropy`).
- **Main device**: encoding of the most used device by the bidder (`encoded_most_used_device`).

These features allow for a more detailed modeling of each bidder's behavior, improving the model's ability to distinguish between normal and anomaly. The total number of columns after the extraction is 35.

4.3.2 Comparison of Models

We compared different models for this task, evaluating them using key metrics such as AUC score (as required by Kaggle), recall, and F1-score. Additionally, we visualized their performance using a confusion matrix.
Due to the limited amount of data, we split the dataset into 90% for training (using k-fold cross-validation) and 10% for testing. This approach allows us to maximize the training data, ensuring that the model is trained as effectively as possible.

Knn

We evaluated the performance of the K-Nearest Neighbors (KNN) model using two approaches. First, we trained the model using all available features to obtain a baseline performance. Then, we applied feature selection using `SelectKBest` with the ANOVA F-statistic (`f_classif`) to retain the most relevant features.
To further optimize the model, we performed hyperparameter tuning through a grid search with 5-fold cross-validation. The parameters tuned were the number of selected features (`k`) and the number of neighbors (`n_neighbors`). The best model was selected based on the AUC-ROC score.
After evaluating both approaches on the test set using AUC-ROC, recall, and F1-score, we also visualized the confusion matrix. In both cases, the model exhibited poor performance, suggesting that KNN is not well-suited for this task.

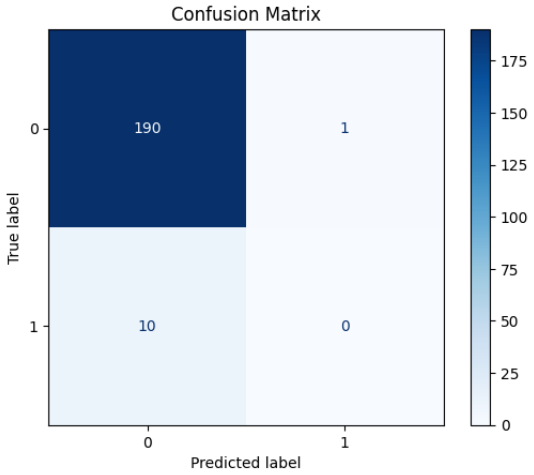

Figure 3: Confusion Matrix of KNN without Feature Selection.

Metric	Score
AUC-ROC Score	0.8623
Recall Score	0.0000
F1 Score	0.0000

Figure 4: Test Performance Metrics.

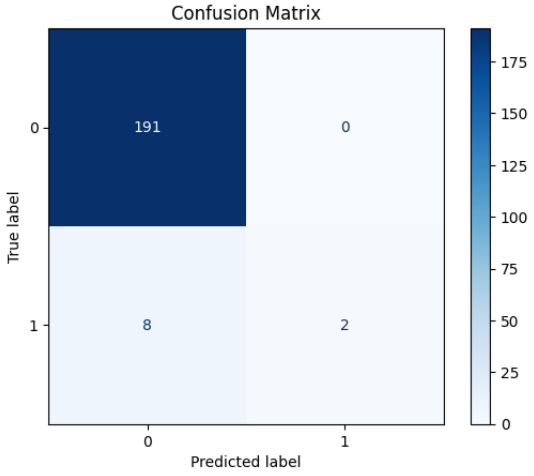

Figure 5: Confusion Matrix of KNN with Feature Selection. The classifier shows slight improvement, but performance is still not good.

Metric	Score
AUC-ROC Score	0.8458
Recall Score	0.2000
F1 Score	0.3333

Figure 6: Test Performance Metrics. Although feature selection was applied, the performance remains low.

Logistic Regression

For the logistic regression model, we employed L1 regularization (*Lasso*), which performs intrinsic feature selection by assigning zero weights to less relevant features. To determine the optimal hyperparameters, we conducted a GridSearchCV over different values of the inverse regularization strength (**C**) while using 5-fold cross-validation and optimizing for the AUC-ROC metric.

Once the best model was identified, we evaluated its performance on the test set by computing the same key metrics as before: AUC-ROC, recall, and F1-score. Additionally, we visualized

the confusion matrix to assess the model's classification performance.

