Detecting Sophisticated Malware with Supervised Learning Methods

Milestone: Project Report

Group 3
Student 1: Siddharthan Singaravel
Student 2: Yue Zhou

singaravel.s@northeastern.edu
zhou.yue5@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: Siddharth S

Signature of Student 2: Yue Zhou

Submission Date: 7th April 2023

Table of Contents

Problem Setting	3
Problem Definition	3
Data Sources	3
Data Description	3
Data Exploration, Visualization, and Data Processing	4
Exploration of Candidate Data Mining Models	9
Performance Evaluation	13
Project Results	20
Impact of Project Outcomes	21
Project Challenges	21

I. Problem Setting

Malware, *malicious software*, is a virus, worm, Trojan horse, or other code-based malicious entity that successfully infects a host. With sophisticated malwares adding layers of abstraction through code encryption and compression, manual methods to detect malwares become complex and time-consuming. This warrants a need to develop *intelligent* models that differentiates benign from seemingly benign software with malicious intent.

II. Problem Definition

The motivation for this project is to utilize multiple supervised learning methods and pick the best-fit model that precisely and accurately separates malware from the rest while not compromising on time complexity which a significant proportion of present models suffer from. To that end, the memory dump obtained from the Volatile Memory Analyzer provides a set of potential features that are hypothesized to be linked with the presence of a malware. We intend to research the causal effect of the features on the response variable and predicate a solution that optimizes for accurate classifications with low-latency.

III. Data Sources

The dataset used in this project is sourced from the website of Canadian Institute for Cybersecurity (CIC), University of New Brunswick [1].

IV. Data Description

The dataset, built by the authors of the publication, intends to simulate a real-world representation of modern malware attacks. With fifty-five independent variables and a response variable, the dataset has an equal split between the classes of interest and the rest. The dataset has ~59k observations comprising spyware, ransomware, and trojan horse.

The features in the dataset can be grouped under five categories - **Malfind** (which detects potential malicious executables that are usually DLLs associated with Trojan malware), **Ldrmodule** (which gives information on potential injected code into the system), **Handles** (type of information in memory and its classification.), **Process View** (process list with information that can be used to find malicious processes), or **Callbacks** (total number of APIhooks of key types).

V. Data Exploration, Visualization, and Data Processing

By comparing aggregate and singular statistics of the dataframe, it is noted that the features of the dataset are not scaled and not appropriate to be fed into a model. Missing value ratio is 0% considering no NaN values are present in the dataset.

	pslist.nproc	pslist.nppid	pslist.avg_threads	pslist.avg_handlers	dlllist.ndlls	dlllist.avg_dlls_per_proc	handles.nhandles	handles.avg_handles_per_proc	handles.nfile
count	58596.000000	58596.000000	58596.000000	58596.000000	58596.000000	58596.000000	5.859600e+04	58596.000000	58596.000000
mean	41.394771	14.713837	11.341655	247.509819	1810.805447	43.707806	1.025858e+04	249.560958	899.119513
std	5.777249	2.656748	1.588231	111.857790	329.782639	5.742023	4.866864e+03	145.999866	3432.351200
min	21.000000	8.000000	1.650000	34.962500	670.000000	7.333333	3.514000e+03	71.139241	266.000000
25%	40.000000	12.000000	9.972973	208.725000	1556.000000	38.833333	8.393000e+03	209.648228	646.000000
50%	41.000000	15.000000	11.000000	243.963710	1735.000000	42.781524	9.287500e+03	247.208951	839.000000
75%	43.000000	16.000000	12.861955	289.974322	2087.000000	49.605280	1.219300e+04	291.355050	1080.000000
max	240.000000	72.000000	16.818182	24845.951220	3443.000000	53.170732	1.047310e+06	33784.193550	807008.000000

8 rows × 52 columns

Fig 1. Data Description Table

Categorical variables have been converted to numerical variables. Distribution of the response variable is split equally between malignant and benign classes.

Standardization allows the features of the dataset to be present within a similar range.

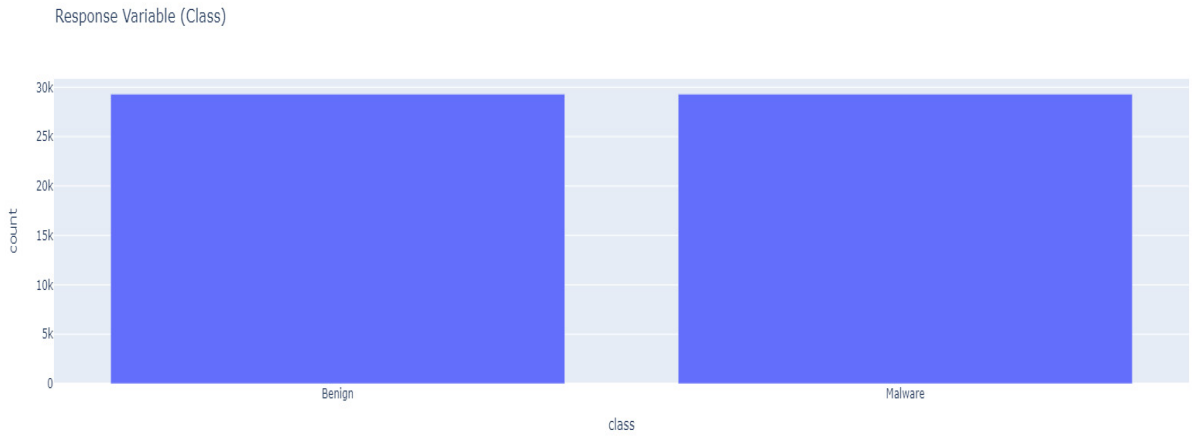

Fig 2. Class Distribution

Distribution Plot

Distribution plot of a select features under handles, psxview, and pslist categories show a skewed/bimodal distribution of the features.

