ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 12 (2022) No. 6, pp. 17-30
DOI:10.33168/JSMS.2022.0602

Comparison of Label Encoding and Evidence Counting for Malware Classification

Min Xuan Low [1], Timothy Tzen Vun Yap [1+], Wooi King Soo [1], Hu Ng [1], Vik Tor Goh [2], Ji Jian Chin [1] and Thiam Yong Kuek [3]

[1] Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia

[2] Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Malaysia

[3] Faculty of Business and Finance, Universiti Tunku Abdul Rahman, 31900 Kampar, Malaysia

[+]*timothy@mmu.edu.my (corresponding author)*

Abstract. Malware is a type of software that is aimed to attack or harm a computer system without the owner's knowledge or permission. However, the traditional methods such as signature-based and behaviour-based malware analysis techniques heavily depend on manual inspection and detection by analysts, and it is nearly impossible, although it is very reliable, given the cons when dealing with techniques like polymorphism, metamorphism and obfuscation. Thus, this project aims to develop models to detect malware in operation. Two different techniques are considered for feature engineering, namely label encoding and evidence counting. Feature selection is applied to isolate less significant features. Machine learning models such as Random Forest (RF), Decision Tree (DT), K-Nearest Neighbour (K-NN) and Support Vector Machine (SVM) Classifiers, Multilayer Perceptron (MLP) and Long-Short Term Memory (LSTM) are applied for comparison of the efficacy of the two feature engineering approaches. The performances of the models are evaluated through accuracy, precision, recall, F1-score, and loss. Tree-based models such as RF and DT seem to be more suitable for mining patterns from labelled data, as their performances are generally better in the label encoding approach. SVM and K-NN tend to not cope very well with labelled data in this study. Deep learning approaches in this study has shown potential in malware classification, with further improvements required in building a robust solution against solving complex real-world malware detention and classification.

Keywords: malware detection, machine learning, deep learning, classification

1. Introduction

In this digital era, malware has affected a vast number of computer devices, as the gadgets and the Internet play a vital part in our lives (Ghadiya et al, 2013; Gilbert, et al., 2020). Malware is a type of software that is aimed to attack or harm a computer system without the owner's knowledge or permission. These hostile threats include but are not limited to programs like rootkits, adware, bots, spyware, ransomware, viruses, worms, trojans and bugs (Ghadiya et al, 2013; Masabo, 2019; Rathore et al, 2018). Without a question, these malwares pose a significant security risk in today's computer world, as people's lives are increasingly linked with digital gadgets (Masabo, 2019), provided the fact that malware threats continue to advance, by horizontally, namely the types and functionality, as well as vertically, which refers to the number and volume (Gilbert et al., 2020).

Or-meir et al. (2019) mentioned that a type of malware, namely ransomware, single handedly seeded damage that cost $8 billion dollars in 2018. Although ransomware attacks have lessened after its peak around 2016, cryptojacking, another type of malware which utilises the victim's computing power to mine cryptocurrencies, has become the mainstream. In the third quarter of 2017, from the materials published by the Kaspersky Lab Solutions, one of them demonstrates the top countries where online resources contain malware from a whopping 277,646,376 attacks blocked in 185 countries. Along with the development of internet and digital products, the threats from malware are also having a quadruple growth from $3.3 billion to $13.3 billion from 1997 to 2006, reported by Computer Economics. Even while scoping into Malaysia, according to Bernama (2021), Trend Micro discovered that a total of 232.70 million assaults were prevented in the first half of 2021, with email threats accounting for over half of them (145.95 million) and has increased 24% from the first half of 2020 in the Malaysia market. From the statistics, it is obvious that the importance of malware analysis or detection has gained the attention of the public, until the point that the definition of “Year of Mega Breach” has to be redefined every few years.

Traditionally, the process of identifying malware before any consequences relies on human intervention, by having analysts to perform malware analysis. Antivirus software, as a common line of defense, depends on signature-based methods to detect malware by identifying known malware with a short sequence of bytes (Chowdhury et al, 2017; Rathore, 2018). However, this can be easily countered by techniques such as polymorphism, metamorphism and obfuscation which automates the generation of different variants for a single malware family (Masabo, 2019; Nari & Ghorbani, 2013; Dahl et al., 2013). Apart from signature based methods, there is also behavior-based approach, which uses API calls instead of sequence of bytes, to tackle malware by their behaviour. However, this method is struggling with a high false alarm rate (Chowdury et al., 2017; Masabo, 2019].