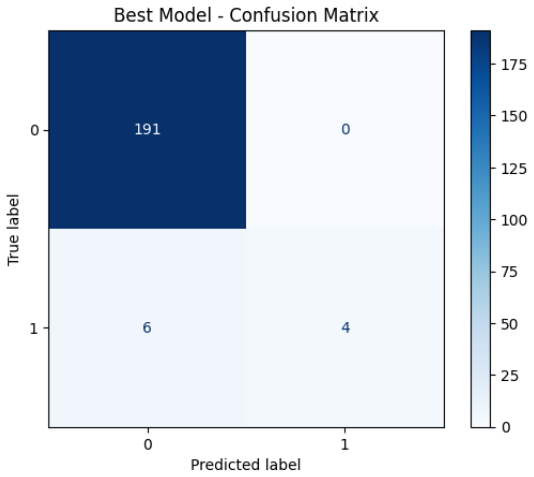

Figure 7: Confusion Matrix with Logistic Regression

Metric	Score
AUC-ROC Score	0.9204
Recall Score	0.4000
F1 Score	0.5714

Figure 8: Test Performance Metrics for Logistic Regression

Random Forest

Initially, the Random Forest was trained without hyperparameter tuning, followed by an optimized version using Grid Search with 5-fold cross-validation. A key advantage of Random Forests is their intrinsic feature selection: each decision tree in the ensemble is trained on a random subset of features, naturally minimizing the impact of irrelevant or redundant ones. This built-in mechanism eliminates the need for explicit feature selection before training.

Here are the results before parameter tuning,

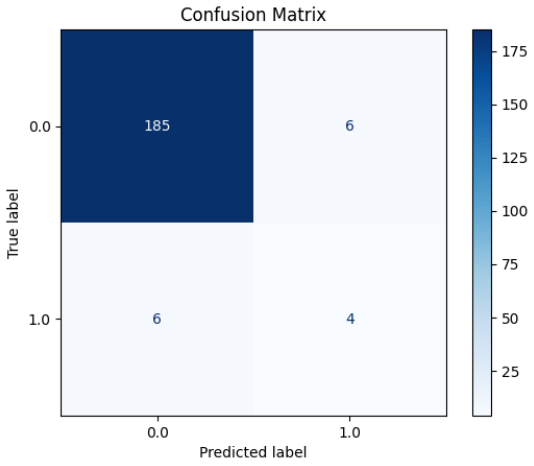

Figure 9: Confusion Matrix of Random Forest without hyperparameter tuning.

Metric	Score
AUC-ROC Score	0.9649
Recall Score	0.4000
F1 Score	0.4000

Figure 10: Test Performance Metrics of the Random Forest model.

and after parameters tuning:

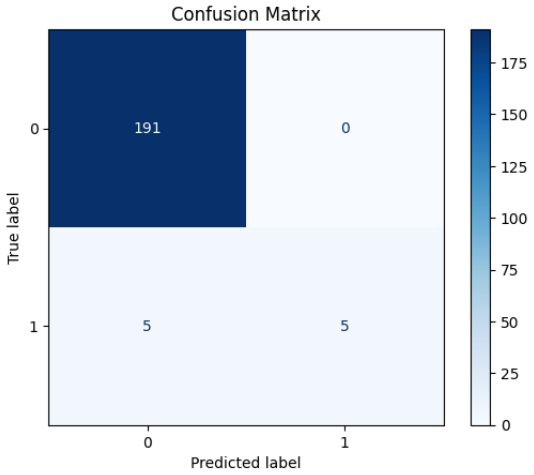

Figure 11: Confusion Matrix of Random Forest with hyperparameter tuning.

Metric	Score
AUC-ROC Score	0.9859
Recall Score	0.5000
F1 Score	0.6667

Figure 12: Test Performance Metrics of Random Forest with parameters tuning.

As expected, after parameters tuning, the Random Forest emerges as the best-performing classifier. To further improve its performance, one potential approach is to apply oversampling, especially considering the unbalanced nature of the dataset.

