JES
Journal of Engineering Sciences

Vol 13, Issue 05, May /2022

ISSN NO:0377-9254

# MALWARE PREDICTION USING XGBOOST AND CATBOOST

C.PRABHAVATHI[1], ALLAMPATI MAHESH[2], SOMA VENKATA AJAY[3], RAMIREDDY MYTHILI[4]

[1] *Assistant Professor (Adhoc) Dept. of Computer Science and Engineering,JNTUA College of Engineering,Pulivendula .*

[2,3,4] *Students, Dept. of Computer Science and Engineering, JNTUA College of Engineering,Pulivendula.*

[1]prabhavathi1231@gmail.com,[2]allampatimahesh3@gmail.com, [3]ajay482000@gmail.com,

[4]ramireddymythili@gmail.com

## Abstract

Malware is actually a generic definition for all kinds of computer threats. Malware is divided into two categories: file infectors and individual malware. Worms, backdoors, Trojans, rootkits, spyware, and adware are examples of malware that are classified depending on their distinct actions. Malware is actually a generic definition for all kinds of computer threats. A simple classification of malware consists of file infectors and individual malware. We are building a model to detect malicious files.

Machine learning has revolutionized many fields, including cybersecurity. Traditional approaches to machine learning can be divided into two main groups according to the type of analysis: static approaches and dynamic approaches. The main difference between the two is that the static approach extracts feature from static malware analysis whereas the dynamic approach extracts feature from dynamic analysis.

We propose a general framework that uses various machine learning algorithms to successfully distinguish between clean and malicious files with the goal of minimizing false positives. Reduce data training time and find maximum accuracy between models.

**Objectives:**

1. The main objective is to build a model that uses machine learning models to detect malicious files.
2. The second objective is to reduce the amount of time to train the data.
3. The final objective is to find the best model for malicious file detection by extracting detailed comparisons between different machine learning algorithms.

## 1. Introduction

Malware is defined as software designed to penetrate or damage computer systems without the prior consent of the owner. Malware is actually a generic definition for all kinds of computer threats. A simple classification of malware consists of file infectors and individual malware. Worms, backdoors, Trojans, rootkits, spyware, and adware are examples of malware that are classified depending on their distinct actions.

Today, technology is so advanced that everything is being coordinated digitally rather than manually. Technology has its pros and cons. We make life easier and provide access to personal information to anyone who can misuse it, which can lead to cyberattacks and data loss. Therefore, device security is very important in today's cyber world.

Internet use is increasing day by day. One of the downsides of the widespread use of the Internet is that many computer systems are vulnerable to attacks and malware. Malicious code, such as malware, malicious code, or malicious executable files, has many different names.

Malicious software is malicious software used to violate a computer system's security policies regarding data confidentiality, integrity, and availability.

System data may be changed and deleted without the user's knowledge in order to harm the system.

Malicious software, commonly known as malware, is computer software that is intended to disrupt, damage, or gain unauthorised access to a computer system or network**.** Depending on their purpose and distribution system, malware can be divided into various non-exclusive categories.

# 2. Implementation

## 2.1 Dataset

Malicious software is software that is expressly designed to cause harm to a computer system, or to obtain unauthorised access to it. A legitimate file is software that does not act like malware and is useful and harmless to users.

Statistical analysis was performed on these files and mainly consisted of extracting PE information and calculating entropy for various sections of these files.

## 2.2 Feature Extraction

PE files have many format features, but most of them do not help distinguish malware from harmless software. Based on empirical research and in-depth analysis of PE file format characteristics, we have extracted 54 characteristics that can distinguish malicious software from malicious software from a given PE file. These features are summarized in . In the discussion below, the features extracted in this study are briefly explained, and 9 important features are also classified.

## 2.3 PE File Header

Like other executable files, PE files have a set of fields that define the rest of the file. The header, as previously stated, comprises information such as the code's location and size. The first few hundred bytes of a typical PE file are occupied by an MSDOS stub. PE files are found by indexing the e_ifanew MS DOS header. Because e_ifanew just gives the file an offset, it adds the file's memory address to get the real memory address**.** There are several main subsections defined in the header section itself. they are listed below

**Signature:** Only the signature is included for easy understanding by the Windows bootloader. Letter P.E. Anything followed by two zeros says it all.