Manual inspection and detection by analysts are nearly impossible, although it is very reliable (Or-meir at al., 2019), given the fact that Microsoft is receiving over 150,000 new and unknown files every day to be analysed (Dahl et al., 2013), leaving manual intervention to be trounced in terms of scalability. In view of this, analysts and researchers need to constantly improve and revise techniques in cybersecurity to combat the ever-growing malware threats (Gilbert et al., 2020), as Or-meir et al. (2019) laid out that malicious threat countering is all about preventing malicious code from executing, given the fact that it is difficult to handle unknown binary code compared to known malware. These lead to active research by the researchers and analysts in implementing machine learning and deep learning in anti-malware and malware detection in recent years (Dahl et al., 2013; Pascanu et al., 2015; Rathore et al., 2018). Machine learning and AI-powered malware detection tools generally caught attention due to their ability to allow learning to improve the scanning engine (Gilbert et al., 2020) and detecting new malware variants (Chowdury, et al., 2017). In order to implement machine learning in malware detection and classification, the feature is an essential factor to be studied. A well-processed feature can maximize the efficiency and accuracy of the machine learning models, as it explains the behaviour of the malware better. Feature engineering or transformation is a core component in implementation of machine learning in malware detection (Masabo, 2019). Therefore, this project aims to implement machine learning and deep learning models to detect malware in operation using proper data acquisition and preprocessing methods.

2. Literature Review

2.1. Traditional Malware Analysis

Malware analysis involves studying malicious files and determining the purpose and functionality of a given malware sample in order to gain a better knowledge about several elements of malware like malware behavior, evolution over time, and their selected target (Sihwail et al., 2018; Ghadiya et al., 2013) The development of malware analysis is affected by the desire of the analysts to understand the behavior of the given sample, while on the other hand, the malware authors, or the attackers, who try to disguise the malicious intents of their creations. The competition between the two parties unintentionally motivates the development of malware analysis as the analysis techniques becomes more elaborate while new and creative evasion techniques continue to rise (Egele et al., 2008). Consequently, malware analysis is crucial for developing effective detection techniques for malicious code and critical to any business and infrastructure that responds to security incidents (Gilbert, et al., 2020). To wrap up the techniques in malware analysis, static analysis and dynamic analysis are the general approaches that classify current techniques based on the process of analyzing. Static analysis is a technique that analyzes a software program or examines the structure of the executable file without running it (Sihwail et al., 2018; Ghadiya et al., 2013). As a result, static analysis is more secure than dynamic analysis, as malware is not executed (Masabo, 2109). Meanwhile, dynamic analysis involves executing the given malware sample with a controlled environment and monitoring its behavior for the sake of analyzing the malicious behaviour (Ghadiya et al., 2013; Gilbert et al., 2020).

2.2. Feature Selection

Feature selection is a process of acquiring a subset from an original feature set according to a certain feature selection criterion, which selects the relevant features of the dataset (Cai et al., 2018). In other words, it reduces the number of columns or variables that are redundant and irrelevant in a dataset normally before constructing the predictive models. The purpose of implementing feature selection is to improve model's accuracy, save computational time and cost, as the machine learning methods will have difficulty in dealing with the large number of input features (Kumar & Minz, 2014) and causing overfitting problems.

According to Ang et al. (2015), supervised feature selection utilizes the labeled data in the feature selection process, and it is categorized to three types of methods: filter, wrapper and embedded. Its goal is to select a subset of features that can discriminate samples from different classes (Li et al., 2017). The filter method is known to be very efficient and computationally faster hence more easily increased up to big databases compared to wrapper methods (Ang et al., 2015). Filter method can work well with classification models as it is independent of any learning algorithm, as a result, it can provide generic solutions for a wide range of classifiers (Ang et al., 2015).

2.3. Modelling Techniques

Firdausi et al. (2010) focused on the analysis of machine learning techniques used in behaviour-based malware detection, an Attribute-Relation File Format (ARFF) file was created after several steps of data pre-processing. The accuracy of the 5 different classifier algorithms with feature selection are 92.9% for K-NN, 92.3% for Naive Bayes, 92.9% for SVM, 92.3% for DT and 91.0% for MLP.