4.4 Working directly on bids

4.4.1 Pre-analisys

In the first part, we analyze bidders present in both of the dataset resulting in the following table:

Dataset	Number of Bidders
Bidder Dataset	2013
Bids Dataset	6614

Table 4: Number of Bidders in Different Datasets

The table 4 shows that the *bids* dataset contains more bidders than the *bidder* dataset. This discrepancy arises because the *bids* dataset includes bidders used by Kaggle for model evaluation during the competition. These particular bidders are not utilized in our analysis, as we do not have access to their labels because Kaggle retains them for evaluation purposes.

After this pre-analysis process, we start doing the main parts indicated here:

1. Feature Extraction
2. Use PCA for better performance
3. Test different models

4.4.2 Feature Extraction

As we said before, the feature that characterised the dataset are not helpful to train a model able to find a bot bidder, so we start the feature extraction process in order to obtain new worthwhile features. Below are the features extracted regarding the single bid. They are all binary, 0 or 1.

- **Time elapsed between bids:** measures the time interval between consecutive bids placed by the same bidder in the same auction. A bidder who places many temporarily close bids could be a bot.
- **Time until end of auction:** calculates the remaining time before the auction closes for each bid. A bidder who places bids close to the end of the auction could be a bot.
- **Time entropy per auction:** measures the temporal dispersion of a bidder's bids in a specific auction. A higher entropy value indicates that the bids are spread out more evenly over time, while a lower entropy means that the bids are concentrated within a specific time window.
- **Bidding speed:** time difference between bids from the same bidder within an auction.
- **Concurrent speed:** number of bids placed simultaneously in multiple different auctions. This indicates that a bidder could be a bot, because it is able to place bids at the same time in many auctions.
- **Winning bid in an auction:** indicates whether a bidder has placed the winning bid in an auction. Useful to distinguish bot strategies from human ones, as humans often do not win all auctions they participate in.
- **Percentile and quartile bids:** calculates the percentile and quantile positions of a bidder's bids relative to all bids in an auction. Helps understand whether a bidder tends to place bids after the half of an auction.

We exploited some of the features extracted in the previous method, such as ratio metrics and basic characteristics.
Next, we extrapolate some characteristics regarding the bidder's outcome. In order to achieve this, we merge the bidders and bids datasets to obtain the relative outcome for each bid.

The *calculate_feature_bot_probability* function computes the probability that a bid is from a bot given a specific feature. The specific features are *device, url, ip, country, merchandise, auction.*

4.4.3 PCA

Principal Component analysis was apply to the bids dataset to reduce its dimensionality and extract the most important features. We scaled the extracted features using the `StandardScaler` and subsequently applied PCA with three different explained variance: `0.80`, `0.90`, and `0.95`. The resulting datasets were saved separately to evaluate which variance threshold performs best.

PCA retains 8 components for 80% variance, 13 components for 90%, and 16 components for 95%.

4.4.4 Comparison of Models

Dataset splitting

After performing Principal Component Analysis (PCA) on the bidder data to reduce dimensionality and capture the most relevant features, we proceeded to split the dataset into three subsets: *training*, *validation*, and *test* sets. The goal of this split is to ensure that the model can be evaluated on unseen data and prevent overfitting.
Next, to ensure that the model does not learn patterns that are specific to individual bidders (which could lead to overfitting or biased predictions), we merged these splits with the previously transformed PCA datasets. This merging process ensures that each dataset (training,

validation, and test) includes data from different bidders.
This process ultimately enhances the model's ability to generalize well to new, unseen bidders, making the predictions more robust.

Dataset	Proportion of Total
Training set	42%
Validation set	28%
Test set	30%

We compare the results of 2 different models, respectively **Random Forest** and **Logistic Regression** for each of the number of Principal Components calculated before.
To evaluate the model, we first used the training set and the validation set. Once the best-performing model is identified, we assess its final performance on the test set to ensure its generalization to unseen data.