**Machine:** A number that identifies the machine type of the target machine, such as Intel, AMD, etc. It targets the same basic architecture as Intel, as shown below.

0x14d Intel i860

**NumberOfSections:** This determines the size of the partition table immediately following the header.

**SizeOfOptionalHeader:** Between the optional header's top and the partition table's first row. This is the size of the executable's optional header, which is necessary. For an object file, its value must be 0.

## 2.4 Training and Testing Data

Splitting the data into a training set and a test set is an important part of evaluating data mining models. In general, if you split the data set into a training set and a test set, most of the data is used for training and less data is used for testing. Machine learning algorithms learn from data.

| Testing Data (80%) | Testing Data (20%) |
|---|---|

They find relationships, develop understanding, make decisions, and evaluate confidence based on

the learning data they receive. And the better the training data, the better the model will perform. The more you train the model, the more accurate it becomes. Therefore, it is always advisable to have a large amount of data as training data.

# 3. Algorithms

## 3.1 XGBoost

Gradient Boosted Decision Trees are implemented in XGBoost. Decision trees are generated in a sequential manner using this approach. Weights play an important role in XGBoost. Every independent variable is assigned a weight and entered into a decision tree that predicts an outcome. The weights of the mispredicted variables in the tree are increased and those variables are passed to the second decision tree. These individual classifiers/predictors are then combined to create a more robust and accurate model. Can be used with regression, classification, ranking, and custom forecasting tasks.

## 3.2 CatBoost

CatBoost is a gradient boosting implementation that uses a binary decision tree as its default predictor.

Catboost can be used to solve problems such as regression, classification, multi-class classification, and ranking. The modes differ in the objective function to minimize in gradient descent. Catboost also has proactive metrics to measure model accuracy.

Catboost introduces two significant algorithmic improvements. It is an implementation of order boosting, a permutation-based alternative to existing algorithms, and an innovative categorical feature processing algorithm.

Both methods use random permutations of training examples to avoid prediction bias due to certain types of target leaks present in all existing implementations of gradient boosting algorithms.

## Categorical Feature Handling

**Ordered Target Statistic:**

This is a basic but effective method for encoding each category feature using a category-provided estimate of the intended target 'y'. Well, an incorrect application of this encoding (average of 'y' over the same category of training examples) leads to target leakage.

CatBoost employs a more efficient technique to avoid this shift in projections. It is based on the principle of order and is inspired by an online learning algorithm that receives training examples sequentially over time. In this case, the TS values for each example depend only on the observed history.

To apply this idea to a standard offline setup, Catboost introduced an artificial "time". This is a random permutation of the *'σ1'* training example.

Then, for each example, we use all available "history" to compute the target statistic.

Using only one random permutation results in a higher variance of the target statistic in the previous example than in the subsequent example. For this, CatBoost uses different permutations for the different stages of gradient boosting.

**Ordered Boosting:**

CatBoost has two tree-structured selection modes: Ordered and Plain. Normal mode corresponds to a combination of the standard GBDT algorithm and aligned target statistics.

We perform random permutations of the training example *'σ2'* in ordinal boost mode and have n different supporting models *M1, . . . , Mn* causes the model *Mi* to be trained using only the first i samples of the permutation.

At each step, use the *Mj−1* model to obtain the residuals for the jth sample.

Unfortunately, this algorithm is not feasible for most real-world problems as it has to support n different models, which increases the complexity and memory requirements by n times. Catboost implements a modification of this algorithm based on the gradient boosting algorithm, using a single tree structure common to all models to be built.

To avoid prediction bias, Catboost uses a permutation such as $\sigma 1 = \sigma 2$. This ensures that target-*yi* is not used to train *Mi* to compute target statistics or to estimate gradients.

