Spring 2023

Application of Adversarial Attacks on Malware Detection Models

Vaishnavi Nagireddy
San Jose State University

Writing Project

Application of Adversarial Attacks on Malware Detection Models

Final Report

Author

Vaishnavi Nagireddy

CS 298

05/16/2023

Advisor

Fabio Di Troia

A Writing Project Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment of the Requirements for the Degree: Master of Science

The Designated Committee Approves the Master's Project Titled

Application of Adversarial Attacks on Malware Detection Models

By

Vaishnavi Nagireddy

Approved for the Department of Computer Science

San José State University

May 2023

Fabio Di Troia Date
Department of Computer Science

Nada Attar Date
Department of Computer Science

Genya Ishigaki Date
Department of Computer Science

Acknowledgements

I would like to express my gratitude to my project advisor, Fabio Di Troia, for his support, expertise, guidance and encouragement throughout the project. I am also grateful to my defense committee members, Nada Attar and Genya Ishigaki, for their time, inputs and suggestions. Finally, I would like to thank my family members and friends for their unconditional love and support.

Abstract

Malware detection is vital as it ensures that a computer is safe from any kind of malicious software that puts users at risk. Too many variants of these malicious software are being introduced everyday at increased speed. Thus, to guarantee security of computer systems, huge advancements in the field of malware detection are made and one such approach is to use machine learning for malware detection. Even though machine learning is very powerful, it is prone to adversarial attacks. In this project, we will try to apply adversarial attacks on malware detection models. To perform these attacks, fake samples that are generated using Generative Adversarial Networks (GAN) algorithm are used and these fake malware data along with the actual data is given to a machine learning model for malware detection. Here, we will also be experimenting with the percentage of fake malware samples to be considered and observe the behavior of the model according to the given input. The novelty of this project is given by the use of adversarial samples that are generated by the implementation of word embeddings produced by our generative algorithms.

Index Terms – **Malware Detection, Adversarial Attacks, Machine Learning, Fake Malware Generation.**

TABLE OF CONTENTS

1 Problem Statement

The advancements in technology made software devices available to everyone. As there is expeditious improvement in the field of information technology, usage of these software devices has become inevitable. It is crucial to secure computer systems as they are prone to cyberattacks, which have the potential to cause significant damage such as data robbery, financial distress, system failure and so on. Malicious software, which is otherwise known for the term called malware, is designed to perform malicious activities like stealing sensitive information, disrupting normal operations and even demand ransom from victims in order to restore access to their systems. Therefore, to avoid these types of activities and to safeguard computer systems, it is necessary to come up with malware prevention techniques. A traditional method to perform malware detection was through signature scanning as it could successfully detect different types of malware but it was futile when novel or variants of existing malware were encountered [1]. Another common method used to detect malware was by performing behavioral based techniques but this technique yields a significant number of false positives. When faced with sophisticated types of malware, both of these conventional methods for detecting malware prove ineffective [2]. Hence, there was a need to explore other techniques for malware detection and this is when machine learning for spyware recognition emerged. Machine Learning (ML), which is a part of Artificial Intelligence (AI) has made its way towards cyber security by detecting malware files and helps to keep the computer systems safe. Now-a-days, ML is being widely used by profit-oriented organizations for anti-malware applications. X. Sun et al. and H. Zhang et al. in [3] and [4] respectively, have conducted a literature review on different machine learning approaches involving supervised and unsupervised algorithms.

Advanced Machine Learning engine (AML) was developed by Symantec to classify a file as benign or malware by analyzing the nature of file [5] but identifying a zero day malware has been a challenging task because as per the survey conducted by McAfee, around 60 million novel malicious software samples were generated only in the span of three months[6]. Application of machine learning for detecting zero day malware was illustrated by TrendMicro [7]. Recent investigations witnessed the fact that both ML and Deep Learning (DL) techniques are highly effective in detecting latest and unprecedented malware. This is due to their ability to analyze vast amounts of data and identify patterns that might be difficult for humans to detect. However, despite their success, these techniques are not infallible. One of the biggest challenges that both ML and DL models are facing is the threat of adversarial attacks. These attacks involve malicious actors who deliberately modify legitimate inputs to trick the models into making incorrect predictions. Adversarial examples can be created by slightly and carefully perturbing inputs that cause the model to misclassify them.

Adversarial attacks were studied primarily in the field of computer vision, specifically in the context of image classification. Nevertheless, these attacks have since been found to be applicable to other domains as well, including Natural Language Processing (NLP), audio recognition and even malware detection. These adversarial attacks can be highly effective and can have serious consequences, particularly in the context of cybersecurity as they can be used to bypass security measures and gain unauthorized access to sensitive systems and data. Numerous researches on different methods like filtering out adversarial samples and techniques to strengthen ML and DL models against these attacks are being conducted to develop new techniques to detect and mitigate the risk of adversarial attacks. As the field of machine learning and deep learning continues to evolve, it is likely that new approaches will be developed to address these critical security issues. In this project, we will explore problems that cause threats to systems and techniques to overcome these problems in the field of cybersecurity. We will focus on different machine learning models and neural network methods for malware detection with and without adversarial samples. By this means, we address the questions: *How accurate the machine learning models are in identifying malicious data? How accurate the machine learning models are in identifying the families of fake malware data? How adversarial data can affect the performance of the machine learning models? How are bert embeddings useful for malware detection? Which machine learning model can be helpful for malware detection when the data is in the form of bert embeddings?*

In this work, we will explore different machine learning models and neural networks for malware classification with no adversarial samples and compare the performance of these models when increasing percentages of adversarial samples are given to these models.

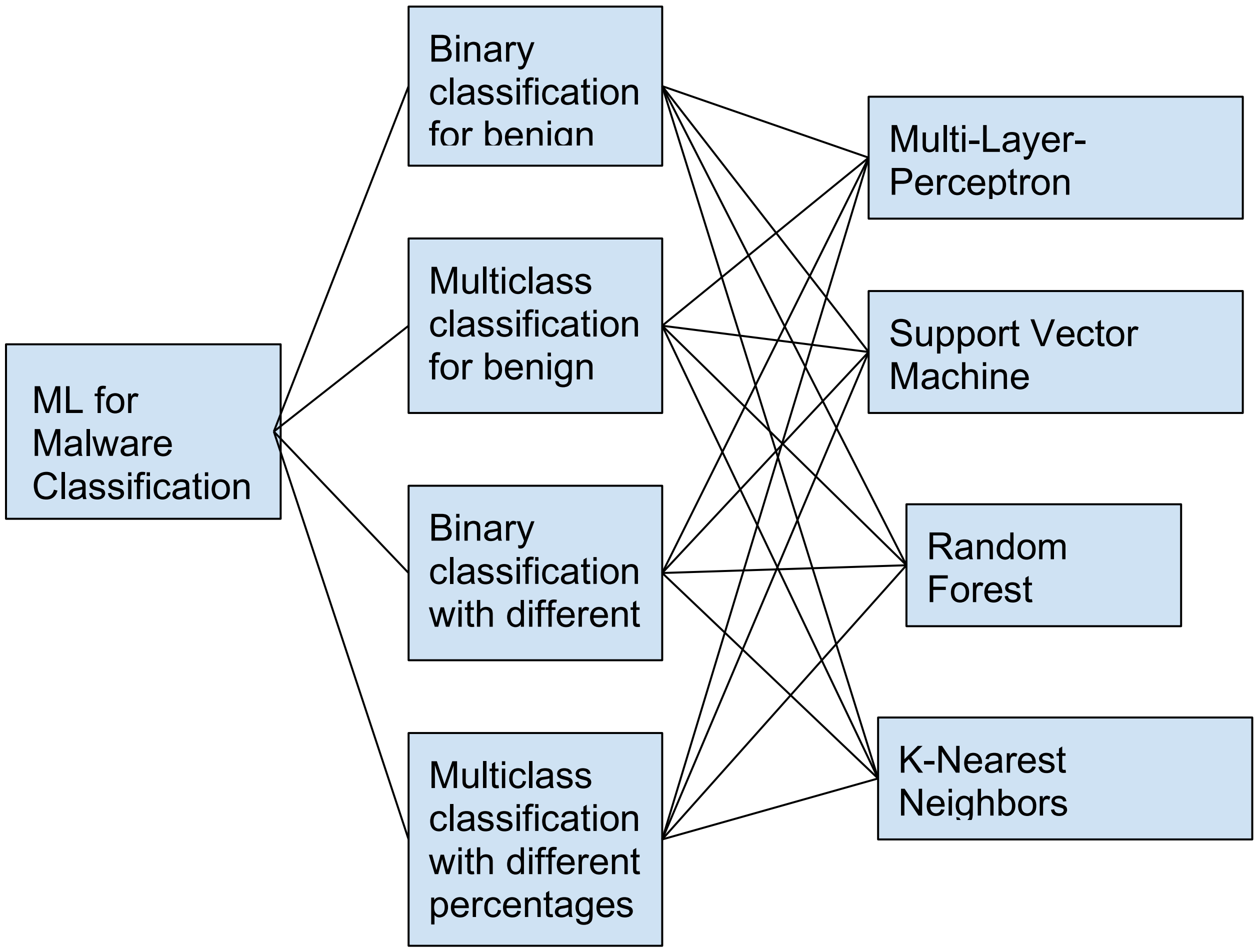

ML for Malware Classification
Binary classification for benign
Multiclass classification for benign
Binary classification with different
Multiclass classification with different percentages
Multi-Layer-Perceptron
Support Vector Machine
Random Forest
K-Nearest Neighbors

Fig. 1 Overview of the Project

2 Related work

In this section, we will explore some of the aspects that need to be considered when it comes to malware detection. Before diving into malware detection using different machine learning techniques, we shall look into the traditional methods used for malware detection, Emergence of machine learning for malware detection, Feature engineering for ML models and the vulnerabilities of these machine learning models caused by adversarial attacks.

2.1 Malware Detection

Before the emergence of advanced models for malware detection, signature based detection was the most widely used method. This method involved comparing the incoming file or data packets with a database of known malware signatures to identify known threats. However, this method had limitations in detecting new and unknown malware. As a result, heuristic-based detection was introduced, which involved identifying suspicious behavior that could indicate the presence of malware. This method could detect unknown threats, but it also had the potential to generate false positives. Behavior-based detection, which is another method focused on monitoring the system's behavior to identify any abnormal activity that senses the existence of malicious software. Sandboxing involved running a program in an isolated environment to observe its behavior without infecting the system. Detection of malicious activity was also done by Intrusion Detection Systems (IDS) through monitoring the network traffic [8].

However, these traditional methods had limitations, including high false positive rates and inability to detect the latest and sophisticated malware threats. Therefore, this behavior encouraged the researchers to explore machine learning for malware detection as they are capable of analyzing substantial amounts of data and could identify patterns. According to the survey conducted by S.A. Aljawarneh et al. and S. Pandey et al. the first ML-based malware detection was introduced in 1995, using a neural network to detect viruses. Since then, various ML models have been developed for malware detection, including decision trees, Support Vector Machines (SVM), Naive Bayes, K-Nearest Neighbors (KNN) and deep learning [9][10].

Decision trees are used to classify data by partitioning it into smaller subsets based on certain attributes whereas Support vector machines are used to classify data identifying a hyperplane that separates the data into different classes. On the other hand, Naive Bayes is a probabilistic method that assesses the probability of a particular class by considering the existence of specific features and KNN algorithm classifies data based on the majority of its k-nearest neighbors. Algorithms such as Convolution Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) which are from deep learning are also used for malware detection as they are capable of learning hierarchical representations of data and analyzing sequential data [9, 11]. S. Aljawarneh et al. in [12] stated that among different approaches for malware detection, deep learning models have shown promising results in malware detection due to their ability to automatically learn features from raw data and detect previously known threats. CNNs have been used to detect malware in binary files by analyzing their byte-level representation, while RNNs have been used to detect malware in network traffic by analyzing its temporal patterns [12]. Additionally, deep learning models have been used in conjunction with other techniques, such as static analysis and dynamic analysis, to improve the accuracy of malware detection [12, 13].

Therefore, machine learning approaches have shown great potential in malware detection with various techniques being applied in research and industry. The choice of approach depends on various factors like the availability of data, desired accuracy and the availability of computational resources. Deep learning models have shown promising results in detecting previous unknown malware but they also have certain limitations such as their need for large amounts of labeled data and their vulnerability to adversarial attacks. Other machine learning approaches are also quite effective but they have their unique set of advantages and drawbacks [9, 13].

2.2 Feature Engineering

Feature engineering plays a crucial role in malware detection as it involves selecting and extracting the most relevant and discriminative features from the data. Researchers have used various techniques for feature selection and extraction, such as manual feature selection, static analysis, dynamic analysis and hybrid approaches [14, 15]. Manual feature selection involves hand-crafting a set of features that are likely to be useful for malware detection. In static analysis, static features such as opcode sequences, readable strings, control flow graphs and similar features are extracted from malware files. Therefore, in this type of analysis, features are extracted without running the program. These features can be considered solitarily or can be combined with other features for malicious software detection. M. Christodorescu et al. in [16] have performed malware detection based on static analysis where control flow graphs are used and were able to achieve better accuracy. M. K. Shankarpani et al. in [17] have considered both opcode sequences and API call sequences to find if a part of code is similar to any malware. On the other hand, Dynamic analysis requires extraction of dynamic features such as API calls, system resource status, system calls, memory writes and so on by executing the program in environments like sandbox or virtual machines. Models were built by C. Kolbitsch et. al. in [18] to detain malware behavior using system calls. Malware detection is performed in [19] using API sequences and in [20] the same is performed by analyzing the frequencies of API call sequences. Both static and dynamic analysis when combined together give rise to more efficient approaches and this hybrid techniques when used appropriately could lead to more accurate results.

Advancements in the machine learning field gave us machine learning algorithms for feature engineering and these techniques can also be used for selecting features in order to detect malware. Principal component analysis (PCA), recursive feature elimination (RFE), and genetic algorithms can be used for feature selection. PCA transforms the original set of features into a smaller set of uncorrelated features that capture most of the variability in the dataset. Whereas, RFE recursively removes the least significant features from the dataset until the desired number of features is obtained. Finally, genetic algorithms use an evolutionary approach to select the most relevant features based on a fitness function that measures the performance of the classifier. PCA, which is one of the machine learning approaches, is used in [21, 22].

2.3 Adversarial Attacks on Machine Learning Models

Machine learning and deep learning techniques have been proven to be very useful in different fields like cyber security, natural language processing, computer vision, speech recognition and many more, but these models were proven to be vulnerable to adversarial attacks. These attacks impact the accuracy of the models and make them less reliable, which is why these attacks are significantly concerning. Inputs which are carefully crafted are introduced to the ML models and this leads to inaccurate predictions. Therefore, it is very crucial to identify these adversarial examples to build reliable models.

Many researches have been performed and some techniques like inclusion of adversarial training and detection methods came to light to mitigate the impact of adversarial attacks performed on machine learning models [23, 24]. Extensive study on adversarial attacks on ML models were performed over the past few years and work done by Grosse et. al. in [25] was a breakthrough as he illustrates that adversarial samples can be generated to bypass malware detection models based on machine learning. To prove this, Grosse et. al. have used adversarial examples to trick malware detectors based on signature and machine learning. Kolosnjaji et. al. in [26] have performed research and came up with a technique called "black-box adversarial attack" which tricks malware detection models that use behavior analysis. In this work, adversarial samples were created using reinforcement learning to exhibit how the adversarial samples even when considered in very small amounts can affect the performance of the model. Feinman et. al. in [27] conducted a research on visualization of decision boundaries of machine learning model for detection of adversarial samples and showcased that this technique can detect samples generated from state-of-the-art attacking techniques. Many researches have been conducted and are still going on to counteract the impact of adversarial attacks and build more reliable models.

3 Malware Classification using Machine Learning Techniques

3.1 Artificial Neural Networks (ANN)

Neural Networks, which are popularly known as Artificial Neural Networks(ANNs) and are also known by the name Simulated Neural Networks (SNNs) belong to machine learning and are very essential to deep learning which is a member of machine learning. These networks got their name from the biological neural networks of the human brain and are evolved to be performed in the way the brain of human beings would by sending and receiving signals from neurons. Unlike human brains, these artificial neural networks process numeric data and to process information which are in the form of speech, images and texts, different neural networks like CNNs and RNNs are used. ANNs have multiple layers where each layer contains multiple nodes, which are otherwise called neurons. The neurons in each layer are interconnected with neurons in the next layer and these neurons are used to process the information given to the network. The layers of the neural network are categorized into input, hidden and output layers. Input layer, which is the first layer of the network, receives the information that needs to be processed and passes this information to hidden layers of the network along with weights. On the other hand, the output layer, which is the last layer of the network, is where the predictions are produced. There can be as many as hidden layers between input and output layers and these hidden layers perform complex computations which aid in processing the input data.

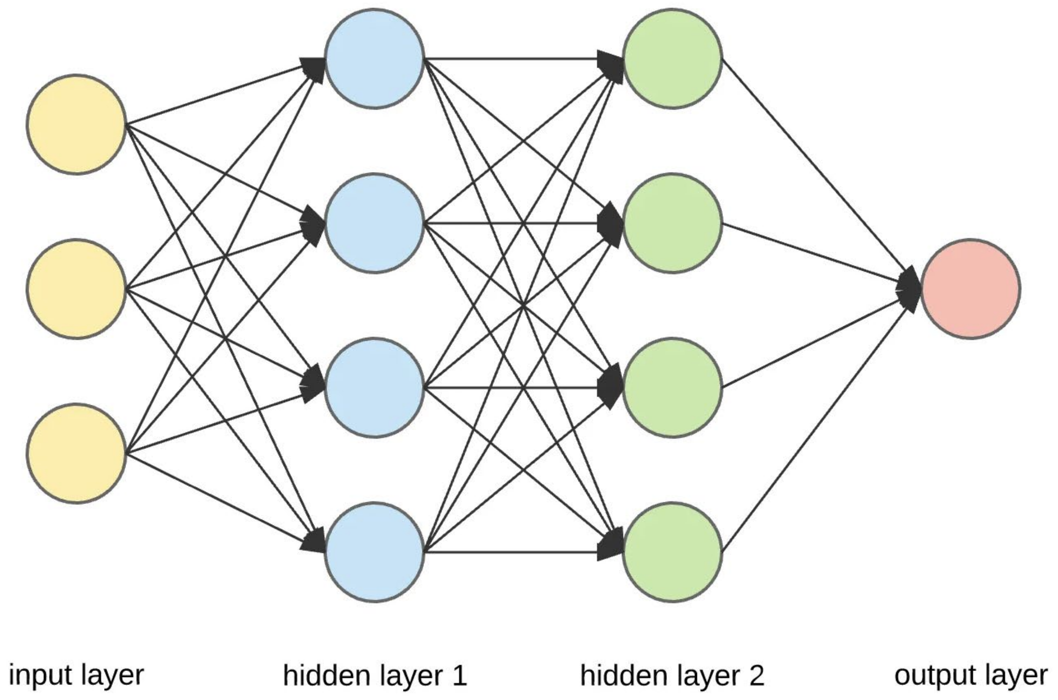


Fig. 2 Neural Network [28]

ANNs are used to perform a variety of tasks such as solving regression, prediction and classification problems by learning complex relationships between input data and output data. These networks are trained by presenting a set of labeled data as input and the weights of connection between neurons are adjusted in a way that the difference between the actual output and the predicted output is minimum. Generally, the training in artificial neural networks is performed in two phases called Forward Propagation and Backwards Propagation. In forward propagation, the input data is multiplied by their corresponding weight and this product is added to a bias. This weighted sum is then given to the activation function to add non-linearity. The output obtained from the activation function is then given as input to the neuron in the next layer and this procedure continues till the output is obtained.

