The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

Report
Machine Learning
final project

Team 12

Shir Berson 206550857

Michal Gogol 207932823

The objective of this project was to develop a machine learning model for classifying malware files as malicious or benign. As the threat of malware continues to increase, the need for robust cybersecurity measures becomes crucial.

PART 1-EXPLORATION: we conducted exploratory data analysis to gain insights into the dataset and better understand its characteristics. We began by loading the train dataset, splitting into train and test in order to check ourselves in the end, separating into numeric and categorial features and analyzing the data basic properties.

By getting the descriptive statistics we conclude that the features size, vsize, and printables exhibit large values and significant differences between their minimum and maximum values. These characteristics suggest the need for mathematical transformations to normalize the data and potentially improve the performance of machine learning algorithms later on.

By using histograms, we can visually assess if there are noticeable differences in the distribution of each feature between malicious and benign files. Our conclusions were:

* Feature A has a bell-shaped or symmetric distribution, which suggests that the data is evenly distributed around the mean, which indicates a normal distribution of values.

* Feature B looks like Negatively Skewed (Left-skewed). The tail of the histogram extends towards the left, indicating a concentration of larger values. It suggests that the majority of data points are larger, and there may be some outliers with smaller values.

* Feature file_type_prob_trid is suspected as Multimodal Distribution.

* We can also suspect that the features: size, vsize, imports, exports, paths, numstrings, symbols, urls, MZ, registry and printables, has a similar distribution though in a different scale.

* We can see from features size, numstrings, imports, exports, MZ, urls and printables histograms that there are samples that are significantly different from the majority of the data points. Therefore they are maybe Outliers.

* It looks like in feature avlength all the samples are labeled as malicious. The fact that the feature's histogram exclusively contains values associated with malicious files suggests that there may be a strong association between that feature and the label. This indicates that the feature has discriminatory power and can potentially be informative in distinguishing between malicious and benign files.

* It looks like in features exports and registry most of the samples which their value is higher than 0, are labeled as benign.

We created a correlation map and recognized a very high correlation of 0.89 between features size and numstrings. In addition, we find high correlation of 0.79 between features size and MZ. We noted to ourselves to maybe remove one of them later on.

We also checked how many unique values were in each categorial features, plotted it and revealed the presence of rare unique values with low frequency in the dataset. These rare categories need to be carefully considered during analysis and modeling to ensure accurate predictions. In addition, we used box plot for feature A due to its normal distribution observed in the histogram. Box plots are effective in detecting outliers and that allowed us identified potential outliers in feature A.

PART 2-PREPROCESSING: We started in finding missing values in both train and test datasets, searching for features or samples with a large number of missing values that can potentially be removed from the datasets. Unfortunately, samples and features like this weren't found.

The next stage was detecting and removing outliers which was done using techniques that weren't taught in class: Z-Score and LOF. From the visual inspection that we did, in some of the features we found individual data points that are far away from the bulk of the data, which we suspect them as outliers.

Based on the conclusion from the exploration part, we chose to remove outliers from feature A using z-score method, which relies on the assumption of normal distribution. In addition, it is widely used, simple, and easy to interpret. Z-Score is a statistical measure that quantifies how far a data point deviates from the mean of a distribution. It indicates the number of standard deviations a data point is away from the mean. The formula of z-score is: $z = (x - \mu) / \sigma$. In outlier detection, if a data point has a z-score that is significantly different from zero (typically greater than a certain threshold), it suggests that the data point is an outlier. By using z-scores, we can identify data points that are unusually far from the mean and are likely to be outliers. We chose a threshold of 3 because it is a commonly used value in outlier detection, it corresponds to approximately 3 standard deviations away from the mean. This threshold is often considered a conservative choice, capturing data points that deviate significantly from the average and are considered extreme outliers. We found and removed 310 outliers in feature A.

In order to prepare the data for LOF algorithm, we needed to fill missing values and perform mathematical transformations. As for handling missing values, we utilized the SimpleImputer which based on a specified strategy. For the numeric features, we opted for the "mean" strategy, which replaces the missing values with the average of the available values. By using the mean value, we can preserve the overall distribution and central tendency of the data. For the categorical features, we chose the "most frequent" strategy, which fills in the missing values with the most commonly occurring category. By using the most frequent category, we can maintain the majority class distribution and preserve the existing patterns within the categorical data. As for mathematical transformations, from the exploration part, we observed that certain features have very large values and substantial differences between the minimum and maximum values. We applied the logarithm function on these identified features.