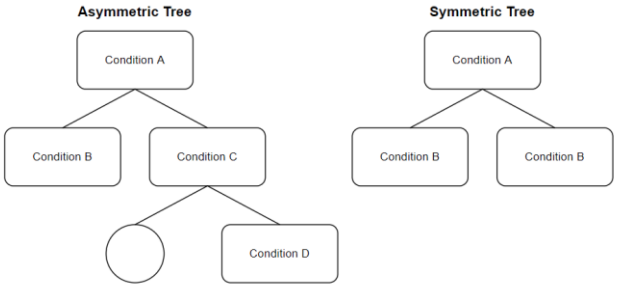



**Figure : Asymmetric and Symmetric Trees**

In CatBoost, a symmetric tree or balanced tree represents an agreed split condition for all nodes of the same tree depth. On the other hand, XGBoost leads to an asymmetric tree, which means that the split conditions for each node at the same depth can be different.

### Logloss

Log loss is an important metric that defines the divergence of numeric values between the estimated probability label and the true label, expressed as a value between 0 and 1. Log loss is measured in units, and different values may be better or worse depending on the size and type of case and problem.

### Accuracy

Accuracy is usually a metric that describes how a model performs in all classes. This is useful when all classes are equally important. The number of right guesses divided by the total number of forecasts yields this figure.

## 4.1 Results Part 1

Figure 1 shows that the log loss decreases with increasing n_estimators. This means that log loss is inversely proportional to n_estimators. For CatBoost, logloss is less than 0.1 when n_estimators = 100.

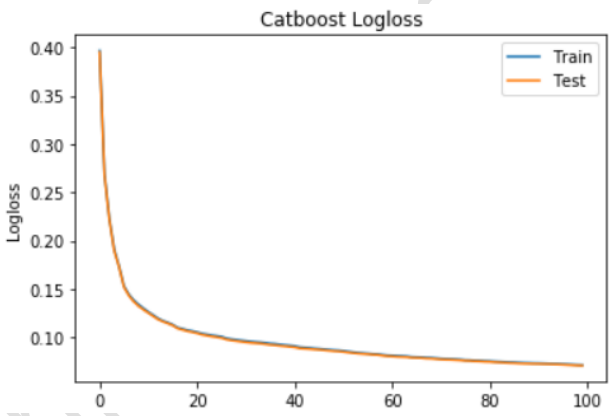



Figure 1: CatBoost Logloss for n_estimator = 100

Figure 2 shows that the accuracy increases with increasing n_estimators. For CatBoost, the accuracy value is almost 97% when n_estimators = 100.

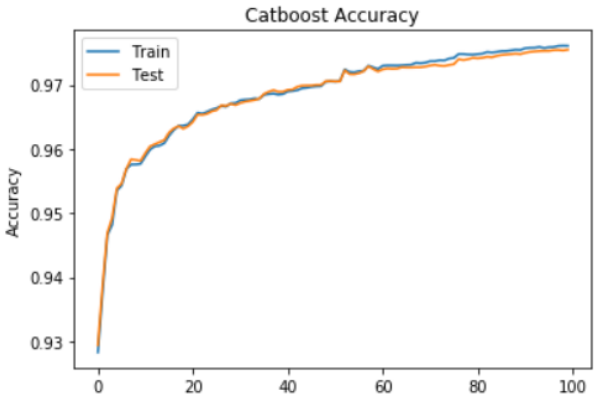



Figure 2: CatBoost Accuracy for n_estimator = 100

## 4.2 Result Part 2

Figure 3 shows that when n_estimator is increased to 700, the logloss value is less than 0.05. This means that the loss decreases as n_estimators increase.