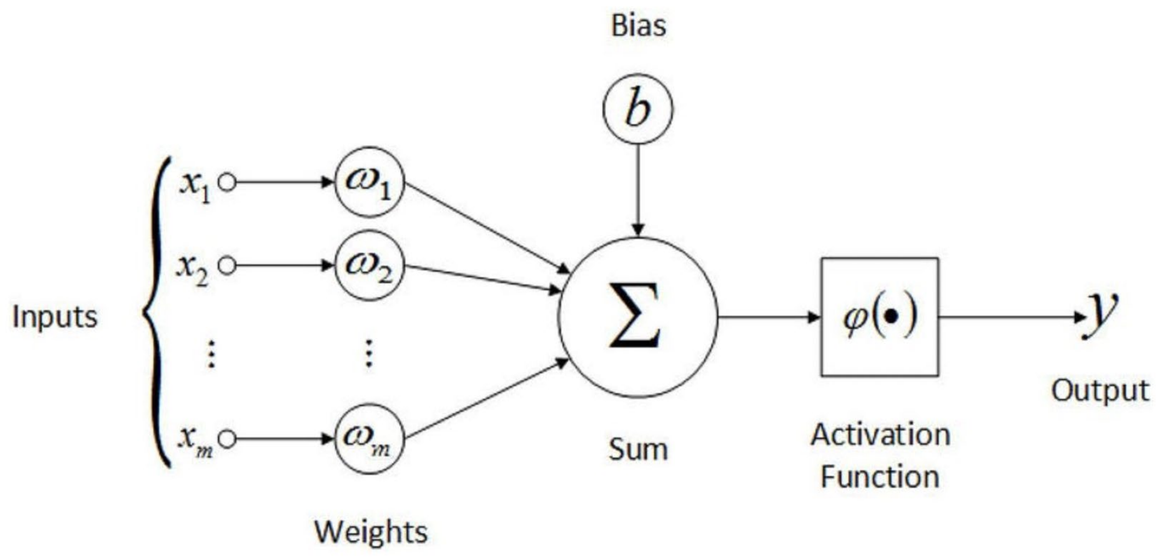


Fig. 3 An Artificial Neuron [29]

Once the output is generated, it is compared with the expected output from the training data and error is calculated using loss function and then backward propagation is performed. This is a preferred method to adjust the bias and weights as the rate of convergence is higher since the data is passed backwards from the output layer to the hidden layer. This procedure is performed until minimal loss is obtained. After training is completed, the model can be used to make predictions on new data.

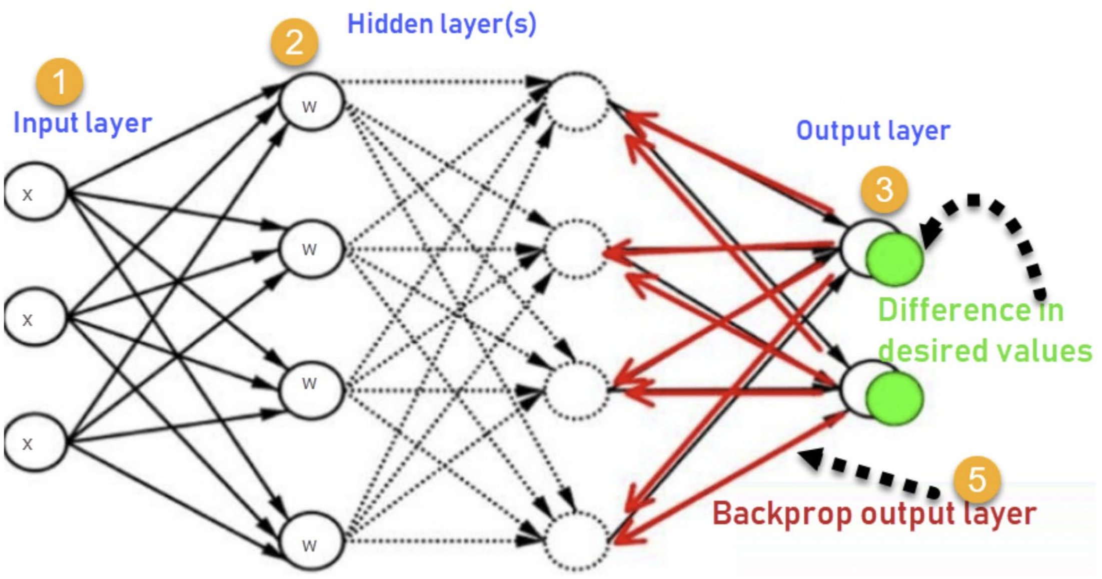


Fig. 4 Backpropagation in neural networks [30]

Non-linear activation functions are used in neurons to add nonlinearity to the neural networks and these functions transform the weighted sum of inputs as the output of that neuron. Hence, complex relations between input and output data is learned by the network. Some of the most common action functions used to add nonlinearity are Sigmoid, Softmax and Rectified Linear Unit (ReLU). The performance of the model can be impacted by the choice of activation function. Therefore, it is very important to decide on the activation function as different functions work better with respect to different types of data.

3.1.1 Sigmoid Function

Sigmoid function is one of the popular non-linear activation functions which can be viewed as a step function but with smooth transition which results in different gradients at different points. This function takes real-values as inputs and compresses them into the range of [0, 1] and is defined as

$$f(x) = \frac{1}{1+e^{-x}}$$

Where x is the input given to the function and e is Euler's number, which is a mathematical constant. Most of the time, this function is used in the output layer and is often used for binary classification where the model tries to predict if a particular input belongs to the majority or minority class. This function outputs the probability of given input falling into the positive class. In the context of malware detection, this activation function gives the probability of the input being malicious.

3.1.2 ReLU Function

The Rectified Linear Unit function is another commonly used activation function in neural networks for introducing non-linearity, which aids in learning complex patterns in data. This function is defined as $f(x) = max(0, x)$ where x is the input given to the function. It is much simpler than the sigmoid activation function and it is a popular choice in artificial neural networks as it is rare for the output to saturate at either ends of the range. The efficiency of the neural network is improved when ReLU activation function is used as this function is easy to evaluate, so it reduces the number of computations and calculating its derivative can be done either analytically or numerically. Hence, it is easy to integrate them into backpropagation. Despite its advantages, it comes with a limitation as it comes with a problem called "dying ReLU". This problem occurs when the inputs fall below zero as this does not trigger the neurons in the network. This can be caused either while training if the learning rate is considered to be too high or if the weights are not initialized appropriately. Leaky ReLU, Exponential Linear Unit (ELU) are the variants of ReLU which were introduced to solve this problem.

3.1.3 Softmax Function

Softmax activation function is particularly used in the output layer of artificial neural networks to accomplish tasks belonging to multi-class classification. A vector of real numbers is taken as input and normalizes these inputs into probability distributions, which adds up to one. The softmax function is defined as

$$\frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

Fig. 5 Formula for Softmax activation function [31]

Where n is the length of the input vector and x_i is the ith element of the input vector. Using this activation function the input vector is mapped to the probability distribution over n classes. Therefore, each and every element of the output vector indicates the probability of input falling into that particular class. If the probability is higher for any class then that class is considered as the predicted output. Softmax function is at advantage as it guarantees that the sum of probabilities in the output vector is always equal to one and this is crucial as predictions are performed based on these probabilities. The outputs of the neural network are converted into class probabilities in an efficient and differentiable way and this aids in training the neural network by using backpropagation. Softmax function can cause numerical instability when large values are given as inputs. So, to address this issue, variants such as LogSoftmax function and MaxSoftmax function have been introduced. These variants consider maximum elements rather than using sum of elements to normalize the input vector.

3.2 Support Vector Machine (SVM)

Support Vector Machines come from the family of supervised machine learning and are one of the highly preferred techniques as the requirement for computational power is less. SVMs are useful to tackle both classification and regression problems but are widely known for solving classification tasks as this method has the capability to analyze data and perform pattern recognition. These patterns are then used to categorize the input data to the corresponding class. SVMs are at advantage when compared with other classification algorithms as they have the ability to handle high dimensional data, work with non-linear data and the size of the dataset does not have to be very large.

SVM algorithm aims in finding a hyperplane in an n-dimensional space (where n stands for the number of features) that separates all the data points in a way that the data belonging to different classes are segregated accordingly in the feature space. This algorithm tries to find the best ground where the margin between the nearest datapoint of each class and the hydroplane is maximized. Reinforcement is provided by maximizing the margin distance as the model can perform better even with new data and hence this increases the performance of the model. Mapping of the original feature space to a higher dimensional space can also be done by SVMs using kernel functions and doing this could lead to a better separation boundary which aids in better accuracy of the model.

Applications such as image classification, text classification and so on use SVM. This technique is also used in fields like bioinformatics for gene classification and many more. In the context of cybersecurity, SVMs are well known for their performance in malware detection as they yield in low false positive rates and higher accuracies.

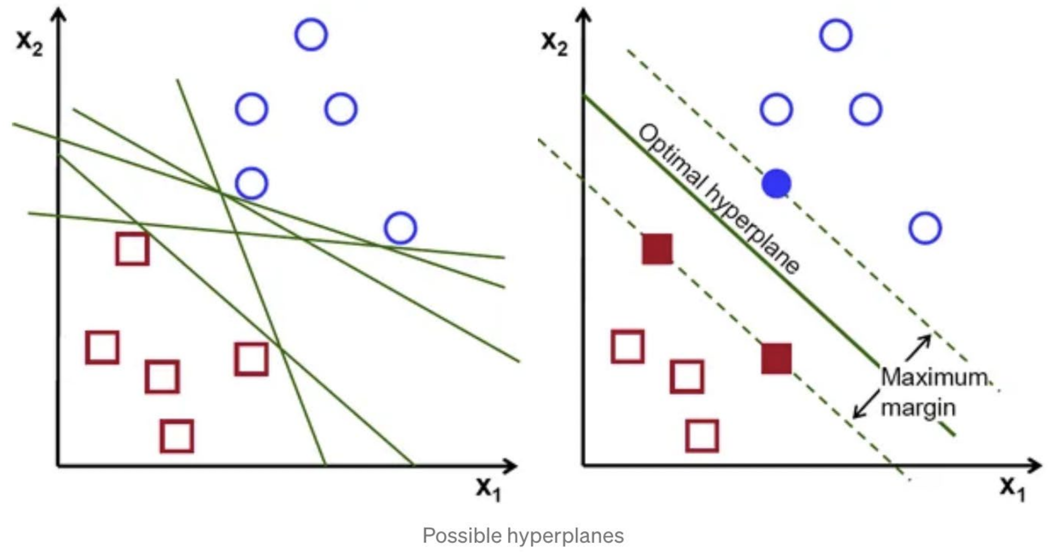


Fig. 6 Possible Hydroplanes - SVM[32]

3.3 Random Forest

Random Forest, being a member of supervised learning techniques, is used to solve problems of both classification and regression. It is known as an ensemble method as multiple decision trees are combined inorder to make a prediction. Subsets of features from the input data that belong to the training split are selected randomly in order to construct decision trees in the forest. High dimensional data can be handled with ease and overfitting can also be avoided by constructing multiple trees in this manner.

Based on the selected features, prediction of a class is performed by every single tree in the forest. Once each tree has made its decision, voting is performed to conclude the prediction and each and every single tree in the forest should participate in this process. The predicted class with the highest votes is concluded as the final decision. This way of electing the predicted class helps to improve the performance of the model and also reduces noise. This algorithm estimates important features which aid in identifying the most important features for classification. Random Forest is a powerful algorithm which is being used in a wide range of applications like image, text classification and is also being used for malware classification. Despite its advantages, this algorithm is sensitive to the hyperparameters such as number of trees in forest, maximum depth of trees that are used for building the model. From the description of the model, it is evident that this algorithm is computationally expensive when huge datasets are considered.

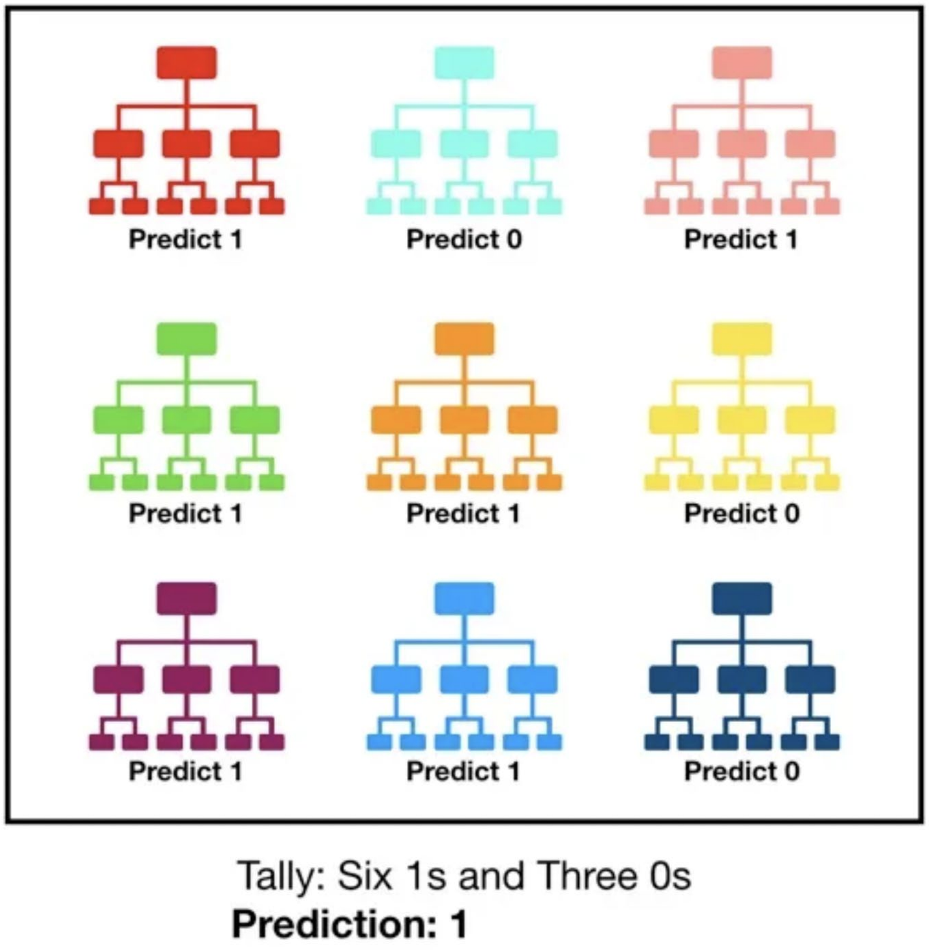


Fig. 7 Visualization of Random Forest Model[33]

3.4 K - Nearest Neighbors

K Nearest Neighbors (KNN) algorithm is also a supervised learning technique which can be used to solve classification and regression problems. This algorithm does not encourage making assumptions on the underlying data distribution and hence it is said to be a non-pragmatic algorithm. The prediction of a new datapoint belonging to a particular class is based on the class that its k nearest neighbors belong to. Here, k is the hyperparameter that needs to be tuned before training is performed. In this algorithm, classification of a new datapoint is done by calculating the distance between the new data point and all the training data points; k nearest neighbors with shortest distance to the new data point are selected and class to which this new data point belong to is based on the majority class of its k nearest neighbors. The average value of the k nearest neighbors will be the predicted value of the new data point in case of regression problem. The simplicity of this algorithm is its advantage and hence it can be implemented with ease. Assumptions on the underlying data distribution are not performed by this algorithm and non-linear decision boundaries are handled easily. When the value of k is set to be large, it leads to underfitting and on the other hand when the k value is small, it leads to overfitting. Therefore, it is very crucial to come up with an appropriate value for k as this has the potential to impact the performance of the model. Despite it being a simple algorithm, it is computationally expensive when huge datasets are considered as the distance between all the training data points and the new data point will be calculated to assign a class to the new datapoint.

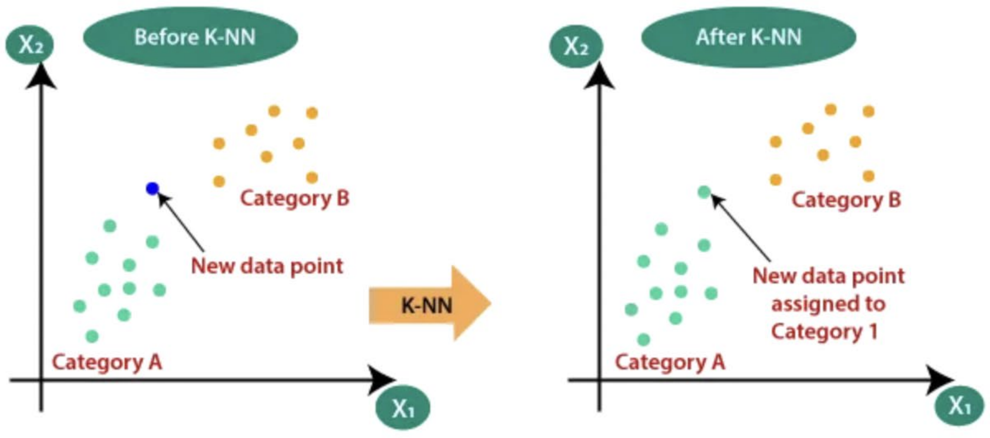


Fig. 6 Overview of KNN [34]

3.5 Classification of Malware using Deep Learning and Machine Learning

The process through which different malware types are categorized based on different attributes and their behavior is called malware classification. As the complexity and the magnitude of malware is being escalated over periods of time, traditional methods for malware detection and classification have become ineffective. To overcome this issue, plenty of research has been done and came up with machine learning and deep learning techniques, which have proven to be quite efficient in detecting malware and classifying them. These models are very efficient in learning and detecting patterns of malware based on their behavior and characteristics when trained with larger amounts of malware data. Moustafa et. al in [35] has conducted a study on malware classification using deep neural networks and achieved an accuracy of 99%. The model was trained on a large dataset of malware samples, and this helped in recognising the patterns of malware and to achieve such accuracy. A hybrid approach was proposed by Patel et. al in [36], which uses a combination of machine learning and data mining techniques to perform malware classification. This study also came up with surprising results with an accuracy of 98.73 percent. Using machine learning and deep learning for malware detection and classification is important as these models have the capability to identify novel and unknown malware, which are difficult to recognise by traditional signature-based approaches. These models also assist with developing effective defense mechanisms by providing insights on characteristics and behavior of various malware types. Usage of these machine learning and deep learning techniques for malware detection and classification have demonstrated higher accuracies and thus lowered the risk of malware attacks. With the increasing threat of malware, it is crucial to explore more techniques that help in malware detection and classification. In the below sections, we will look into malware classification performed by a few machine learning and deep learning techniques.

3.5.1 Malware Classification using Neural Networks

Multi-Layer Perceptron (MLP) is a very popular technique that deep learning has to offer to solve regression and classification tasks. This type of neural network contains multiple layers of nodes that are interconnected for processing input data and hence it has the multilayer perceptron. MLPs have been intensively used for performing tasks such as malware detection and classification of the malware according to their family. According to the study performed by Hans et. al. and Qiao et. al. in [37, 38], multilayer perceptrons have achieved good accuracy for malware detection and therefore demonstrates the effectiveness of using MLPs for malware detection and classification. For any deep learning model, selection of features plays an important role as the model learns the patterns using these features and will perform prediction. Similarly, the performance of the MLP model also depends on the input features. Several studies have been performed to explore the impact on these models when different sets of features are used for training. The authors in [39] have conducted a study by considering byte n-grams and opcode sequences as features to train MLP model and evaluate the performance of it.

