Journal of Applied Science and Education, 2024,
Vol. 04, Iss. 03, S. No. 071, pp. 1-13
ISSN (Online): 2583-1372

Ransomware Detection using ML

Ujjwal Singh[1], Pawan Singh[2]

[1,2]Amity School of Engineering & Technology, Amity University Uttar Pradesh, Lucknow, India
singhujjwal161@gmail.com[1], pawansingh51279@gmail.com[2]

How to cite this paper: U. Singh and P. Singh "Ransomware Detection using ML," *Journal of Applied Science and Education (JASE)*, Vol. 04, Iss. 03, S. No. 071, pp 1-13, 2024.

https://doi.org/10.54060/a2zjournals.jase.71

Received: 06/08/2024
Accepted: 20/10/2024
Online First: 25/11/2024
Published: 25/11/2024

Abstract

This study investigates the effectiveness of three machine learning classifiers—Random Forest, SVM, and XGBoost—in detecting benign files using a dataset of file features. The dataset is cleaned by removing rows with missing values and duplicates, followed by feature scaling with StandardScaler. Correlation heatmaps and confusion matrix visualizations are used to explore data relationships and model performance. The classifiers are trained and evaluated based on accuracy, precision, recall, F1-score, and ROC-AUC. Results indicate the comparative performance of each model, highlighting their strengths and weaknesses in distinguishing between benign and non-benign files. This comprehensive approach provides insights into the most suitable classifier for this specific detection task.

Keywords

Machine Learning, Random Forest, Support Vector Machine (SVM), XGBoost, Correlation Analysis, Cybersecurity.

1. Introduction

Ransomware, a particularly harmful type of malware, has emerged as a formidable adversary in the digital realm, wreaking havoc on individuals, businesses, and governments alike. This insidious software encrypts files or locks users out of their systems, preventing them from accessing their data until a ransom, typically in cryptocurrency, is paid. Ransomware has progressed from simple, opportunistic attacks to sophisticated, highly targeted campaigns orchestrated by cybercriminals with malicious intent. The consequences of ransomware are staggering, with victims suffering not only financial losses from ransom payments, but also reputational damage, operational disruptions, and legal liabilities. Ransomware, which has become a pervasive and persistent threat in the cybersecurity landscape, affects all entities, from small businesses to multinational corporations. Traditional defense mechanisms, such as antivirus software and firewalls, are frequently ineffective against

ransomware, which uses sophisticated evasion techniques to avoid detection. As a result, organizations are increasingly turning to novel approaches, such as machine learning, to strengthen their defenses against ransomware attacks [1].

Machine learning, a subset of artificial intelligence, holds promise in cybersecurity because it allows computers to analyze massive amounts of data and identify patterns that indicate malicious activity. Researchers and cybersecurity professionals can develop robust models capable of detecting ransomware with high accuracy by training machine learning algorithms on a variety of datasets that include both benign and malicious samples. In this report, we'll look at machine learning techniques for detecting ransomware. To train and evaluate multiple machine learning models, we use a comprehensive dataset that includes features extracted from ransomware samples as well as legitimate software. Our goal is to evaluate the effectiveness of various algorithms, such as Random Forest, Support Vector Machine (SVM), and XGBoost, in distinguishing ransomware from benign software [7][6].

The purpose of this report is to thoroughly investigate and assess the effectiveness of various machine learning algorithms, such as Random Forest, XGBoost, and Support Vector Machine (SVM), in detecting ransomware. The report aims to accomplish the following objectives through a comparative analysis of these algorithms:

- Comprehending Ransomware Characteristics: Offer a synopsis of the unique attributes and actions of ransomware, enabling a more profound comprehension of the threat environment and the difficulties linked to ransomware identification.
- Examining Machine Learning Techniques: Investigate the use of Random Forest, XGBoost, and SVM, three machine learning algorithms, in ransomware detection. Analyze each algorithm's fundamental ideas, advantages, and disadvantages in relation to cybersecurity.
- Algorithm Implementation and Training: Using pre-prepared datasets, implementing and train the Random Forest, XGBoost, and SVM algorithms. Examine evaluation metrics, model optimization, and parameter tuning to determine how well each algorithm performs in terms of computational efficiency, false positive rate, and detection accuracy.
- Comparative Analysis and insights: Examine how well the Random Forest, XGBoost, and SVM algorithms perform in terms of performance metrics. Determine the advantages, disadvantages, and trade-offs of each algorithm, offering information to guide the choice of algorithms and optimization techniques for ransomware detection.

2. Literature Review

Signature-based detection techniques identify and stop malicious files or activities by using predefined patterns or signatures of known ransomware strains. Usually, these signatures are created using attributes related to known ransomware variants, like file names, hashes, and encryption algorithms. Although efficient against known threats, signature-based methods are reactive by nature and cannot identify zero-day or previously unseen attacks, leaving them open to new strains of ransomware and polymorphic malware.