The Local Outlier Factor Algorithm (LOF) is an unsupervised anomaly detection technique that measures the local density deviation of a data point in relation to its neighboring points. It can effectively identify outliers that exhibit different characteristics compared to their surrounding data points. One advantage of using LOF is that it does not assume any specific distribution of the data, making it applicable even when the feature's distribution is unknown like in our case. A detailed explanation of the parameters we chose and the features on which we performed the algorithm, can be found in the appendices.

By removing outliers, we improve the model's ability to learn patterns and make accurate predictions, as it can now focus on the majority of the data that represents the underlying patterns and relationships.

The next step, as part of preparing the data for subsequent analysis and modeling steps, we conducted feature exclusions. We Removed the features sha256 and numstrings. sha256 feature serves as a unique identifier for each sample and doesn't provide meaningful information about the samples themselves. numstrings feature exhibits a high correlation with size feature in the dataset. Highly correlated features can introduce redundancy and increase the dimensionality of the data.

Going forward, we created 5 new features but in the final work flow we used only 4 of them. Through feature engineering techniques, we aim to extract valuable information from the existing variables, enabling us to better understand and identify malicious files. The new features names and the rationale behind them are detailed in the appendices.

We continued to normalization. In our project, the data is not normalized, therefore we need to normalize it because it can be important in many machine learning problems for the following reasons: eliminating scale differences, improving convergence of optimization algorithms, facilitating comparison of features, handling distance-based algorithms and ensuring robustness to outliers. A broader explanation of each of the reasons can be found in the appendices. We wanted to normalize each feature in a way that best suits it. We started with the numeric features for which we haven't identified a particular distribution, using Min-Max Scaling. Then, we normalized feature 'B' using the Box-Cox transformation. This method is effective for normalizing skewed data and can handle both positive and negative skewness (a distribution that fits what we identified in the exploration).

Next, we proceeded with the normalization of the categorical features: file_type_trid and C. It's important to note that the remaining categorical features are binary, so there is no need for further scaling it. We have chosen to use target encoding which is a technique that wasn't covered in our course but we have decided to utilize it based on its potential benefits in capturing valuable information from categorical features. Target encoding is a technique that involves replacing categorical values with the mean of the target variable for each category. It provides a way to represent categorical information in a numerical format that can be more easily processed by machine learning algorithms. We ensured that our code can handle unseen data and avoid any leakage of information from the test data into the training process.

In the final part of the preprocessing phase, we conducted dimensionality reduction. High dimensionality refers to a large number of features in a dataset. In our opinion, the dimensionality of the problem is too great because it leads to the curse of dimensionality, which can cause several issues. With a large number of features, the available data becomes sparse, making it difficult to find meaningful patterns and relationships. Additionally, the computational complexity increases exponentially, making analysis and modeling more time-consuming and resource-intensive. High dimensionality also raises the risk of overfitting. Interpretation and visualization become challenging as well, as it becomes harder to comprehend and visualize data in high-dimensional spaces. To recognize that the dimensions of the problem are too large, we can consider the following factors: The ratio of the number of samples to the number of features, Computational limitations and Lack of meaningful patterns or predictive power. We used Backward selection technique which led to a final set of 24 features that achieved the lowest Mallows Cp score. The reduction in dimensionality can bring several benefits to the model, such as: improved generalization, increased interpretability and enhanced computational efficiency. A broader explanation of each of the benefits can be found in the appendices.

PART 3-MODELING: We applied KNN and Logistic Regression as part of the initial models, and Ada boosting and SVM as part as the advanced models. We performed hyperparameter tuning for each of the models above using grid search. The meaning of the hyperparameters in each of the models and how they affect the model in terms of variance and bias are found in the appendices.

As for the importance of the features, by analyzing the feature importance, we can uncover the key drivers behind the model's predictions and gain insights into the underlying relationships between the features and the target variable. In our analysis, the feature importance scores were obtained using Ada Boost algorithm. Based on the feature importance scores, the 7 top features are detail in the appendices. Significance and Interpretation: The top features identified provide valuable insights into the model's decision-making process. Their high importance scores indicate their strong influence on the classification of malicious and benign files. Furthermore, the inclusion of engineered features highlights the significance of feature engineering in improving the model's performance. These engineered features capture specific aspects of the file structure and relationships between file attributes, contributing to the model's ability to differentiate between malicious and benign files.