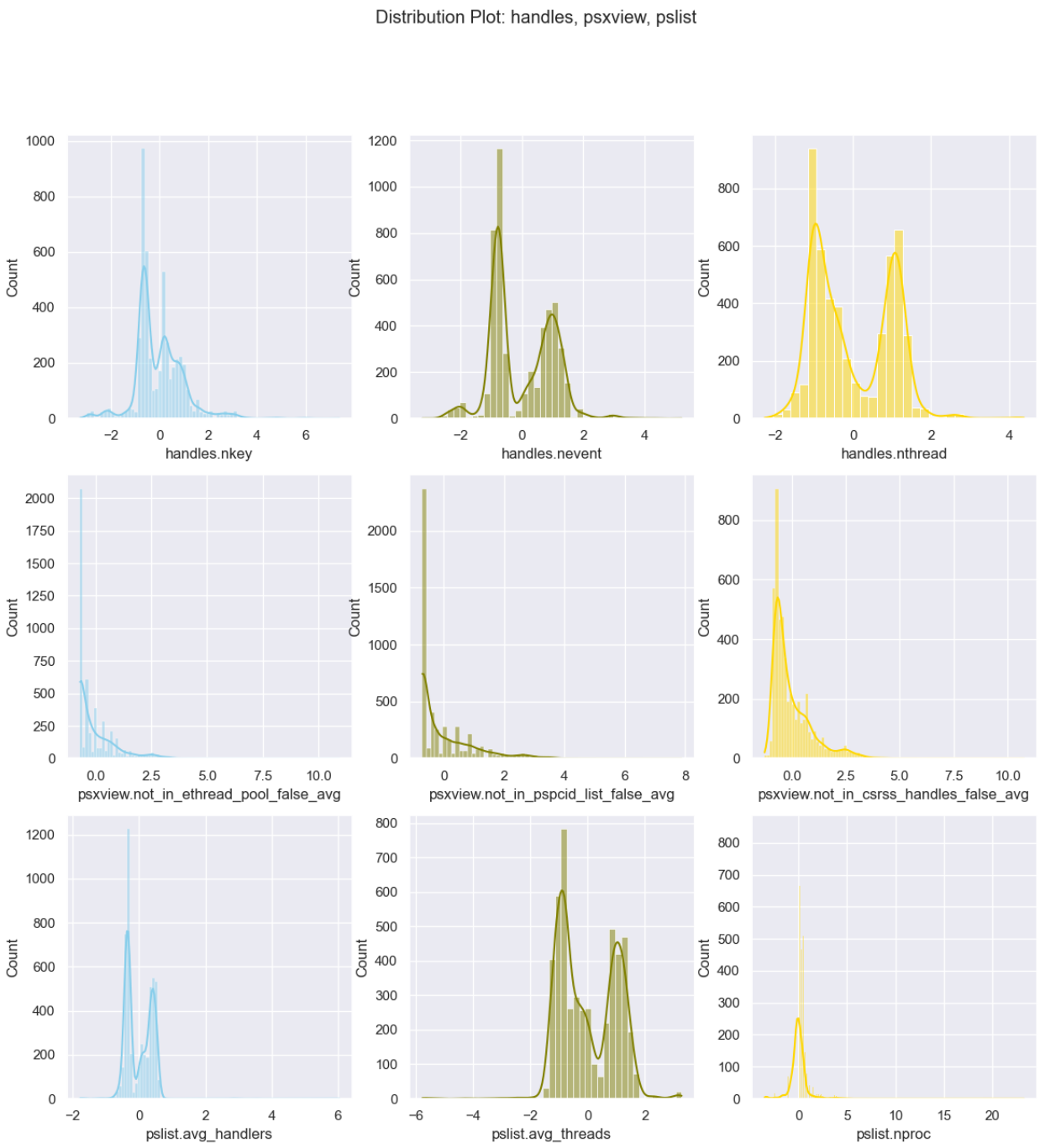

Fig 3. Distribution Plot

High Covariance Filter

By developing a correlation matrix, we observe the linear relationships among the features of the dataset. High correlation implies redundant storage of information; hence the dimension of the dataset can be reduced using a high correlation filter. Variables under the handles, psxview, and svscan exhibit high correlation amidst themselves.

Fig 4. Correlation Heatmap

With a correlation filter threshold of 0.90, 29 features of the dataset can be removed to reduce data sparsity. Table below shows the features that can be dropped. At the Modeling stage, the most optimal threshold can be picked.

Correlation Filter Threshold	Droppable Features
0.80	35
0.85	33
0.90	29
0.95	21

Table 1. Correlation Filter Threshold

Low Variance Filter

Variance of almost all features is distributed in a single band, and so a low variance filter would not be the best method to enhance feature selection.

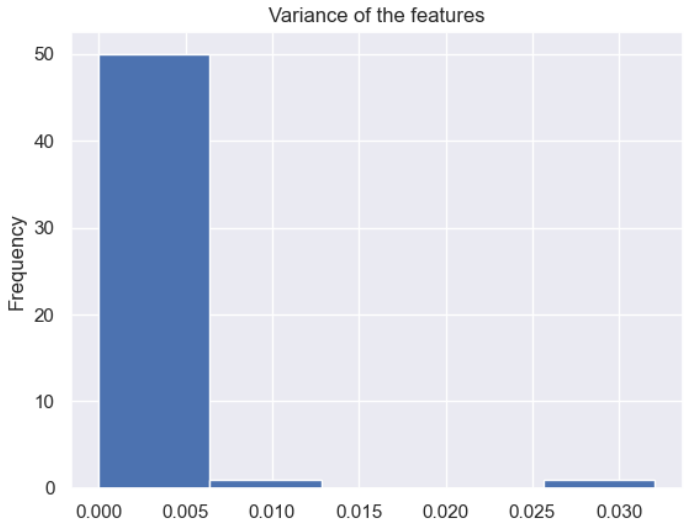

Fig 5. Variance (Histogram) of the features

Principal Component Analysis (PCA)

We use Principal Component Analysis to reduce the dimensions of our dataset. By varying the *n_components* hyperparameter, we see the proportion of variance captured in the linearly transformed components. However, the lack of interpretability of the new components is a drawback. In the below figure, we chart the variance captured against the number of components used.