In contrast, heuristic-based detection techniques examine the features and actions of files or processes to spot possible ransomware activity. These techniques use rules or heuristics that are based on behaviors that can be observed, like file encryption, file modification, and patterns of network communication, to identify suspicious activity that could be a sign of ransomware activity. Heuristic-based techniques are more flexible and adaptive than signature-based techniques, but they can also produce false positives and have trouble telling the difference between malicious and benign activity, which could present operational difficulties for cybersecurity teams. Both signature-based and heuristic-based detection techniques are essential to ransomware mitigation and detection strategies, despite their differences. Signature-based techniques are especially good at thwarting known threats and offer a dependable way to identify known ransomware variants. However, they are ineffective against zero-day attacks and have limitations due to their reliance on predefined signatures.

Machine learning techniques offer a data-driven approach to ransomware detection, using algorithms to extract patterns

and features from large datasets of both benign and malicious samples. Unlike traditional methods that rely on predefined rules or signatures, machine learning models can adapt and generalize to new and evolving ransomware variants based on previously learned patterns and features. Several machine learning-based approaches have been proposed to detect ransomware, including:

- Feature-Based Detection: To train machine learning models for ransomware detection, feature-based techniques extract pertinent features from files or processes, such as file metadata, API calls, and opcode sequences. These characteristics allow the models to discern between malicious and benign samples by capturing the behavior and traits of ransomware. Gradient boosting machines, random forests, and decision trees are popular algorithms for feature-based detection.
- Anomaly Detection: These methods look for departures from the norm and mark them as possibly ransomware activity. Neural networks, clustering, and classification are examples of machine learning algorithms that can learn the typical patterns of system behavior and identify anomalies that point to ransomware attacks. While anomaly detection techniques can be especially useful in identifying novel or zero-day ransomware variations, they may need a lot of labeled training data to function at their best.
- Ensemble Learning: To increase the precision and resilience of ransomware detection systems, ensemble learning blends several machine learning models. To improve classification accuracy and lower false positives, methods like AdaBoost, Gradient Boosting, and Random Forest combine the predictions of individual models. Ensemble learning techniques have proven successful in reducing the negative effects of class imbalance and enhancing ransomware detection systems' overall functionality.

3. Machine Learning Algorithms for Ransomware Detection

3.1. Random Forest Classifier

Random Forest is a powerful ensemble learning method that can be used for both classification and regression in machine learning. During training, it generates many decision trees and outputs their mode (for classification) or mean prediction (for regression) [2]. Here's an explanation of how Random Forest works:

- Decision Trees: Random Forest is built on the concept of decision trees. Decision trees are hierarchical structures with internal nodes representing feature-based decisions and leaf nodes representing class labels or numeric values (in regression).
- Bagging: Random Forest uses a technique called bagging (Bootstrap Aggregating). Bagging is the process of creating multiple subsets of the training dataset using random sampling with replacement (bootstrap samples) and then training each decision tree on a separate subset.
- Random Feature Selection: In addition to using bootstrapped datasets, Random Forest adds randomness to the decision tree by taking only a random subset of features at each split. This helps to decorate the trees and increase diversity, resulting in a more robust and accurate model.
- Voting or Averaging: Once all of the trees have been built, for classification tasks, each tree "votes" for a class, and the class with the most votes becomes the Random Forest prediction. For regression tasks, the average of all three predictions is used to calculate the final prediction.

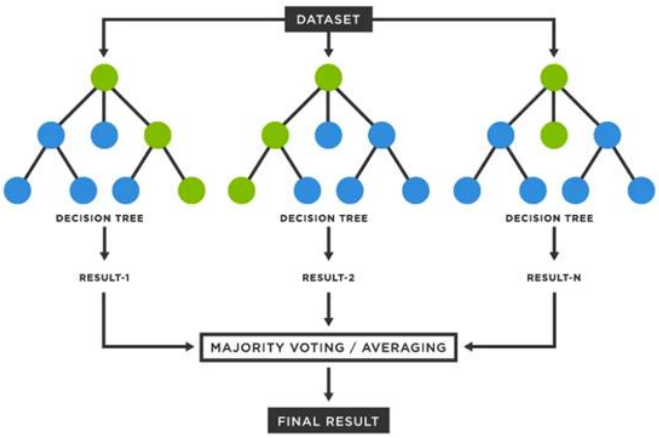


Figure 1. Random Forest

3.2. Support vector Machine (SVM)

The Support Vector Machine (SVM) is a versatile and powerful supervised machine learning algorithm used for classification and regression. It is especially useful for binary classification problems, where the goal is to divide data points into two classes based on their characteristics. SVM seeks to identify the optimal hyperplane that best separates the classes in the feature space while maximizing the margin between them. At its core, SVM works by determining the decision boundary (hyperplane) that maximizes the margin between the support vectors, which are the data points closest to the decision boundary for each class. The hyperplane is defined by a set of weights (coefficients) and a bias term, with the goal of determining the weights that maximize the margin while correctly classifying as many data points as possible.

