Malware Detection Using Different Supervised Learning Methods

Kailin Lyu[1, †], Fengning Yang[2, †], and Luning Zhang[3,4, †]

[1] CIA FIRST International School, Sen Sok, Phnom Penh, 12101, Cambodia, kailin.lyu@ciaschool.edu.kh
[2] Living Word Shanghai, Shanghai, 200000, China, y2532562185@outlook.com
[3] Canada Qingdao Secondary School, Qingdao, 266000, China, LuningZhangCQSS@cseec.education
[4] Author to whom any correspondence should be addressed
* All authors contributed equally

Abstract
Malware that can threaten cyber security is now a problem that needs to be addressed. Malware detection has been significantly developed through the use of machine learning techniques. However, simple supervised learning is rarely used for malware detection. The goal of this paper is to compare the accuracy of different supervised learning models in malware detection. The experiment will load a dataset with malware and corresponding characteristics, train four models separately using a decision tree, logistic regression, and Naïve Bayes, and compare their train scores, test scores, and test time. The experimental result shows that the decision tree model has the best in malware detection, following the Logistic regression model and the Gaussian Naïve Bayes model which has similar scores. In contrast, the Multinomial Naïve Bayes model has lower scores. Moreover, the logistic regression model is significantly slower than other models. The results of the experiments reflect the different efficiency and accuracy of different supervised learning methods when used for malware detection. A possible reason for the differences between the methods is due to the complex dataset, and in conclusion, supervised learning could play a crucial role in cyber security.

Keywords
Supervised learning · Malware detection · Decision tree · Logistic regression · Naïve Bayes

1. Introduction

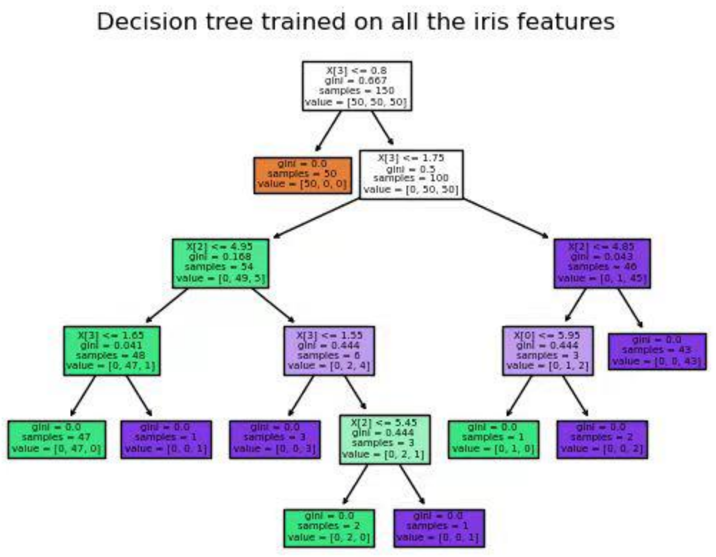
Nowadays, malware which is malicious software perpetrators dispatch such as ransomware, worms, and trojan, is a threat that cannot be ignored to cyber security, they are able to infect individual computers or an entire organization’s network. Nevertheless, the malware detection technique, which was first created in 1949 by John von Neumann, is an efficient tool for dealing with malware [1]. Malware detection is the process of scanning the computer and files to detect malware, and it involves multiple tools and approaches. Applying machine learning for malware detection is ubiquitous, but simpler supervised learning is not as widely used. Therefore, this paper aims to verify if supervised learning models show great accuracy when dealing with a complex dataset about malware. In the experiment, multiple supervised learning methods are used to build different models, and their performance is compared.

2. Methods

In order to achieve the goal mentioned, an experiment based on the anomaly detection is going to be done with a dataset and several representative supervised learning algorithms for binary classification related tasks. In the next few sections of this chapter, various algorithms are decided to be used for experimentation and a database containing samples of attacks are going to be introduced. Several figures are going to be displayed in order to explain the methods and the dataset better. Several symbols are going to be used: DT (Decision Tree), LR (Logistic Regression), GNB (Gaussian Naive Bayes), MNB (Multinomial Naive Bayes), [x, y] (Between x and y), P(y|x) (The probability of y based on condition).