PART 4 - MODELS EVALUATION: The chosen Model is ADA BOOST. among the models evaluated through k-fold cross-validation, AdaBoost achieved the highest AUC score of 0.94. This indicates that AdaBoost outperformed the other models and exhibited strong predictive capabilities for the given dataset. At first we used Random Forest and then, after obtained better results on the test dataset using AdaBoost, we replace it because it suggests that it was able to generalize well and make accurate predictions on unseen data. Even though in the AUC validation for the random forest score we got 0.98, in the test AUC we got much better result with AdaBoost. We created a Validation Confusion Matrix for AdaBoost. Each cell of the matrix represents:

* True Positives (TP): This cell indicates the number of files (3754) that are truly malicious (class 1) and were correctly predicted as positive by AdaBoost model. These are the instances where the model made correct predictions of positive instances, correctly identifying them as malicious.

* False Positives (FP): This cell represents the number of files (771) that are actually benign (class 0) but were incorrectly predicted as positive by AdaBoost model. These are the instances where the model made an incorrect prediction by classifying benign files as malicious.

* False Negatives (FN): This cell indicates the number of files (663) that are truly positive (class 1) but were incorrectly predicted as negative by AdaBoost model. These are the instances where the model made an incorrect prediction by failing to classify malicious files as positive.

* True Negatives (TN): This cell represents the number of files (3587) that are actually negative (class 0) and were correctly predicted as negative by AdaBoost model. These are the instances where the model made correct predictions of negative instances, correctly identifying them as non-malicious.

Based on these observations, the following conclusions can be made about the model's performance in this context:

1. Accuracy: The accuracy of the model can be calculated by summing the number of true positives and true negatives and dividing it by the total number of instances. In this case, the accuracy would be 0.8366. It indicates the overall correctness of the model's predictions.

2. Precision: Precision is the proportion of true positive predictions out of all positive predictions (both true positives and false positives). It can be calculated as TP / (TP + FP). A higher precision value (0.8296) indicates a lower rate of falsely labeling benign files as malicious. It shows how well the model identifies truly malicious files among the predicted positive instances.

3. Recall (Sensitivity or True Positive Rate): Recall is the proportion of true positive predictions out of all actual positive instances (both true positives and false negatives). It can be calculated as TP / (TP + FN). A higher recall value (0.8499) indicates a lower rate of falsely labeling malicious files as benign. It demonstrates the model's ability to capture the majority of truly malicious files.

4. F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of both precision and recall. It can be calculated as 2 * (Precision * Recall) / (Precision + Recall). A higher F1 score (0.8396) indicates a better balance between precision and recall, implying a better overall performance of the model in identifying both malicious and benign files accurately.

Comparing the confusion matrices of the validation and train sets can provide insights into the performance gaps between running the AdaBoost model on these two datasets. It can also help determine if the model is overfitting and suggest potential steps to improve its generalization ability. **Performance Gaps:**

* Accuracy Gap: The accuracy on the train set (0.8397) is slightly higher than the validation set (0.8366), indicating a small performance gap between the two datasets. This suggests that the model may be slightly overfitting, but the difference is not significant.

* Precision Gap: The precision on the train set (0.8293) is slightly lower than the validation set (0.8296). The precision remains relatively consistent between the two sets.

* Recall Gap: The recall on the train set (0.8517) is slightly higher than the validation set (0.8499). The recall values also show a small difference, but it is not substantial.

* F1 Score Gap: The F1 score on the train set (0.8404) is slightly higher than the validation set (0.8396). This suggests that the model's ability to balance precision and recall is slightly better on the train set.

Overall, the performance gaps between the train and validation sets are relatively small. This indicates that the model is not significantly overfitting. However, there is still room for improvement to enhance its generalization ability. Initially, the model exhibited overfitting with a high validation accuracy of 0.84 and an exceptionally high training accuracy of 0.999. **To reduce the gap between training and validation accuracies and mitigating overfitting and enhance the model's ability to generalize, several steps were taken:**

1. Feature Selection: instead of using PCA, which may not have been capturing the most informative features, backward selection was employed. Backward selection helps identify the most relevant features by iteratively removing the least important ones. By focusing on the most influential features, the model can potentially make better predictions and generalize more effectively.

