Business process model with “RandomForest” algorithm

Anatoly Kukartsev[1,2,*], *Alexandra* Fedorova[1], *Yuriy* Danilchenko[1], *Konstantin* Lobkov[2,3] and *Evgeniia* Korepanova[2]

[1]Siberian Federal University, 79, Svobodny Av., Krasnoyarsk, 660041, Russian Federation
[2]Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., 660037 Krasnoyarsk, Russian Federation
[3]Krasnoyarsk State Agrarian University, 2 Kirenskogo, Krasnoyarsk, 660074, Russian Federation

Abstract. Currently, the areas of application of machine learning are multifaceted: artificial intelligence, financial applications, bioinformatics, intellectual games, speech and text recognition, computer language processing, medical diagnostics, technical diagnostics, text search and rubrication. Machine learning is an area of scientific knowledge associated with learning-capable algorithms. The use of machine learning methods is explained by the fact that for most intelligent complex tasks (for example, speech recognition, etc.) it is almost impossible to develop an obvious algorithm for their solution. However, you can teach a computer to learn how to solve such problems. In our article, we propose a model based on the machine learning algorithm "RandomForest", which allows one to recognize bots by HTTP sessions. The chosen algorithm provides many advantages: non-iterative learning; high quality of the resulting models (comparable to neural networks and ensembles of neural networks); a small number of adjustable parameters. It works well with missing data (retains good accuracy); internal assessment of the generalizability of the model; able to work with raw data without preprocessing. The algorithm was trained on a dataset of more than 5000 sessions. The prospects of this direction are obvious, since robotic traffic accounts for more than 40% of the total Internet traffic.

1 Introduction

The new digital world assumes a gradual transition of business to the online space. The 2020 quarantine measures accelerated this process. The concept of “transferring activity to the online” appeared, thus the business underwent a digital transformation [1-3]. It helped companies that had succeeded in the quick restructuring to stay afloat, to take the audience of less prepared competitors. But has the activity of people only increased in the online space? After reviewing data from traffic data, Imperva'sThreatResearchLab [1] produced a report for 2020 that shows an increase in network activity of automated means of interacting with content on websites. In this case, the share of human activity stands out - 62.8%; good bots - 13.1% and bad bots - 24.1% [4-6]. It should be noted that bad bot traffic has grown significantly and, at the moment, it is almost a quarter of all Internet traffic [7-9]. The damage is done to both businesses and individuals. In this regard, it is imperative to develop processes for filtering unwanted activity on websites [10, 11].

A bot is software designed to simulate the behavior of a real user on online social networks [2]. Many methods have been proposed for detecting and removing them. Research is carried out on the characteristics, detection and assessment of the impact of bots.

There are various techniques for detecting web bots in network traffic: limiting the frequency of requests to a host; checking blacklists of IP addresses; parsing the value of the User-Agent HTTP header; taking prints of the device; CAPTCHA implementation; behavioral analysis of network activity using machine learning [3].

Bot traffic analysis tools have become a must for any website.

2 Materials and Methods

The process of detecting bots by HTTP session. Session is a sequence of requests from one node (unique value of the IP address and the User-Agent field in the HTTP request) in a fixed time interval [3, 12]. The session will be evaluated offline using the data uploaded to the file, which will be divided into testset and trainingset, see Figure 1. This model uses key concepts:
-A random sampling of samples from a dataset when constructing trees.
-When dividing nodes, random sets of parameters are selected.

“RandomForest” optimization can be done through random search using the Randomized Search CV in Scikit-Learn. Optimization involves finding the best hyper parameters for the model on the current dataset (Figure 2) [14].

[*] Corresponding author: kempf@sibsau.ru

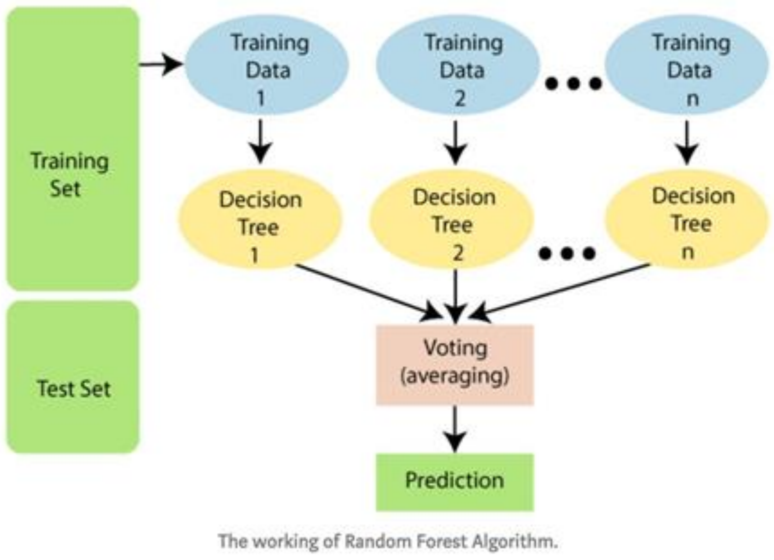

Fig. 1. Schematic diagram of the operation of the “RandomForest” algorithm.