2.1. Decision Tree (DT)

The Decision Tree model is a non-parametric supervised learning algorithm. It consists of a tree-like graph shown in Figure 1 with decision nodes and possible results. Features with if/else statements exist in the nodes, and such if/else statements decide which node is going to be passed next. Starting from the root node and passing through different nodes, the leaf node with a possible result that fits to the features in nodes passed will be approached [2, 3]. This method is very simple, easy to understand, and is widely-used for binary classifications. Meanwhile, it has a higher accuracy when the features are non-linear. However, an overfit is more likely to appear during training while there are too many nodes in a tree.



Decision tree trained on all the iris features
X[3] <= 0.8 gini = 0.667 samples = 150 value = [50, 50, 50]
gini = 0.0 samples = 50 value = [50, 0, 0]
X[3] <= 1.75 gini = 0.5 samples = 100 value = [0, 50, 50]
X[2] <= 4.95 gini = 0.168 samples = 54 value = [0, 49, 5]
X[2] <= 4.85 gini = 0.043 samples = 46 value = [0, 1, 45]
X[3] <= 1.65 gini = 0.041 samples = 48 value = [0, 47, 1]
X[3] <= 1.55 gini = 0.444 samples = 6 value = [0, 2, 4]
X[0] <= 5.95 gini = 0.444 samples = 3 value = [0, 1, 2]
gini = 0.0 samples = 43 value = [0, 0, 43]
gini = 0.0 samples = 47 value = [0, 47, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]
gini = 0.0 samples = 3 value = [0, 0, 3]
X[2] <= 5.45 gini = 0.444 samples = 3 value = [0, 2, 1]
gini = 0.0 samples = 1 value = [0, 1, 0]
gini = 0.0 samples = 2 value = [0, 0, 2]
gini = 0.0 samples = 2 value = [0, 2, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]

Figure 1. Decision tree trained on all the iris features

2.2. Logistic Regression (LR)

Logistic regression is a machine learning algorithm based on linear regression. Different from the linear regression, as a classifier, it has discrete values as outputs, and is used to solve binary classification tasks. In order to make classifications with inputs, a sigmoid function converting inputs into outputs with value between section was used to represent the posterior probability P(y=1|x) shown in Figure 2 [4]. This model performs better when the features are discrete, since the model becomes faster to train and more stable while the features are discrete.

$$P(Y=1|x)=\frac{1}{1+e^{-(w^T x+b)}}$$

Figure 2. Using sigmoid function to represent posterior probability

2.3. Naïve Bayes

The Naive Bayes classifier is a simple probabilistic classifier based on the Bayes Principle. However, it actually still works without the principle since in many actual applications a Maximum Likelihood Estimation method is used to estimate the value [5, 6]. The Naive Bayes method assumed the features are independent to each other and affect the results independently, which greatly increased the simplicity of this model. For that reason, only a few training data is required during the training of this method. Beside its simplicity, the high stability and efficiency are also advantages of this model. On top of that, it tends to have relatively high accuracy while using a larger database. However, the attribute conditional independence assumption lowered the effect of Bayes classifier algorithm.

2.4. Gaussian Naive Bayes (GNB)

The Gaussian Naive Bayes classifier is a certain type of Naive Bayes classifier whose conditional probability follows the Gaussian (normal) distribution, example presented in Figure 3 [7]. It's good at handling continuous values, and works similarly to the linear regression classifier.