2. Normalizing categorial features: in the previous approach, one-hot encoding was used for the categorical features, which resulted in a significant increase in the number of columns, leading to a sparse dataset. To overcome this, a threshold was implemented to select only the most frequent unique values in the feature. Later on, we decided to switch from one-hot encoding with a threshold to target encoding. Our decision was driven by the aim to reduce dimensionality, mitigate the curse of dimensionality, and improve the model's generalization capability. Before changing to target encoding, the backward selection chose 29 features, and after it chose 24 features.

3. Feature Engineering: New features were created as detailed in the preprocessing part. The aim was to provide the model with more relevant and discriminative features.

4. Model Selection: The model was changed from Random Forest to AdaBoost. AdaBoost is an ensemble learning algorithm that combines multiple weak learners to create a strong classifier. It is known for its ability to handle complex relationships between features and the target variable. This change in model algorithm can potentially improve the model's generalization capabilities.

5. Regularization: Experiment with different regularization techniques, such as adjusting the learning rate or incorporating regularization parameters, to prevent overfitting and improve generalization.

6. Cross-Validation: by utilizing cross-validation techniques like k-fold cross-validation, we obtain a more comprehensive assessment of the model's generalization ability. It helps us gauge the model's performance across various subsets of the data, detect overfitting, and optimize hyperparameters.

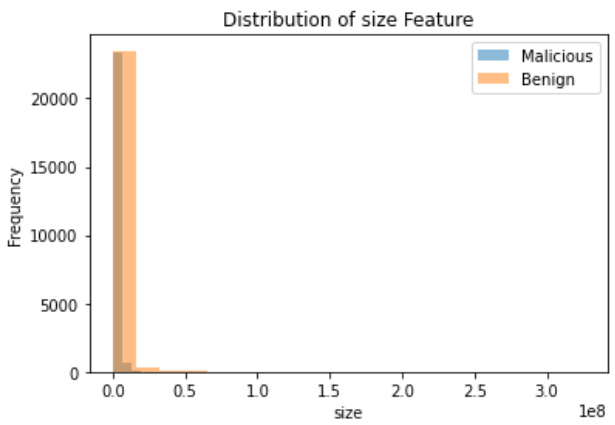
In order to evaluate the generalization performance of our model, we applied our trained model to our split test dataset and calculated the AUC score: 0.809.

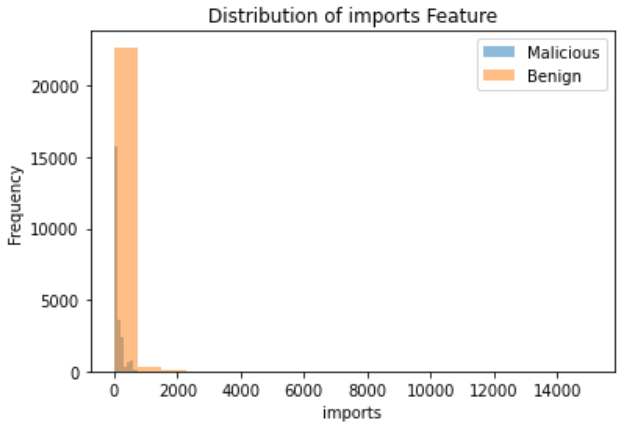
PART 5: PREDICTION: We developed a preprocessing function that encapsulates all the actions we performed in the preprocessing part. Later, we used it as part of the pipline function we created as well. The pipline function runs the selected and final model on the real test dataset and performing predication.