Considering these features together, higher accuracy was obtained by the model. On the other hand, a similar study was conducted by S. Jang et. al. in [40], which includes API calls, opcode n-grams, and system calls as features. The accuracy in this case has increased significantly. Feature selection plays an important role in building robust malware detection systems as the choice of features impacts the performance of MLP models. Considering the same features does not always lead to better performance as some features perform better in some situations. Therefore, there is always a trade-off between computational complexity for feature extraction and accuracy of the model. The model's architecture can also influence the performance of the model. Therefore a thorough study on the architecture and features to be considered is very important to obtain better accuracy.

MLPs with certain features given as inputs are at advantage due to their ability to process complicated nonlinear relationships between features and targets, which is very essential for malware classification as the relationship between malware families and its features are nonlinear and complex. MLPs can process inputs with high-dimensionality and can be trained on huge datasets. Therefore, MLPs are suitable for malware classification and aids in improving computer systems security.

3.5.2 Malware Classification using SVM

SVM works on the principle of finding an optimal hydroplane that can be used to separate data points into different classes based on their features. SVMs have the capability to handle huge sets of features and thus avoids overfitting when compared with other machine learning algorithms. Many researches have been performed for usage of SVM for malware detection and classification and there has been enough proof that this model comes up with better accuracy when used for malware classification. Kumar et. al. in [41] have performed malware classification using opcode sequences as features and this model surprised everyone with a very good accuracy even with usage of obfuscation techniques. A similar study was conducted by the authors in [42] but instead of using opcodes sequences as features, they have considered API calls. Research to use support vector machines in order to classify trained Hidden Markov Models (HMMs) have been performed in [43]. Hence, support vector machines have been extensively used for malware identification and family type classification.

3.5.3 Malware Classification using Random Forest

Random Forest, which is said to be an ensemble learning method, because multiple decision trees will be used to make a prediction, can be one of the good fits to perform malware classification. This algorithm has the potential to handle many features and also captures the relation between the input features and their corresponding target classes, and this is very important to solve any classification problem. Many researches have been performed to get machine learning to a platform where problems from different fields can be solved and one such area is cybersecurity. Not all machine learning algorithms can be used to solve malware classification but Random Forest is one such method which can help in classification of malware. Gracia et. at. in [44] worked on multiclass classification of malware families using Random Forest. In this work binaries of malware files are converted into images and these images are given as input to the model and this model was effective in detecting malware as the accuracy obtained was around 96 percent. A hybrid decision model based on machine learning was discussed by the authors in [45], which is aimed to achieve a low false positive rate and high accuracy. This model from the above discussion is a combination of deep learning model with 12 hidden layers and random forest, is experimented to determine if a given file belongs to benign class or malware class. Hence, Random Forest is one of the most efficient algorithms used for malware classification as it can handle huge datasets, deals with data imbalances and also avoids overfitting of the model.

3.5.4 Malware Classification using K-Nearest Neighbors

KNN is popularly used for classification tasks and therefore it can be used to perform malware classification; a new malware sample is classified based on its similarity to samples of training data. A comparison on malware classification performed by deep learning and KNN is performed in the research conducted by authors in [46]. Another research is performed in [47] by Aparna Sunil Kale et. al., where opcode sequences are considered as features and feature engineering is performed by training HMMs and Word2Vec to obtain HMM2Vec and Word2Vec embeddings. The vectors are then given to different machine learning models including KNN. Increase in the size of data increases the computational complexity and this is a concerning drawback of this model and the value for k must be chosen wisely because KNN is sensitive to this hyperparameter as it impacts the performance of the model.

4 Dataset

Now, as we are aware of the different approaches that we will be using to conduct our experiments, let's explore the dataset that we will be used to perform experiments for generation of adversarial malware samples and to perform malware detection. Our experiments are performed on seven different malware families and benign samples. The malware data is taken from two different datasets. One is Malicia dataset [48] and the other is VirusShare dataset [49]. In the Malicia dataset, there are more than fifty different types of malware families and from this dataset, we have selected three different malware families namely, Winwebsec, Zbot and Zeroaccess as only these families have over a thousand samples per family. The remaining four families are taken from VirusShare, which is in fact a very large dataset that is about one hundred gigabytes and contains over one hundred and twenty thousand executable malware files. The four families that are selected from this dataset are BHO, OnLineGames, Renos and VBInject. We have considered these four families as there are over thirteen hundred samples per family.

Before moving on to further details, let us have a sneak peek on the description of the malware families that were selected for our experiments.

- BHO - This malware type presents a variety of dangerous behaviors programmed by the aggressor [50].
- OnLineGames - Login credentials of users are robbed by this type of malware and this also keeps track of keystroke activities [51].
- Renos - This malware can be awarded as a best actor as it convinces the user that the system is infected and pressurizes the user to pay more in order to remove malware that does not exist [52].

- VBInject - Obfuscated content is packed to hide other malwares and itself from being identified [53].
- Winwebsec - This is a trojan which fabricates concerns and claims that they have solutions to these problems and these solutions are projected as anti-virus softwares [54].
- Zbot - This malware is a trojan that robs highly sensitive information by breaching into systems with Microsoft Windows software [55]
- Zeroaccess - This is a trojan that exploits a system with windows software to corrupt them and also performs evil activities [56]
- Benign files are considered from the dataset available in [57]

Name of the Malware Family	Malware Type	Number of Data Points
BHO	Trojan	1405
OnLineGames	Password Stealer	1513
Renos	Trojan Downloader	1568
VBInject	Worm	2694
Winwebsec	Rogue	4360
Zbot	Information Stealer	2136
Zeroaccess	Corrupting Devices	1305

Fig. 7 Overview of Number of Samples [34]

5 Malware vs Benign Classification

In this section, we will look into the techniques that are used for data processing and conduct several experiments by inducing different percentages of adversarial samples to the training data and evaluate the performance of the model.

5.1 Data Pre-Processing

In order to get the data ready for using them in machine learning and deep learning algorithms, we need to perform some preprocessing. Quality of data will be improved and through this, machine learning and deep learning models achieve a better performance. According to the data we have, techniques such as data transformation and/or integration, data cleansing and so on are performed.

5.1.1 Data Collection and Malware Family Selection

Malware Data that we need to conduct our experiments has been collected from two different datasets, which are amongst the popular datasets that are used to perform various researches in the field of malware detection and cybersecurity. A pack of four families are taken from the VirusShare dataset where the data is in the form of executable files. The families that are considered from this dataset are BHO, OnLineGames, Renos and VBInject and we particularly choose these four families because there are an abundant number of samples per family in the dataset. The remaining three families, Winwedsec, Zbot and Zeroaccess are taken from Malicia dataset which contains the data in the form of opcode sequences. The reason why these three families are selected is the same as earlier. It is always encouraged to have as many data samples as possible for the models to perform better prediction. Non-malware samples are taken from a benign dataset, which holds a decent amount of samples. These benign samples are also in the form of opcode sequences. Finally, we have gathered the data but it is not ready to give as inputs for ML models.

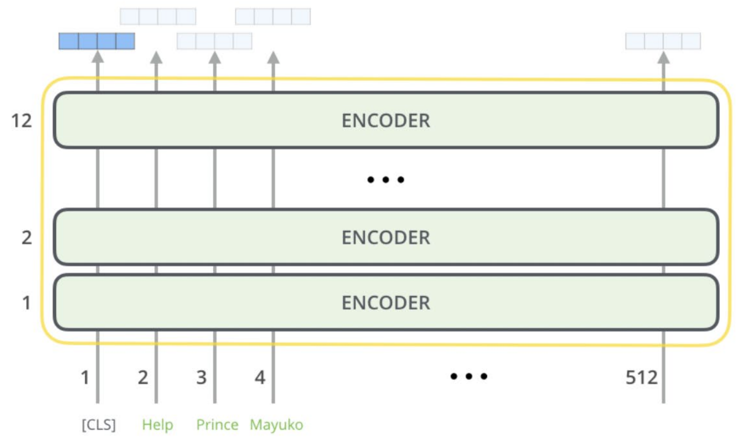
5.1.2 Abstraction of Opcode Sequences from Executable Files

The data we have considered is a mixed variety of executable files and opcode sequences. Hence, to get the data in similar format, we extract the opcode sequences of malware files that are in the form of executables. Before diving into the procedure to extract opcode sequences from malware files, let's have a small introduction to opcodes. Opcodes is an abbreviation of Operational Code and it does exactly the same. Commands like performing arithmetic calculations, moving data to different registers, and so on are given to a processor to perform tasks and opcodes are similar commands that are given to the processor to get a job done. A sequence of commands that are said to be executed are called opcode sequences. In order to get the opcode sequences, we have used objdump, which is a command-line utility that aids in disassembling executable files and gives the instructions in assembly language, and hence mnemonic opcodes are extracted. Once these are extracted, unnecessary information regarding addresses, registers and so on are eliminated using “sed” and “cut” commands, which also belong to command-line utilities.

5.1.3 Bert Embeddings of Opcode Sequences

In this project, we will be using Bidirectional Encoder Representation from Transformers (BERT), which is based on Natural Language Processing (NLP) techniques to get the word embeddings from the opcode sequences for both malware and benign samples. The reason for considering NLP for feature engineering is that these techniques generate word embeddings by extracting prosperous information from input text. Using these word embeddings, multiple tasks like paraphrasing a given sentence, identifying masked words in a sentence, and so on can be performed. In a phrase, the relationship of the word with every other word is captured and then akin words are clustered together in space with higher dimensionality.

BERT, which is one of the most popular models from the ground of natural language processing, has proven its effectiveness in solving strenuous language based problems like sentiment classification and prediction of masked words in a given sentence. BERT model provides better word embeddings as the context of each individual word is considered while generating these embeddings. Sequences of words are given as inputs for the BERT model and are propagated in upward direction as seen in the picture below. Self-attention is applied in each layer and results are passed to the neural network. After passing the results to the feed forward neural network, the output is then passed to the adjacent encoder. The output of the BERT model is a 768 length vector from each position.



In the research done by Kale et. al., in [47] mnemonic opcode sequences were subjected to natural language processing techniques, where different methods like BERT, Word2Vec, HMM2Vec and ELMo were used to generate word embeddings. In this research, it was proved that using NLP for feature extraction aids in better accuracy and amongst these models, usage of BERT embeddings for malware classification obtained the highest accuracy of 96 percent in comparison with other models. It is expensive to execute these huge pretrained NLP modes as they require huge computational capacity which is very expensive. Therefore, in this project, we have considered DistilBERT, which reduces the size of BERT model by 40 percent through the process called knowledge distillation that is performed at the phase of pretraining. It is effective as it retains over 97 percent of the abilities to understand the language and is faster by 60 percent [59].

5.1.4 Generation of Fake Malware Samples

Our experiments in this project are based on benign samples, actual malware samples and fake malware samples as we will be evaluating the performance of machine learning and deep learning models when adversarial attacks are performed. Generative Adversarial Networks (GANs) were first introduced in 2014 by Ian Goodfellow and his fellow researchers. New data samples that are similar to genuine data samples are generated by these GANs using generative modeling approaches. The architecture of GANs include a couple of highly competitive neural networks that can capture the data and perform an analysis on the variations in the data. These two networks of GANs are named as “Generator” and “Discriminator” and they do exactly the same.

In this project, fake malware samples that look very similar to actual malware samples are generated by using Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP), which is a variant of GAN. WGAN-GP is considered as it comes with a set of advantages when compared to its predecessor. Training process of the generator has become more stable when WGAN-GP is used and enlightening gradients are provided as Wasserstein distance is used. In traditional GANs, the discriminator is more powerful than the generator and this condition has been addressed when WGAN-GP is used. Over-powerful discriminator in GAN is an issue as this causes generation of inferior samples. Generation of fake malware samples were performed based on the research conducted in [60]. An overview of the set of parameters that have been used in WGAN-GP are presented in the table below:

Network	Parameters	Values
Generator	activation kernel size Conv1D layer filter 1 Conv1D layer filter 2 Conv1D layer filter 3 Conv1D padding BatchNorm momentum	TanH 3 64 32 16 same 0.8
Critic	activation kernel size Conv1D layer filter 1 Conv1D layer filter 2 Conv1D layer filter 3 Conv1D padding	LeakyReLU 3 64 128 256 same

Fig. 9 Parameters used for WGAN-GP [60]

From the Real malware executable files, opcode sequences are extracted and then these opcode sequences are given to a BERT model to get word embeddings. The word embeddings that are generated by BERT are given to WGAN-GP in order to generate fake malware samples. Below is the flowchart of the procedure for fake malware generation.

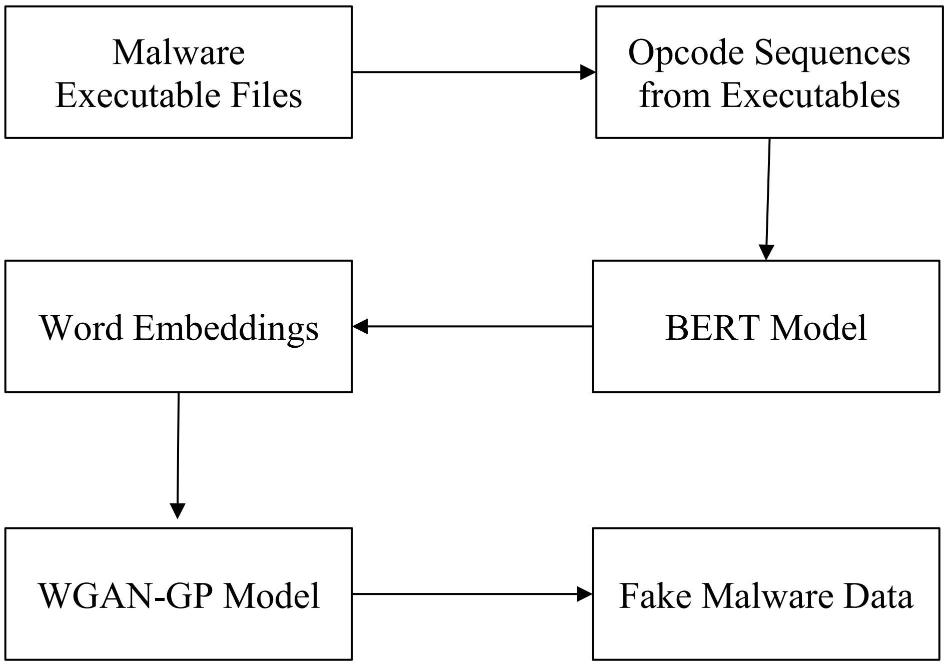


Fig. 10 Flowchart for Generating Fake Malware Data

5.2 Experiments

Now, we have the data ready to train deep learning and machine learning models. The experiments were conducted on Google Colaboratory Notebook using python programming language. Sklearn, Kersas, Tensorflow, Pandas, Numpy packages were used for data analysis and machine learning; Matplotlib is used for visualization. Experiments are conducted starting from benign vs malware classification and then later by considering different percentages of fake malware samples in the training dataset.

5.2.1 Benign vs Malware Classification with no Fake Samples

We have considered Multi Layer Perceptron, Support Vector Machine, K Nearest Neighbors and Random Forest to perform benign vs malware classification. The input data to these models are word embedding of benign samples and malware samples that are obtained from the BERT language. The word embedding for each sample is of length 768.

5.2.1.1 Architectural Setup for MLP Model and Results:

Multi Layer Perceptron model that we have considered in our experiments has the architecture that as mentioned below:

- Input Layer: One input layer with input size of 768
- Hidden Layer 1: Fully connected dense layer with 100 neurons and uses RELU as activation function
- Hidden Layer 2 : Fully connected dense layer with 200 neurons and uses RELU as activation function
- Hidden Layer 3: Fully connected dense layer with 300 neurons and uses RELU as activation function.
- Hidden Layer 4: Fully connected dense layer with 400 neurons and uses RELU as activation function
- Hidden Layer 5: Fully connected dense layer with 500 neurons and uses RELU as activation function
- Hidden Layer 6: Fully connected dense layer with 600 neurons and uses RELU as activation function
- Hidden Layer 7: Fully connected dense layer with 600 neurons and uses RELU as activation function

- Hidden Layer 8: Fully connected dense layer with 500 neurons and uses RELU as activation function
- Hidden Layer 9: Fully connected dense layer with 400 neurons and uses RELU as activation function
- Hidden Layer 10: Fully connected dense layer with 300 neurons and uses RELU as activation function
- Hidden Layer 11: Fully connected dense layer with 200 neurons and uses RELU as activation function
- Hidden Layer 12: Fully connected dense layer with 100 neurons and uses RELU as activation function
- Output Layer: Fully connected dense layer where the output size is 2 and used softmax as activation function.
- "Sparse_categorical_crossentropy" and "Adam" are used as loss function and optimizer respectively

Results for MLP**:**

- The model outperformed with an accuracy of 99% on testing data.
- Training, Testing and Validation splits are in the ratio of 70, 20, 10 respectively.
- The precision, recall and F1 score for the model are 0.993, 1.0 and 0.996 respectively

5.2.1.2 Support Vector Machine Setup and Results:

Hyper parameter tuning was performed to get the best parameters to train the SVM model. The parameters that were tuned are as follows:

- Kernel: RBF, Poly and Linear

- Gamma: 0.0001, 0.001, 0.01
- C: 0.5, 1.0

Amongst these parameters, best parameters were found using GridSearchCV and the best parameters were: C - 0.5, Gamma - 0.0001 and Kernel - Linear

Results for SVM:

SVM model gave best accuracy in classifying malware and benign samples. The accuracy obtained by this model is 99% . The precision, recall and F1 score for the model are 0.99, 1.0 and 0.99 respectively. SVM, has proven its place in malware classification in many previous researches. 70% and 30% of data is used for training and testing respectively.

5.2.1.3 Random Forest Model and Results:

Data preprocessing for the Random Forest model was the same as procedure considered for other machine learning models. Here 70% of data is used for training and 30% of data is used for testing. With the default parameters, this model gave an accuracy of 99%.

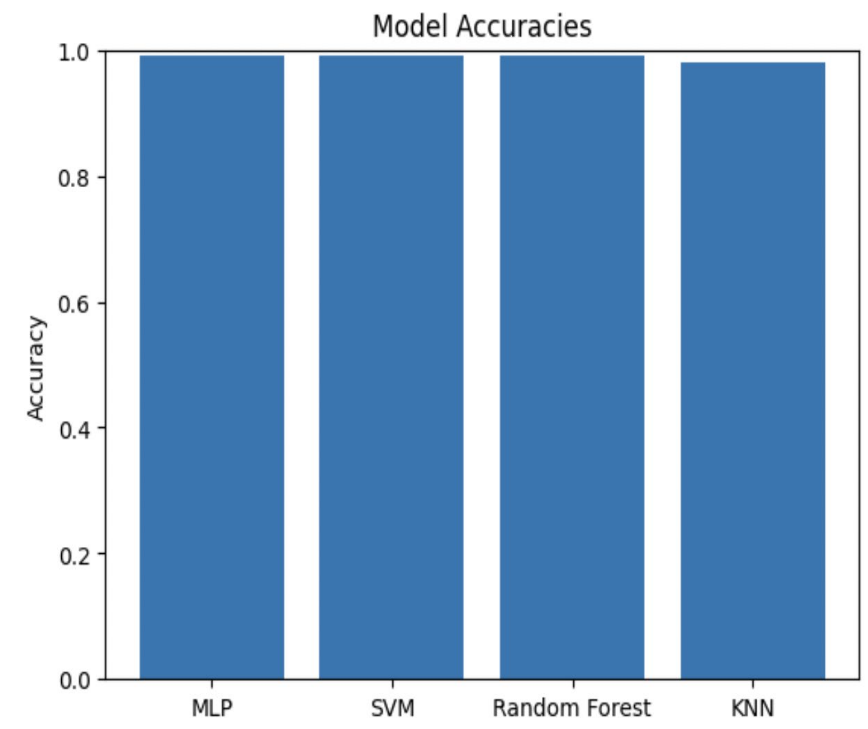
5.2.1.4 K Nearest Neighbors Model and Results:

The number of neighbors to be considered while making a prediction is tuned to the value 10 in this model. With this parameter the model performed really well and thus the accuracy for this model is 98%. The precision, recall and F1 score for this model are 0.98, 0.99 and 0.98 respectively. The training and testing split considered for this model is 70-30 split.