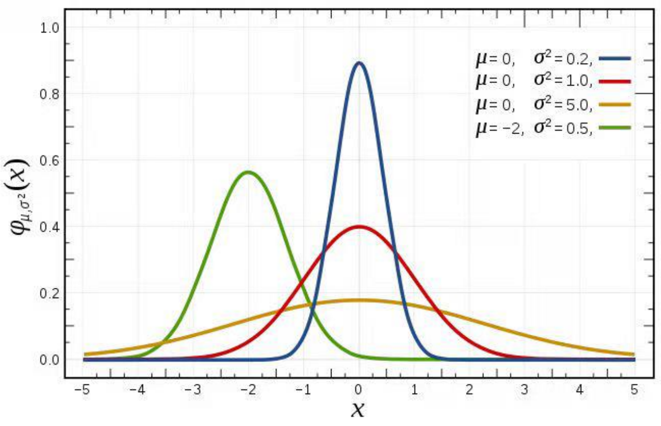


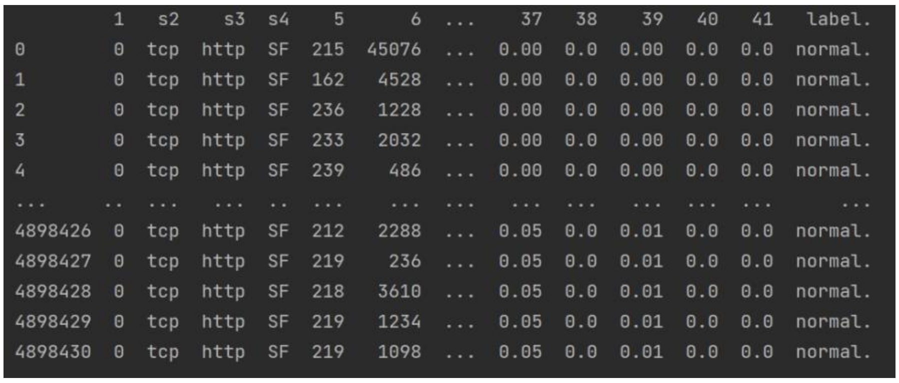
Figure 3. Example of graph following Gaussian Distribution

2.5. Multinomial Naive Bayes (MNB)

The Multinomial Naive Bayes classifier is the Naive Bayes classifier that uses multinomial distribution for each feature. However, this method is different from many other Naive Bayes classifiers, since it doesn't assume the features as conditionally independent. In its principle, it assumes that the prior probability of a feature is a multinomial distribution. Moreover, it's good at handling problems with discrete features [8]. The results of the Multinomial Naive Bayes classifier are specific, the features involved are often discrete positive integers.

2.6. KDD'99 Dataset

Since it's nearly impossible to find actual examples of attacks for training, a public dataset called KDD'99 available on the internet including several types of attacks is used for our test. This dataset was initially used in the KDD Cup 1999, and was then widely used for research. To create this dataset, 9 weeks of internet connections were captured and some simulated attacks were added to part of the samples. Beside that, another simulated two weeks of connections were added to the dataset with additional types of attacks as test data [9]. The full dataset includes 4898431 labeled samples of data, and for this test, only 10 percent of this dataset is used. Each of the samples has 41 features and a label identifying whether it's normal or is a certain type of attack, shown in Figure 4. Four main categories (DoS, Probing, R2L, U2R) of attacks and 39 different subtypes of simulated attacks exist in the dataset [10].



```
         1   s2    s3  s4    5      6  ...    37   38    39   40   41   label.
0        0  tcp  http  SF  215  45076  ...  0.00  0.0  0.00  0.0  0.0  normal.
1        0  tcp  http  SF  162   4528  ...  0.00  0.0  0.00  0.0  0.0  normal.
2        0  tcp  http  SF  236   1228  ...  0.00  0.0  0.00  0.0  0.0  normal.
3        0  tcp  http  SF  233   2032  ...  0.00  0.0  0.00  0.0  0.0  normal.
4        0  tcp  http  SF  239    486  ...  0.00  0.0  0.00  0.0  0.0  normal.
...     ..  ...   ...  ..  ...    ...  ...   ...  ...   ...  ...  ...      ...
4898426  0  tcp  http  SF  212   2288  ...  0.05  0.0  0.01  0.0  0.0  normal.
4898427  0  tcp  http  SF  219    236  ...  0.05  0.0  0.01  0.0  0.0  normal.
4898428  0  tcp  http  SF  218   3610  ...  0.05  0.0  0.01  0.0  0.0  normal.
4898429  0  tcp  http  SF  219   1234  ...  0.05  0.0  0.01  0.0  0.0  normal.
4898430  0  tcp  http  SF  219   1098  ...  0.05  0.0  0.01  0.0  0.0  normal.
```