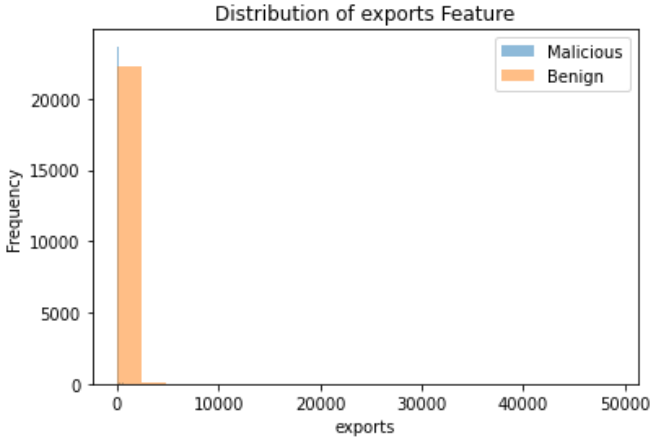
Appendices

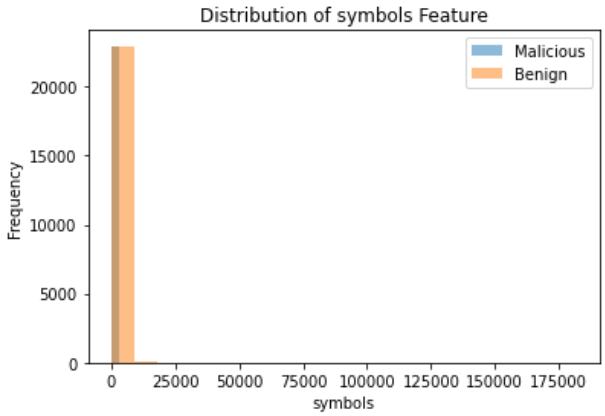
PART 1-EXPLORATION:

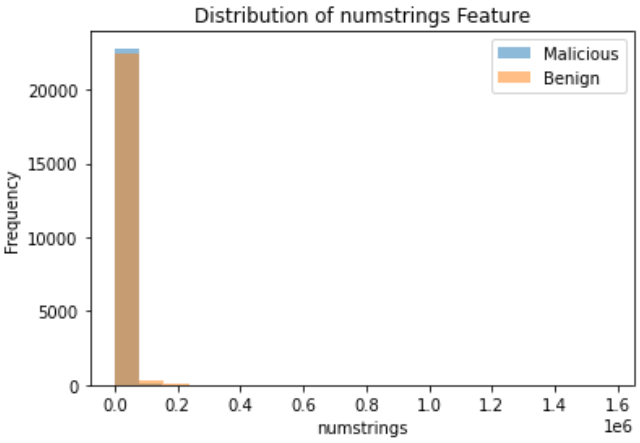
1. Histograms of Each numeric feature which separates between the target variable (malicious and benign files)