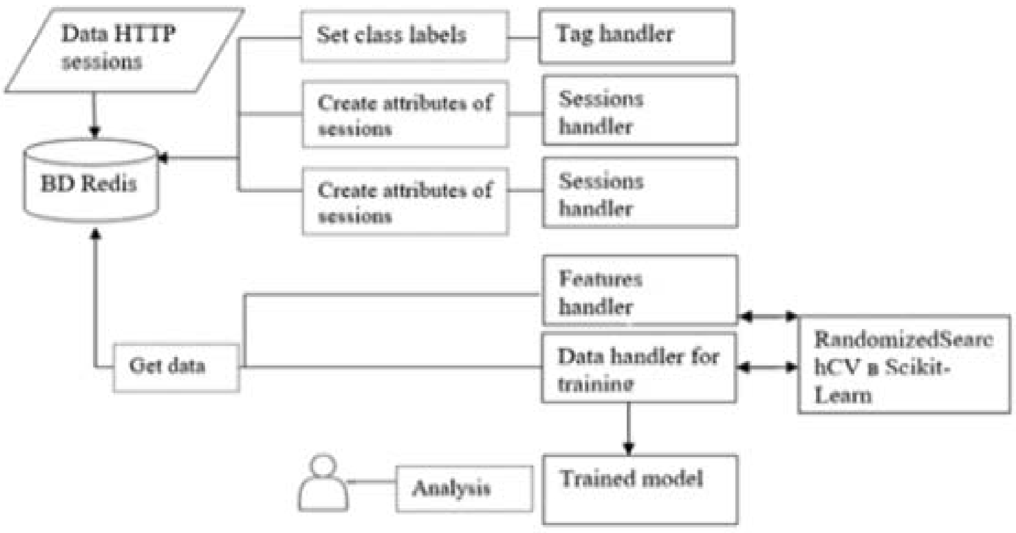

Fig. 2. Model of the process of working with data from HTTP sessions.

There is a significant difference from other models ; in this case, feature weighting is not used ; therefore any monotonous transformations will not affect the result (up to a numerical error) [4].

The algorithm for constructing a RandomForest consisting of N trees is as follows: For each n = 1, ..., N: Generate Xn fetch with Bootstrap.

Build the decision tree bn from the sample Xn:

According to a given criterion, we choose the best feature, make a partition in the tree according to it, and so on until the sample is exhausted.

The tree is built until each leaf contains no more than nmin objects or until we reach a certain tree height.

At each partition, firstly, m random features are selected from the initial ones and the optimal division of the sample is sought only among them.

3 Results and Discussion

The final classifier is a (x) = 1N∑i = 1Nbi (x), in simple words - for the classification problem we choose the solution by voting by the majority, and in the regression problem - by the average [9].

It is recommended to take m = n in classification problems, and m = n3 in regression problems, where n is the number of features. It is also recommended in classification problems to build each tree until each sheet contains one object, and in regression problems - until each sheet contains five objects.

Thus, a RandomForest is Boots trapaggregating over decision trees, during training of which, for each partition, features are selected from some random subset of features [5].

Let's create a simulation model for calculation with the main metrics of the unit-economy (Figure 3) [13-15].