Figure 4. Overview of the KDD'99 dataset

However, before the test starts, some additional work is required to be done to this dataset to make it usable. First at all, the features and outputs of the dataset are divided into two parts so that they could be used as X and Y values in the fit method. Secondly, a train-test-split method is used to split 20 percent of the database for test data and make the rest parts become train data. Finally, a One-Hot encoding method is used to make the discrete values in the dataset readable by certain fit functions.

3. Results

Based on the dataset and algorithms introduced in the last chapter, the experiment can finally start. This experiment is mainly made with python, which makes it easier to import and train the model. In this chapter, the results received from the models will be presented, as well as analysis and discussions on the results. All of the algorithms are using the same train data and test data, making it easier to compare the results of different methods. Moreover, figures and tables are used in order to make the result more obvious. A group of most typical data is chosen to make the tables and figures from various experiments made under different conditions such as proportions between train and test sets in the dataset. In the table, the accuracy chosen algorithms have on both train and test data and the time each algorithm cost on training is presented numerically while only the accuracy of each methods' prediction on test data is shown graphically in a bar chart on the figure.

3.1. Decision Tree

The percentage of accuracy of DT in predicting both train and test data is presented by Table 1 below. According to the results, the DT method has an extraordinarily high accuracy on such anomaly detection tasks. According to Table 1, this model has both a high accuracy on training data and testing data, which proves that it performs very well on this task without an overfit or underfit. Moreover, DT also costs a relatively short time for training, and is even the fastest trained algorithm in some other trials of this experiment. Various indications indicate that this is the best model for anomaly detection among the models selected for this experiment.

Table 1. Score of the decision tree model

Method	Train	Test	Time

DT	0.999997	0.999747	5.296875

3.2. Logistic Regression

During the experiment, at most of the time it takes a relatively long time to train a LR model. According to this representative data chosen from various tests, the time it costs to train a LR model is about the time consumed while training a DT model. However, according to Table 2 shown below, this long training time doesn't bring a relatively high accuracy compared to the other models. This model's accuracy of predicting test data only has had an insignificant decrease compared to the accuracy of train data, which led to the result that it doesn't overfit during this experiment. To recap, according to the above, it doesn't seem to be a reasonable choice in the network security related tasks such as this one due to its low speed and medium accuracy.

Table 2. Score of the logistic regression model

Method	Train	Test	Time
LR	0.94128	0.939952	759.8281

3.3. Gaussian Naïve Bayes

The results produced by GNB are presented on Table 3. The GNB classifier has an accuracy similar to but a little bit higher than LR. However, a significant gap exists between the training speed of LR and GNB, which the LR classifier takes much longer to train. The accuracy of GNB on predicting Test data seems to be similar as the accuracy of predicting training data, representing that the model doesn't have an obvious overfit. In conclusion, since GNB fits better than most of the other algorithms, it's still a reasonable choice on solving this kind of problems though both of its accuracy and speed are quite inferior compared to DT.

Table 3. Score of the Gaussian NB model

Method	Train	Test	Time
GNB	0.947221	0.945863	9.421875

3.4. Multinomial Naïve Bayes

From Table 4, this model has a poorer accuracy than the previous ones. It has the highest training speed in this trial among the algorithms used, but both its score for predicting training data and testing data is relatively low. According to the aspects introduced in chapter 2, it's out of expectation that it has a lower score in this experiment. Some possible reasons causing this strange phenomenon are going to be analyzed in the next few sections. As a conclusion, at least according to the results received from the experiment, using MNB for anomaly detection is not a reasonable choice due to its low accuracy compared to other supervised machine learning algorithms.