One of the key concepts in SVM is the maximum margin, which refers to the distance between the decision boundary and the nearest data points in each class. Maximizing the margin improves the model's generalization and robustness against noise and outliers. SVM seeks to identify the decision boundary that not only separates the classes but also maximizes the margin. When the classes are not linearly separable in the original feature space, SVM can use the "kernel trick" to map the data to a higher-dimensional space where linear separation may be possible. Polynomial kernels, radial basis function (RBF) kernels, and sigmoid kernels are all common kernel functions used in support vector machines. These kernels enable SVM to capture complex relationships between features and handle non-linear separation tasks efficiently.

Regularization, which discourages big coefficients during optimization and helps avoid overfitting, is another idea incorporated into SVM. The trade-off between maximizing the margin and minimizing the classification error is adjusted by the regularization parameter (C), which also sets the penalty for misclassification. While a larger value of C decreases misclassifications but may produce a narrower margin, a smaller value of C permits a wider margin but may result in more misclassifications. The optimal hyperplane that maximizes the margin and minimizes the classification error is the goal of the SVM optimization problem. Usually, methods like gradient descent, quadratic programming, or convex optimization are used to solve this kind of optimization problem. By analyzing which side of the hyperplane new data points fall on, SVM can classify them once the ideal hyperplane has been identified during the training phase.

All things considered, SVM is renowned for its efficiency in high-dimensional spaces, resilience to overfitting, and capacity to manage tasks involving both linear and non-linear separation. It is a well-liked and adaptable machine learning algorithm with applications in a wide range of fields, such as finance, bioinformatics, image recognition, and text classification [2].

3.3. XGBoost (Extreme Gradient Boosting)

XGBoost, which stands for eXtreme Gradient Boosting, is a sophisticated and frequently used machine learning algorithm that excels at handling structured/tabular data accurately, scalably, and efficiently for both regression and classification tasks [4].

Its outstanding performance is the result of a combination of various crucial elements and techniques:

- Gradient Boosting Framework: XGBoost functions inside the gradient boosting framework, an ensemble learning technique in which a series of weak learners—typically decision trees—are gradually trained to correct mistakes made by the models that came before them. Their predictions are combined in an iterative process to create a strong learner. The core of the algorithm is the iterative minimization of residuals, or the differences between expected and actual values, which refines the model's predictions.
- Decision Trees as Base Learners: Decision trees are the main base learners in XGBoost because of their ease of use and interpretability. The purpose of these shallow trees is to prevent overfitting while also making the model more intricate and precise. In order to maximize the performance of the model, XGBoost builds decision trees level-wise, sequentially determining the feature that offers the most informative split at each node.
- Regularization: Regularization techniques are incorporated into XGBoost in order to prevent overfitting and improve the model's generalization ability. To deter overly complex models, regularization penalties are incorporated into the loss function. L1 regularization (Lasso) and L2 regularization (Ridge) are two common regularization terms that penalize models according to the magnitudes of the coefficients and encourage simpler models.
- Gradient Descent Optimization: XGBoost uses gradient descent optimization to reduce a particular loss function that measures the difference between the actual and predicted values. Iteratively updating these parameters to minimize the loss involves computing the gradients of the loss function with respect to the model's parameters (e.g., tree structure and leaf scores). The predictive accuracy of the model is improved as a result of this iterative refinement process, which converges toward an ideal solution.
- Tree Pruning: To reduce overfitting, XGBoost uses a method called tree pruning to regulate the size and complexity of each decision tree. Pruning is the process of eliminating splits that have little effect on improving the performance of the model. XGBoost generates models with improved generalization to new data by streamlining the trees.
- Feature Importance: XGBoost provides a method for determining the importance of features in predicting the target variable. It assigns important scores based on the frequency with which features are used throughout all boosting rounds. This capability allows users to identify the most influential features in a dataset, facilitating insights and informed decision-making.
- Scalability and Efficiency: Notably, XGBoost is designed for scalability and efficiency, making it ideal for processing large datasets containing millions of samples and features. It uses parallel and distributed computing paradigms to harness the computational power of multicore CPUs and distributed computing frameworks such as Apache Spark. This scalability ensures XGBoost's efficiency and effectiveness across a wide range of datasets and computing environments.

4. Methodology

4.1. Data Collection

The dataset used in this research is made up of several features that have been taken out of files with the intention of identifying ransomware. The dataset contains samples of ransomware as well as benign files [1]. The features are different executable file attributes and characteristics, including size, metadata, version information, and debugging, exporting, and importing characteristics. The binary indicator "Benign," which is the target variable, most likely indicates whether each file is malicious (1) or benign (0). The dataset was obtained from kaggle.com.1) For author/s of only one affiliation: To change the default, adjust the template as follows.

4.2. Data analysis and Visualization

The 'pairplot' function in Seaborn is indeed a powerful tool for data visualization. It gives users an easy way to see the relationships between various variables in a dataset by enabling them to create a grid of pairwise plots. It is simple to spot patterns, correlations, and trends because each plot in the grid shows the relationship between two variables.

