Comparative Analysis of Decision Tree-Based Algorithms for Detecting Anomalies in Web Traffic

Student paper

Veljko Lončarević*, Savo Šućurović**
Second cycle students
University of Kragujevac, Faculty of Technical Sciences
Čačak, Serbia
Email: *veljkoloncarevicharry@gmail.com, **savos977@gmail.com

Abstract— **This research delves into the performance assessment of decision tree-based machine learning algorithms—specifically Decision Tree, Random Forest, XGBoost, LightGBM, and CatBoost—in the context of anomaly detection in web traffic. The study employs key metrics like accuracy, precision, recall, and F1 score, juxtaposed with the computational efficiency measured by model training times. Notably, LightGBM demonstrates very efficient training with competitive performance, while CatBoost, despite longer training, outshines others in predictive accuracy.**

Keywords- machine learning, decision tree, anomaly detection, web traffic, comparative analysis, model training time

I. INTRODUCTION

Anomalous behavior refers to events that significantly deviate from the expected or normal, withing a given context. Such behavior often indicates unusual, potentially harmful or noteworthy occurrences. Identifying anomalies in web traffic helps in detection of security threats, such as hacking attempts, unauthorized access, fraudulent transactions, issues related to the performance and health of the web application, potential service disruptions or distributed denial-of-service (DDoS) attacks. Monitoring for anomalies is crucial for preventing financial losses, ensuring the availability of web applications, and mitigating potential risks, as well as protecting the web application and its users. It is also often necessary for compliance with privacy and security regulations. When it comes to anomaly detection in general, finance, healthcare and e-commerce are an example of sectors where monitoring and protecting sensitive data is particularly important [1, 2].

There is a large variety of detection methods being applied in different industries, some of which are statistical methods, rule-based systems, classification and clustering machine learning algorithms, and neural networks. Machine learning algorithms enable automated identifying of patterns in complex datasets. They can be adapted and evolve over time based on new data, which can be of great benefit in dynamic environments where new patterns may emerge. They also allow analysis of complex, multidimensional relationships in data, where even subtle differences or anomalies can be captured, whereas the same might be more difficult for traditional systems based on rules. The objective of this paper is to compare the efficacy and accuracy of various decision tree-based machine learning classification algorithms, including decision trees, random forests, and gradient boosting. Machine learning algorithms will be trained on a dataset gathered from web server log files and user behavior data. The primary metrics to be assessed in this study encompass the training time required for machine learning models, as well as key classification accuracy metrics, including accuracy, precision, recall, and F1 score, which are to be compared at the end.

II. THEORETICAL FOUNDATIONS

A. Decision Trees

Decision tree is a machine learning algorithm used for both classification and regression tasks. The general idea behind the algorithm is to recursively split the dataset based on the features, making decisions at each node to arrive at a final outcome. In the beginning, the algorithm considers the entire dataset, and selects the feature that best separates the data into distinct group. This feature becomes the root node of the tree. The dataset is split into subsets based on the chosen feature, and the process is repeated for each subset. At each step, the algorithm selects the feature that best separates the data in the current subset. The decision at each node is based on a criterion, usually by calculating the Gini impurity for classification tasks [3]. The process continues until a stopping criterion is met, such as a predefined depth of the tree or a minimum number of samples in a node. The final nodes are called leaf nodes, and they represent the prediction. For classification tasks, the majority class in a leaf node is assigned as the predicted class. Gini impurity quantifies the likelihood of misclassifying an element in a set chosen randomly. It is calculated in the following way:

$$f(p) = 1 - \sum_{i=1}^{C} p_i^2 \tag{1}$$

where C is the number of classes and pi is the probability of an element of class i in the node being chosen randomly. Gini impurity ranges between 0 and 1, where 0 represents perfect purity (all elements in the node belong to the same class), and 1 represents maximum impurity (an equal distribution of elements across all classes). One common issue decision trees face is overfitting, where a model captures noise or random fluctuations that do not generalize well to new, unseen data. Decision trees can become overly complex, creating nodes that are tailored to the training data's peculiarities, instead of capturing the underlying patterns. Such models tend to perform poorly on new data. A technique often used to mitigate overfitting in decision trees is pruning, which involves removing parts of the tree that provide no significant predictive power. Pruning includes setting a stopping criterion before the tree is fully grown, such as maximum depth for the tree, minimum number of samples required to split a node or a minimum number of samples in a leaf node. If a node does not meet these criteria, it is not split further. Due to the way they work, decision trees are highly interpretable, they can model non-linear relationships in the data, automatically select most informative features at each node, and are robust to outliers by isolating them in separate nodes. Decision trees can handle both numerical and categorical data, without the need for feature scaling. Decision trees automatically handle missing values by evaluating the available data and choosing the best split based on the information present, without the need for imputing missing values. They can also create separate branches for missing data. Decision trees can be part of ensemble methods like Random forests and Gradient boosting, which can improve predictive performance and mitigate overfitting.