5.2.1.5 Observations and Accuracy Plot:

The accuracy obtained by the multi layer perceptron model and rest of the machine learning models were pretty high as we have considered only real malware samples and benign samples.

We have considered an equal number of malware and benign samples to avoid the problem of imbalanced data, which can be essential to obtain better results. There was ample data for the model to learn and make a prediction. Sometimes even better accuracies can be concerning as this takes us to the thought of overfitting. Regularization techniques can be used to avoid such problems. From the graph given below, it is clear that MLP, SVM and Random Forest classifiers have performed extremely well but KNN is no less. KNN's accuracy is just 1% less when compared to others.



5.2.2 Benign vs Malware Classification with 10% of Fake Malware Data

The data preprocessing is the same as explained in the above experiment and remains the same for the rest of the experiments except that the training data now consist of 10% of fake malware data. Initially 10% of actual malware data is removed from the training dataset after splitting the data into training, testing and validation splits. Now, exactly the same amount of fake malware data has been added to the training dataset and then the models were trained based on this data.

5.2.2.1 Architectural Setup for MLP Model and Results:

Not too many but model with different architectures were considered and the architecture of the model with best accuracy is given below:

- Input Layer: One input layer with input size of 768
- Hidden Layer 1: Fully connected dense layer with 256 units and uses RELU as activation function
- Hidden Layer 2: Fully connected dense layer with 128 units and uses RELU as activation function
- Hidden Layer 3: Fully connected dense layer with 64 units and uses RELU as activation function.
- Hidden Layer 4: Fully connected dense layer with 32 units and uses RELU as activation function
- Hidden Layer 5: Fully connected dense layer with 16 units and uses RELU as activation function
- Output Layer: Dense Fully Connected Layer with 2 units in and used Softmax activation function
- Sparse_categorical_crossentropy is used as loss function and "Adam" is used as optimizer.

Results for MLP:

This model gave an accuracy of 98.9%, which indicates that the model actually performed better even if 10% of fake malware samples were used for training.

5.2.2.2 Support Vector Machine Model and Results:

Hyper parameter tuning was performed to get the best parameters to train the SVM model. The parameters that were tuned are as follows:

- Kernel: RBF, Poly and Linear
- Gamma: 0.0001, 0.001, 0.01
- C: 0.5, 1.0

Amongst these parameters, best parameters were found using GridSearchCV and the best parameters were: C - 1.0, Gamma - 0.0001 and Kernel - Linear

Results for SVM:

The accuracy of the model with the best parameters obtained after hyperparameter tuning is 99.7%. The data used for training includes 10% of fake malware data and is induced after train-test split is performed. A 70-30 split is considered for training and testing this model.

5.2.2.3 Random Forest Model and Results

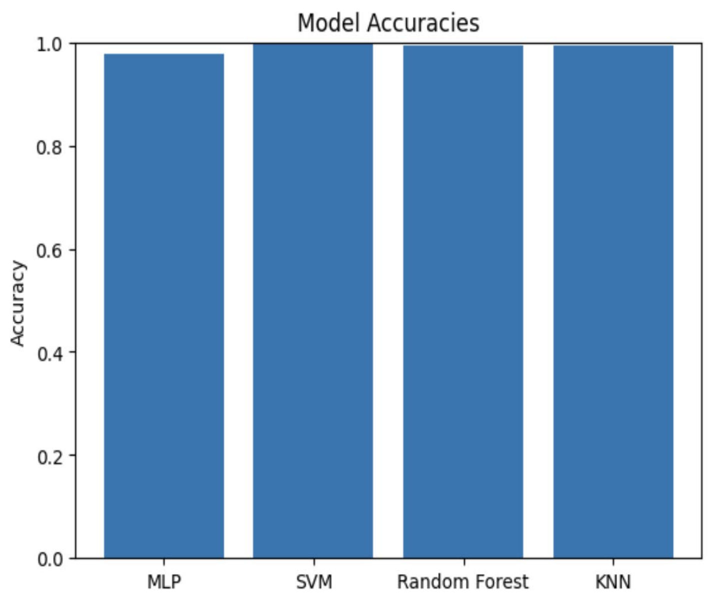
Different parameters can be considered to tune this model which leads to better accuracy but with the default parameters, this model has obtained an accuracy of 99.5%, which means that the model has performed a lot better even with the inclusion of fake malware data. A 70-30 split has been considered to perform training and testing respectively.

5.2.2.4 K Nearest Neighbors Model and Results:

The model setup for KNN is the same as the setup used in the previous experiment. The value for the number of neighbors to be considered for making a decision, which is a hyper parameter that can be tuned to get better accuracy, has been set to 10. The accuracy, precision, recall and F1 score for this model is the same value , which is 0.995. The accuracy of the model is 99.5%.

5.2.2.5 Observation and Accuracy Plot:

There is not much difference in the architectures of the models except for MLP and SVM. For the MLP model, different numbers of hidden layers and neurons have been considered as this change gave better accuracy than the parameters considered in previous experiments. The accuracy for the MLP model dropped to 97% from 99%. In the SVM model, the value for C has been changed to 1.0 as this was the best parameter chosen after performing the gridsearch. The accuracy of the SVM model does not change a lot and the same behavior is observed for Random Forest. KNN model's accuracy dropped by a very negligible amount. Therefore, even with 10% of fake malware samples, all the models have given better accuracy.



5.2.3 Benign vs Malware Classification with 20% of Fake Malware Data

After preprocessing the data, 70-10-20 split has been considered for training, validation and testing purposes. After these splits have been done, 20% of real malware data in the training set has been removed and the same percentage of data is replaced with fake malware data.

5.2.3.1 Architectural Setup for MLP Model and Results:

The architecture of the model is similar to the architecture used in previous experiments but it has one additional input layer. The setup is as shown below:

- Input Layer: One input layer with input size of 768
- Hidden Layer 1**:** Fully connected dense layer with 512 neurons and uses RELU as activation function.
- Hidden Layer 2: Fully connected dense layer with 256 neurons and uses RELU as activation function.
- Hidden Layer 3: Fully connected dense layer with 128 neurons and uses RELU as activation function.
- Hidden Layer 4: Fully connected dense layer with 64 neurons and uses RELU as activation function.
- Hidden Layer 5: Fully connected dense layer with 32 neurons and uses RELU as activation function.
- Hidden Layer 6: Fully connected dense layer with 16 neurons and uses RELU as activation function.
- Hidden Layer 7: Fully connected dense layer with 8 neurons and uses RELU as activation function.

- Output Layer: Fully connected dense layer with output size 2 and used softmax activation function.
- "Sparse_categorical_crossentropy" and "Adam" are used as a loss function and optimizer respectively.
- Batch size and maximum number of epochs is initialized to 100 and early stopping has been enabled with patience 2.

Results for MLP:

The model obtained an accuracy of 98.62% on testing data, which indicates that the model has performed well even with 20 percent of fake malware data. MLP models were built with other hyperparameters such as hidden layers, number of neurons in each layer but this model gave better accuracy compared with others.

5.2.3.2 Support Vector Machine Model and Results:

The setup for the SVM model is the same as the setup from the previous experiment. Even after performing GridSearchCV, the best parameters have not changed. Therefore we have considered the same hyperparameters and performed training. 70% of the data has been used for training and 30% of the data has been allocated for testing.

Results for SVM:

Even with a 10% increase in the amount of fake malware data, the SVM model has an accuracy of 99.7%.

5.2.3.3 Random Forest Model and Results:

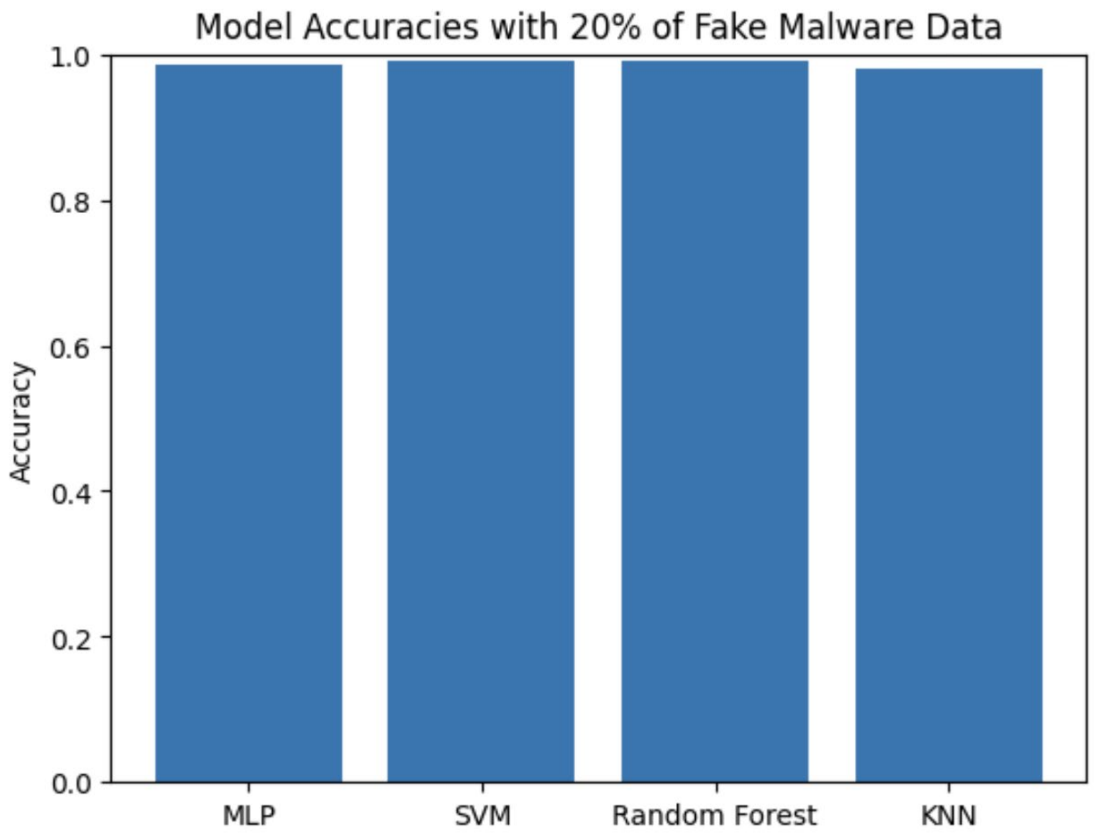
The Random Forest model setup has not been changed and remains the same as considered in previous experiments. It is surprising to notice that the random forest model has the capacity to classify benign and malignant samples effectively with its default parameters. In this case, intensive hyperparameter tuning can lead to overfitting of the model. The accuracy of the model when 20% of fake malware data has been considered is 99.7%

5.2.3.4 K Nearest Neighbors Model and Results:

The setup that has been used for training and evaluating KNN models has not been changed and remains the same as used in previous experiments. Even with the infusion of 20% of fake malware data, the KNN model came up with a testing accuracy of 98.3%. The precision, recall and F1 score for this model are 0.99, 097 and 0.98 respectively.

5.2.3.5 Observations and Accuracy Plot

Even with the increase in the amount of fake malware data, the models have performed better. It is true that there has been a bit of decrease in the accuracy but this is negligible. Below is the bar graph which displays the accuracies of all the models. From all the models that have been used for classification, KNN has a bit lower accuracy than other models, yet it has an accuracy more than 95%.



5.2.4 Benign vs Malware Classification with 30% to 90% of Fake Malware Data

The procedure for preprocessing remains the same even while considering different percentages of fake malware data. The entire data is split into 70-10-30 training, validation and testing splits for MLP model and 70-30 split for remaining models. Once the data has been separated, real malware data is removed from the training split and in its place, the same amount of fake malware data is replaced. The models are then trained based on these adulterated training dataset. Testing is performed on the testing split, which holds just the true malware samples and benign samples. Validation in MLP models also has pure malware and benign data.

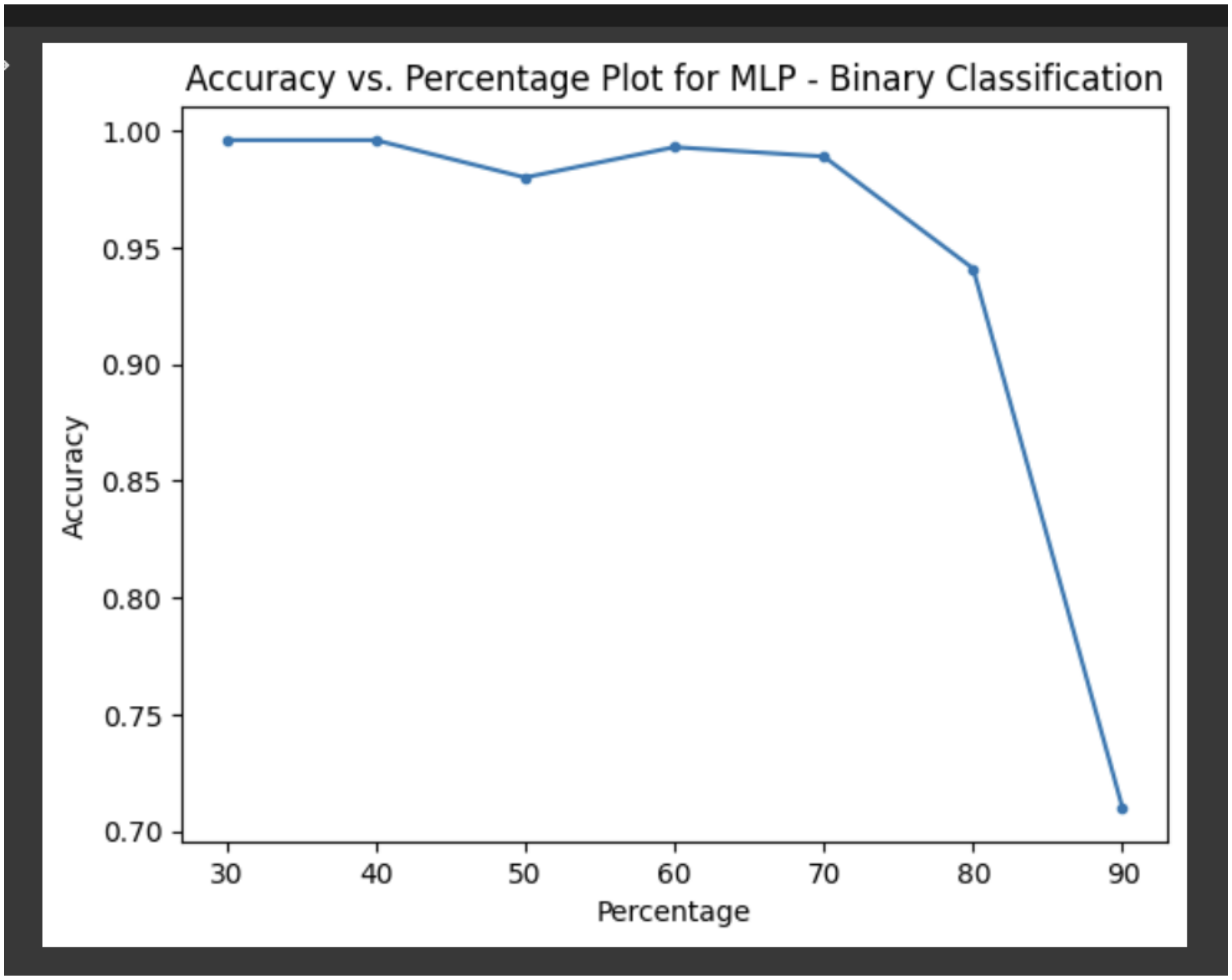
5.2.4.1 Architectural Setup of MLP and Results for inclusion of 30% to 90% fake malware data and Results

The architecture for the Multi Layer Perceptron has not been changed much and remains similar to the architecture described in previous experiments. In some rare cases, to check if the accuracy of the MLP models mitigates based on parameters like number of neurons in each hidden layer, number of hidden layers, these simple changes are made but apart from that no major changes have been made on the architecture of the model. The activation function used in the hidden layers is “relu” and in the output layer, it is “softmax”. The loss function used is “sparse_categorical_crossentropy” and the optimizer used is “adam”. Early stopping has been enabled and patience is set to 2. In some rare cases, the patience is set to 5.

Results:

There has not been a huge difference in the accuracies obtained when percentages of fake malware samples have increased from 30 to 90. This is a very interesting observation which says that even with fake malware samples, most of the real and adversarial samples are classified properly.

Decrease in the accuracy of the model has been observed with an increasing number of fake malware samples but we can see that the accuracies obtained by these intelligent models are still pretty high. Even with 90% of fake malware samples, the accuracy obtained by multi layer perceptron is around 71%.



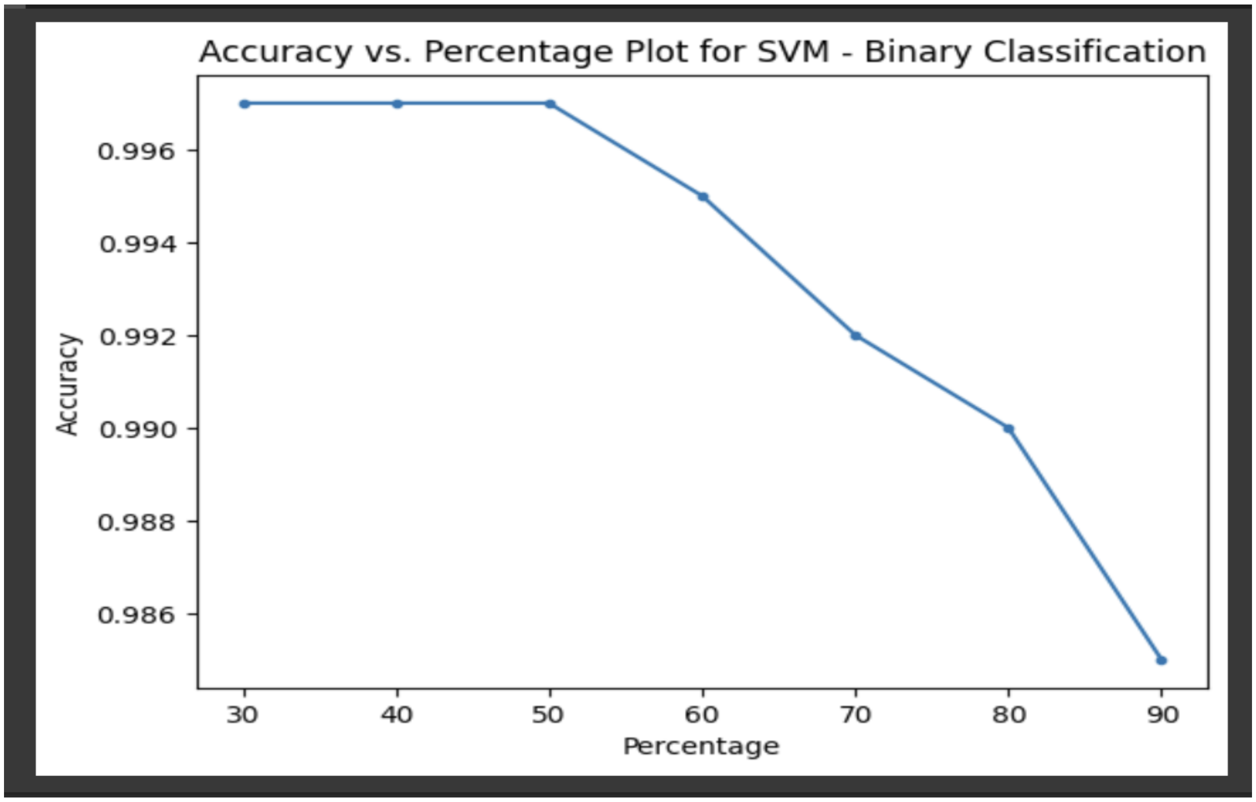
5.2.4.2 Support Vector Machine Setup and Results

The setup remains the same for the SVM model in most of the experiments. The best hyperparameters are obtained each and every single time by using GridSearchCV and these best parameters are used to train and evaluate the model.