By default, Seaborn creates scatter plots for every pair of variables when utilizing the 'pairplot' function. These scatter plots show the marginal histograms for each variable along the diagonal of the grid, as well as the relationship between the variables. Pairwise relationships and the distributions of individual variables can both be comprehensively visualized with this arrangement. The 'pairplot' function can also be used to explore relationships in both numerical and categorical data because it allows the integration of categorical variables using different plot styles or color-coded markers.

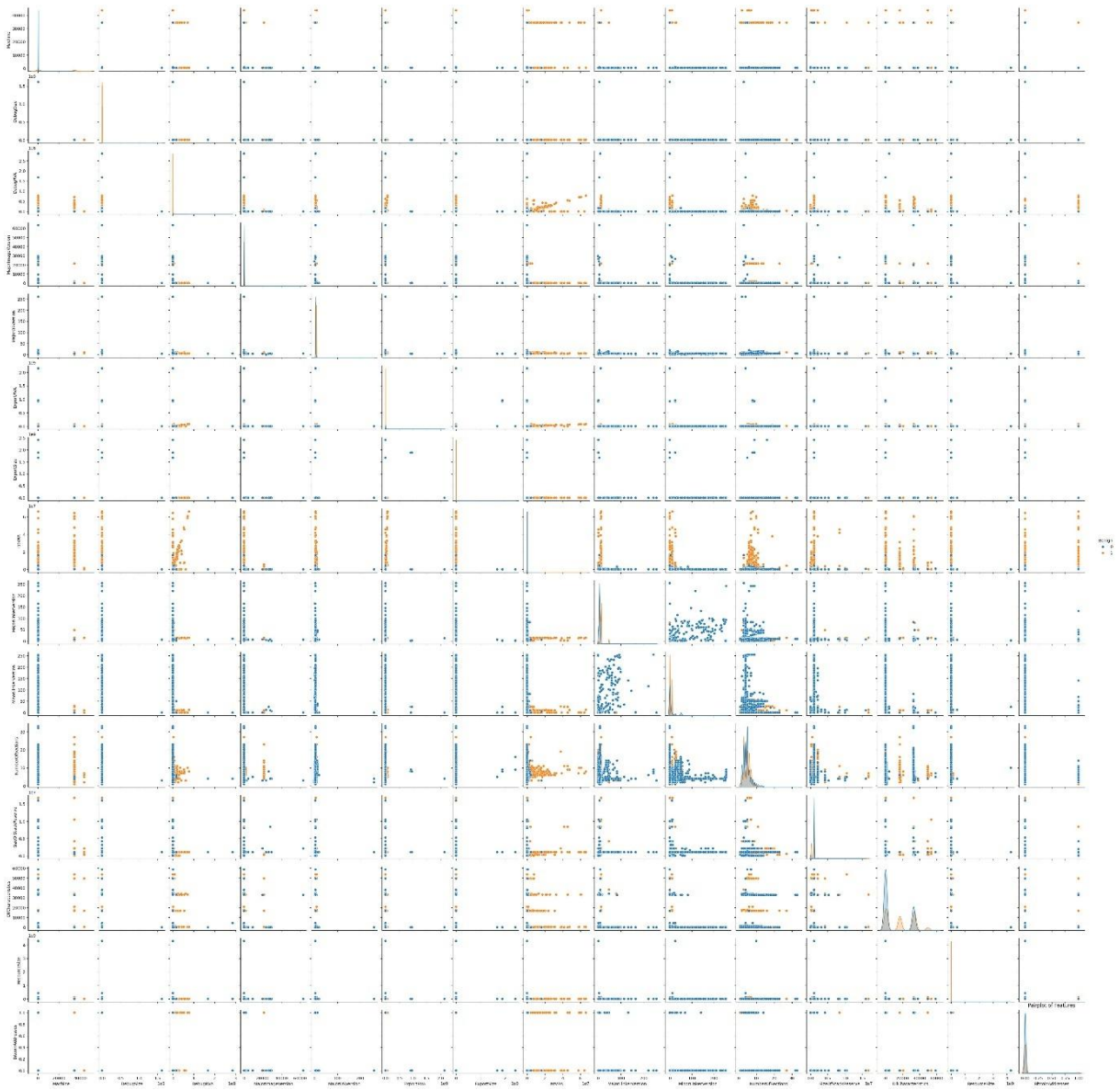
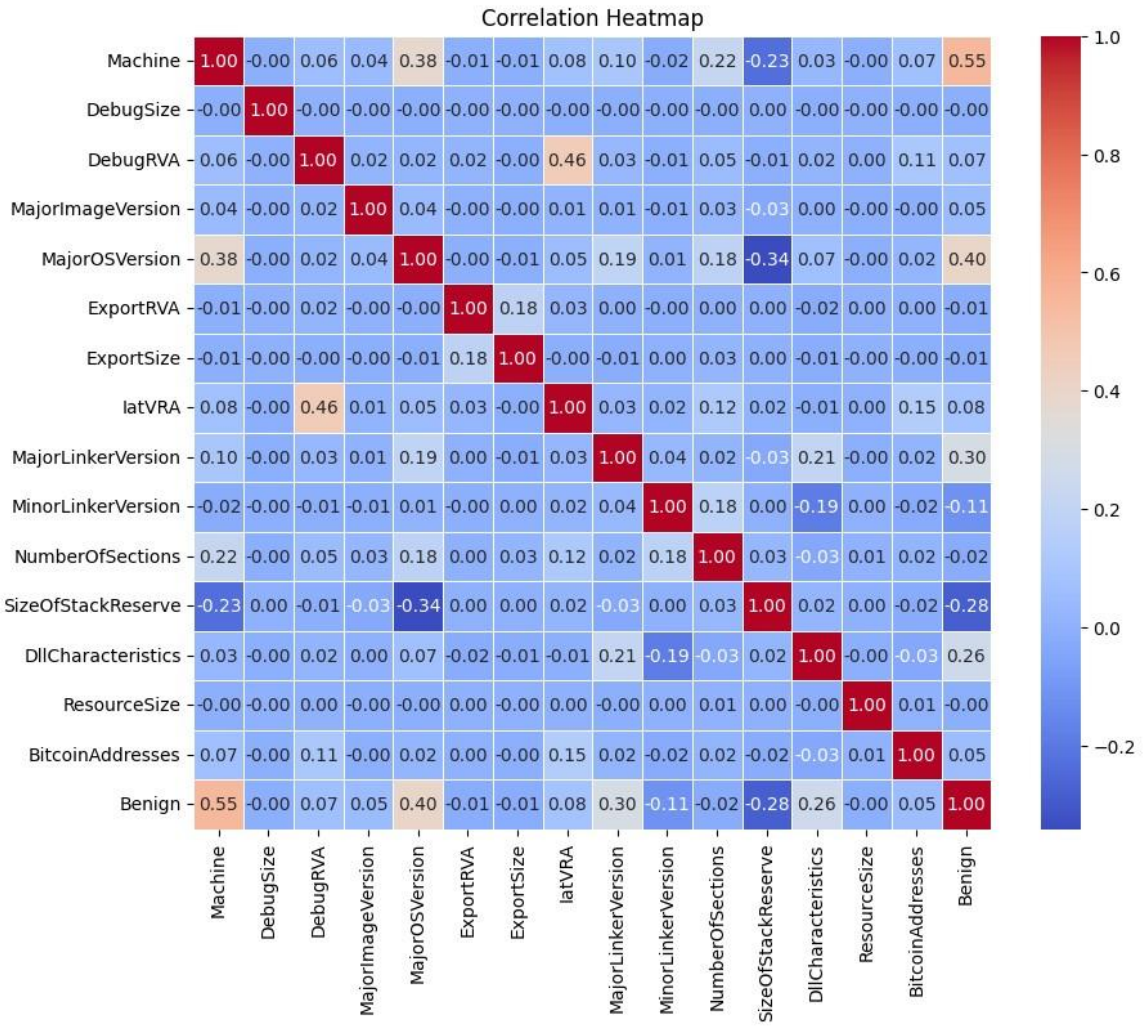