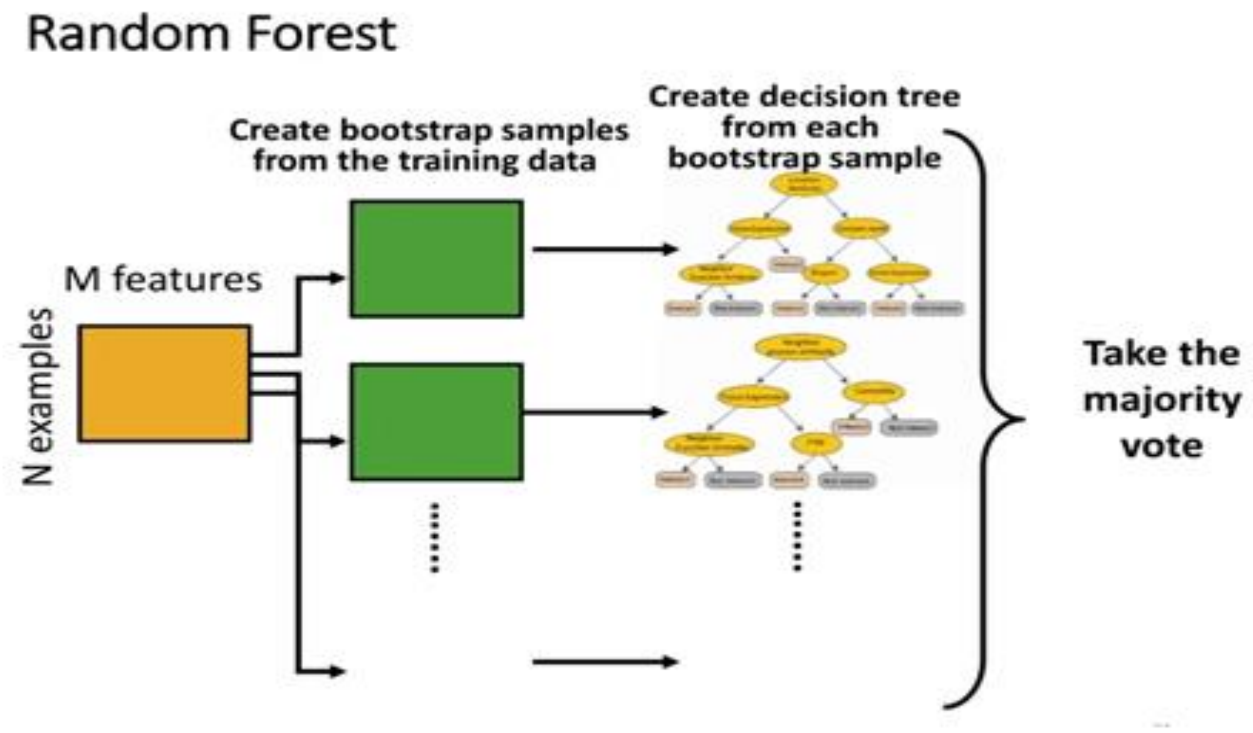
Random Forest
Create bootstrap samples from the training data
Create decision tree from each bootstrap sample
M features
N examples
Take the majority vote

Fig. 3. Schematic diagram of the "RandomForest" algorithm.

We use session characteristics as parameters (Table 1).

Table 1. Session characteristics

n	Number of requests per session
n_pages	Number of requests per session in pages (URI ends with .htm, .html, .php, .asp, .aspx, .jsp)
n_static_request	Number of requests per session in static pages
n_sec	Session time in seconds
n_unique_uri	Number of requests in a session with a unique URI
headers_cnt	Number of headers per session
has_cookie	There is a cookie header
has_referer	There is a Referer header
(avg)mean_time_page	Average time per page per session
(avg)mean_time_request	Average time per request per session
(avg)mean_headers	Average number of headers per session

We will take 0 BOT and 1 PERSON as labels; that is, we will build a clear binary classification model: network activity is created by a web bot (bot label); network activity is created by a human (human label) [10].

In this case, we use quality metrics: accuracy, completeness and F-measure (Table 2, Figure 4).

The number of sessions is over 5000.

Data are divided into two subgroups: training and test. From the training data, we get the model.

To create and train the algorithm, we use:

fromsklearn.ensembleimportRandomForestClassifier
# Create a forest model from hundreds of trees
model = RandomForestClassifier(n_estimators=100,
bootstrap = True,
max_features = 'sqrt')
# Train on training data
model.fit(train, train_labels)

After a few minutes of training, the model is ready to make predictions for the test data:

# Current classification
rf_predictions = model.predict(test)
# Probabilities for each class
rf_probs = model.predict_proba(test)[:, 1]

Table 2. Training result

	Average accuracy	Average fullness	Average F-measure
bot	0.84	0.8	0.86
human	0.96	0.95	0.95

We will calculate the classification predictions (predict) along with the predictive probability (predict_proba).

from sklearn.metrics import roc_auc_score

# Calculate rocauc

roc_value = roc_auc_score(test_labels, rf_probs).

To test the effectiveness of the model, we consider the results..

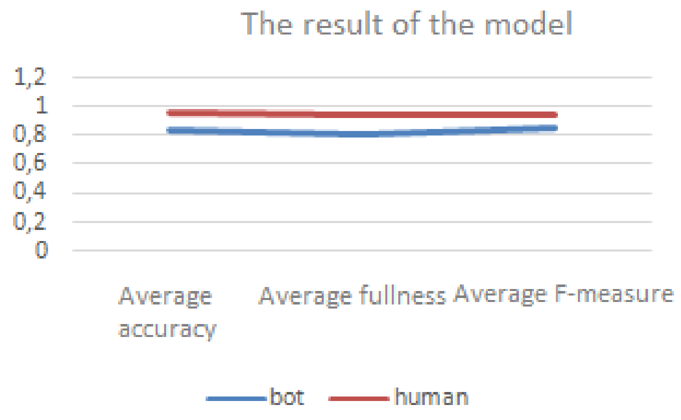

Fig. 4. Operation of the model with lazy fetch.