Chowdury et al. (2017) focused on malware analysis and detection using data mining and machine learning classification, The researchers compiled a total of 52,185 executable files, with 41,265 being current malicious files and the rest are benign files. The accuracies for all the machine learning models are above 88%.

Selamat and Ali (2019) performed comparison of malware detection techniques using the three types of machine learning algorithms which are K-NN, DT and SVM for malware detection by using portable executable (PE) information as part of the feature extraction. The result of the experiment shows that the DT performs best with 99% detection accuracy and is effective in detecting the malware, the performance of K-NN has 94% detection accuracy while the SVM model obtained 91% detection accuracy.

Narayanan et al. (2016) proposed a novel technique that combines the best of both worlds to substantially boost malware performance of the classifier. The experimental results obtained 96.6% accuracy for the Linear K-NN classifier, Quadratic Kernel SVM obtained 94.6% accuracy, SFN1 and SFN2 obtained 95.6% and 94.2% respectively, and finally DFN obtained 95.5% accuracy.

Masabo (2019) proposed a novel feature engineering approach for a better classification and detection of polymorphic malware based on structural and behavioural features. The highest accuracy for the results that only use the static and dynamic features among the models is the Gradient Boost Model (GBM), which has 82% accuracy. The second highest accuracy is the Linear Discriminant Analysis (LDA) model with 75% accuracy and followed by the K-NN model, which has only 56% accuracy.

In a summary, K-NN, SVM and DT are widely used by the researchers in malware detection and classification. K-NN became the choice in most of the findings, due to the algorithm's ability to identify patterns, which allows for learning of the behaviours of the malwares. SVM on the other hand is good at learning from the examples and generally performs well on labelled data, which in this case, the labelled dataset of malware families. Looking at DT, although it makes strong inference by continuous splitting according to certain parameters, however the large variance can be a disadvantage. Hence, RF which learns around multiple decision trees is another suitable approach to overcome the disadvantage. Therefore, the K-NN, SVM, DT, and RF algorithms are considered and as part of the methodology in this study. While deep learning models tend to be the state-of-the-art solutions, models such as MLP and LSTM are proposed as additional models in the methodology. Table 1 shows the models used by researchers in recent years.

Table 1: Models in recent work

Author	*KNN*	*Naïve Bayes*	*SVM*	*Decision Tree*	*MLP*	*Random Forest*	*Neural Networks*	*LDA*	*GBM*	*Logistic Regression*
Firdausi et al., 2010	✓	✓	✓	✓	✓					
Chowdury et al., 2017		✓	✓	✓	✓	✓				
Selamat & Ali, 2019	✓		✓	✓						
Narayanan et al, 2016	✓		✓				✓			
Masabo, 2019	✓							✓	✓	
Masabo et al., 2018			✓							
Nari & Ghorbani, 2013				✓						
Dahl et al., 2013							✓			✓
Pascanu, et al., 2015					✓					✓
Rathore et al., 2018						✓	✓			

3. Methodology

The methodology involves data pre-processing, feature selection, model construction and evaluation. After the raw data is acquired, data pre-processing is implemented to clean, integrate and transform the data to an analytical dataset for further analysis. Feature selection is performed to select the significant features that contribute to the performance of the classification models. After the classification models are constructed, the results are recorded for evaluation.

3.1. Dataset Description

The raw data is provided by Ramilli (2016). In this malware dataset, there are multiple features extracted from various executable malware by using sandboxes tools. Every feature extracted constitutes a specific characteristic or a behaviour of malware. There are no benign files in this dataset which means the dataset does not contain files that are not malware. There are different types of malware samples in the dataset. The five malware families have been used to develop classification models. The five types of malwares extracted from the dataset are Advanced Persistent Threats (APT), Crypto, Zeus, Locker and Shadowbrokers.

Advanced Persistent Threats (APT) malware uses continuous and sophisticated hacking techniques to take advantage of flaws and obtain access to a system and stay within for an extended amount of time, causing effects that are harmful. This malware can be controlled remotely and can attack a certain target repeatedly.

The Crypto malware and locker malware are types of ransomware. This type of malware attacks computer systems by blackmailing the victims. The recipient will receive an email with an attachment, if the attachment is opened the computer will be infected and the files inside will be encrypted. The user has to pay for the encryption key to get the data back.

Zeus malware is designed to steal the banking information of users by monitoring and collecting keystrokes from users. The malware detects and records the keystrokes while the user is logging in their account on a banking website.