Figure 2. Data Visualization

Correlation Heatmap: A correlational heatmap is a graphical representation of the correlation matrix that displays the correlation coefficients for all pairs of variables in a dataset. It is a popular tool for data analysis and visualization that allows you to quickly identify patterns and relationships between variables.



4.3. Preprocessing

Data Cleaning: Data cleaning is an important step in the data preprocessing pipeline that focuses on identifying and addressing any inconsistencies, errors, or missing values in the dataset. The goal is to make sure the data is correct, reliable, and appropriate for analysis or modelling. The data cleaning steps are:

- Checking for Missing Values
- Dropping Rows with Missing Values
- Removing Duplicates

Dataset Splitting: Data splitting is a fundamental step in machine learning model development for several important reasons:

- Model Evaluation: We can train the model on a subset of the dataset and assess its performance on a different subset by dividing the dataset into distinct training and testing sets. This enables us to evaluate the model's generalization performance to new data. The model's performance could be overestimated if the entire dataset was used for training, as this

would essentially amount to memorization of the training data rather than the acquisition of well-generalized patterns.

- Preventing Overfitting: When a model learns to capture random fluctuations and noise in the training data instead of the underlying patterns, this is known as overfitting. Overfitting can be found by testing the model on a different testing set. Overfitting is suggested if the model's performance on the testing set is noticeably worse than on the training set.
- Hyperparameter Tuning: We can perform hyperparameter tuning without contaminating the testing set by splitting the data. Hyperparameters, such as the number of trees in a random forest, are settings that influence the model's learning process but are not learned by the model itself. To determine the ideal hyperparameter values without affecting the testing set, we can apply strategies such as cross-validation on the training set.
- Generalization Assessment: The primary goal of machine learning models is to generalize well to new, previously unseen data. By evaluating the model on a testing set that it did not see during training, we can get a better estimate of how well it will perform in the real world.