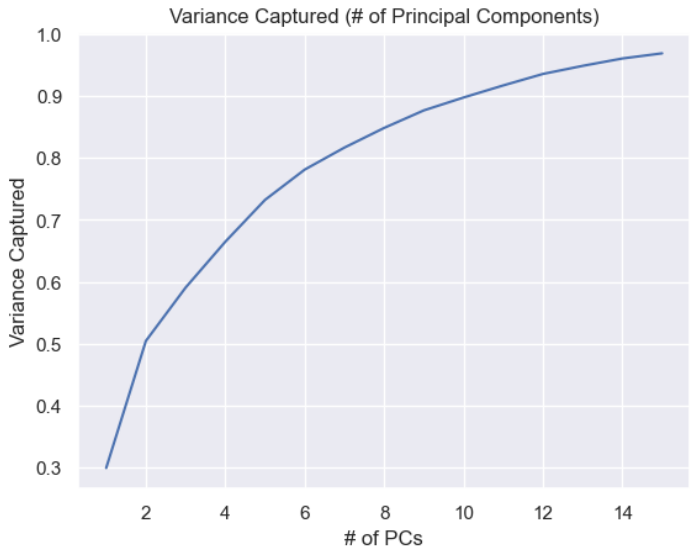

Fig 6. Variance captured (# of Principal Components)

With just 10 components, we can capture 90% of the variance of the original 54-feature dataset. The below plot outlines the classification with just two Principal Components that captures nearly half of the total variance of the data matrix.

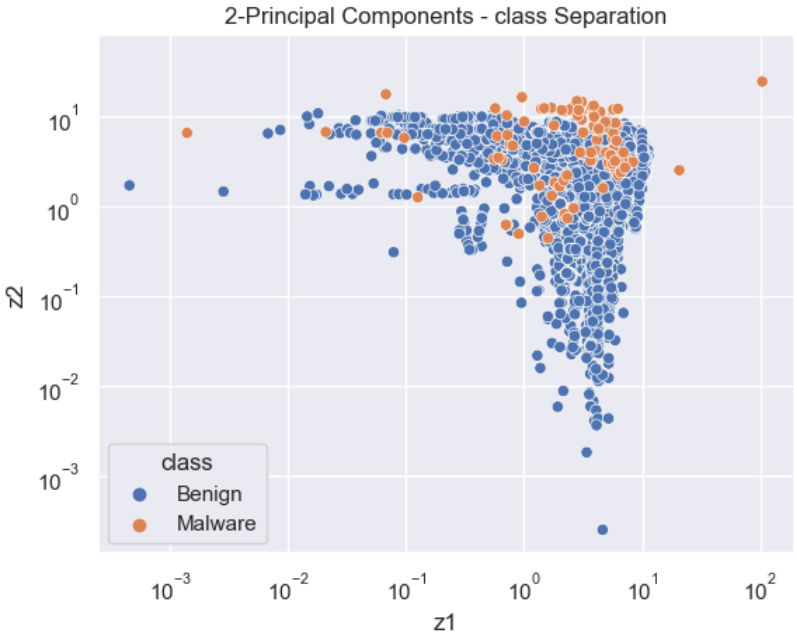

Fig 7. 2-Principal Component Classification

VI. Exploration of Candidate Data Mining Models

Data Partitioning

Data was standardized and pre-processed in the previous exercise. Using PCA, the number of features were reduced to 15 retaining 97% of the dataset's variability. The normalized dataset was proportioned with a 70:30 train-test ratio randomly. X_train (Training data matrix) has 41,017 records whereas X_test (Testing data matrix) has 17,579 records.

Candidate Mining Models

We have used six machine-intelligent classification algorithms to train the classifier.

1. Logistic Regression

Logistic regression models the classifier using the sigmoid function, also known as the logistic function. Logit can describe the relationship between a dependent categorical variable (binary, in this case), and multiple nominal independent variables.

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$cost(w) = (-1/m) \sum_{i=1}^{i=m} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

$$w_i = w_j - (\alpha * dw_j)$$

where the two parameters β0 and β1 denote intercept and inverse scale parameter respectively. A base model developed with sklearn generates the below confusion matrix for the test dataset.

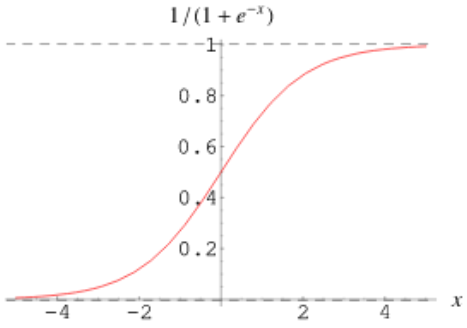

Fig 8. Sigmoid Function

Logistic regression is interpretable, easy to implement, fast, and makes no assumptions about the dataset. The downsides with logit regression is its assumption to construct linear boundaries. The independent variables may not be linear with respect to the response variable which is an assumption made for the logistic regression.

2. k-Nearest Neighbors

k-NN is a non-parametric supervised learning method for both classification and regression. Class membership of a record is dependent on the majority vote of the neighbors; the record is classified to the class among its k nearest neighbors. kNN makes no assumptions about the data and is easily extensible to multiple classes. The downside is that it might become computationally expensive. Distance can be any of the norms. In practice, it is mostly Manhattan/Mahalanobis.

k-NN has a simple implementation considering that it does not have a dedicated training phase, and adding new data would not affect the model. With large models, distance calculations become tedious and computation heavy. The algorithm is also susceptible to noise and outliers.

3. Decision Trees

Decision trees can be developed for both regression and classification tasks. The objective of decision trees is to infer decision rules based on the training data starting from the base (root) of the data. Trees generate discernible rules and are computation efficient.