Logistic Regression

Metric	PCA 0.80 score	PCA 0.90 score	PCA 0.95 score
AUC-ROC Score	0.9887	0.9936	0.9937
Recall Score	0.9105	0.9389	0.9376
F1 Score	0.8984	0.9376	0.9389

Table 5: Linear Regression performance with different PCA variance thresholds

Random Forest

Below the results using Random Forest Classifier as model. The hyperparameter *n_estimator* was set to 5.

Metric	PCA 0.80 score	PCA 0.90 score	PCA 0.95 score
AUC-ROC Score	0.9359	0.9003	0.9199
Recall Score	0.7307	0.5677	0.7318
F1 Score	0.7877	0.6005	0.7721

Table 6: Random Forest performance with different PCA variance thresholds

4.4.5 Best model on test data

The previous results lead to select as best classifier the **Logistic Regression** model.
Now, we apply this model to the test data to evaluate its predictive performance and assess its generalization capability. Here is the resulting confusion matrix and metrics:

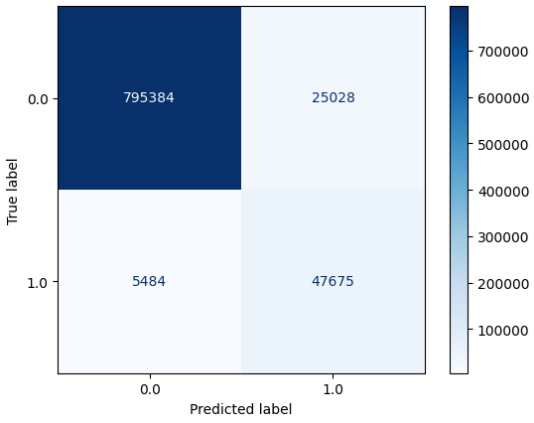

Figure 13: Confusion Matrix on test data with Logistic Regression (PCA 0.95)

Metric	Score
AUC-ROC Score	0.9770
Recall Score	0.8968
F1 Score	0.7576

Figure 14: Test Performance Metrics with Logistic Regression (PCA 0.95)

As observed, the selected model shows strong performance on the test data. At this stage, we analyzed each encoded bidder ID by counting the number of bids classified as either bot or human. A bidder is classified as a bot if the majority of their bids are labeled as such; otherwise, they are considered human. Finally, we compared these results with the labels provided in the bidder dataset to assess the effectiveness of our classification approach, computing *true positive*, *true negative*, *false positive*, *false negative* and, consequently, precision, recall and F1-score:

Metric	Score
Precision Score	0.9375
Recall Score	0.9677
F1-Score	0.9523

Table 7: Performance Metrics on bidder

4.4.6 Overlap checking

To ensure that our results are obtained on completely new data, we verify that there is no overlap between bidders in the training and test sets. This ensures that the selected model is evaluated on unseen data, allowing for a more reliable estimate of its generalization ability.

5 Results

As the final part of our work, we evaluate the models selected from both methodologies and test them on the same test data to determine which one has the best performance.

Metric	Score
Recall Score	0.3000
F1-Score	0.4615

Table 8: Performance Metrics of first methodology

Metric	Score
Recall Score	0.9500
F1-Score	0.9268

Table 9: Performance Metrics of second methodology

6 Conclusions

In conclusion, our results indicate that the methodology that directly analyzes bids, by counting bid classifications for each bidder, combined with Logistic Regression, PCA, and feature extraction, outperforms the approach that relies on the bidder dataset.
However, the approach that centers on the bidders may require oversampling due to the limited amount of data, as highlighted in the state-of-the-art notebooks.

References

[1] Zhenjie Chong. *Human or Robot? Analyzing Bidding Behaviors in Online Auctions*. Kaggle Notebook. 2021. URL: https://www.kaggle.com/code/chongzhenjie/humanrobot.

[2] Small Yellow Duck. *fba-sub9.py*. Accessed: 2025-02-18. 2020. URL: https://github.com/small-yellow-duck/facebook_auction/blob/master/fba-sub9.py.

[3] Kaggle. *Facebook Recruiting IV: Human or Bot*. Accessed: 2025-02-19. 2015. URL: https://www.kaggle.com/competitions/facebook-recruiting-iv-human-or-bot.