Normalization: A preprocessing method called normalization is used to scale numerical features inside a dataset to a uniform range. Normalization aims to achieve this by giving all features comparable scales, usually with a mean of 0 and a standard deviation of 1. This procedure is crucial because similar-scale features improve the performance of many machines learning algorithms. Normalization guarantees that every feature contributes equally to the performance of the model and helps prevent features with larger magnitudes from controlling the learning process. Here's how normalization works:

1. Mean Centering: The first step in normalization is to calculate the meaning of each feature across the dataset. For each feature, the mean value is subtracted from every data point. This centers the distribution of each feature around 0.
2. Scaling to Unit Variance: After mean centering, the next step is to scale each feature such that it has a standard deviation of 1. This is achieved by dividing each feature by its standard deviation. As a result, the variance of each feature becomes

Mathematically, the normalization formula for each feature X can be expressed as:

$$X_{\text{normalization}} = \frac{X - \mu}{\sigma} \tag{1}$$

Where:

- X is the original feature.
- X normalized is the normalized feature.
- μ is the mean of the feature.
- σ is the standard deviation of the feature.

By normalizing the features in this way, we ensure that each feature has a similar scale, making it easier for machine learning algorithms to learn patterns from the data.

4.4. Model Training and evaluation.

Training a classifier involves teaching it to recognize patterns in data so it can make accurate predictions on new, unseen data. In the code provided, three different classifiers are trained: Random Forest, Support Vector Machine (SVM), and XGBoost. Let's delve into each classifier's training process briefly:

Random Forest Classifier (rf_clf): An ensemble learning technique called Random Forest builds several decision trees during training and produces a prediction that is the class mode. The following are involved in the training process:

- Creating a RandomForestClassifier object with specified hyperparameters like the number of trees (n_estimators) and maximum depth of each tree (max_depth).
- Fitting the classifier to the training data. During this process, each tree in the forest is grown using a bootstrapped sample

of the training data, and at each node, the best split is chosen among a random subset of features. This randomness helps prevent overfitting.

- The random_state parameter ensures reproducibility by fixing the random seed.

Support Vector Machine Classifier (svm_clf): SVM is an effective supervised learning algorithm that can be applied to regression and classification problems. The way it operates is by identifying the hyperplane that divides data points into various classes the best. The following are involved in the training process:

- Creating an SVC object with specified parameters such as the kernel type (kernel) and gamma value (gamma), which controls the flexibility of the decision boundary.
- Fitting the classifier with the training data. SVM learns the optimal hyperplane that maximizes the margin between classes while minimizing classification errors. The kernel trick is used to map the input features into a higher-dimensional space where classes become separable.
- Setting probability=True enables the classifier to provide probability estimates, which is useful for certain evaluation metrics like ROC curves.

XGBoost Classifier (xgb_clf): Gradient boosting, a machine learning technique that builds an ensemble of weak learners (usually decision trees) sequentially, is implemented by XGBoost. The following are involved in the training process:

- Creating an XGBClassifier object.
- Fitting the classifier to the training data. XGBoost builds trees sequentially, with each tree learning from the errors of its predecessors. It uses gradient descent optimization to minimize a loss function, such as logistic loss for classification tasks.
- The random_state parameter ensures reproducibility by fixing the random seed.

The classifiers are prepared to predict new data after training. As demonstrated in the code, they can be assessed using a variety of performance metrics, including recall, accuracy, precision, F1-score, and ROC curves. This evaluation aids in determining the best model for the given task by evaluating the performance of the classifiers.

5. Results and Discussion

5.1. Performance Metrics evaluation

Metric	Random Forest	SVM	XGBoost
Accuracy	0.996639	0.964632	0.996639
Precision	0.997581	0.971238	0.997211
Recall	0.994620	0.946011	0.994991
F1-score	0.996098	0.958459	0.996100

Table 1. Model Performance Comparison

We examined the outcomes of the Random Forest Classifier, SVM, and XGBoost to find which model performs the best for malware detection. Their performance metrics are displayed in the following table 1.

When all these metrics are considered, Random Forest and XGBoost perform extremely similarly. Nonetheless, Random Forest outperforms XGBoost in terms of precision, recall, and F1-score, albeit very narrowly. Consequently, Random Forest might be regarded as the top-performing model for this dataset based on these evaluation metrics. Yet because of its scalability, efficiency, and performance, XGBoost frequently emerges as a formidable contender for big and complicated datasets. Ultimately, though, the decision between Random Forest, SVM, and XGBoost comes down to the particulars of the dataset, the available computational power, the need for interpretability, and the intended trade-off between computational complexity and accuracy.