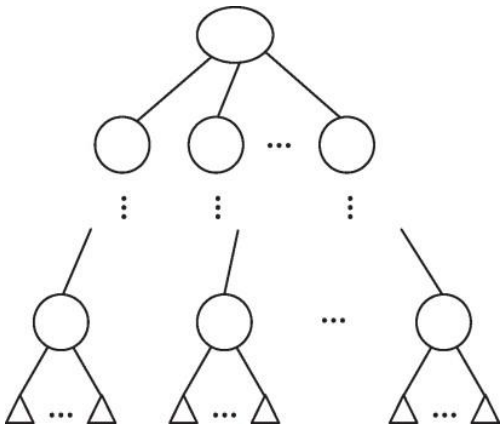

Fig 9. Root and Nodes Tree Structure

We have implemented the CART model from sklearn as part of the classification. Decision trees does not require a normalized/standardized data to be fed. Decision tree models are easily interpretable. On the downsides, it takes a significant amount of time to compute the trees.

4. Random Forest

Random decision forests is an ensemble learning method for prediction as well classification. An advantage over decision trees is the ability of the model to alleviate the effect of overfitting observed from decision trees. Random forests work by generating multiple decision trees and then merging them for higher accuracy (by majority voting).

Since the model has to work with more trees, the algorithm is memory intensive.

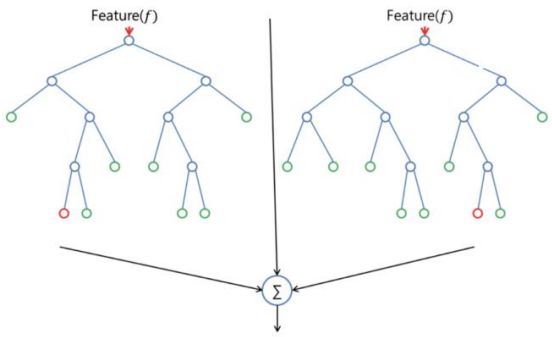

Fig 10. Merging of Decision Trees

Random forests is able to handle missing values. Also, the dataset doesn't have to be normalized for the algorithm to work. But the model creates a lot of trees, thereby increasing complexity.

5. Support Vector Machines (SVM)

SVM works by maximizing the width of the gap between two classes to be classified. By defining a hyperplane (which acts as the decision boundary), objects are assigned to classes based on where they fall. SVM is less prone to overfitting.

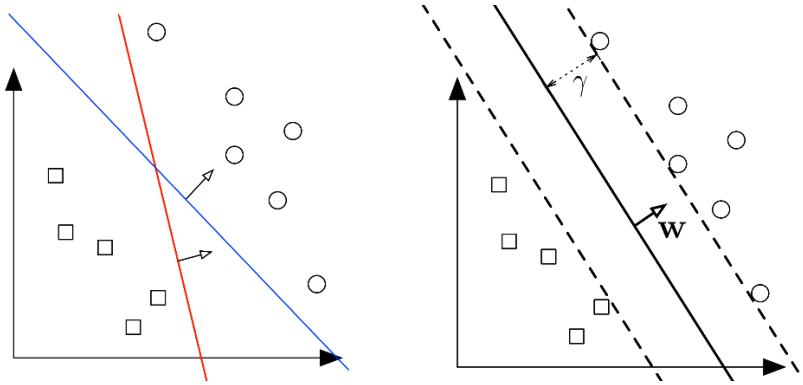

Fig 11. Max Margin Support Vectors

SVM is dimension agnostic and are very effective in high dimensional spaces but is a computation heavy algorithm.

6. Gaussian Naïve Bayes (GNB)

A variant of Naïve Bayes, this assumes normal distribution of the continuous features (likelihood) in the data matrix. The decision boundary for GNB is a linear function (hyperplane).

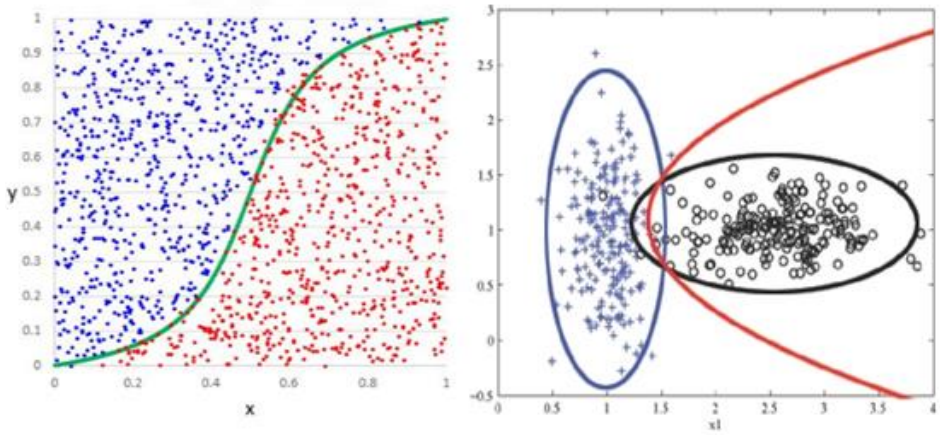

Fig 12. Gaussian Distribution of Binary Class

VII. Performance Evaluation

1. k-Nearest Neighbors

By varying the hyperparameter (k) for this discriminative machine learning model, we vary the number of neighbors for assimilating a point to its class by adopting either ball_tree or kd_tree algorithm. Euclidean L2 norm has been employed to calculate the point distances. k-NN estimates a near perfect model with the highest recall being 0.993 until 5 neighbors.