On lazy sampling, the values of the quality metrics are almost identical to the values of these metrics when the model requirements are met. It follows from this that the model based on these data can generalize the knowledge gained during training [7, 15].

Let's consider errors of the 1st kind. If these data are marked out expertly, then the error matrix will change significantly. This means that some errors were made when marking up the data for the model, but the model was still able to recognize such sessions correctly (Table 3, Figure 5) [15].

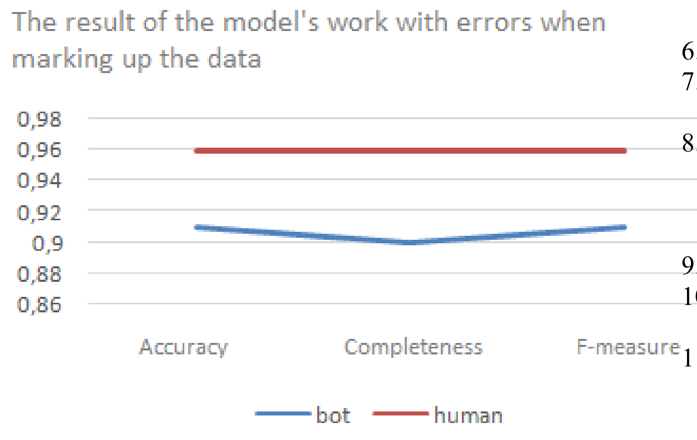

Fig. 5. Operation of the model with lazy fetch.

Table 3. Training result.

	Accuracy	Completeness	F-measure	Number
bot	0.91	0.9	0.91	1223
human	0.96	0.96	0.96	4220

5 Conclusions

As a result, an approach to the detection of web bots using a machine learning algorithm and the use of statistical features is described. The "RandomForest" machine learning algorithm is selected as the most appropriate. Implementation of the similarity of the model for recognizing bots by session in Python using the scikit-learn machine learning library [6]. A classical measure is used to assess the accuracy. 11 features were used to train the algorithm. The trained algorithm was tested on a random data set and the F-metric value was 0.98 [8].

References

1. E. Roberts, Imperva blog (2020)
2. A. Cabri, *IEEE 20th Int. Conf. on HPC. and Com.* (2018)
3. Z. Chu, S. Gianvecchio, H. Wang, *LNCS.* **6**, 11170 (2018)
4. A.A. Boyko, V.V. Kukartsev, E.S. Smolina, V.S. Tynchenko, Y.I. Shamlitskiy, N.V. Fedorova, *Jour. of Phys.: Conf. Ser.* **1353**, 012124 (2019)
5. P. De Meo, E. Ferrara, G. Fiumara, A. Provetti, *ACM.* **11**, 78 (2014)
6. N.D. Lyfenko, *Cyber security issues* **5**, 17 (2014)
7. F. Pedregosa, G. Varoquaux, A. Gramfort, *The J. of Machine Learning Research* **12**, 2825 (2011)
8. A.V. Kukartsev, A.A. Boyko, V.V. Kukartsev, V.S. Tynchenko, V.V. Bukhtoyarov, S.V. Tynchenko, *IOP Conf. Ser.: Mat. Sci. and Eng.* **537**, 042009 (2019)
9. L. Demidova, M. Ivkina, *SUMMA.* **359** (2020)
10. Y. Su, K. Weng, C. Lin, Z. Zheng, *IEEE Access* **9**, 9142 (2021)
11. A.A. Boyko, V.V. Kukartsev, D.V. Eremeev, V.S. Tynchenko, V.V. Bukhtoyarov, A.A. Stupina, *Jour. of Phys.: Conf. Ser.* **1353**, 012123 (2019)
12. I.R.H.T. Tangkawarow, J. Waworuntu, *IOP Conf. Ser.: Mat. Sci. and Eng.* **128**, 012010 (2016)
13. T. Zhu, *Jour. of Phys.: Conf. Ser.* **1607**, 012123 (2020)
14. A.A. Boyko, V.V. Kukartsev, V.S. Tynchenko, D.V. Eremeev, A.V. Kukartsev, S.V. Tynchenko, *Jour. of Phys.: Conf. Ser.* **1353**, 012138 (2019)
15. V. Khareva, O. Voronova, T. Khnykina, *IOP Conf. Ser.: Mat. Sci. and Eng.* **940**, 012057 (2020)