Results:

There has been a decrease in the accuracy of the model with an increasing percentage of fake malware data but the difference in the accuracy is not significant. We can notice that even at

90% of fake malware data infused in the training dataset, the accuracy of the model is around 98%.

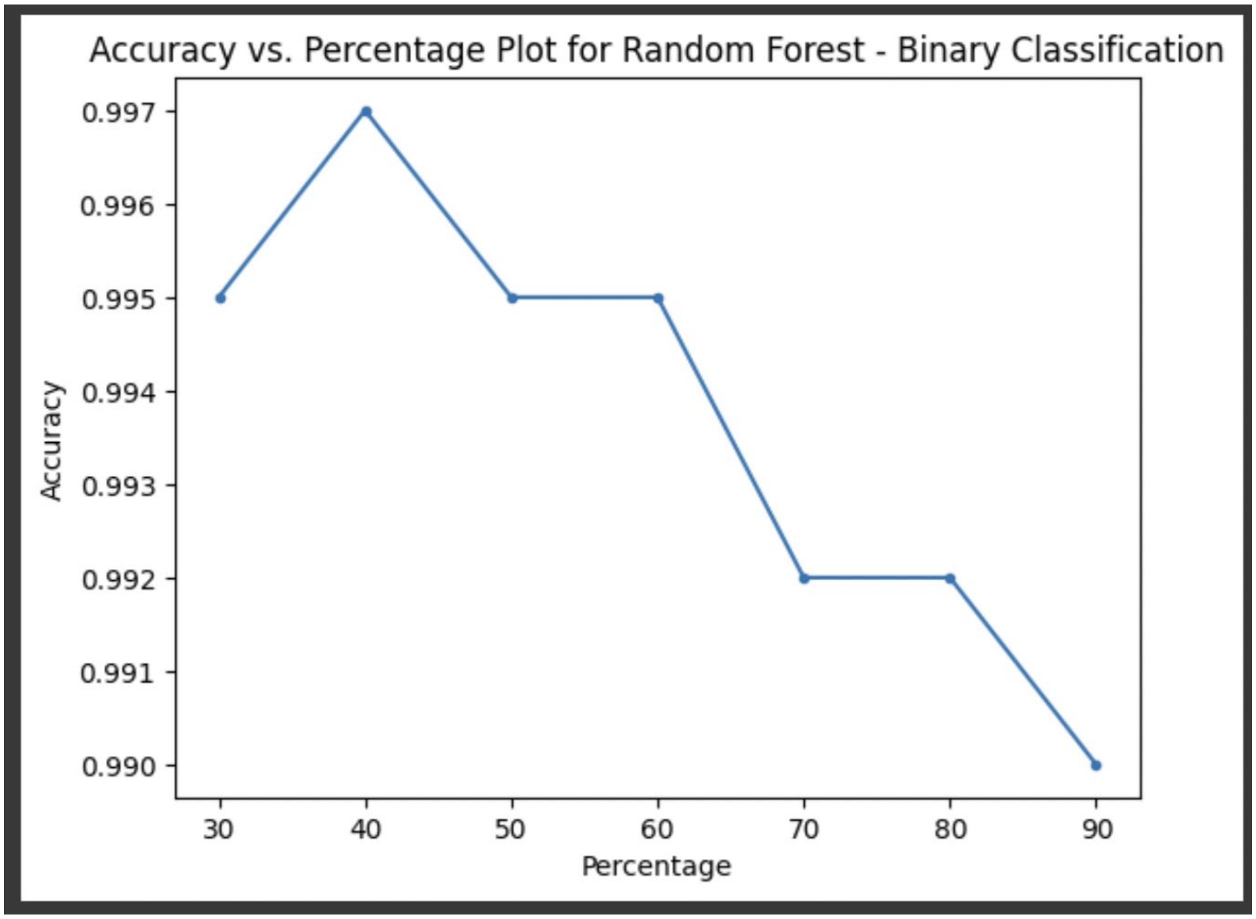


5.2.4.3 Random Forest Model Setup and Results

The Random Forest model uses the same parameters that have been used in the previous experiments and the setup did not change for these sets of experiments.

Results:

The results are quite surprising. There has been an increase in the accuracy when the number of fake samples is increased from 30% to 40% and then drops at a very negligible rate. We can observe the pattern of decreasing accuracy but surprisingly the accuracy never dropped lower than 99%. The highest accuracy obtained till now from all the experiments is 99.7%.

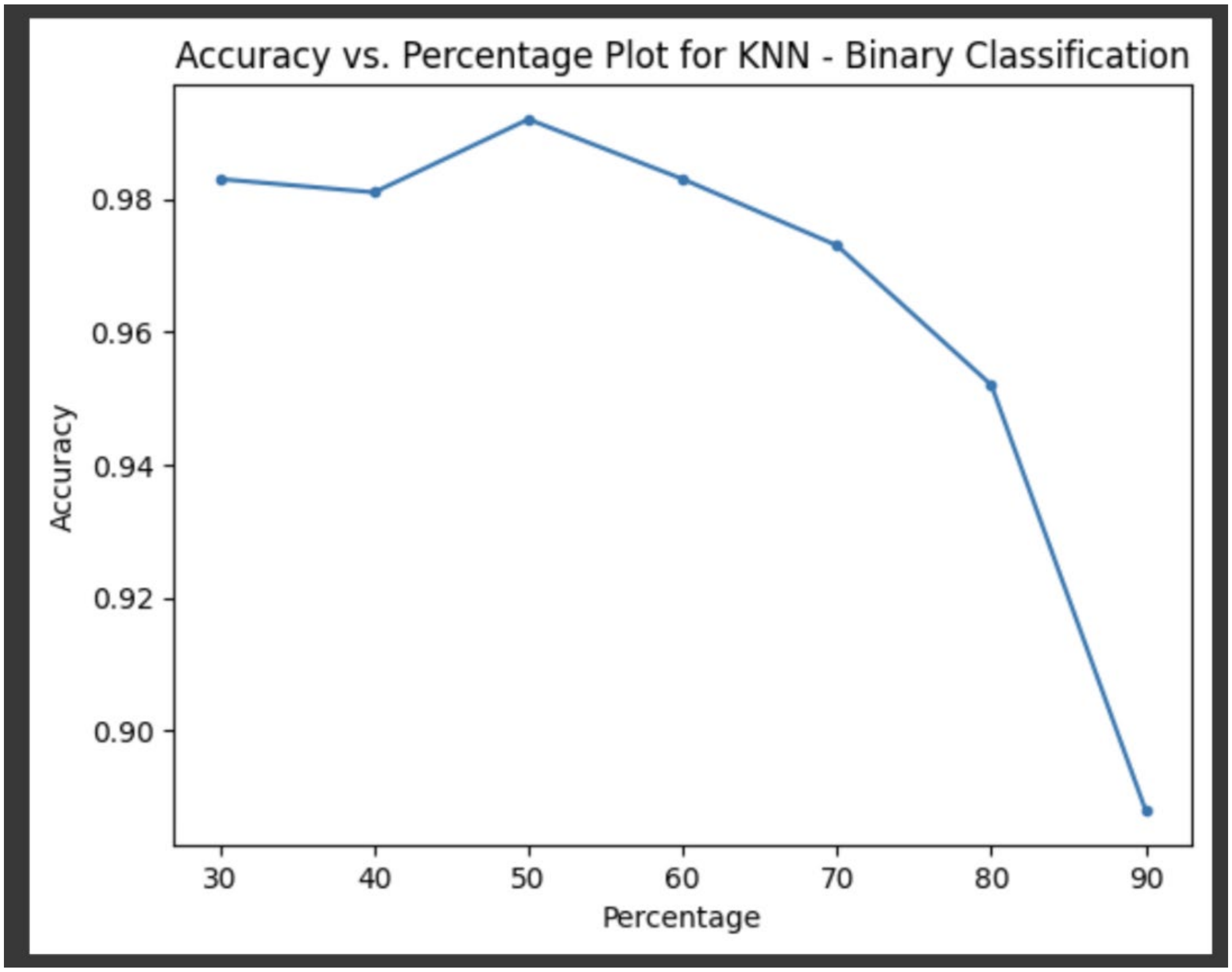


5.2.4.4 K Nearest Neighbors Model and Results

The setting of KNN has not changed as the model continues to give exceptional results.

Results:

Initially, it looked like the performance of the model was going downhill but when the percentage of fake malware samples was increased to 50, there was some increase in the accuracy. This behavior is not very common and is also observed in previous models. Since that point, we can notice a gradual decrease in the performance of the model. Even with decrease in the accuracy, we can say that the model performed extremely well as the accuracy of KNN model was around 88% when 90% of fake malware samples are infused into the training dataset.



5.2.4.5 Observations

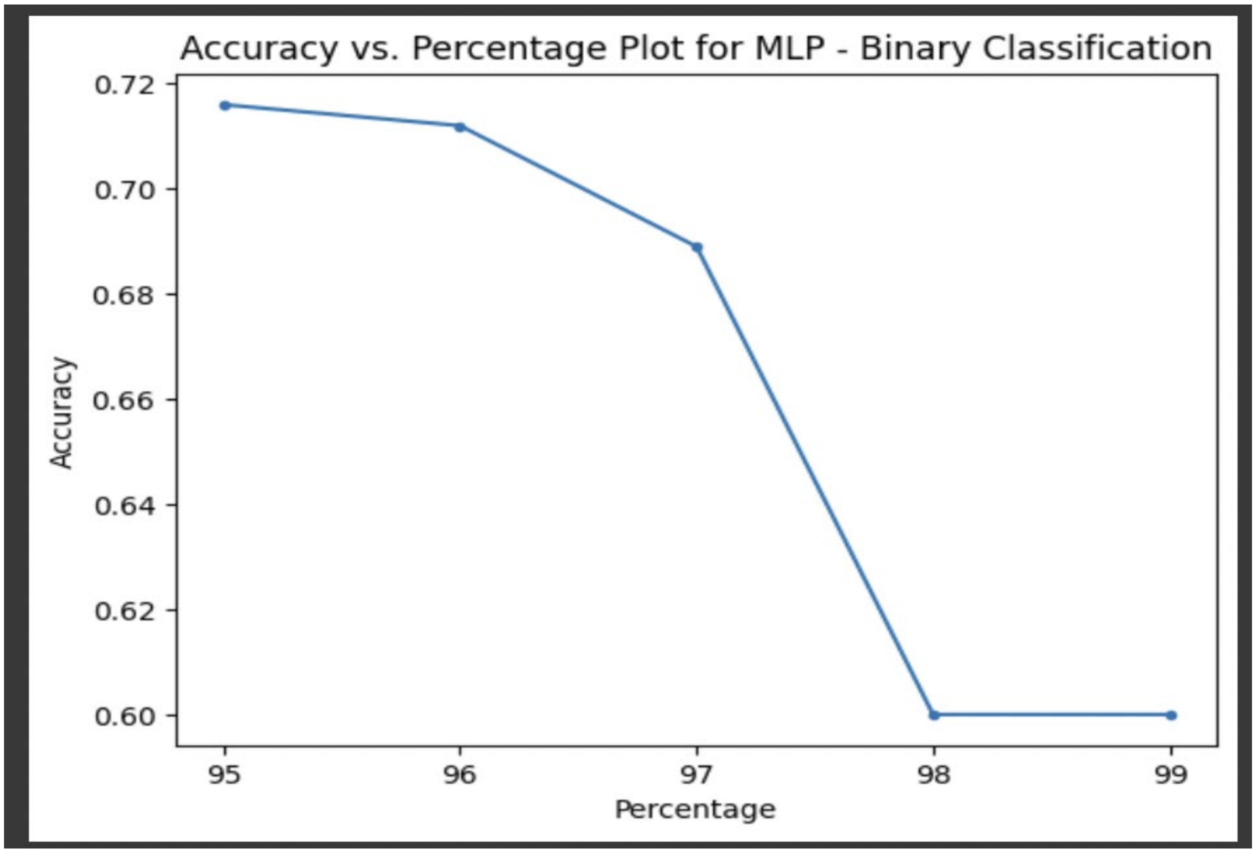
When we compare the performance of MLP, SVM, Random Forest and KNN, we observe a pattern of decreasing accuracy with increase in fake malware data. There is a negligible amount of decrease in the accuracies of SVM and Random Forest. On the other hand, we can observe a significant amount of decrease in the accuracies of MLP and KNN models. Both SVM and Random forest has accuracy over 98% even with 90% of fake malware data where MLP and KNN have the accuracies around 70% and 88% respectively.

5.2.5 Benign vs Malware Classification with 95% to 99% of Fake Malware Data

The architectural setup has not been changed for any of the models. Small changes in parameters like number of hidden layers, number of neurons in each layer and sometimes the value of patience has been tuned in order to get better results in the MLP model. KNN and Random Forest remain as is and no changes have been made. In the case of SVM, best parameters obtained by using GridSearchCV are considered to train and evaluate the model.

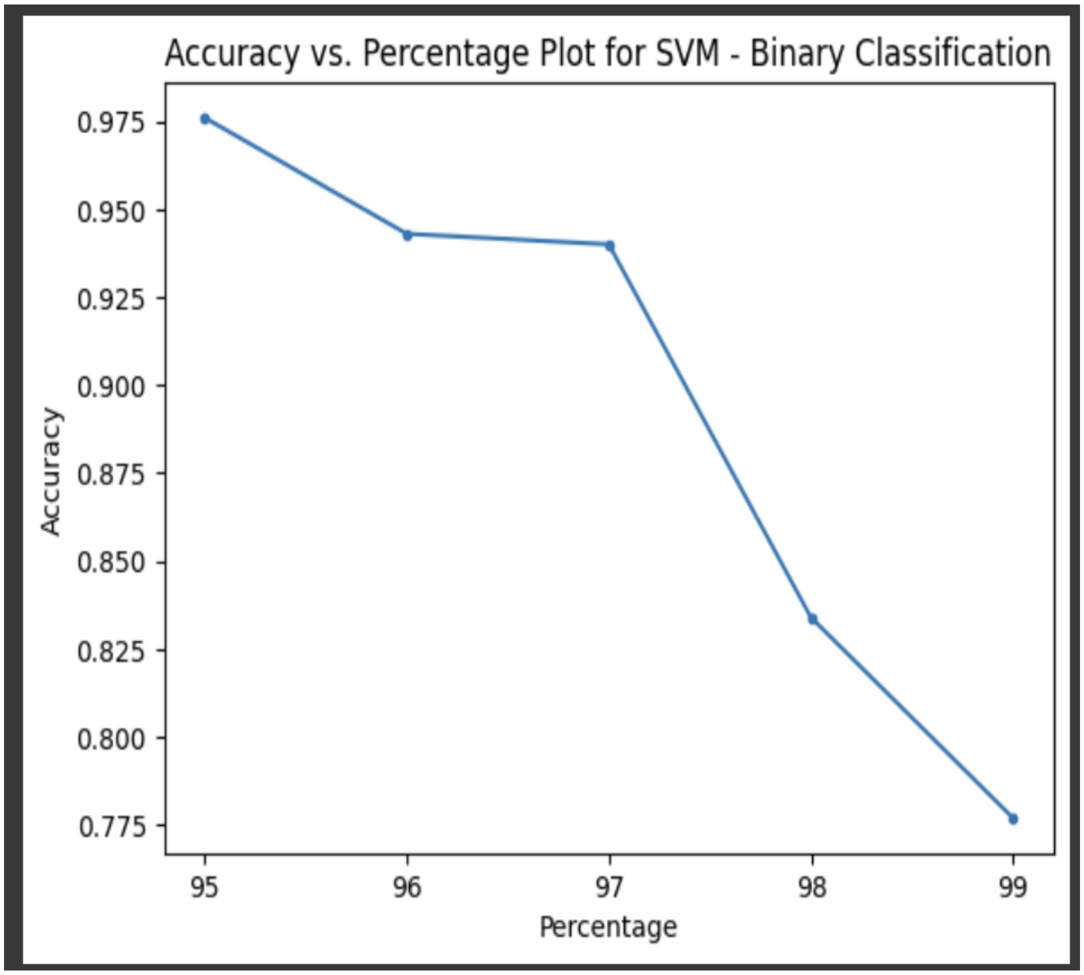
5.2.5.1 Multi Layer Perceptron Model Results

There is a gradual decrease in the performance of the model with an increase in the percentage of fake malware data from 95% to 99%. The accuracy was dropped from 71% to 60% when the fake malware samples infused were around 95% and 99% respectively.



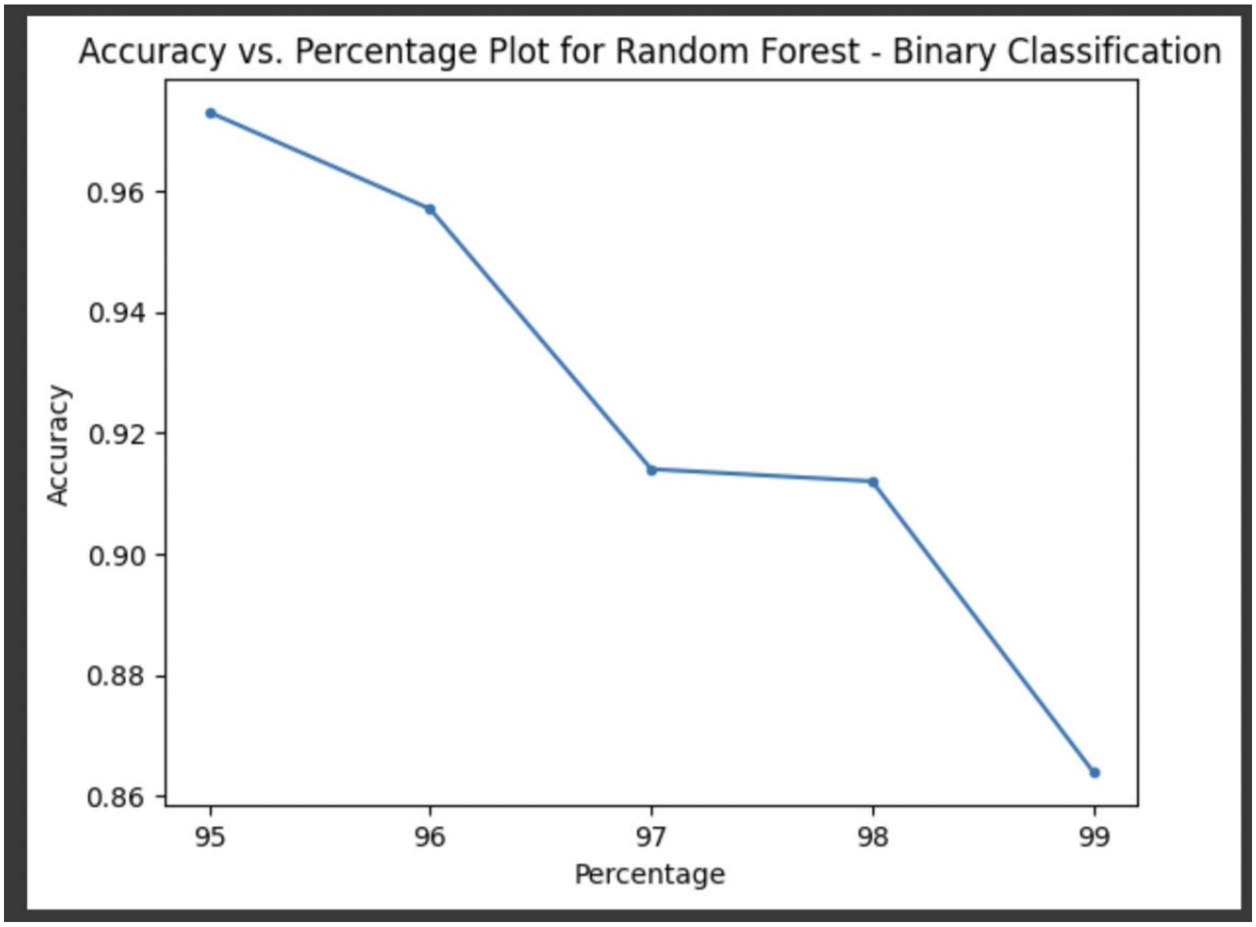
5.2.5.2 Support Vector Machine Model and Results

The performance of the model gradually decreased with an increase in the number of fake malware samples in the range of 95% to 99%. When 95% of fake samples were considered, the accuracy of the model was 97% and the accuracy dropped to 77% when 99% of fake malware data was considered. It is surprising to notice that very small increases in the number of fake malware samples have drastically decreased the performance of the model on the scale of 95 to 99 percent.