Locker malware is mainly used to lock and block access to the files and data stored in a computer system. The data and files can only be accessed until a ransom is paid.

The Shadow Brokers (TBS) is a group of hackers that is meant to exploit system vulnerabilities and can be monitored remotely. For surveillance reasons, they employ plugins to collect webcam and microphone output, log keystrokes, and access other discs.

Table 2 shows the malware families used in this project. The variable refers to the type of evidence that the author has constructed in this dataset, which include but not limited to resolved API, command execution, accessing, deleting, reading of files, as well as network activities and the requests of dns and http. The samples refer to the number of samples provided in each family of malware.

3.2. Data Pre-processing

Data pre-processing steps are crucial prior to any insightful analytics for the sake of drawing and generating meaningful information from the data source. The steps of data pre-processing that are going to utilize the raw data are data integration, data cleaning and data transformation.

In this project, the process of data integration can be shown in the generation of a flat dataset from every individual JSON file. Each JSON file represents a sample which contains information related to the respective malware, which in terms of dataset, refers to an individual row of data. The process of data integration is described in Algorithm 1.

Data cleaning needs to be performed as the dataset contains substantial number of null entries in the properties column. Common data cleaning steps include handling missing values, cleaning the inconsistent format of data, as well as identifying outliers. As this project focuses on pre-constructed malware samples data, there exists no noisy or outlier data within the dataset obtained. However, upon checking, there are null values in some of the features. As these descriptive features are variables that explain characteristics of a particular malware sample, the method of handling the missing values is to fill in a constant value instead of dropping them.

Table 2: Description of malware samples

Malware	Variables	Samples
APT1	64	292
Crypto	113	3020
Zeus	114	2019
Locker	128	431
Shadowbrokers	70	1270

Data transformation is a process of changing the format and structure of the dataset. In this project, the features inside the column properties are representing the specific activity or a characteristic of the labelled malware. Each feature value provided evidence that the malware was performing a certain task or possessed a specific trait. There were certain traits that were absent from proof. This indicates that the specific activity depicted by this feature has no effect on a single sample. Meanwhile, some other samples contain more than one evidence for each feature. For instance, for the feature file_write that possess a value of “45b96339 0976bdd4 e34d514f”, this can be inferred as there were 3 pieces of evidence for the activity of writing or creating a file, part of an evidence that describe the behaviour of that particular malware sample.

In the process of data transformation, the feature values, also known as the evidence, are transformed by:

- Counting the number of available pieces of evidence. The outcome of this transformation will be an integer value n, where $n \geqslant 0$, and zero indicates no evidence (Algorithm 2).
- Label encoding - converting the evidence into a numeric form so as to convert them into the machine-readable form

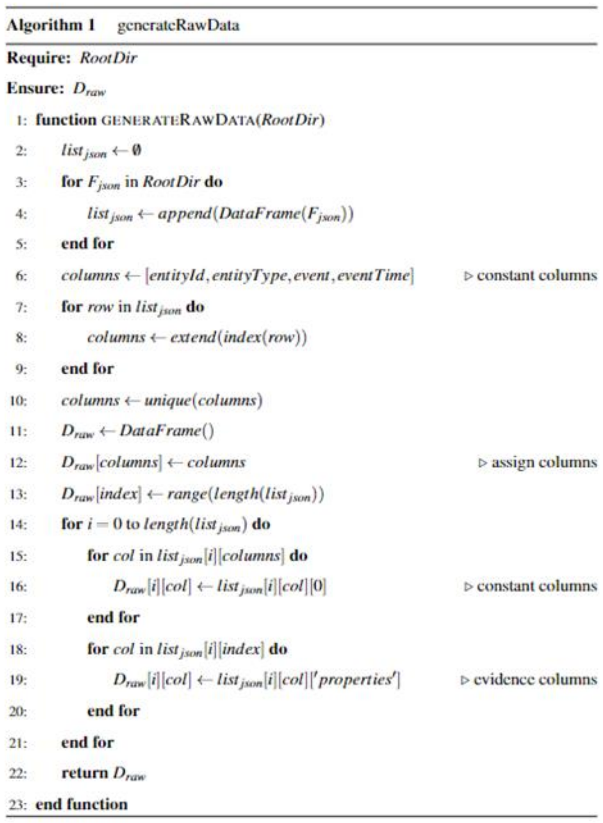
3.3. Feature Selection

A wrapper feature selection approach, namely Boruta, is used in this study. In this study, the features selected to train the models are those with an importance score > 0.5 generated by Boruta. The importance score was selected after a series of preliminary tests which shows consistent acceptable sampling.