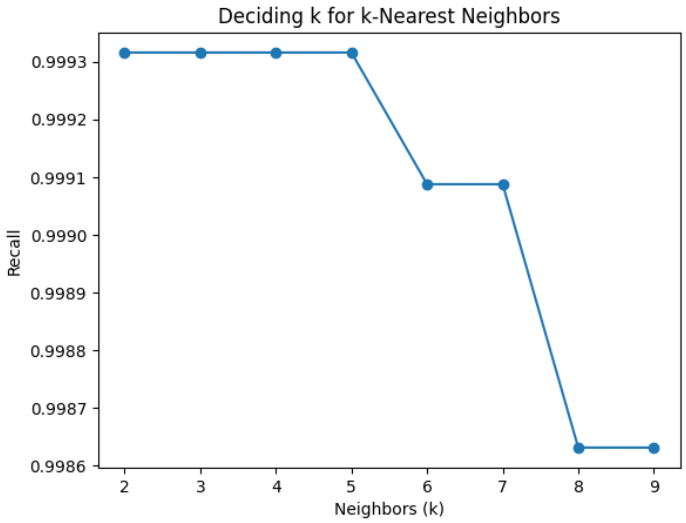

Fig 13. Varying k for k-Nearest Neighbors

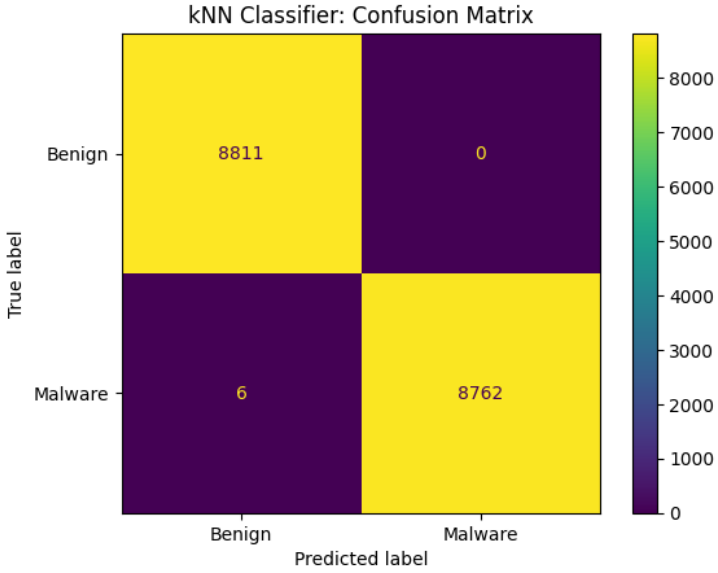

Fig 14. Confusion Matrix: kNN Classifier

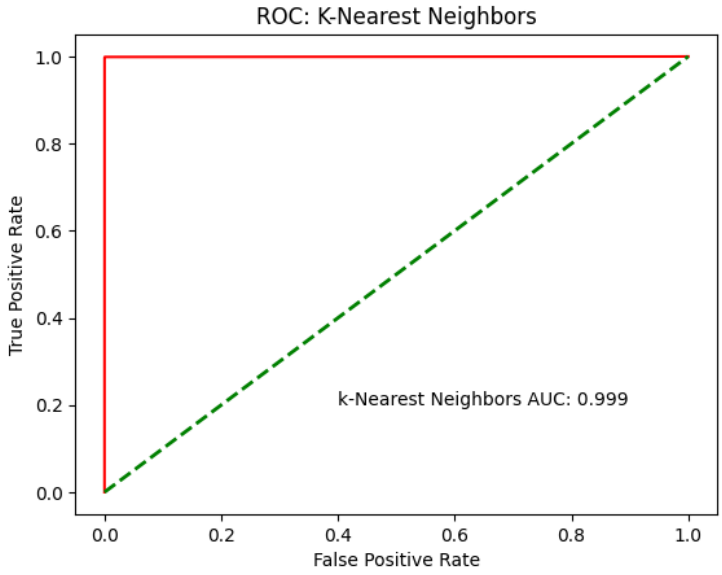

Fig 15. ROC Curve: k-NN

2. Decision Trees

One hyperparameter for Decision Trees is the max depth of the model. The highest recall for this model is obtained at max_depth = 7, impurity_method = 'gini'. The ROC curve is near-perfect with an AUC of 0.999.

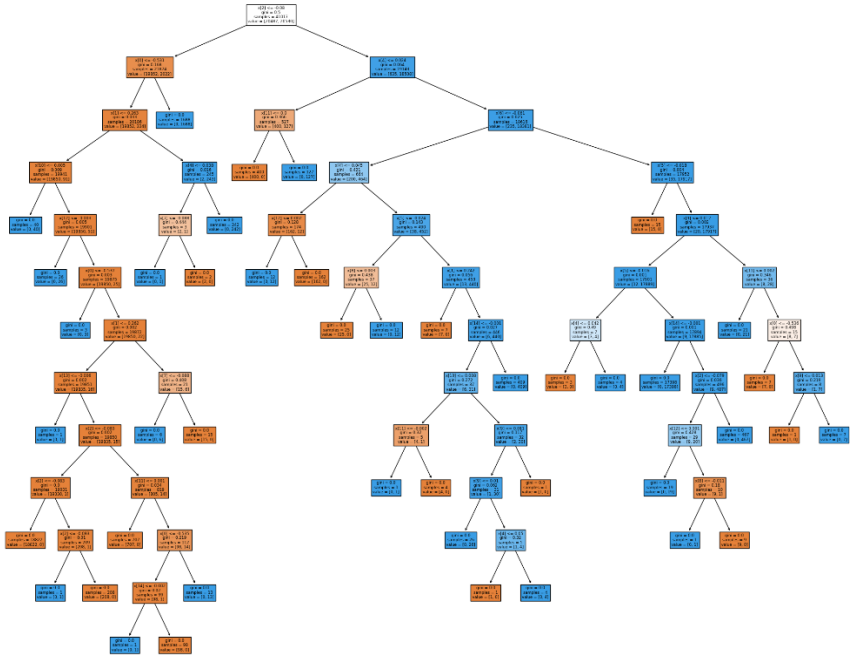

Fig 16. Decision Tree for ObfuscatedMalware Model (Default)