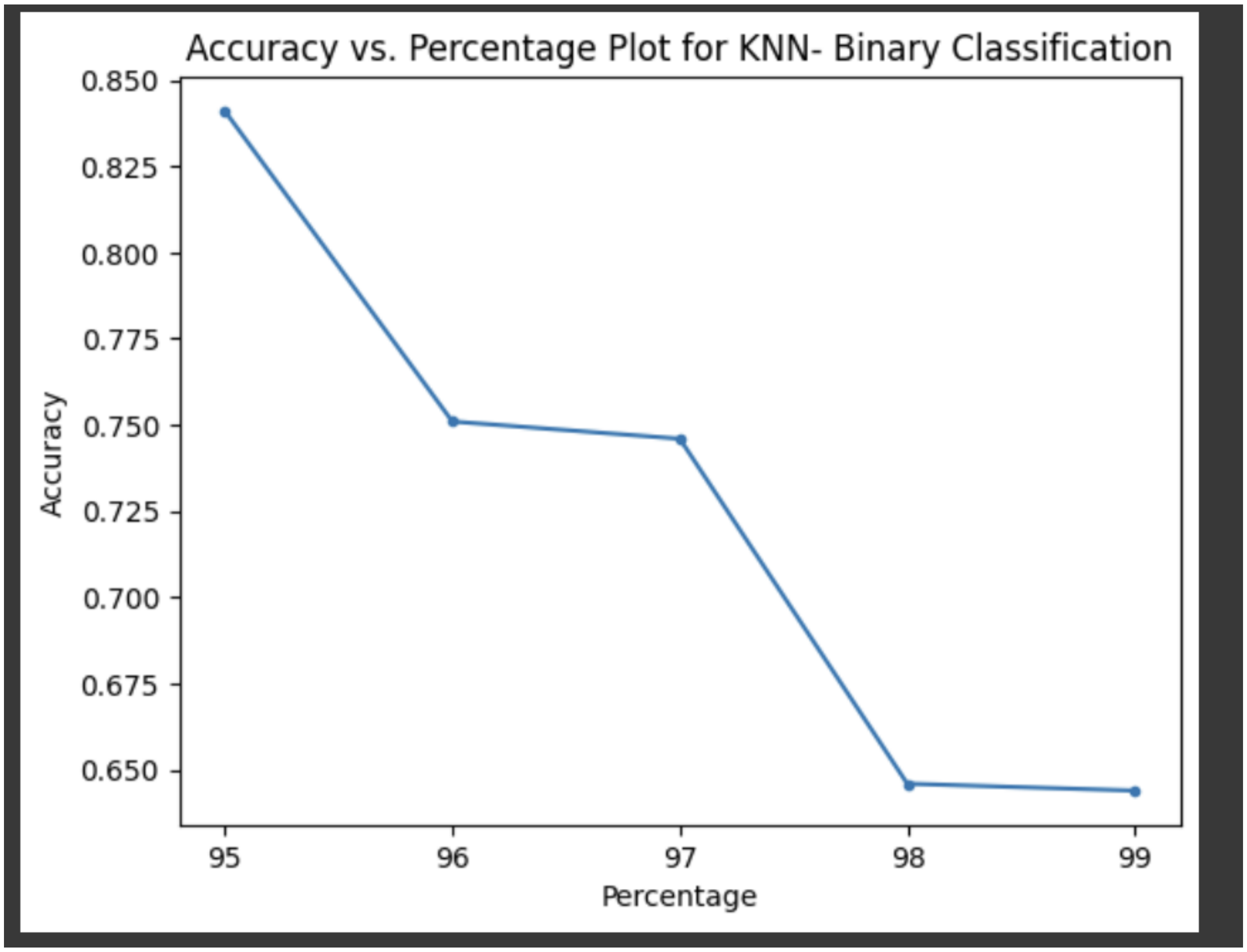
5.2.5.3 Random Forest Model and Results

The accuracy of the model with percentage of fake malware data in the range of 95 to 99 are being decreased and the same can be observed from the below picture. Random Forest outperforms the rest of the models by maintaining better accuracies than remaining ones. There has been a dip in the performance of the model but the lowest accuracy recorded for this model till this experiment is 86%, which can still be considered as a better accuracy.



5.2.5.4 K Nearest Neighbors Model Results

There has been a steady decrease in the performance of the model with increasing amounts of fake malware data. At 95% of fake malware data, the accuracy of the KNN model was around 84% and the accuracy dropped significantly to 64% when 99% of fake malware samples were considered.

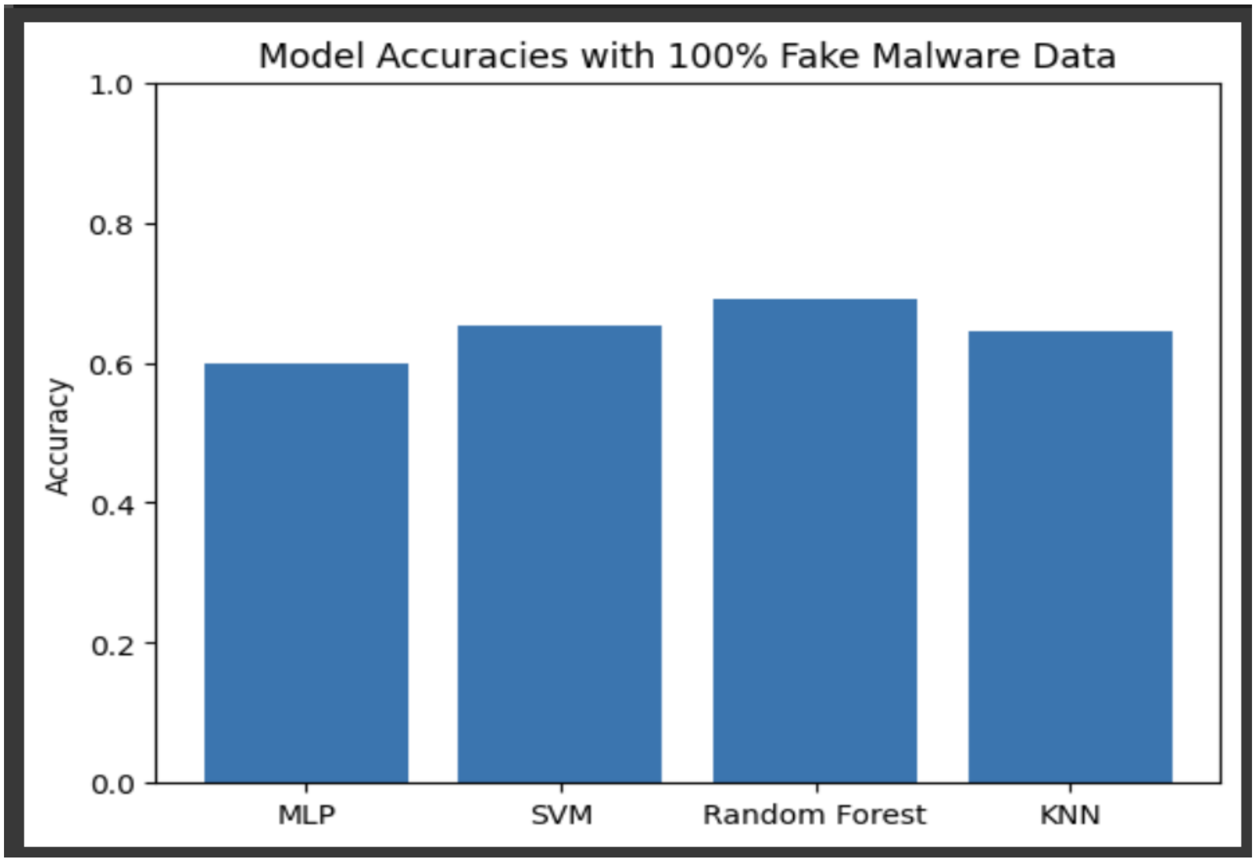


5.2.5.5 Observations

Amongst all the models, Random Forest has better accuracy. For percent range between 95-99, there has been a significant decrease in the performance of the models. Random Forest, followed by SVM then KNN and finally MLP is the order of models with better performance.

5.2.6 Benign vs Malware Classification with 100% Fake Malware Data

Data preprocessing and setup of the machine learning and deep learning models remain unchanged. Just a few parameters are adjusted when needed to increase the performance of the model. Considering 100% fake malware data means that we do not have any genuine malware samples available in the training dataset. Therefore, the model is trained on benign data and fake malware data. Validation and testing are performed with the datasets that contain benign and actual malware data. We can observe that the accuracy has dropped significantly. It is surprising to notice how a 1% increase in the fake malware data has brought significant change in the performance of SVM and Random Forest Model. There is a slight decrease in the accuracies of MLP and KNN model but this behavior could be expected unlike the behavior of other two models.



6 Malware Family Type and Benign Classification

In this part, we will perform a multi-class classification to identify benign samples and the family of the given malware sample. We have classified the data into eight classes as we have considered seven different malware families and benign samples to conduct our experiments. The malware families that were considered are BHO, OnLineGames, Renos, Zbot. Winwebsec, Zeroaccess and VBInject. Multi-class classification is a bit difficult when compared to binary classification.

6.1 Data Pre-Processing

The data, after collecting from various sources, has been segmented into their corresponding family types and then opcode sequences are obtained from the data which are in the form of executables files. These opcode sequences are given to the BERT model, which comes from the family of natural language processing techniques and this model outputs word embeddings of length 768. At this point we have benign data and actual malware data ready but as we are experimenting with different percentages of fake malware data, we will generate enough fake malware samples using WGAN-GP, which is a variant of Generative Adversarial Network. The generated fake malware samples are very similar to the actual malware samples. Thes fake samples are also in the form of bert embeddings and are used to train our intelligent models and evaluate them.

The data has been split evenly among all the classes to avoid the problem of data imbalances. Therefore, we have the same number of samples in each class. While inducing different percentages of fake malware samples, it is made sure that the data is removed evenly from all the malware families and then the same amount of fake malware samples are induced to the training set. The data is divided into 70-10-20 splits for the MLP model where 70% of the data is used for training the model, 10% is used for validation and 20% of the data is used for model evaluation. For the rest of the machine learning models, the split considered is 70-10 where 70% of data is used for training and the rest is used for testing. The fake malware samples are included only in the training dataset after the partition is done.

6.2 Experiments

6.2.1 Multiclass Classification with 0% to 90% of Fake Malware Data

In this section, several experiments have been conducted with increasing amounts of fake malware data to evaluate the performance of machine learning and deep learning models and the accuracies of these models are recorded.

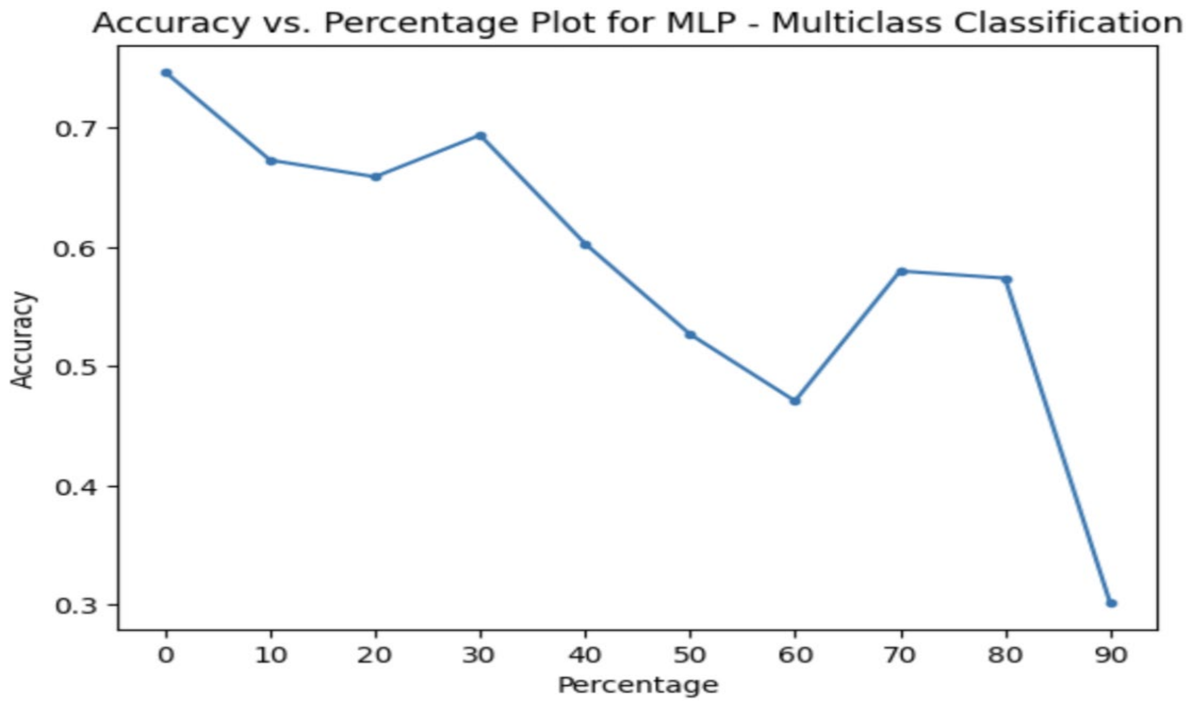
6.2.1.1 Architectural Setup for MLP Model and Results

- Input Layer: Dense layer with 768 input units
- Hidden Layer 1**:** Dense layer with 512 neurons and uses ‘ReLU’ activation function
- Hidden Layer 2 : Dense layer with 256 neurons and used ‘ReLU’ as activation function
- Hidden Layer 3: Dense layer with 128 neurons with ‘ReLU’ as activation function
- Hidden Layer 4: Dense layer with 64 neurons and this layer uses ‘ReLU’ as activation function.
- Hidden Layer 5: Dense layer with 16 neurons and ‘ReLU’ activation function.

- Hidden Layer 6: Dense layer with 8 neurons and this layer uses 'ReLU' as activation function.
- Output Layer: Contains 8 output neurons and uses 'Softmax' as activation function.
- 'Saprse_categorical_crossentropy' and 'Adam' are used as loss function and optimizer respectively.

Architectural setup remains the same in most of the experiments but parameters like number of hidden layers, number of neurons in each layer, considering dropout layers in between dense layers, number of epochs and batch size have been tweaked to check if the model can perform better. Accuracy is the metrics used to evaluate the performance of the model.

The model started off with the performance of 75% when no fake malware data but has decreased by a bit when the fake malware data has been increased to 20%. Surprisingly, the model came up with a better accuracy when 30% of fake malware data has been infused. Similarly, the accuracy has decreased since then till 60% of fake malware data was considered. At 70% fake malware data, the accuracy increased and then dipped significantly when 90% of fake malware data is considered.

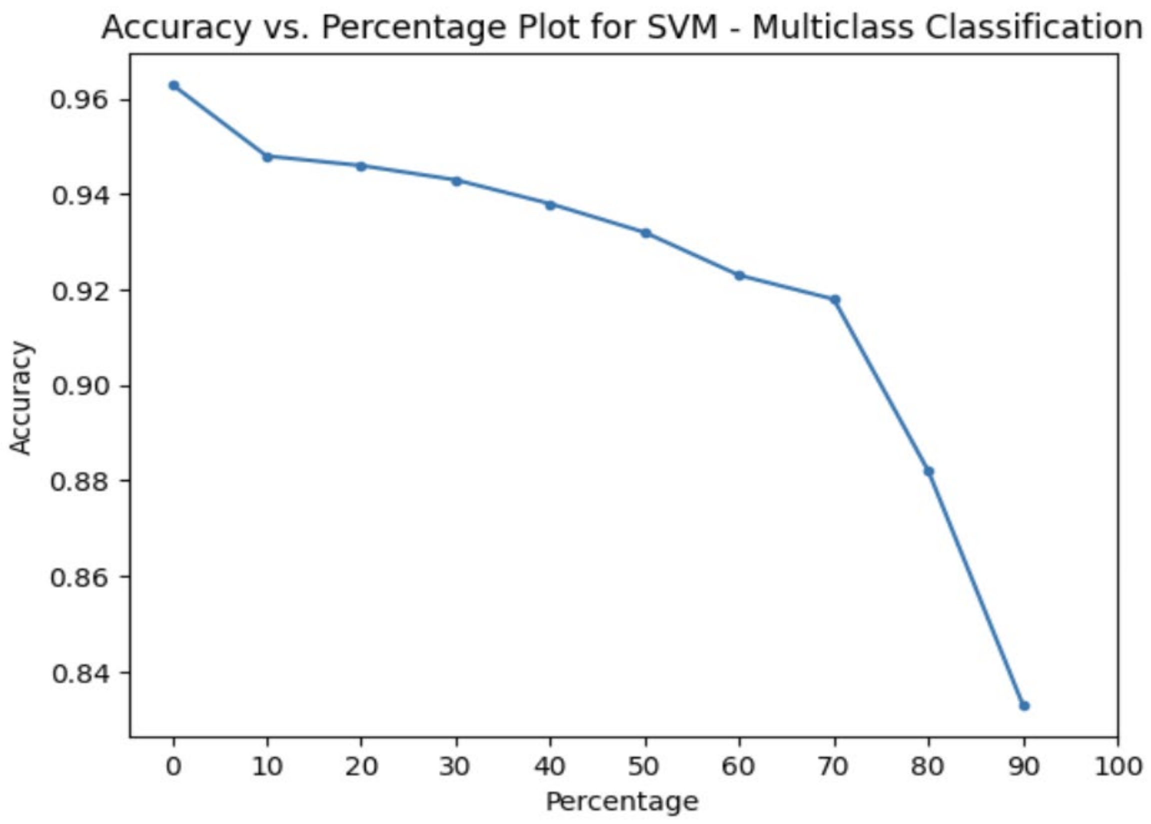


6.2.1.2 Support Vector Machine Setup and Results

Hyper parameter tuning was performed to get the best parameters to train the SVM model. The parameters that were tuned are as follows:

- Kernel: RBF, Poly and Linear
- Gamma: 0.0001, 0.001, 0.01
- C: 0.5, 1.0

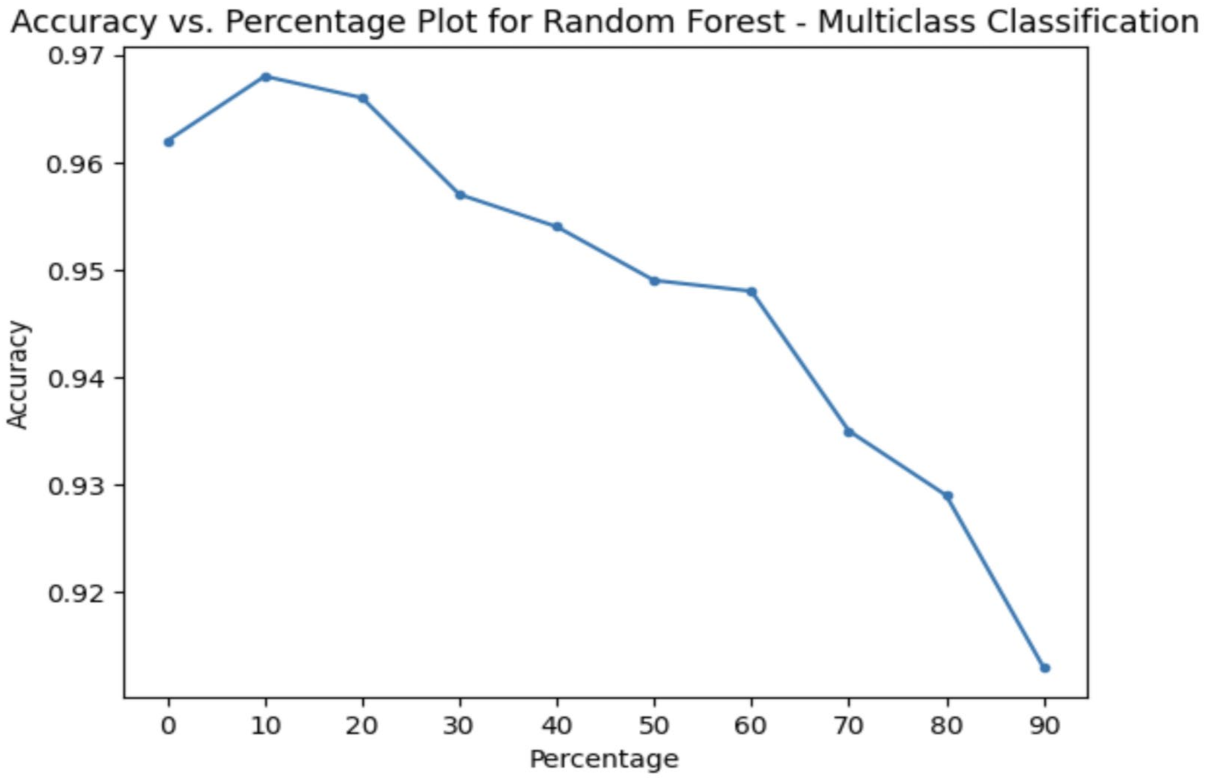
Amongst these parameters, best parameters were found using GridSearchCV and the best parameters were: C - 1.0, Gamma - 0.0001 and Kernel - Linear for most of the experiments. In some rare cases, the parameters were changed according to the best parameters obtained by GridSearchCV.



From the above graph, there is a decrease in the model performance with an increase in the fake malware data. But unlike the MLP model, this deterioration in the performance of the mode is more expected but we can see that even at 90% of fake malware data, the model exhibited good performance with the accuracy of 83%.

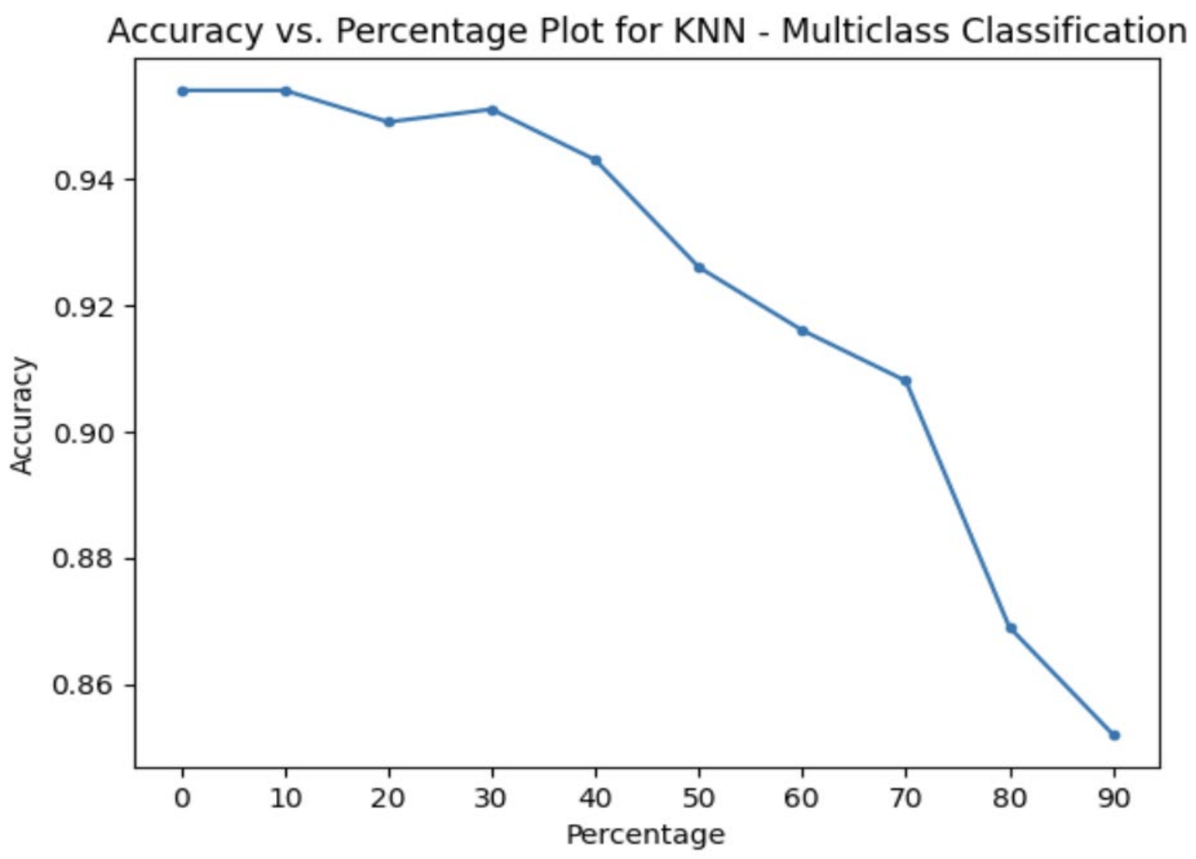
6.2.1.3 Random Forest Model Setup and Results

The architectural startup for random forest model has not been changed and it is the same as the model we have considered in the binary classification. The difference is in the data, in binary classification we have only two classes but here, we have seven classes. Even with the default setup, the model has outperformed when compared to the rest of the models as it obtained the highest accuracy of 91% even with 90% of fake malware data.



6.2.1.4 K Nearest Neighbor Setup and Results

The model setup for KNN is the same as the setup used in the previous experiment. The value for the number of neighbors to be considered for making a decision, which is a hyper parameter 'k' that can be tuned to get better accuracy, has been set to 10. When the fake malware samples that were considered are at its lowest, the accuracy of the model was around 96% but then increased a bit when fake samples were introduced, which is an unpredicted behavior. Since then, there has been a decline in the accuracy of the model with an increase in the percentage of fake malware data. Even Though, the accuracy of 85% has been obtained by the model with 90% of fake malware data.

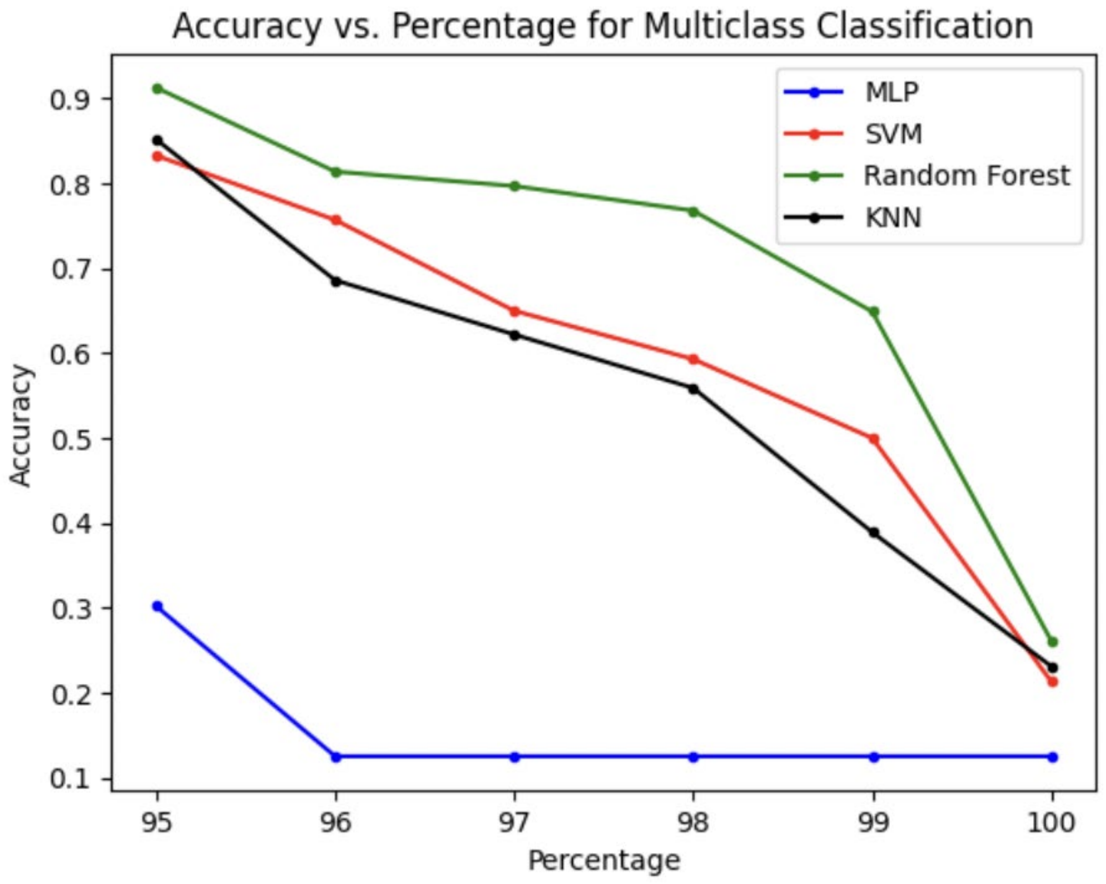


6.2.1.5 Observations

It is surprising to notice that all the models have performed better than expected by providing exceptional accuracy. Amongst all the models, Random forest has the best performance followed by KNN and then comes SVM. It is astonishing to see that the MLP has the least performance when compared to the rest of the models.

6.2.2 Multiclass Classification with 95% to 100% of Fake Malware Data

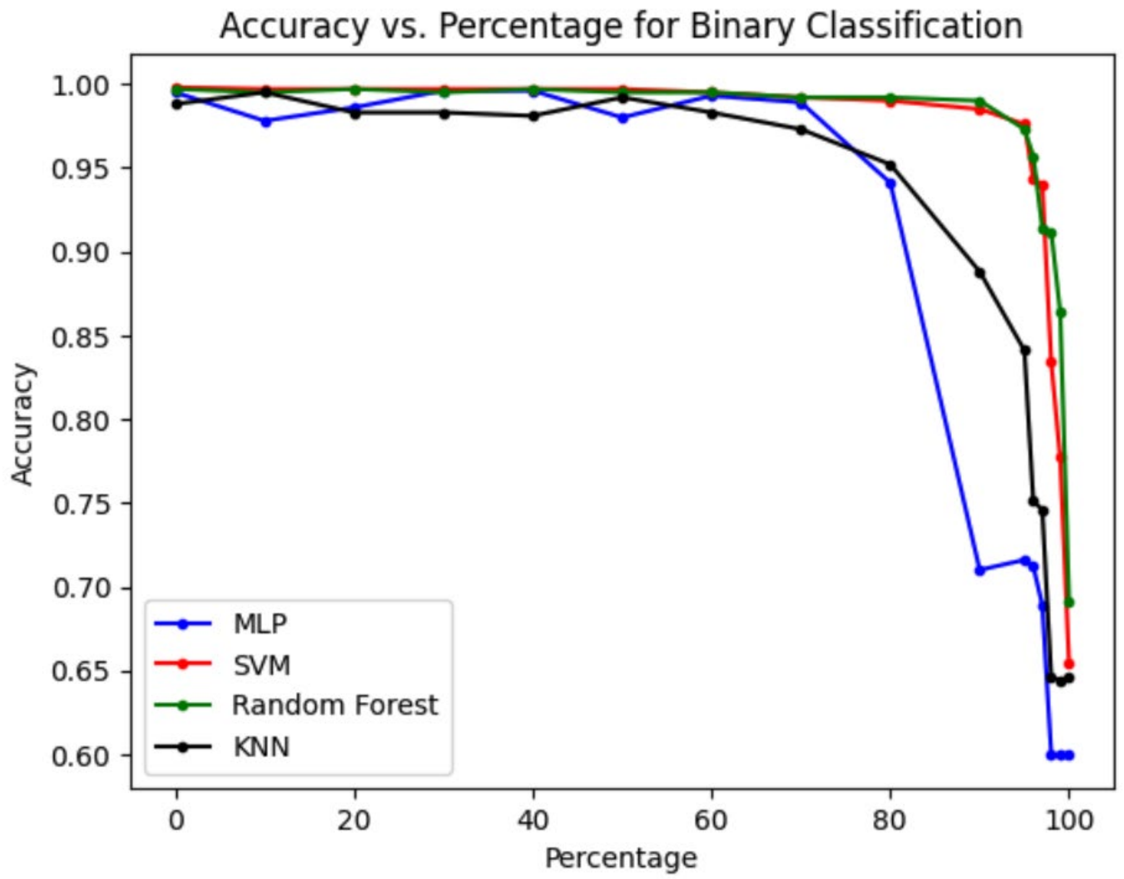
The architectural setup for machine learning and deep learning models have not been changed for these sets of experiments. These experiments have to be performed to specifically observe the behavior of the model when fake malware samples are increased gradually by 1%. The accuracy of the MLP model has dropped by a significant percentage. MLP model has obtained an accuracy of 30% when 95% of fake malware data is considered. The performance of the other models have not decreased in such a rate but we can see a decreasing pattern in the accuracies for other models as well. Random Forest has achieved an accuracy of 91%, whereas SVM and KNN have the accuracies of 83% and 85% respectively. On the other hand, when 100% fake malware samples are considered, the performance of the models have significantly decreased. The accuracies of MLP, SVM, Random Forest and KNN at this rate of fake malware samples is 12%, 21%, 26% and 23% respectively.

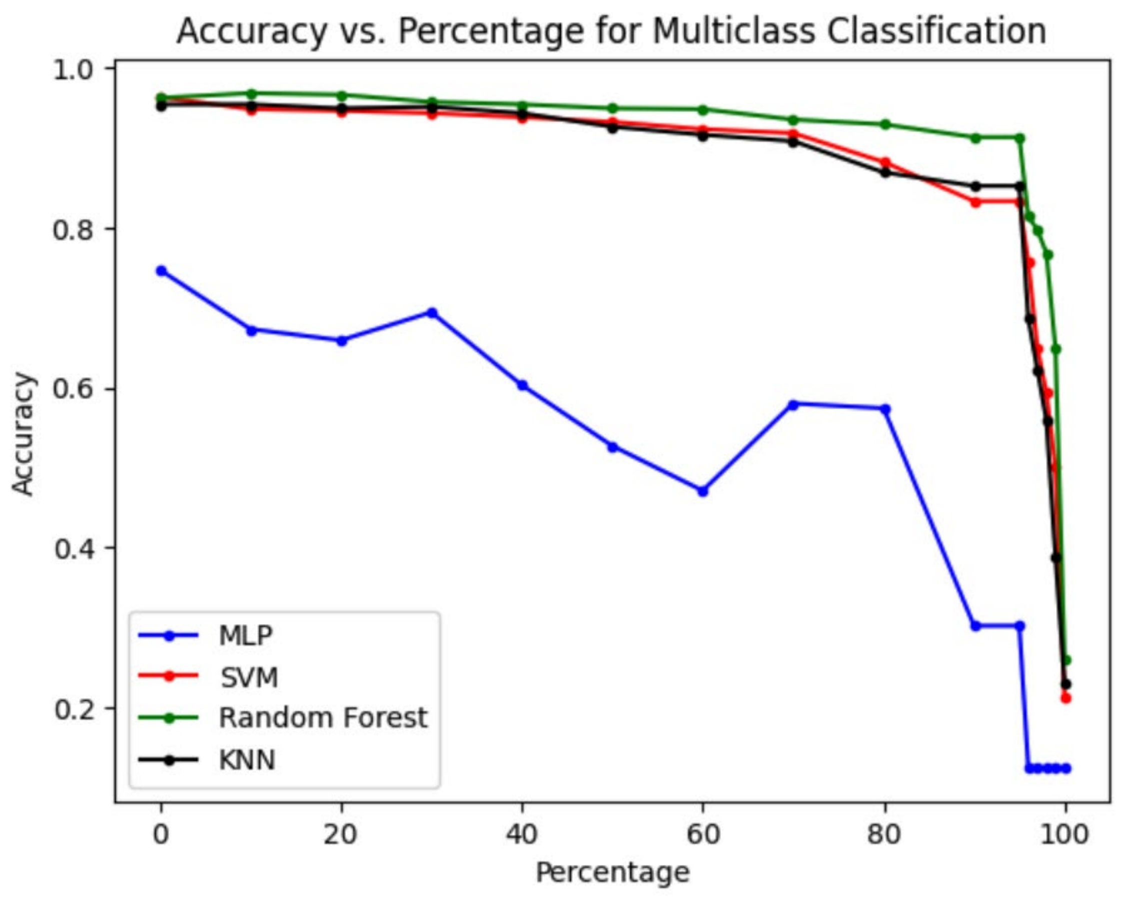


MLP model performance has not changed since 95% of fake samples have been considered. Whereas other models performance has been decreased with increase in the percentage of fake malware samples. The highest accuracy was obtained by random forest and the lowest accuracy was obtained by MLP.