The reasons of choosing Boruta in feature selection:

- Boruta can handle multi-variable relationships, interaction in variables, and the changing nature of randomness.
- It can work well with classification problems
- It is an advance on the RF variable importance measure, which is a widely used variable selection approach.

Algorithm 1: Conversion of JSON files in the dataset



```
Algorithm 1  generateRawData
Require: $RootDir$
Ensure: $D_{raw}$
function GENERATERAWDATA($RootDir$)
    $list_{json} \leftarrow \emptyset$
    for $F_{json}$ in $RootDir$ do
        $list_{json} \leftarrow append(DataFrame(F_{json}))$
    end for
    $columns \leftarrow [entityId, entityType, event, eventTime]$        ▷ constant columns
    for $row$ in $list_{json}$ do
        $columns \leftarrow extend(index(row))$
    end for
    $columns \leftarrow unique(columns)$
    $D_{raw} \leftarrow DataFrame()$
    $D_{raw}[columns] \leftarrow columns$        ▷ assign columns
    $D_{raw}[index] \leftarrow range(length(list_{json}))$
    for $i = 0$ to $length(list_{json})$ do
        for $col$ in $list_{json}[i][columns]$ do
            $D_{raw}[i][col] \leftarrow list_{json}[i][col][0]$        ▷ constant columns
        end for
        for $col$ in $list_{json}[i][index]$ do
            $D_{raw}[i][col] \leftarrow list_{json}[i][col]['properties']$        ▷ evidence columns
        end for
    end for
    return $D_{raw}$
end function
```

Algorithm 2: Evidence counting

```
Algorithm 2  countEvidence
Require: $D_{raw}$
Ensure: $D_{transform}$
 1: function COUNTEVIDENCE($D_{raw}$)
 2:     $D_{transform} \leftarrow D_{raw}$
 3:     for $row$ in $D_{raw}$ do
 4:         for $column$ in $row$ do
 5:             if $D_{raw}[row][column] =$ "" then
 6:                 $D_{transform}[row][column] \leftarrow 0$
 7:             else
 8:                 $S \leftarrow D_{raw}[row][column].split(" ")$
 9:                 $S_{len} \leftarrow length(S)$
10:                 $D_{tranform}[row][column] \leftarrow S_{len}$
11:             end if
12:         end for
13:     end for
14:     return $D_{tranform}$
15: end function
```

3.4. Model Construction

The classification models that were constructed in this project are K-NN classifier, DT classifier, RF classifier, Support Vector Classifier, Multilayer Percepton (MLP) and the Long- Short Term Memory (LSTM). Before performing the classification models, the dataset is split into 70% and 30% using the train_test_split() function from scikit-learn. From there, the portion of 70% will be chosen for model training, meanwhile the remaining 30% is testing data which are used to evaluate the trained models. Hyperparameter tuning is performed using a grid search method to find the right combination of hyperparameters that improves the performance of models. The models will be trained on two cases with different feature engineering methods, namely label encoding and counting the evidence approaches.

In this dataset, the class samples suffer from imbalances for each malware family. Therefore, data sampling technique was applied to treat the imbalanced data, namely polynomial fitting Synthetic Minority Over-sampling Technique (SMOTE). By adding an optimal number of synthetic samples to minority data, this approach rebalances the training data using polynomial fitting functions, while marginalization of the minority class is avoided. To further improve the model performance on the evidence counting technique, an additional normalization step is performed, utilizing the Yeo-Johnson transformation, transforming the numeric variables to make the target variables more normally distributed. This reduces the skewness in the dataset.

3.5. Model Evaluation

For classification, four outcomes that could be obtained are the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), and these are used to determine the metrics to evaluate the classifiers, namely accuracy, precision, recall and F1 score.