B. Random Forests

Random forest is an ensemble method that builds multiple decision trees during training and outputs the mode of the classes in classification tasks. It introduces randomness in the tree-building process, which makes it more robust and less prone to overfitting compared to individual decision trees. It builds multiple trees using a technique called bootstrapped sampling, where multiple subsets of the original dataset are created by sampling with replacement, after which each tree is trained on one of these subsets. Random forest introduces randomness by considering only a random subset of features for splitting, which helps with tree decorrelation and makes sure that each tree focuses on different aspects of the dataset. For classification tasks, the final prediction is determined by a majority vote from the individual trees. The ensemble nature of the method helps generalize well to new data. Building each tree in a Random forest is done independently, which is suitable for parallelization, helping with algorithm scaling and increasing efficiency, particularly with large datasets.

C. Gradient Boosted Trees

Gradient Boosted Trees (GBT) is an ensemble method that combines predictions of multiple decision trees to create a strong predictive model [4]. Unlike Random forests, which build multiple trees independently, GBT builds trees sequentially and corrects errors made by the preceding ones, in an iterative process that minimizes a loss function by adding trees to the ensemble. It starts with a simple model, often a single decision tree referred to as a "weak learner". This initial weak learner is fitted to the data. The next weak learner is added to correct the errors made by the first one. The algorithm pays more attention to the instances that were misclassified or had higher residuals in the previous iteration. This process is repeated for a specified number of iterations or until a stopping criterion is met. Each new weak learner is trained to minimize the overall loss of the combined model. The final prediction is a weighted sum of the predictions from all weak learners. The weights are determined during the training process. GBT often achieves higher predictive accuracy compared to individual decision trees and, in some cases, even Random forests.

D. Performance metrics

Accuracy score is used to evaluate the performance of a classification model. It represents the ratio of correctly predicted instances to the total instances in the dataset. It is calculated using the formula

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}. \tag{2}$$

Accuracy score is expressed as a percentage ranging from 0% to 100%, where a higher percentage indicates better performance. Although accuracy score can be useful, it still has some limitations, especially in situations where classes are imbalanced (where one class significantly outnumber the others). In such a case, a model can achieve high accuracy score by simply predicting the majority class, even if it performs poorly on the minority classes. Precision, denoted as the ratio of true positive predictions (correctly predicted positive values) to the total predicted positives (correctly and incorrectly predicted positive values), is especially valuable in scenarios where the cost of false positives is high. This measure, represented by the formula

$$Precision = \frac{TP}{TP+FP}, \tag{3}$$

quantifies the accuracy of positive predictions. Here, TP stands for the number of true positive predictions, and FP represents the number of false positive predictions. Recall, on the other hand, gauges the model's proficiency in capturing all positive instances, and it becomes particularly pertinent when the cost of false negatives is a significant concern. It is calculated using the formula

$$Recall = \frac{TP}{TP+FN}, \tag{4}$$

where TP denotes the number of true positive predictions, and FN signifies the number of false negative predictions. The F1 score, serving as the harmonic mean of precision and recall, offers a balanced assessment of the model's performance, especially in situations characterized by an uneven class distribution [5]. Expressed by the formula

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{5}$$

it provides a single metric that considers both precision and recall. Implementations of Gradient boosting algorithms include XGBoost, LightGBM and CatBoost. Extreme Gradient Boosting (XGBoost) integrates regularization techniques to prevent overfitting and parallel processing to speed up model training. It supports various objective functions and custom loss functions, making it highly versatile and applicable across a wide range of tasks [6]. LightGBM, which was developed by Microsoft, is renowned for its efficiency, utilizing a histogram-based learning approach to construct decision trees. It uses a technique called "Gradient-Based One-Side Sampling", which selects the optimal split points more efficiently, which makes LightGBM particularly suitable for large datasets and high-dimensional feature spaces [7]. CatBoost, developed by Yandex, is another gradient boosting algorithm designed to handle categorical features seamlessly. It employs a unique strategy known as "ordered boosting", which reduces overfitting by sorting categorical variables and incorporating the information from the sorted order [8].