7 ANALYSIS

Multiple experiments have been conducted for binary classification and multiclass classification, starting from considering 0% fake malware samples and constantly increasing the percentage of fake malware samples to 100%. When binary classification was performed, all the models that were considered in our experiments have shown exemplary results. All the models have given the accuracy more than 90%, which indicated that the algorithms were able to classify malware and benign files with utmost confidence, which can be achieved through thorough learning of the patterns by analyzing the relationships between the features and targets. To avoid the problem of imbalances, data has been distributed evenly with the same number of samples per class for all training and testing splits. In the case of binary classification, an equal number of samples were selected from each family and grouped them together to form malware class. About the same number of malware and benign samples were considered in the dataset for the model to analyze the data correctly and make a prediction. The accuracies were pretty high even with increasing percentages of fake malware data and we could see a significant drop in the accuracies only from 95% of fake malware for all the intelligent models that we have used in our work. In multiclass classification, MLP model has the least performance even when no fake malware samples were given. All other models (SVM, Random Forest, KNN) had the accuracies of over 98% when 0 fake samples were considered. With the increase in the amount of fake malware samples, there was a very negligible amount of decrease in the performance of SVM, Random Forest and KNN models until around the consideration of 70%, 90% and 70% of fake malware data respectively. MLP has been an outlier by exhibiting unexpected accuracies and still remains to have lower accuracies compared to rest of the models and the performance of the model neither mitigates nor deteriorates after a certain point. For the rest of the models, when 90 to 100 percent of fake malware samples were considered, the performance of the models were deteriorating and displayed lower accuracies. There was a significant drop in the accuracies of the models from considering around 95% of fake malware data and the intuition behind it could be that the models were able to analyze the data even with slight amounts of actual malware data ; the reason behind this could be that the fake malware samples considered in the experiments are very close to the actual malware samples. Hence, a certain amount of fake malware samples can be considered to boost the performance of the ML and DL models when there are insufficient number of real malware samples.





Below is the table to compare the performance of the MultiLayer Perceptron, Support Vector Machine, Random Forest and K-Nearest Neighbors models with increasing percentages of fake malware samples. Binary classification and multiclass classification is performed using these algorithms.

Percentage of Fake Malware Samples	Binary Classification Accuracy Score				Multiclass Classification Accuracy Score			
	MLP	SVM	Random Forest	KNN	MLP	SVM	Random Forest	KNN
0	0.995	0.998	0.998	0.998	0.747	0.963	0.968	0.954
10	0.978	0.997	0.997	0.995	0.673	0.948	0.968	0.954

20	0.986	0.997	0.997	0.983	0.659	0.946	0.962	0.949
30	0.996	0.997	0.997	0.983	0.694	0.943	0.966	0.951
40	0.996	0.997	0.997	0.981	0.603	0.938	0.954	0.943
50	0.98	0.997	0.997	0.992	0.527	0.932	0.949	0.926
60	0.993	0.995	0.995	0.983	0.471	0.923	0.948	0.916
70	0.989	0.992	0.995	0.973	0.580	0.918	0.935	0.908
80	0.941	0.990	0.992	0.952	0.574	0.882	0.929	0.869
90	0.716	0.985	0.990	0.888	0.302	0.833	0.913	0.852
95	0.716	0.976	0.973	0.841	0.302	0.883	0.913	0.852
96	0.712	0.943	0.957	0.751	0.125	0.757	0.814	0.686
97	0.689	0.940	0.914	0.756	0.125	0.650	0.797	0.622
98	0.60	0.834	0.912	0.646	0.125	0.593	0.768	0.559
99	0.60	0.777	0.846	0.644	0.125	0.500	0.649	0.389
100	0.60	0.654	0.691	0.644	0.125	0.213	0.261	0.231

The random forest model has performed better when compared to the other models used in these experiments. So, below are the confusion matrices of the random forest model for multiclass classification when 95%, 99% and 100% of fake malware samples are considered.

Confusion Matrix when 95% of Fake Malware Samples:

There are 212 samples per class in the test split and from the below confusion matrix and we can see that except 1, all the benign samples (class - 0) are classified correctly and 53, 11, 13, 13, 1, 33 and 22 malware samples, which belong to the classes 1 through 7 respectively are misclassified.

ACTUAL LABELS

```
          0      1     2     3     4     5     6     7
   ([[211,    0,    0,    1,    0,    0,    0,    0],
1    [  5,  159,    0,   13,   10,   10,    6,    9],
2    [  0,    0,  201,    1,    5,    1,    3,    1],
3    [  0,    1,    0,  199,    1,    0,   10,    1],
4    [  1,    3,    0,    1,  199,    7,    0,    1],
5    [  0,    0,    0,    0,    0,  211,    1,    0],
6    [  1,    4,    0,    6,    7,   15,  179,    0],
7    [  0,    1,    0,    0,   14,    4,    3,  190]])
```

PREDICTED LABELS

Confusion Matrix when 99% of Fake Malware Samples:

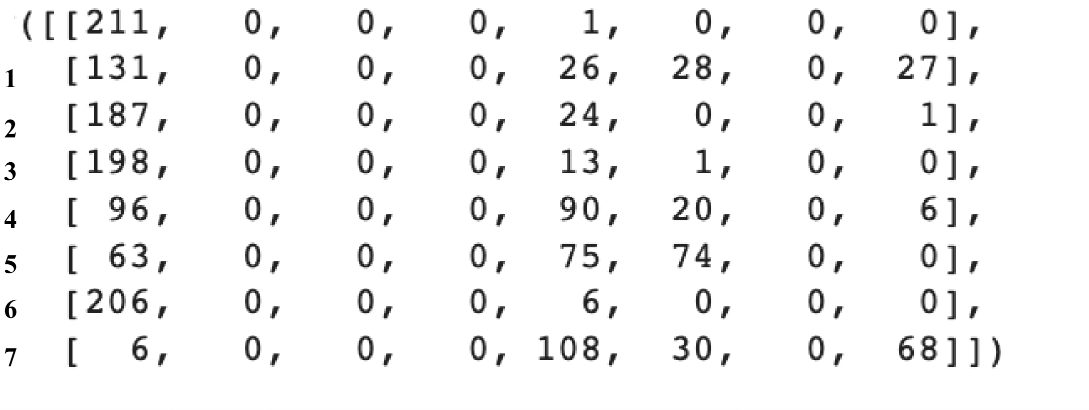
ACTUAL LABELS

```
          0      1     2     3     4     5     6     7
   ([[211,    0,    0,    1,    0,    0,    0,    0],
1    [105,    4,   12,    3,   27,   30,    0,   31],
2    [  3,   10,  178,    2,   12,    5,    1,    1],
3    [ 11,    0,   13,  159,   27,    0,    1,    1],
4    [ 17,    1,    0,   11,  157,   18,    0,    8],
5    [  9,    0,    0,    0,   23,  180,    0,    0],
6    [ 57,    2,    0,   34,   50,    0,   69,    0],
7    [  1,    0,    0,    1,   54,   12,    0,  144]])
```

PREDICTED LABELS

When 99% of fake malware samples are considered, we can see that most of the malware samples were misclassified. Highest number of samples (208) belonging to class 1 are misclassified, followed by class 6 (143). More than 30 samples per remaining class were misclassified and this can be observed in the above picture.

Confusion Matrix when 100% of Fake Malware Samples:



```
                   0       1      2      3      4      5      6      7
A              ([[211,     0,     0,     0,     1,     0,     0,     0],
C         1      [131,     0,     0,     0,    26,    28,     0,    27],
T         2      [187,     0,     0,     0,    24,     0,     0,     1],
U         3      [198,     0,     0,     0,    13,     1,     0,     0],
A         4      [ 96,     0,     0,     0,    90,    20,     0,     6],
L         5      [ 63,     0,     0,     0,    75,    74,     0,     0],
          6      [206,     0,     0,     0,     6,     0,     0,     0],
L         7      [  6,     0,     0,     0,   108,    30,     0,    68]])
A
B
E
L
S
```

PREDICTED LABELS

When 100% of fake malware samples are considered we can see that the majority of samples belonging to class 1 through class 7 are misclassified. From the above picture, we can see that samples of classes 1, 2, 3 and 6 are completely misclassified and 90, 74 and 68 samples belonging to classes 4, 5 and 7 respectively were only classified correctly. Most benign samples were classified correctly when compared to the rest of the samples.

From the above three confusion matrices, we can see that samples belonging to class 0, which are benign samples, are classified correctly and the number of misclassifications for the rest of the classes, which are malware samples belonging to different families, have been increased with increase in the percentage of fake malware samples. When the maximum number of fake malware samples are considered, only 232 malware samples were correctly classified. This value is considerably small when compared to the total number of malware samples considered in the test split, that is 1484.

8 CONCLUSION

Cybersecurity is very crucial as the enhancements in the technology gave birth to a digital world in which humans mostly rely on software gadgets and the internet, which can always be at risk of cyber attacks. Therefore, there is ample room for research to develop defensive techniques for the protection of computer systems. The room for research in this area keeps expanding as long as new methodologies evolve to threaten security. Now-a-days, a boom in the area of artificial intelligence came up with more reliable techniques that aids in cybersecurity by accomplishing various tasks like anomaly detection, malware classification and many more. As machine learning techniques make a decision by learning the patterns based on huge amounts of data that is given as inputs, the hackers came up with adversarial samples, which are given to the machine learning models to bypass malicious softwares. As these models will be trained based on these adversarial data, they cannot classify suspicious data properly. This is a potential threat to cybersecurity and hence many researches are being performed to build defensive models to aid cybersecurity. In this project, we have seen how adversarial samples are generated using generative algorithms and observed how these samples can impact the training process of machine learning models. We have noticed that even with high amounts of adversarial samples, some of the models performed extremely well but the performance deteriorates after a certain point. Therefore, to enhance the performance of the machine learning and deep learning models, a certain amount of fake malware samples can be used when there are small amounts of actual malware samples available per family. According to our experiments, we can come to the conclusion that the models were able to analyze the data, classify them accordingly even with small amounts of actual malware data and the reason behind this could be that fake samples generated by the generative algorithm are similar to the actual malware data. From our experiments, we can say that the Random Forest classifier has performed extremely well in classifying the data accordingly even in binary and multiclass classification.

8.1 Further Experiments

There is a scope for development in this study. Experiments were conducted using the Multi Layer Perceptron, Support Vector Machine, Random Forest and K Nearest Neighbors algorithms and the study can be expanded by considering other machine learning and deep learning techniques. From the above experiments, we can see that the performance of the MLP model is deficient and to enhance its performance, techniques like data augmentation and transfer learning can be used. Another set of experiments can be performed by poisoning the models in a different way like removing some percentage of actual malware data, replacing them with fake malware data and then label them as benign samples. In this scenario, the same percentage of actual malware samples have to be added to the dataset to avoid the problem of data imbalances. This method can come with a problem on how you split the training and testing datasets as the adversarial samples are only added to training split and when you try to do the above procedure to training set, then, the percentage of train-test split may vary, which could lead to very small amounts of data for testing. This issue needs to be analyzed carefully to come up with a solution. In this work, word embeddings obtained using the BERT model are used as features. Different data preprocessing and feature engineering techniques can be used to observe the performance of the models.

References

[1] J. Aycock, Computer Viruses and Malware, *Advances in Information Security*, Springer-Verlag, New York, 2006.

[2] Igor Santos, Felix Brezo, Javier Nieves, Yoseba K. Penya, Borja Sanz, Carlos Laorden, and Pablo G Bringas. Idea: Opcode-sequence-based malware detection. In *International Symposium on Engineering Secure Software and Systems*, pages 35–43, 2010.

[3] X. Sun, X. Li, K. Ren, J. Song, Z. Xu, J. Chen, Rethinking compact abating probability modeling for open set recognition problem in cyber-physical systems, J. Syst. Archit. 101 (2019) 101660, http://dx.doi.org/10.1016/j.sysarc. 2019.101660.

[4] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, Classification of ransomware families with machine learning based on N -gram of opcodes, Future Gener. Comput. Syst. 90 (2019) 211–221, http://dx.doi.org/10.1016/j.future.2018.07. 052.

[5] Symantec, "How does Symantec Endpoint Protection use advanced machine learning?", 2018. [Online].

[6] Christiaan Beek et al. McAfee labs threats report. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf, August 2019.

[7] J.J. Oliver, et al. TrendMicro Zero Day Malware Scanner. 2013, US, TrendMicro, Inc., Tokyo (JP). Art.no. 8375450

[8] R. Sharma, A. Kumar and A. K. Singh, "Malware detection techniques: A comprehensive survey," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-6, doi: 10.1109/ICCCNT.2017.8204083.

[9] S. A. Aljawarneh, H. M. Abdalla, H. Aljawarneh and R. N. Al-Khawaldeh, "Malware detection techniques using machine learning: A survey," 2019 4th International Conference on Computer and Communication Systems (ICCCS), Singapore, 2019, pp. 294-299, doi: 10.1109/CCOMS.2019.8724443.

[10] S. Pandey and S. K. Sood, "Machine learning for malware detection: A survey," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8945976.

[11] A. N. Toosi, M. A. Razak and B. A. Y. Al-Dabbagh, "A review on machine learning techniques in malware detection," 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), Muscat, Oman, 2017, pp. 147-152, doi: 10.1109/C-CODE.2017.8273914.

[12] S. Aljawarneh, M. A. Al-Zou'bi and M. Qabajeh, "An efficient deep learning approach for malware detection," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 10, pp. 3915-3924, 2019, doi: 10.1007/s12652-019-01415-2

[13] X. Liu, Z. Wu, Y. Chen, S. Xue and W. Li, "A comparative study on machine learning techniques for android malware detection," 2017 29th International Conference on Software Engineering and Knowledge Engineering (SEKE), Pittsburgh, PA, 2017, pp. 307-312, doi: 10.18293/SEKE2017-143.

[14] Fabrício Ceschin, Heitor Murilo Gomes, Marcus Botacin, Albert Bifet, Bernhard Pfahringer, Luiz S Oliveira, and André Grégio. 2020. “Machine Learning (In) Security: A Stream of Problems”. (2020). arXiv preprint arXiv:2010.16045.

[15] Edward Raff and Charles Nicholas. 2020. “A Survey of Machine Learning Methods and Challenges for Windows Malware Classification”. (2020). arXiv preprint arXiv:2006.09271.

[16] M. Christodorescu, S. Jha, “Static analysis of executables to detect malicious patterns”, USENIX Security Symposium, 2023.

[17] M. K. Shankarpani, S. Ramamoorthy, R. S. Movva, and S.Mukkamala, “Malware detection using assembly and API call sequences”, *Journal of Computer Virology,* 2(7):107-119, 2011

[18] C. Kolbitsch, et al, Effective and efficient malware detection at the end host, UNSENIX Security Symposium, 2009.

[19] Y. Ye, D. Wang, T. Li, D. Ye, and Q.Jiang, An intelligent PE-malware detection system based on association mining, *Journal in Computer Virology,* 4(4):323-334

[20] Y. Qiao, J. He, Y. Yang, and L. Ji, Analyzing malware by abstracting the frequent itemsets in API call sequences, Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 265-270, 2013

[21] R. K. Jidigam, T. H. Austin, and M. Stamp, Singular value decomposition and metamorphic detection, to appear in *Journal of Computer Virology and Hacking Techniques*

[22] P. Deshpande, Metamorphic detection using function call graph analysis, Master's report, Department of Computer Science, San Jose State University, 2013, http: //scholarworks.sjsu.edu/etd_projects/336/

[23] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

[24] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against deep learning systems using adversarial examples. Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.

[25] Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2017). Adversarial examples for malware detection. arXiv preprint arXiv:1705.07263.

[26] Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Arp, D., & Rieck, K. (2018). Adversarial malware binaries: Evading deep learning for malware detection in executables. Proceedings of the 34th Annual Computer Security Applications Conference.

[27] Feinman, Curtin, R. R., Shintre, S., & Gardner, A. B. (2017). Detecting Adversarial Samples from Artifacts. https://doi.org/10.48550/arxiv.1703.00410

[28] https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6 [Accessed May 05]

[29] https://www.analyticsvidhya.com/blog/2021/10/feed-forward-neural-networks-intuition-on-forward-propagation/ [Accessed May 05]

[30] https://www.guru99.com/backpropogation-neural-network.html [Accessed May 05]

[31] https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9 [Accessed May 05]

[32] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Accessed May 05]

[33] https://towardsdatascience.com/understanding-random-forest-58381e0602d2 [Accessed May 05]

[34] https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4 [Accessed May 05]

[35] Moustafa, Nour, et al. "A deep learning approach for malware classification using autoencoders." IEEE Access 6 (2018): 20716-20731.

[36] Patel, Priyanka, and Vaishali Patel. "Hybrid approach for malware classification." 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (2018): 1-4.

[37] Hans, Ahuja, L., & Muttoo, S. K. (2017). Detecting redirection spam using a multilayer perceptron neural network. *Soft Computing (Berlin, Germany)*, *21*(13), 3803–3814. https://doi.org/10.1007/s00500-017-2531-9

[38] Qiao, Zhang, B., & Zhang, W. (2020). Malware Classification Method Based on Word Vector of Bytes and Multilayer Perceptron. ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 1–6. https://doi.org/10.1109/ICC40277.2020.9149143

[39] S. Kia, Y. Yang, J. Wang, X. Qin, and L. Li, "A Malware Classification Method Based on Improved Deep Neural Network," in Proceedings of the 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2020, pp. 1-8.

[40] S. Jang, S. Y. Kim, and Y. Kim, "Comparing the Performance of Machine Learning Classifiers for Malware Detection Based on Different Features," in Proceedings of the 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 82-87.

[41] Kumar, R., Jaiswal, A., & Singh, S. K, Malware classification using machine learning techniques: A survey. 2017 2nd International Conference on Telecommunication and Networks (TEL-NET), 1-6.

[42] Saxena, A., Garg., D, "A survey of machine learning techniques for malware classification". 2018 4th International Conference on Computing Sciences (ICCS), 1-5.

[43] Akriti Sethi. Classification of malware models. Master's thesis, San Jose State University, Department of Computer Science, 2019. https://scholarworks.sjsu.edu/etd_projects/703/.

[44] Garcia, & MugalI, F. P. (2016). Random Forest for Malware Classification. https://doi.org/10.48550/arxiv.1609.07770

[45] Yoo, Kim, S., Kim, S., & Kang, B. B. (2021). AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. Information Sciences, 546, 420–435. https://doi.org/10.1016/j.ins.2020.08.082

[46] Bhodia, Prajapati, P., Di Troia, F., & Stamp, M. (2019). Transfer Learning for Image-Based Malware Classification. https://doi.org/10.48550/arxiv.1903.11551

[47] Kale, Di Troia, F., & Stamp, M. (2021). Malware Classification with Word Embedding Features. https://doi.org/10.48550/arxiv.2103.02711

[48] Nappa, A., Rafique, M.Z., Caballero, J.: The malicia dataset: identification and analysis of drive-by download operations. International Journal of Information Security 14(1), 15–33 (2015)

[49] Roberts, J.M.: VirusShare.com - Because Sharing is Caring. http://www.virusshare.com (2011)

[50] Microsoft Security Intelligence. BHO. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/BHO.BO, 2020.

[51] Microsoft Security Intelligence. Onlinegames. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=PWSÄ%3AWin32Ä%2FOnLineGames,2008.

[52] Microsoft Security Intelligence. Renos. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/RenosÄ&threatId=16054, 2006.

[53] Microsoft Security Intelligence. VBInject. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=VirTool:Win32/VBInjectÄ%26ThreatID=-2147367171, 2010.

[54] Microsoft Security Intelligence. Winwebsec. https://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32Ä%2fWinwebsec, 2010.

[55] Microsoft Security Intelligence. Zbot. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=win32Ä%2Fzbot, 2010.

[56] L. Asher-Dotan, “What is ZeroAccess malware,” Cybereason I Cybersecurity Software To End Cyber Attacks, 16-May-2016. [Online]. Available: https://www. cybereason.com/blog/what-is-zeroaccess-malware.

[57] Benign Dataset: https://1drv.ms/u/s!At7JwojdDuh9jL9U4DaINfbroZvH0g?e=WiLciE

[58] https://jalammar.github.io/illustrated-bert/ [Accessed May 06]

[59] V. Sanh, L. Debut, J. Chaumond, T.Wolf, ‘DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter’, ArXiv, abs/1910.01108, 2019.

[60] Quang Duy Tran and Fabio Di Troia, “Word Embeddings for Fake Malware Generation”, Department of Computer Science, San Jose State University.