Max Depth	Recall
2	0.97582
3	0.99487
4	0.99692
5	0.99795
6	0.99875
7	0.99886

Table 2. Varying Max Depth for Decision Trees

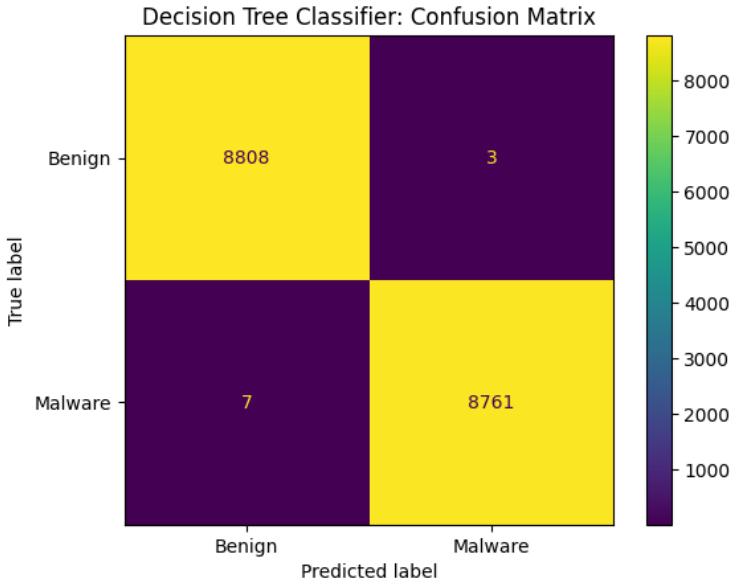

Fig 17. Confusion Matrix: Decision Trees

Decision Trees provides the best classification metrics for the ObfuscatedMalware dataset with only 10 misclassifications out of ~17,000 observations.

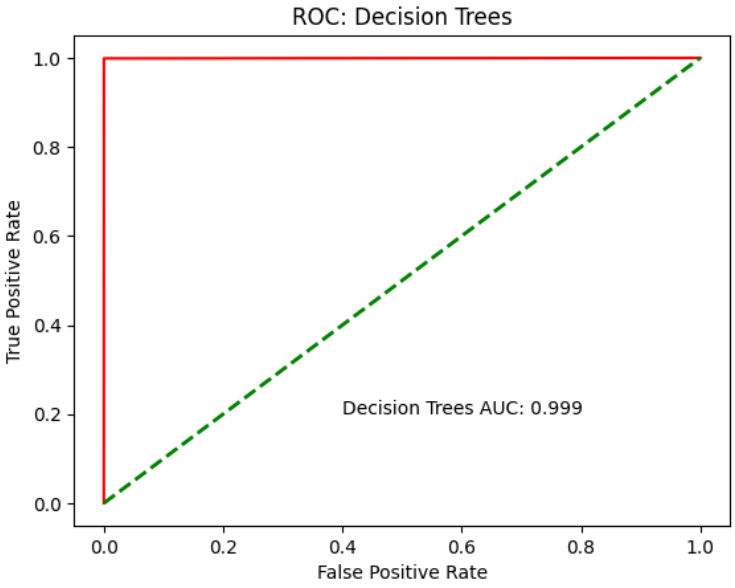

Fig 18. ROC Curve: Decision Trees

3. Gaussian Naïve Bayes

Gaussian Naïve Bayes classifier, a continuous variant of Naïve Bayes that assumes the features of the data matrix are independent shows the poorest classification metrics. This model assumes the features are distributed normally with the feature's mean and standard deviation.

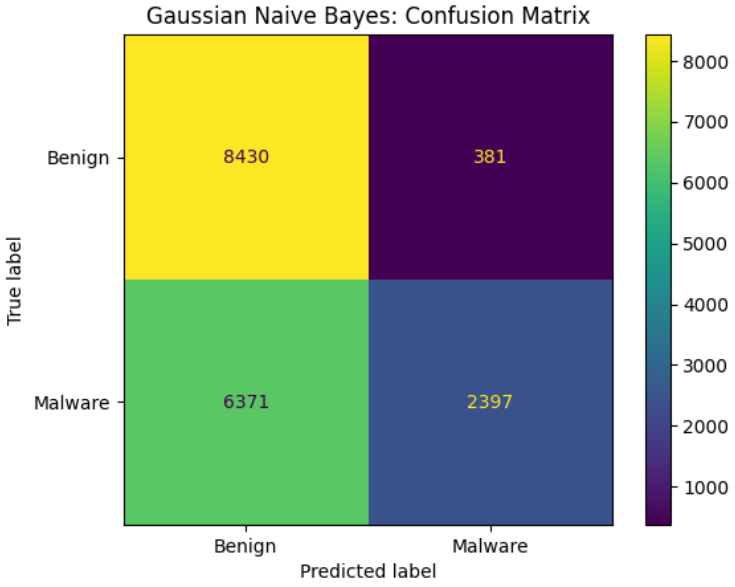

Fig 19. Confusion Matrix: Decision Trees

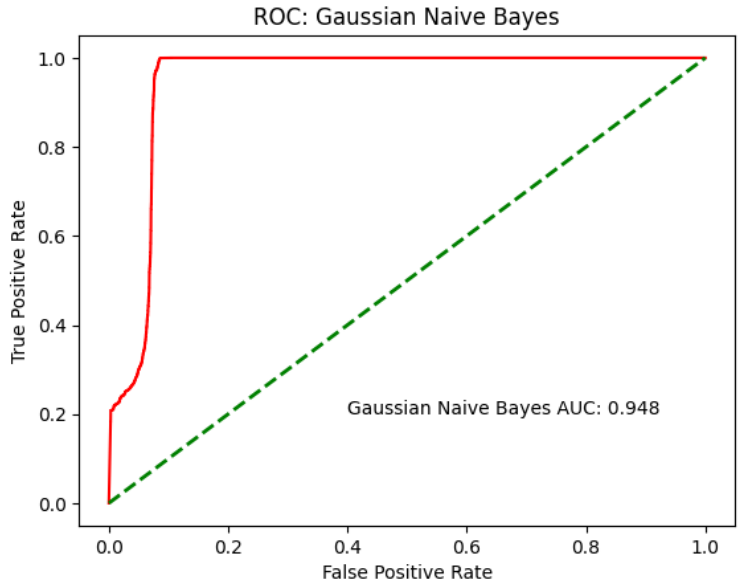

Fig 20. ROC: Gaussian Naïve Bayes

4. Random Forest Classifier

Random decision forests, an extension of decision trees, is an ensemble learning metho. By varying max_depth, we can fine-tune performance. At higher max_depth, the model produces perfect precision scores.