III. Methodology

The main goals of this research paper are as follows:

- Evaluate and compare the model training times of different decision tree-based machine learning algorithms.
- Assess and compare the accuracy performance of the selected algorithms on a specific dataset.

In the course of this research, we gathered data from author's private website (www.veljkoloncarevic.in.rs), spanning the period from September 2022 to August 2023. The dataset comprises 18 columns, of which the target column is "Anomalous". The complete list of columns (features) is given in Table I. With approximately 15,000 rows of data, our dataset provides a substantial and representative sample of the web traffic recorded during the specified time frame.

TABLE I. List of collected features

#	Feature	#	Feature
1.	IP address	10.	Total Requests
2.	Timestamp	11.	Location
3.	Browser	12.	City
4.	Operating System (OS)	13.	Country
5.	Requests Per Second (RPS)	14.	Longitude
6.	HTTP Status Code	15.	Latitude
7.	Session Duration	16.	Device Type
8.	Payload Size	17.	Language
9.	URL Length	18.	Anomalous

An example of the first instance from the dataset is given in Table II, without the potentially identifying information (like IP address, location, longitude and latitude).

The dataset is not publicly available due to privacy considerations and restrictions, in order to preserve the confidentiality of the individuals or entities associated with this data. However, authors of this paper are willing to address any reasonable requests for information that do not compromise the privacy and confidentiality of the individuals or entities involved. For all inquiries, contact Veljko Lončarević at veljkoloncarevicharry@gmail.com.

Features were preprocessed by using a variety of preprocessing techniques. Missing values were replaced with the median value for numerical columns, and the most frequent occurrence for categorical values. Numerical features were scaled using the MinMax scaler technique. Categorical features with less than or equal to five unique values were encoded using One-Hot encoding method, while others were encoded using Target encoding. Target column (Anomalous) was encoded using binary encoding. The following machine learning algorithms were trained on the dataset (using default parameters):

- Decision Tree,
- Random Forest,
- XGBoost,
- LightGBM,
- CatBoost.

TABLE II. First instance trom the dataset (Anonymized)

Feature	Value	Feature	Value
Timestamp	2023-05-12 12:20	URL Length	61
Browser	Chrome 113.0.0	City	Belgrade
OS	Windows 10	Country	Serbia
Requests Per Second	1.27	Device Type	PC
HTTP Status Code	200	Language	EN
Session Duration	32	Anomalous	False
Payload Size	533	Total Requests	5

Prior to training the models, dataset was separated into different train and test splits using the ShuffleSplit technique, after which the models were trained on the train splits, and tested with the test splits, using the cross-validation technique. Test results of each of the performance metrics (Accuracy, precision, recall, F1 score) and time to train each model were recorded and compared. For this analysis, Python programming language was used, with Pandas and NumPy libraries for data handling, matplotlib for result visualization, scikit-learn, XGBoost, LightGBM and CatBoost libraries for their implementations of the necessary decision tree-based algorithms.

IV. Results and Discussion

A. *Results*

Fig. 1 shows the time it took to train a specific model on the dataset using methods described in the previous chapter. The test results for the time required to train various machine learning models reveal significant differences in computational efficiency. Notably, the Decision Tree exhibited the shortest training time at 1.52 seconds, indicating its rapid model building process. LightGBM followed with a still efficient 4.77 seconds, emphasizing its scalability and speed, especially suitable for large datasets. The Random Forest, while providing robust ensemble learning, took 16.20 seconds, signifying a longer training duration compared to individual decision trees. XGBoost demonstrated a longer training time of 31.95 seconds, highlighting the computational resources required for its sophisticated gradient boosting algorithm. Notably, CatBoost exhibited the longest training time among the tested models at 164.08 seconds, underscoring its thorough optimization process and potentially resource-intensive nature.