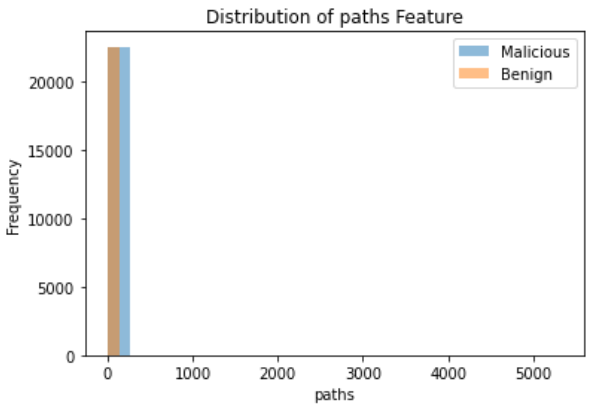


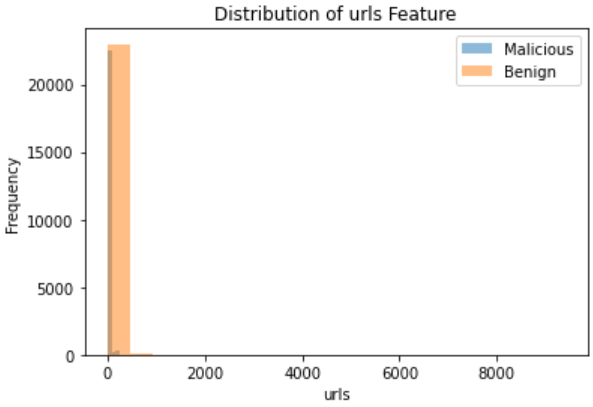


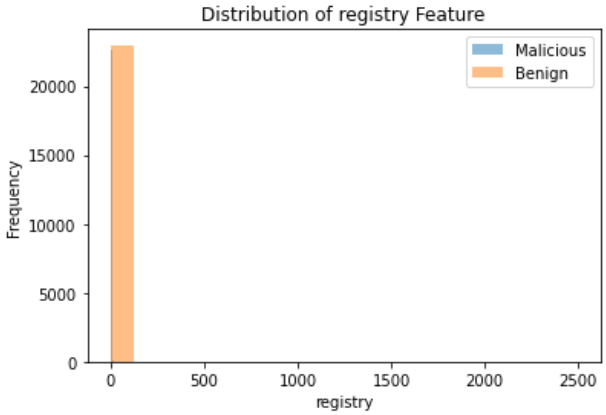


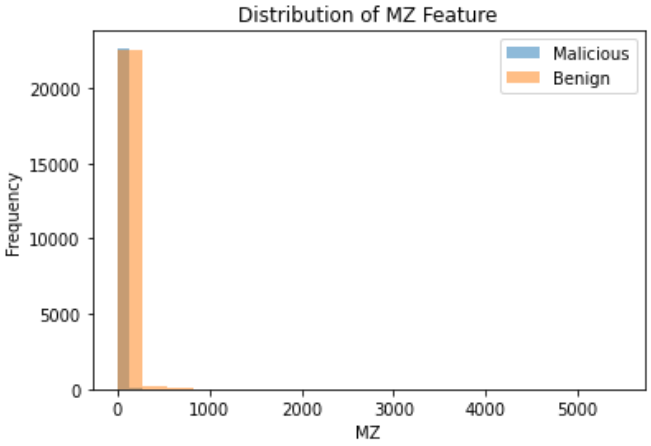


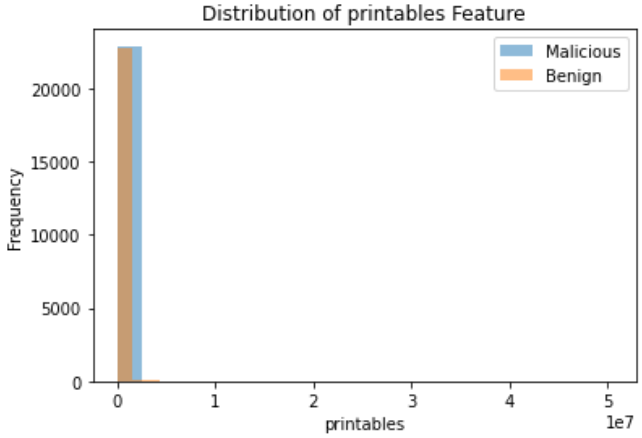


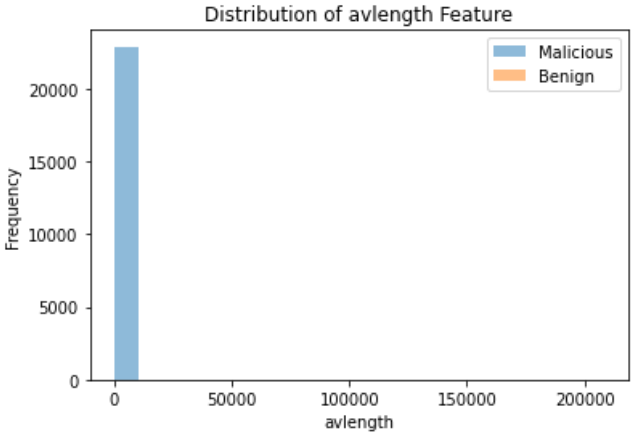

Distribution of paths Feature
Malicious
Benign
Frequency
paths
0
5000
10000
15000
20000
0
1000
2000
3000
4000
5000

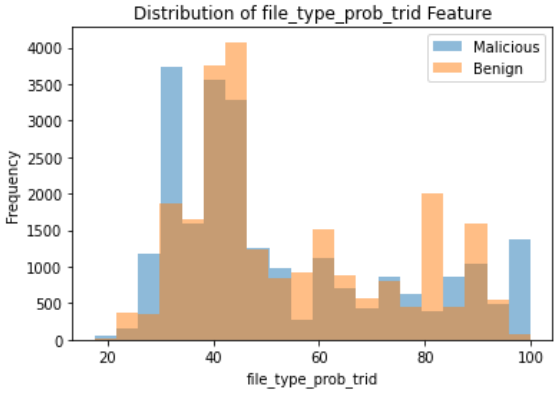

Distribution of urls Feature
Malicious
Benign
Frequency
urls
0
5000
10000
15000
20000
0
2000
4000
6000
8000

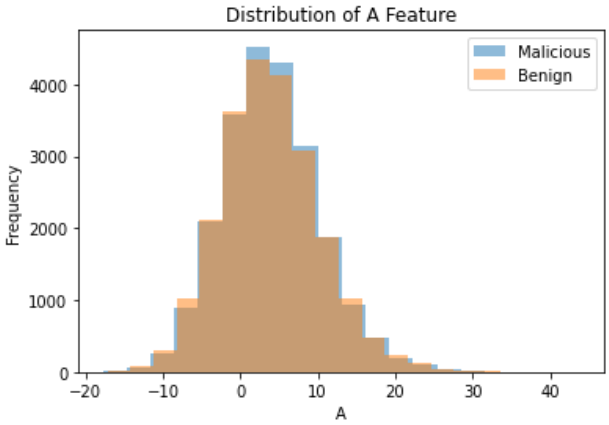

Distribution of registry Feature
Malicious
Benign
Frequency
registry
0
5000
10000
15000
20000
0
500
1000
1500
2000
2500