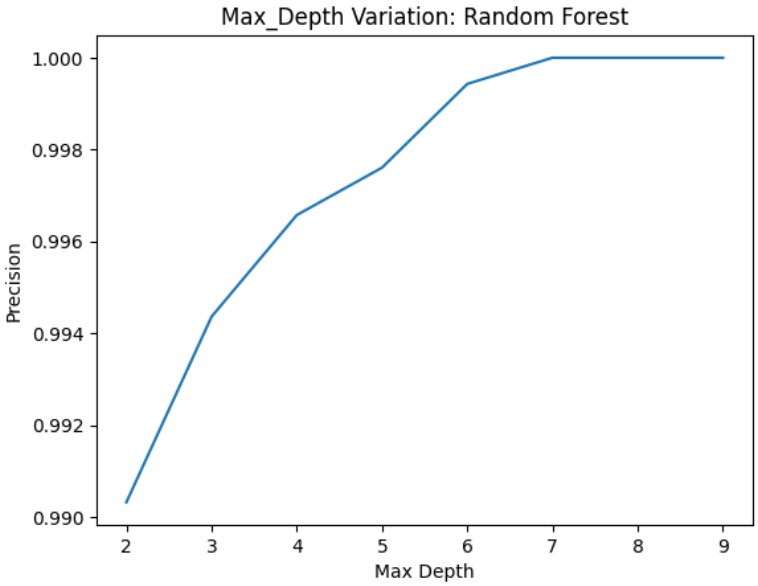

Fig 21. Max_Depth Variation

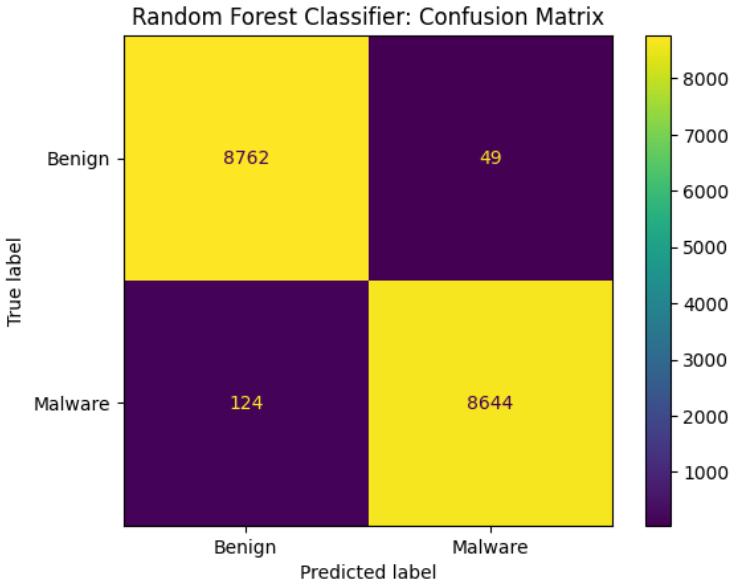

Fig 22. Confusion Matrix: Random Forest

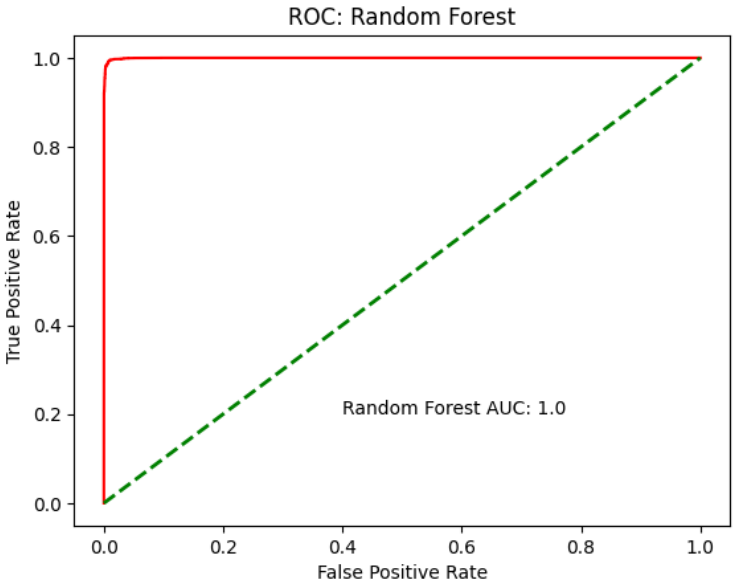

Fig 23. ROC: Random Forest

5. Logistic Regression

Logistic regression works by rewarding points that are further away from the decision boundary, as quantified by a sigmoid function (exponential). In our case, logistic regression is a simple and robust method with decent classification performance.