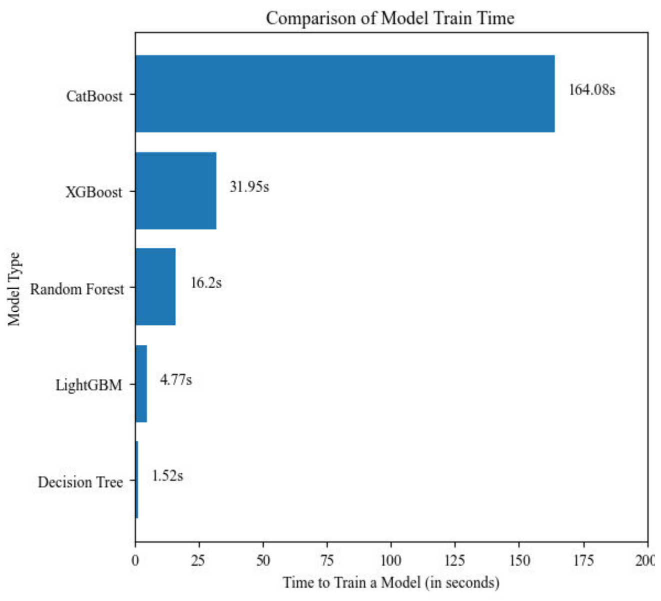

Figure 1. Comparison of Model Train Time

Table III shows the comparison between performance metrics of different machine learning algorithms used in this analysis.

TABLE III. PERFORMANCE METRICS RESULTS

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Decision Tree	82%	78%	85%	81%
Random Forest	89%	93%	86%	89%
XGBoost	96%	94%	94%	94%
LightGBM	95%	96%	94%	95%
CatBoost	97%	98%	98%	98%

The test results of various machine learning algorithms demonstrate distinct performance metrics across accuracy, precision, recall, and F1 score. Notably, CatBoost achieved the highest accuracy at 97%, closely followed by XGBoost at 96%, indicating their effectiveness in overall prediction accuracy. In terms of precision, CatBoost also excelled with a value of 98%, suggesting a small false positive rate. XGBoost, Random Forest, and LightGBM exhibited competitive precision values, emphasizing their ability to make accurate positive predictions. When assessing recall, CatBoost also led with a value of 98%, signifying its capacity to capture a high proportion of actual positive instances. Random Forest demonstrated slightly lower recall value, while XGBoost and LightGBM remained strong in this metric. F1 score, which balances precision and recall, showcased the overall model performance, with CatBoost leading at 98%, closely trailed by XGBoost and LightGBM at 94% and 95%, respectively.

B. *Discussion*

The comparison between the performance metrics test results of machine learning algorithms and their corresponding training times sheds light on the trade-offs between model performance and computational efficiency. Notably, the Decision Tree, despite having the shortest training time at 1.52 seconds, demonstrated a lower performance across accuracy, precision, recall, and F1 score, suggesting its inability to capture more complex relationships in the dataset. On the other end of the spectrum, CatBoost, with the longest training time of 164.08 seconds, exhibited superior predictive performance across all metrics, including the highest accuracy, precision, recall, and F1 score. This indicates that the computational resources invested in training CatBoost were justified by its outstanding predictive capabilities. The time to train models results align with expectations, where the more complex algorithms, such as XGBoost and CatBoost, requiring more substantial training times, demonstrated strong predictive performance. XGBoost, with a training time of 31.95 seconds, achieved competitive results across all metrics, striking a balance between training efficiency and model accuracy. Random Forest, while showing commendable performance, necessitated a more moderate training time compared to CatBoost, emphasizing the versatility of ensemble methods. LightGBM, with a training time of 4.77 seconds, offered an appealing compromise between computational efficiency and model effectiveness, achieving respectable scores across accuracy, precision, recall, and F1 score. The training time for LightGBM is notably shorter than that of CatBoost, demonstrating its efficiency in handling large datasets while maintaining strong predictive capabilities. In instances where paramount emphasis is placed on computational efficiency, the outcomes of this experiment advocate for the adoption of LightGBM. Conversely, when the primary imperative is superior predictive accuracy, the findings strongly advocate the utilization of CatBoost.