Table 4. Score of the Multinomial NB model

Method	Train	Test	Time
MNB	0.826437	0.825464	4.8125

4. Comparison

Individual results have already been presented, but in order to make a more direct and obvious comparison, the results, tables shown in previous lines will be presented in total in this section and a figure based on the models' behavior on predicting test data will be revealed. From both Figure 5 and Table 5, it's easy to come to several conclusions. First at all, the accuracy of DT on both predicting training data and testing data is the highest. After that, the accuracy of GNB and LR is quite similar to each other. On top of that, the LR classifier used the longest time for training, and there's a significant gap between the training time of LR and other models. Moreover, there's no obvious overfitting emerging in the experiment among all of the algorithms used. Finally, the MNB method has the lowest accuracy on both test data and train data, but it costs the shortest time to train. Combining those results with the time consumed to train the model, shows that the DT model performs the best in this area among the chosen algorithms. Except that, since most of the algorithms used have above 90 percent of accuracy and half of them have a high efficiency, machine learning algorithms does have a significant effect on providing network security.

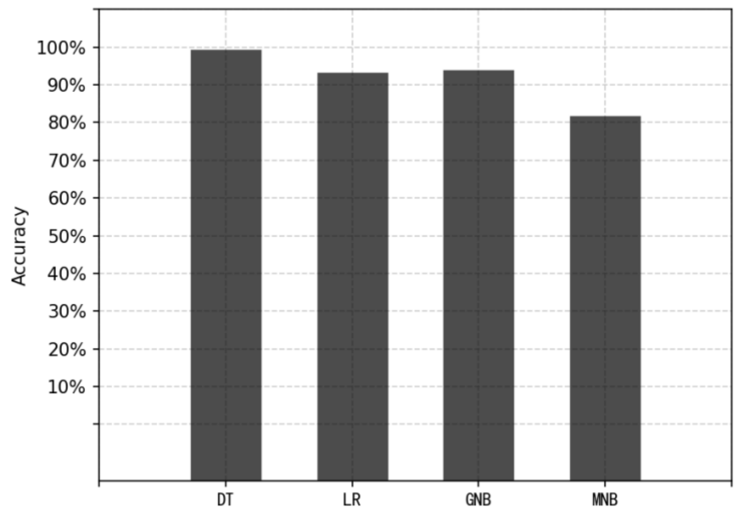


Figure 5. Graph with each algorithms' accuracy on predicting testing data

Table 5. Table with accuracy and training speed of each method

Method	Train	Test	Time
DT	0.999997	0.999747	5.296875
LR	0.94128	0.939952	759.8281
GNB	0.947221	0.945863	9.421875
MNB	0.826437	0.825464	4.8125

5. Analysis

Among the results in previous sections, several notable circumstances emerged. Considering that most of them have a direct connection with the conclusion, it's necessary to make an analysis on those circumstances in order to make more thorough research. The first one is the high accuracy of the DT method. As what is mentioned in the definition of DT at chapter 2, this model tends to have a better performance when various discrete features exist, so it's not surprising to see this model having a higher accuracy. Secondly, is the high similarity between the results produced by LR and GNB. In fact, this phenomenon is mainly caused by the similarities between how the two algorithms work. While the Naive Bayes model's conditional probability follows Gaussian distribution, the form of posterior probability of LR and GNB is the same. Meanwhile, both of the two algorithms are classification models. Finally, is the strange poor behavior of the MNB model. Theoretically, this model tends to have a higher accuracy while the features are discrete, but the results are quite the opposite. One possible source of error might be the using of a one-hot encoder. Since a one-hot encoder is used to deal with certain features, the encoded features are assumed to be independent of each other. Different from other Naive Bayes methods, the MNB classifier doesn't assume that the features affect the result independently, so an error might occur when some features become independent to each other.