5.2. Comparison of confusion matrix

Model	True Positives (TP)	False Negatives (FN)	False Positives (FP)	True Negative (TN)
Random Forest	7095	12	25	5365
SVM	6956	291	151	5099
XGBoost	7092	15	18	5363

Table 2. Comparison of confusion matrix

Among the three models, Random Forest stands out for its exceptional performance, especially when it comes to true positives and false negatives. Its robust capability to accurately identify positive instances while effectively minimizing misclassifications of negative instances is demonstrated by the fact that it consistently displays the highest number of true positives and the lowest number of false negatives. Furthermore, Random Forest continues to have a comparatively low number of false positives, which reinforces its ranking as the best performer in this comparison.

In contrast, Support Vector Machine (SVM) is inferior to Random Forest and XGBoost in a number of ways. It shows a higher number of false negatives and fewer true positives, indicating that it is not very good at correctly identifying positive cases. Furthermore, compared to Random Forest, SVM logs more false positives, which suggests that it is more likely to misclassify negative cases. These findings draw attention to the possible drawbacks of SVM and emphasize how poorly it performs in this situation when compared to Random Forest and XGBoost.

XGBoost performs admirably, but in terms of true positives and false negatives, it lags slightly behind Random Forest. Still, it performs about the same as Random Forest, with slightly fewer false positives. This implies that XGBoost can detect positive instances with a high degree of accuracy, similar to Random Forest, and a somewhat lower rate of misclassifying negative instances. Although Random Forest continues to be the best model, XGBoost's competitiveness highlights how effective it is as a substitute model for comparable classification tasks.

In conclusion, the thorough investigation shows that Random Forest outperforms XGBoost and SVM in precisely detecting positive instances while reducing the number of negative instances that are incorrectly classified. On the other hand, XGBoost performs competitively and can be used as a good substitute for Random Forest. In the end, performance metrics and other factors like interpretability, scalability, and computational efficiency will determine which model is best. When implementing classification models in practical applications, taking these aspects into account will help with well-informed decision-making. The performance of each model is examined in detail, which emphasizes how crucial it is to conduct a comprehensive analysis and comparison in order to determine which model will work best for a given classification task. Although Random Forest performs best in this scenario, the knowledge obtained from examining various models helps make better decisions and opens the door to increased dependability and performance in real-world applications.

5.3. Comparison of ROC Curves

Analysing the Receiver Operating Characteristic (ROC) curves for Random Forest, SVM, and XGBoost reveals that all three classifiers have curves located closer to the top-left corner of the plot. This positioning indicates their strong discriminative ability, implying that these models can effectively distinguish between ransomware and benign files over a wide range of threshold values. While all classifiers perform well overall, subtle differences in the shapes of the curves may provide information about their relative strengths in specific scenarios.

For example, even though all three classifiers show good discriminative ability, subtle information about each classifier's performance characteristics may be revealed by examining the ROC curve's shape. The performance of a Random Forest

classifier is generally consistent across threshold settings; it may display a smoother curve with fewer fluctuations. On the other hand, a steeper initial rise in the curve of an SVM classifier may indicate higher sensitivity at lower false positive rates. An XGBoost classifier, on the other hand, might exhibit a more gradual ascent, perhaps signifying a compromise between specificity and sensitivity across a range of threshold values.

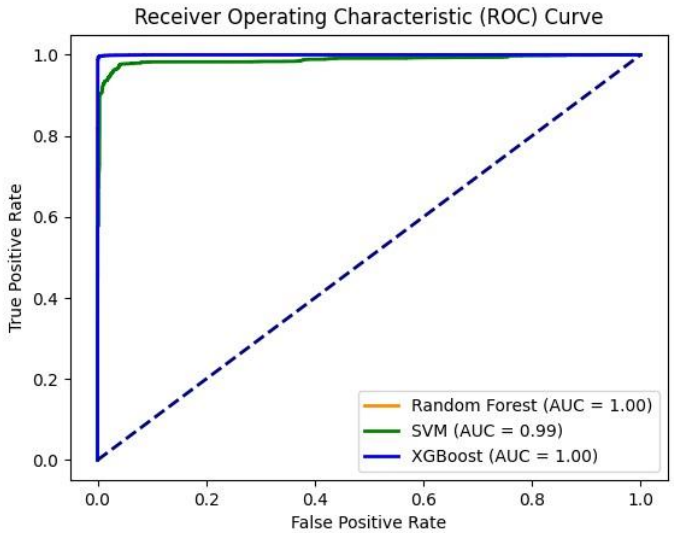


Figure 3. ROC Curve Comparison

Moreover, more information about the relative performance of each classifier can be obtained by utilizing the Area Under the Curve (AUC) metric, which measures each classifier's total discriminative ability. Although the curves for all three classifiers point favourably in the top-left corner, differences in the AUC values might indicate variations in how effective each classifier is. With a value of 0.5 denoting random chance and 1.0 representing perfect discrimination, a higher AUC value usually indicates better performance in differentiating between benign files and ransomware.