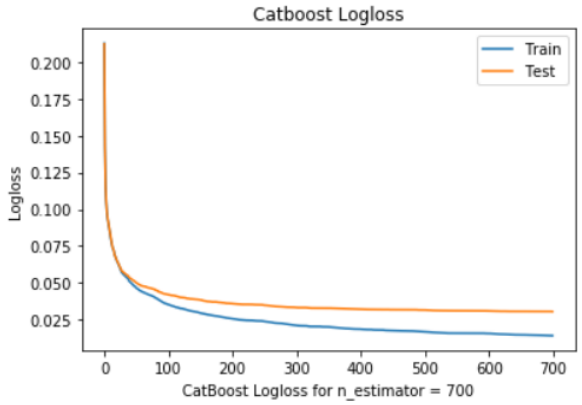



Figure 3: CatBoost Logloss for n_estimator = 700

As can be seen in Figure 4, for CatBoost, when n_estimators = 700, the accuracy increased compared to Figure 2 as n_estimators increased from 100 to 700.

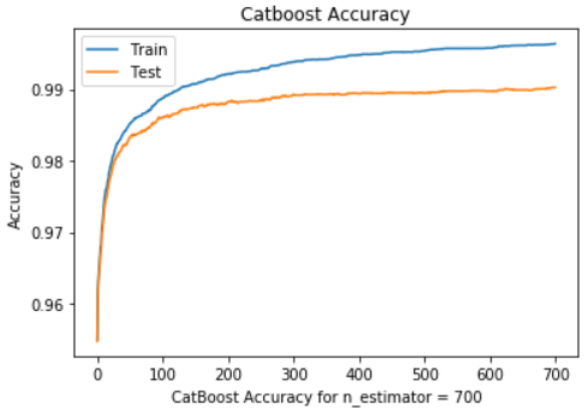



Figure 4: CatBoost Accuracy for n_estimator = 700

## 4.3 Result Part 3

Table 1: Performance of CatBoost with other classification algorithm

| Models | Accuracy (%) | F1_score | Precision | Recall | Time in seconds |
|---|---|---|---|---|---|
| **Random Forest** | 95.09 | 0.93 | 0.94 | 0.92 | 12.99 |
| **Gradient Boosting** | 96.89 | 0.95 | 0.96 | 0.95 | 97.72 |
| **XGBoost** | 97.65 | 0.97 | 0.97 | 0.96 | 9.00 |
| **CatBoost** | 97.55 | 0.96 | 0.97 | 0.96 | 5.74 |

In the table 1 above, CatBoost and XGBoost with n_estimators = 100 have better accuracy than the rest of the models. However, CatBoost has shorter training time.

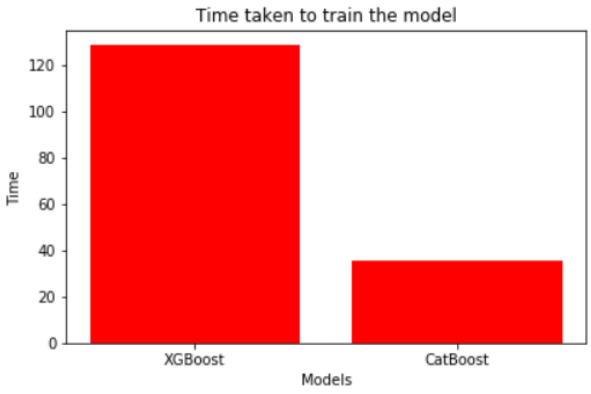
To increase the accuracy and minimize the time taken to train the model. We need to increase the n_estimators.

Table 2: Performance of CatBoost with XGBoost when n_estimators = 700

| Models | Accuracy (%) | F1_score | Precision | Recall | Time in seconds |
|---|---|---|---|---|---|
| **XGBoost** | 99.20 | 0.99 | 0.99 | 0.99 | 128.4 |
| **CatBoost** | 99.02 | 0.98 | 0.98 | 0.99 | 35.64 |

In the table 2 above, CatBoost and XGBoost with n_estimators = 700 achieved nearly 99% accuracy.

XGBoost with n_estimators 700 took more time to train the data. So we want to reduce the training time of the model. To reduce training time, I used a boost called CatBoost.




Here, CatBoost takes 3.5x less time to train the model compared to XGBoost. So, we finally found that the CatBoost is more powerful than the other models. CatBoost achieved 99% accuracy with less model training time.

## 5. Conclusion

The main algorithms we focus on are XGBoost and CatBoost. We found which algorithm is best for training the model with less running time and the best accuracy. This algorithm is CatBoost. This will help determine which ones are malignant and which are not.