Loss is used to evaluate deep learning models, while confusion matrix allows comparison between the models fairly. It is defined as the error of prediction of the neural network and is calculated based on how well the model performs on the interpretation of training and validation. In this experiment, the loss function used is Categorical Cross-Entropy (CCE), also known as Softmax loss. CCE is a combination of Softmax activation and a Cross-Entropy loss that allows each node to output a probability value between 0–1.

$$\mathrm{BinaryCrossEntropy}_n = -\left(P_n(X)\log q_n(x) + (1 - P_n(x))\log(1 - q_n(x))\right) \quad (1)$$

4. Results and Discussions

4.1. Label Encoding Approach

Based on the distribution of model performances in Table 3, RF tops the list at an accuracy of 91.34%, followed by MLP and DT at around 88%. In this approach, K-NN classifier is the one with the lowest accuracy, albeit at 82.05%. By checking the other metrics, the models learn the data well, and are able to describe the data while tree-based models such as RF and DT generally work better with labelled data, even better than LSTM in this case. This could be explained by the properties of tree-based models being deterministic, as opposed to neural networks to being probabilistic. Tree-based models explicitly fit the parameters to direct the information flow, being to the opposite of neural networks that fit parameters to transform the input and indirectly direct the activations of following neurons. Tree-based models tend to lean on simplifications. While observing the loss learning curve of the LSTM model, (Fig. 3), the LSTM model is slightly overfitting, with the training loss and validation loss plateauing over each other. At the end of the epoch, the validation loss is slightly higher than the training loss. Hence, the performance of the LSTM model is comparable with some of the other models, namely SVC and K-NN.

Table 3: Model performance (label encoding approach)

Model	Accuracy	Precision	Recall	F1-Score
RF	0.9134	0.9125	0.9182	0.9135
SVM	0.8250	0.8279	0.8161	0.8209
K-NN	0.8205	0.8134	0.8310	0.8178
DT	0.8801	0.8745	0.8750	0.8747
MLP	0.8810	0.8800	0.8956	0.8818
LSTM	0.8387	0.8217	0.8319	0.8248

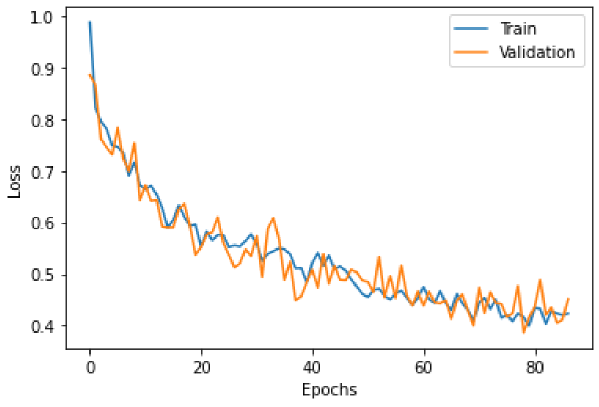


Fig. 1: Loss curve for LSTM (label encoding)

4.2. Evidence Counting Approach

Based on the results in Table 4, MLP, LSTM and SVM obtained the highest accuracies among all the models which are 91.46%, 94.64% and 91.46% respectively. The distribution of performance is different when the two techniques are compared. RF and DT, which are the tree-based models, have a slight disadvantage for evidence counting. This could be due to that tree-based models generally work better on labelled data. In all, the models tend to perform better with the evidence approach counting, compared to label encoding. The models also showed the efficacy of this feature engineering approach, especially observed in the improvements of both SVM and K-NN. Neural nets, namely MLP and LSTM in this case delivered the best performances.

No single model worked best across all possible scenarios. The normalized evidence count contributes to the training of the neural networks, as they tend to work better on the assumption of scaled features, within a similar range. However, from Fig. 2, the LSTM model is slightly overfitting, which is similar to the one in the label encoding experiment. The model's loss began to rise and drop from epoch 4 to epoch 10, and eventually dropped prior to the last epoch. To further measure the true performance of the model, Table 4 presents the other metrics. Nonetheless, LSTM still outperformed the other models.