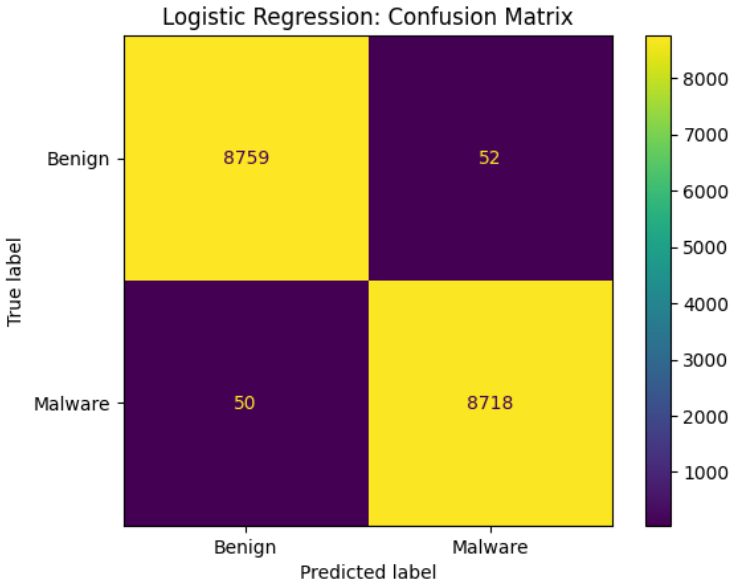

Fig 24. Confusion Matrix: Logistic Regression

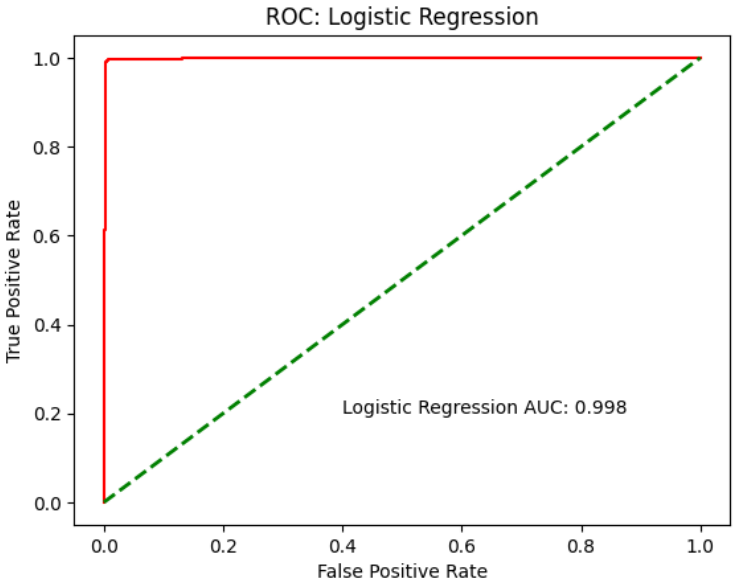

Fig 25. ROC: Logistic Regression

6. Neural Networks

With ReLU activation function, Adam solver, and an adaptive approach towards the learning rate of the neural network produced the best metrics for the model.

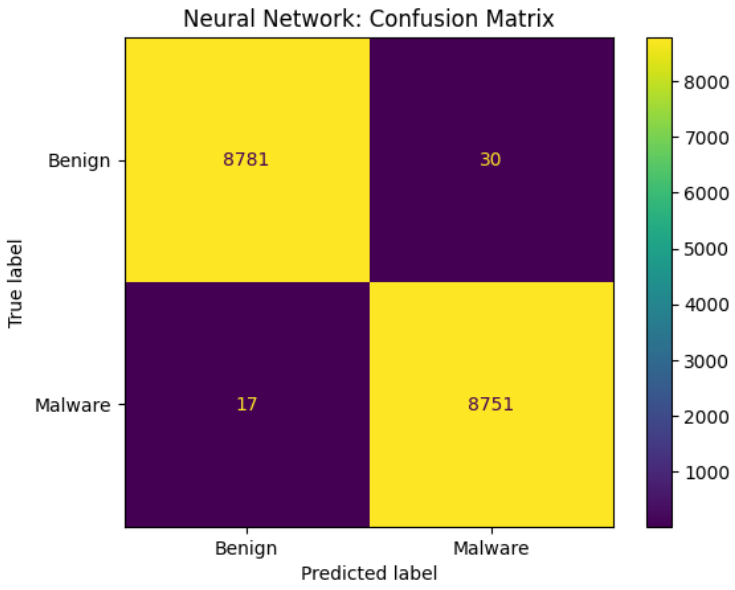

Fig 26. Confusion Matrix: Neural Networks

VIII. Project Results

Post inclusion of Support Vector Machines (Dual, Convex Optimization), the table below shows the performance evaluation metrics for the machine intelligent models that have been employed to detect malwares from benign software.

Model	accuracy	f1	precision	recall
LogisticRegression	0.994198	0.994184	0.994071	0.994297
k-NN	0.999659	0.999658	1	0.999316
DecisionTree	0.999374	0.999373	0.999658	0.999088
RandomForest	0.990159	0.990092	0.994363	0.985858
SupportVectors	0.999829	0.999829	1	0.999658
NeuralNets	0.997725	0.99772	0.997379	0.998061

Table 3. Summary of Models

Though most models were accurate in their predictive capabilities, we chose to go with Logistic Regression for its simplistic and computation friendly attributes. Incorporating orthogonal transformation prior to developing the model helped reduce data sparsity despite loss in data interpretability.

IX. Impact of Project Outcomes

The project objective is to detect obfuscated malware from benign software with data from synthesized features hypothesized to be impacted by presence of malwares. As a well-defined classification problem with binary classes and 50+ inter-dependent features, dimensionality reduction and correlation analysis as pre-processing techniques are required to improve model fitting.

Most models were accurate in their predictive capabilities, but the Logistic Regression model proves to be advantageous in terms of computation cost and time efficiency. k-Nearest Neighbors and Support Vector Machines producing a perfect precision of 1 are next in consideration. Gaussian Naïve Bayes has the poorest performance metrics considering its normally distributed feature assumptions.

X. Project Challenges

High number of features in the dataset could lead to sparsity and thus overfitting. Collecting memory dump from various sources and features is an expensive task. To keep the model updated, the task needs to be performed on a recurring basis.

Conformity within the target class is a problem. Most of the features that are present in the matrix are localized to a specific class of interest which may induce imbalance and/or overfitting. A new record with unique set of characteristics could lead to it being classified a false negative.