6. Conclusion

In order to evaluate the effect different machine learning algorithms have on network security and compare the different performances of algorithms in this field, a test using 10 percent of the publicly available dataset KDD'99 and several representative supervised machine learning algorithms was made. Comparing the behaviors of the machine learning algorithms used in the test, some obvious distinctions are able to be found. Among the methods used, the DT method has a significant high accuracy, and is also the one that has the highest score. The LR and GNB classifiers have a similar relatively high accuracy, while the accuracy of the MNB classifier is quite lower than the others. Some aspects of these algorithms might help to explain the circumstances. First, DT tends to have a better performance when discrete values exist, so it's not surprising that it has a high accuracy in this test. Second, GNB and LR work in a similar way, and are both classifier models, so it's easy for them to have a similar result. However, this research has a serious limitation, which tends to affect the final results of the test and might lower the accuracy of certain algorithms. Since One-Hot encoding is used to deal with discrete values in the KDD'99 dataset, certain features in the dataset become discrete and sparse, and assume the independence between those encoded features. This also helps to explain why the accuracy of LR is a little bit lower than GNB, and why MNB has a lower accuracy in prediction. To recap, from the high accuracy of most of the algorithms used in the experiment, machine learning algorithms have a significant effect on providing network security. Moreover, in network security related tasks like anomaly detection, using the Decision Tree method has tended to be a better choice.

The final goal of this thesis is to evaluate the effect of different machine learning algorithms on network security and compare the different performance of algorithms on this topic. This is helpful to deepen people's knowledge on the effect of artificial intelligence technologies on network security, and make it easier to find which machine learning algorithms are useful in this field. This thesis is also a reference to other researchers in the future. Hope this pioneering work can stimulate more researchers in this area to have deeper research on this topic and help them to make such research.

Finally, some defects and limitations still exist in this research, like the problems caused by One-Hot encoding in previous lines. It's theoretically possible to fix those problems by using other methods such as distributed representation, however, this might cause other unknown issues to emerge. In order to completely remove the limitation and improve the accuracy of the research, massive amounts of work are necessary to be done for having more thorough research in the future.

References

[1] UKEssays. (November 2018). Machine Learning in Malware Detection. Retrieved from https://www.ukessays.com/essays/computer-science/machine-learning-malware-detection-9512.php?vref=1. Accessed 11September 2022.

[2] *"Correction: Decision Tree Analysis."* Research Management, vol. **13**, no. 3, 1970, pp. 181–181. JSTOR, http://www.jstor.org/stable/24115830. Accessed 11 Sep. 2022.

[3] Leo Breiman, Jerome H. Friedman, Charles J. Stone, and Richard A. Olshen. Clas- sification and Regression Trees. Wadsworth and Brooks, Monterey, CA, 1984.

[4] Friedman Hastie, Tibshirani. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second edition. Springer, 2009.

[5] Hand, David J., and Keming Yu. *“Idiot’s Bayes: Not So Stupid after All?”* International Statistical Review / Revue Internationale de Statistique, vol. **69**, no. 3, 2001, pp. 385–98. JSTOR, https://doi.org/10.2307/1403452. Accessed 11 Sep. 2022.

[6] Russell, Stuart J, and Peter Norvig. Artificial Intelligence: A Modern Research. 2nd ed., 2003, p. 499.

[7] Altman, Douglas G., and J. Martin Bland. *“The Normal Distribution.”* BMJ: British Medical Journal, vol. **310**, no. 6975, 1995, pp. 298–298. JSTOR, http://www.jstor.org/stable/29726236. Accessed 11 Sep. 2022.

[8] "Sklearn.Naive_Bayes.Multinomialnb". Scikit-Learn, 2022, https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html.

[9] Mahbod Tavallaee, Ebrahim Bagheri, Wei L, and Ali A. Ghorbani. A detailed anal- ysis of the KDD CUP 99 data set. 2009 IEEE Symposium on Computational Intelli- gence for Security and Defense Applications, 07 2009.

[10] LDhanabalandSPShantharajah.AStudyonNSL-KDDDatasetforIntrusionDetection System Based on Classification Algorithms. International Journal of Advanced Research in Computer and Communication Engineering, 4(6):446–452, 2015.