Table 4: Model performance (evidence counting approach)

Model	Accuracy	Precision	Recall	F1-Score
RF	0.9085	0.9091	0.9198	0.9099
SVM	0.9146	0.9135	0.9265	0.9154
K-NN	0.8842	0.8798	0.8929	0.8836
DT	0.8675	0.8622	0.8649	0.8633
MLP	0.9146	0.9131	0.9261	0.9151
LSTM	0.9464	0.9152	0.9283	0.9173

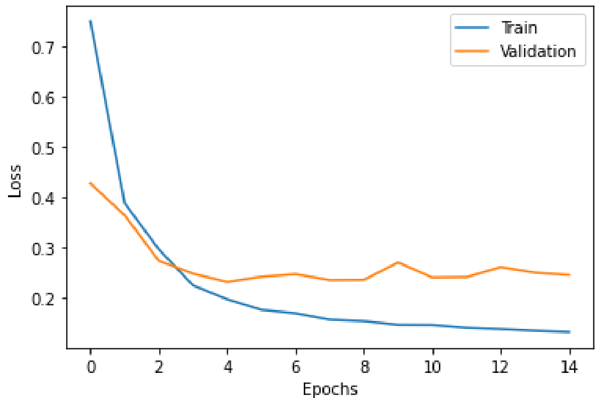


Fig. 2: Loss curve for LSTM (evidence counting)

5. Conclusions

This paper looks at two different feature engineering techniques, namely, label encoding and evidence counting for malware classification. In addition, feature selection was also performed to eliminate variables with less significance. Furthermore, oversampling using SMOTE was performed to overcome imbalance in the dataset. Models, namely RF, DT, K-NN and SVM, MLP and LSTM are constructed and compared between the two feature engineering approaches. From the findings and discussions, tree-based models such as RF and DT performed well on labelled data, and their performances are generally better in the label encoding approach. On the other hand, SVM and K-NN tend to not cope very well with labelled data in this study. The state-of-the-art deep learning approaches in this study have shown their potential in malware classification, with further improvements required in building a robust solution against solving complex real-world malware detention and classification problems.

References

Ang, J. C., Mirzal, A., Haron, H., & Hamed, H. N. A. (2015). Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5), 971–989

Barbarosoglu, G. & Pinhas, D. (1995). Capital rationing in the public sector using the analytic hierarchy process. *The Engineering Economist*, 40, 315-341

Bernama. (2021, Sep). Surge in email threats, malware detected in Malaysia in 2021, says Trend Micro. The Edge Markets (theedgemarkets.com)

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, 300, 70–79

Chowdhury, M., Rahman, A., & Islam, R. (2017). Malware analysis and detection using data mining and machine learning classification. In *International Conference on Applications and Techniques in Cyber Security and Intelligence,* 266–274

Dahl, G. E., Stokes, J. W., Deng, L., & Yu, D. (2013). Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 3422–3426

Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys* (*CSUR*), 44(2), 1–42

Firdausi, I., Erwin, A., Nugroho, A. S., (2010). Analysis of machine learning techniques used in behavior-based malware detection. In *2010 Second International Conference On Advances in Computing, Control, and Telecommunication Technologies,* 201-203

Gadhiya, S., Bhavsar, K., & Student, P. (2013). Techniques for malware analysis. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 2277–128

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153, 102526

Kumar, V. & Minz, S. (2014). Feature selection: A literature review. *SmartCR*, 4(3), 211–229

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys* (*CSUR*), 50(6), 1–45

Masabo, E. (2019). A feature engineering approach for classification and detection of polymorphic malware using machine learning. Unpublished doctoral dissertation, Makerere University

Masabo, E., Kaawaase, K. S., & Sansa-Otim, J. (2018). Big data: deep learning for detecting malware. In *2018 IEEE/ACM Symposium on Software Engineering in Africa (SEIA)*, 20–26

Narayanan, B. N., Djaneye-Boundjou, O., & Kebede, T. M. (2016). Performance analysis of machine learning and pattern recognition algorithms for malware classification. In *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, 338–342

Nari, S. & Ghorbani, A. A. (2013). Automated malware classification based on network behavior. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, 642-647

Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era - a state of the art survey. *ACM Computing Surveys* (*CSUR*), 52(5), 1–48

Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015). Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1916-1920

Ramilli, M. (2016). Malware training sets: a machine learning dataset for everyone. Retrieved from https://marcoramilli.com/2016/12/16/malware-training-sets-a-machine-learning-dataset-for-everyone/ [Online; December 2016]

Rathore, H., Agarwal, S., Sahay, S. K., & Sewak, M. (2018). Malware detection using machine learning and deep learning. *In International Conference on Big Data Analytics*, 402–411

Selamat, N., & Ali, F. (2019). Comparison of malware detection techniques using machine learning algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, 16, 435

Sihwail, R, Omar, K., & Arffin, K. A. Z. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *International Journal on Advanced Science, Engineering and Information Technology*, 8(4-2)