C. *Comparison with Related Research*

While existing research papers, such as [9] and [10], have presented compelling results in training machine learning models for anomaly detection in web traffic, this study contributes to the growing body of knowledge by corroborating and extending these findings. Our experiments align closely with the outcomes reported by [9], which demonstrated the efficacy of an XGBoost ensemble of deep neural networks in achieving a maximum accuracy of 99.5%, coupled with 98% precision and 97% recall. This not only reaffirms the robustness of such ensemble techniques but also establishes a consistent performance baseline across multiple studies. Moreover, our results bear resemblance to those reported in [10], where the authors introduced a novel SVM-C method and achieved accuracy levels ranging from 94% to 97% on diverse datasets. Our contributions lie in the replication and validation of these results in the context of our experimental setup, thereby reinforcing the generalizability and reliability of these anomaly detection methodologies. Looking ahead, there are promising avenues for future work in this domain. One potential direction involves exploring the adaptability and scalability of the identified anomaly detection methods to diverse network architectures and traffic patterns. Additionally, investigating the impact of evolving cyber threats and attacks on the performance of these models could contribute to enhancing their robustness. Furthermore, the integration of advanced deep learning techniques or the exploration of ensemble methods combining various algorithms may unveil novel approaches for even more accurate and resilient anomaly detection systems. These future endeavors aim to refine and extend the current understanding of anomaly detection in web traffic, addressing emerging challenges and optimizing model performance across different real-world scenarios.

V. Conclusion

The juxtaposition of performance metrics and training times for diverse machine learning algorithms provides valuable insights into the intricate trade-offs between model efficiency and predictive accuracy. The Decision Tree, despite its swift training time of 1.52 seconds, exhibited inferior performance metrics, suggesting limitations in capturing intricate dataset relationships. Conversely, CatBoost, with the longest training time of 164.08 seconds, showcased unparalleled predictive capabilities, justifying the computational investment. As anticipated, more complex algorithms like XGBoost and CatBoost, demanding extended training times, demonstrated robust predictive performance. XGBoost, with a balanced 31.95 second training time, emerged as an efficient compromise between speed and accuracy. Random Forest displayed commendable performance with a moderate training time, emphasizing the versatility of ensemble methods. LightGBM, with a brief 4.77 second training time, struck an appealing balance between computational efficiency and model effectiveness. In scenarios prioritizing computational efficiency, LightGBM emerges as a favored choice, while CatBoost stands out for superior predictive accuracy. This research contributes to the broader field by reaffirming and extending findings from previous studies ([9], [10]). The results align with [9], validating the efficacy of an XGBoost ensemble of deep neural networks. Our findings also resonate with [10], where the SVM-C method achieved accuracy levels of 94% to 97%. Our contribution lies in replicating and validating these results within our experimental context, reinforcing the reliability and generalizability of anomaly detection methodologies.

Looking ahead, future research could explore the adaptability of anomaly detection methods across diverse network architectures and evolving cyber threats. Investigating the integration of advanced deep learning techniques and ensemble methods may uncover novel approaches, enhancing accuracy and resilience in anomaly detection systems. These endeavors aim to refine the understanding of anomaly detection in web traffic, addressing emerging challenges and optimizing model performance across varied real-world scenarios.

References

[1] A. Ukil, S. Bandyoapdhyay, C. Puri and A. Pal, "IoT Healthcare Analytics: The Importance of Anomaly Detection," *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Crans-Montana, Switzerland, 2016, pp. 994-997, doi: 10.1109/AINA.2016.158.

[2] W. Hilal, S. A. Gadsden and J. Yawney, "Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances," in *Expert Systems with Applications*, vol. 193, 2022, 116429, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.116429.

[3] R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[4] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189-1232, Oct. 2001. DOI: 10.1214/aos/1013203451.

[5] C. Goutte and E. Gaussier, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation," in *Proceedings of the 27th European Conference on Advances in Information Retrieval Research*, Apr. 2005, Lecture Notes in Computer Science, vol. 3408, pp. 345-359, DOI: 10.1007/978-3-540-31865-1_25.

[6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Aug. 2016. DOI: 10.1145/2939672.2939785

[7] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[8] S. I. Nikolenko et al., "CatBoost: unbiased boosting with categorical features," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS),* 2018.

[9] S. T. Ikram et al., "Anomaly Detection Using XGBoost Ensemble of Deep Neural Network Models," *Cybernetics and Information Technologies*, vol. 21, no. 3, Sofia, 2021, pp. 1-15. DOI: 10.2478/cait-2021-0037.

[10] M. Karuppiah, Q. Ma, C. Sun, and B. Cui, "A Novel Model for Anomaly Detection in Network Traffic Based on Support Vector Machine and Clustering," *Security and Communication Networks*, vol. 2021, pp. 2170788, Nov. 20, 2021. DOI: 10.1155/2021/2170788.