The AUC (Area Under the Curve) values provide insights into the performance of the models in terms of their ability to distinguish between positive and negative instances across various thresholds. Here's an explanation of the AUC values for Random Forest, SVM, and XGBoost:

Random Forest (AUC = 1.00):

- An AUC of 1.00 indicates that the Random Forest classifier achieved perfect separation between the positive and negative instances.
- This suggests that the Random Forest model has excellent discriminative ability and is capable of distinguishing between benign and non-benign instances with high confidence.
- A perfect AUC indicates that the model assigns higher probabilities to positive instances than negative instances across all threshold values.

SVM (AUC = 0.99):

- An AUC of 0.99 suggests that the SVM classifier performed exceptionally well in distinguishing between positive and negative instances, but it might have made a few misclassifications or mis ranked instances.
- While not perfect, an AUC of 0.99 indicates that the SVM model has high discriminative ability and is very effective in separating the classes.
- The slight deviation from a perfect AUC suggests that there might be a small overlap between the distributions of positive

and negative instances, leading to some misclassifications.

XGBoost (AUC = 1.00):

- Similar to Random Forest, an AUC of 1.00 for XGBoost indicates perfect separation between positive and negative instances.
- XGBoost achieved the highest possible AUC, suggesting that it performed exceptionally well in distinguishing between benign and non-benign instances.
- The model assigns higher probabilities to positive instances than negative instances across all thresholds, leading to perfect separation.

5. Conclusion

In conclusion, this project represents a significant step forward in the field of ransomware detection, providing valuable insights and methodologies for developing effective detection systems. Through rigorous evaluation of three prominent classifiers—Random Forest, SVM, and XGBoost—the project has shed light on their respective strengths and limitations, providing a comprehensive understanding of their performance in detecting ransomware.

XGBoost and Random Forest are the top performers; they have proven to be very effective in terms of accuracy, precision, recall, and F1-score, among other important evaluation metrics. These ensemble learning techniques have demonstrated robustness and reliability in differentiating between benign files and ransomware, with high accuracy levels exceeding 99% and precision, recall, and F1-score metrics all falling within the same impressive range. Together with their scalability and adaptability, their ability to leverage the power of ensemble learning to identify intricate patterns and relationships within the data makes them extremely effective tools in the fight against ransomware.

It is important to remember that SVM still performs admirably across all assessed metrics, despite lagging Random Forest and XGBoost in terms of overall performance. Precision, recall, and F1-score metrics all continue to perform well, and accuracy levels are above 96%. For this reason, SVM is still a good choice for ransomware detection, especially when interpretability and computational efficiency are the main concerns.

The final decision regarding which model is "best" is based on several variables, such as the demands of the application, the availability of computing power, the need for interpretability, and the relative weight of various evaluation metrics. To make an informed decision that is specific to their situation, stakeholders need to give careful thought to these factors. Organizations that prioritize robustness and high detection accuracy might choose Random Forest or XGBoost, whereas those that prioritize computational efficiency and interpretability might go toward SVM.

To improve ransomware detection systems, there are several opportunities for future study and development. Investigating cutting-edge methods like behavioral analysis, ensemble learning approaches, and deep learning architectures is one promising avenue. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two examples of deep learning models that have the capacity to automatically extract complex patterns and representations from unprocessed data, thereby doing away with the necessity for manual feature engineering. Researchers can create more advanced and adaptable detection systems that can precisely identify ransomware threats in real-time by utilizing the power of deep learning.[7]

Moreover, combining ensemble learning techniques like stacking and bagging can improve detection resilience and accuracy even more by combining predictions from several models trained on various data subsets or with various architectures. This strategy reduces the possibility of overfitting and increases the capacity for generalization of ransomware detection systems, which in turn promotes a more resilient cybersecurity environment.

References

[1]. A. Bensalah, “ransomware detection data set.” 31-Jul-2022.

[2]. F. Noorbehbahani, F. Rasouli, and M. Saberi, “Analysis of machine learning techniques for ransomware detection,” in 2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), 2019.

[3]. U. Zahoora, A. Khan, M. Rajarajan, S. H. Khan, M. Asam, and T. Jamal, “Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier,” Sci. Rep., vol. 12, no. 1, pp. 1–15, 2022.

[4]. Machinelearningmastery.com [Online]. Available: https://machinelearningmastery.com/save-gradient-boosting-models-xgboost-python/. [Accessed: 20-Apr-2024].

[5]. A. Bendovschi, “Cyber-Attacks - Trends, Patterns and Security Countermeasures,” Procedia Economics and Finance, pp. 24–31, 2015.

[6]. K. Cabaj, Z. Kotulski, B. Księżopolski, and W. Mazurczyk, “Cybersecurity: trends, issues, and challenges,” EURASIP Journal on Information Security, 2018.

[7]. S. Kumar and V. Somani, “Social Media Security Risks, Cyber Threats and Risks Prevention and Mitigation Tech-niques,” International Journal of Advance Research in Computer Science and Management, vol. 4, no. 4, pp. 125–129, 2018.