## 6. References

[1] Malware Types and Classifications, Bert Rankin, 28.03.2018, published in LastLine, last accessed 12.09.2018.

[2] A Brief History of Malware - Its Evolution and Impact, Bert Rankin, 05.04.2018, published in LastLine, last accessed 12.09.2018.

[3] Detecting malware through static and dynamic techniques, Jeremy Scott, 14.09.2017, published in NTT Security, last accessed 12.09.2018.

[4] Hybrid Analysis and Control of Malware, Kevin A. Roundy and Barton P. Miller, International Workshop on Recent Advances in Intrusion Detection, pp. 317-338, 2010, Springer.

[5] Advanced Malware Detection - Signatures Vs. Behavior Analysis John Cloonan Director of Products, Lastline, 11.04.2017, published in Infosecurity Magazine, last accessed 12.09.2018.

[6] D. Devi, S. Nandi, "PE File Features in Detection of Packed Executables," International Journal of Computer Theory and Engineering, (January), 476–478, 2012, doi:10.7763/ijcte.2012.v4.512.

[7] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C. Nicholas, "Malware Detection by Eating a Whole EXE," 2017, doi:10.13016/m2rt7wbkok.

[8] A. Moser, C. Kruegel, E. Kirda, "Limits of static analysis for malware detection," Proceedings - Annual Computer Security Applications Conference, ACSAC, 421–430, 2007, doi:10.1109/ACSAC.2007.21.

[9] M. Alazab, S. Venkataraman, P. Watters, "Towards understanding malware behaviour by the extraction of API calls," Proceedings - 2nd Cybercrime and Trustworthy Computing Workshop, CTC 2010, (July 2009), 52–59, 2010, doi:10.1109/CTC.2010.8.

[10] J.Z. Kolter, M.A. Maloof, "Learning to detect malicious executables in the wild," KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 7, 470–478, 2004, doi:10.1145/1014052.1014105.

[11] Jason Brownlee, XGBoost with Python Gradient Boosted Trees with XGBoost and sci-kit learn. Edition: v1.10

[12] https://www.kdnuggets.com/2018/11/master ing-new-generation-gradient-boosting.html.

[13] https://xgboost.readthedocs.io/en/stable/

[14] https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[15] S. Geetha, N. Ishwarya, N. Kamaraj, "Evolving decision tree rule based system for audio stego anomalies detection based on Hausdorff distance statistics," Information Sciences, 180(13), 2540–2559, 2010, doi:10.1016/j.ins.2010.02.024.

[16] H. Pham, T.D. Le, T.N. Vu, Static PE Malware Detection Using Gradient, Springer International Publishing, 2018, doi:10.1007/978-3-030-03192-3.

[17] https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[18] https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a

[19] https://towardsdatascience.com/catboost-vs-lightgbm-vs-xgboost-c80f40662924

[20] Jonas Nylen, "Gradient Boosting Trees 101: How Do Decision Trees Work:"

[21] Chris Smith, "Decision Trees and Random Forests: A Visual Introduction for Beginners".

[22] Nonita Sharma , "XGBoost. The Extreme Gradient Boosting for Mining Applications".

[23] Alberto Boschetti, Luca Massaron , "Python Data Science Essentials - Third Edition".

[24] Rajdeep Dua, Manpreet Singh Ghotra, Nick Pentreath , "Machine Learning with Spark - Second Edition".

[25] Alex Ronquillo, "A Beginner's Guide to the Python time Module".

[26] https://catboost.ai/en/docs/

[27] E. Raff, J. Sylvester, C. Nicholas, "Learning the PE header, malware detection with minimal domain knowledge," AISec 2017 - Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Co-Located with CCS 2017, 121–132, 2017, doi:10.1145/3128572.3140442

[28]https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[29] https://www.forcepoint.com/cyber-edu/malware

[30] https://en.wikipedia.org/wiki/Malware

[31] Duncan M. McGreggor , "Mastering Matplotlib".