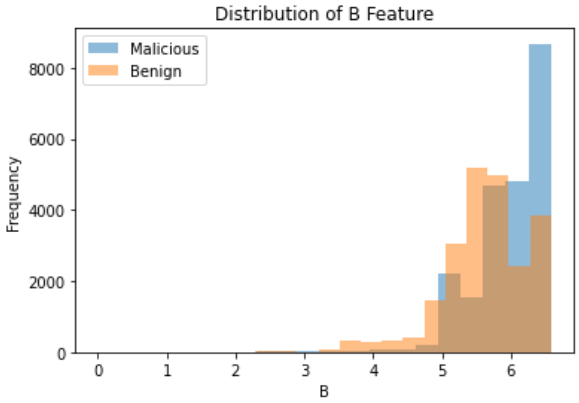

Distribution of MZ Feature
Malicious
Benign
Frequency
MZ
0
5000
10000
15000
20000
0
1000
2000
3000
4000
5000


Distribution of printables Feature
Malicious
Benign
Frequency
printables
1e7
0
5000
10000
15000
20000
0
1
2
3
4
5

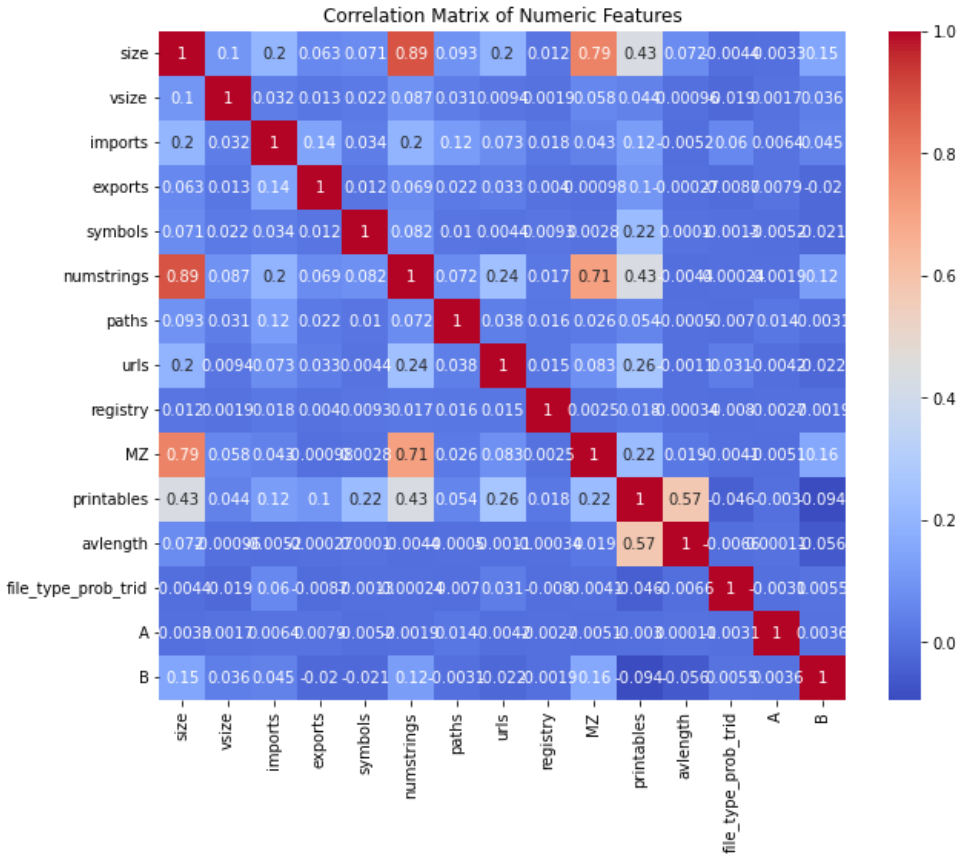

Distribution of avlength Feature
Malicious
Benign
Frequency
avlength
0
5000
10000
15000
20000
0
50000
100000
150000
200000



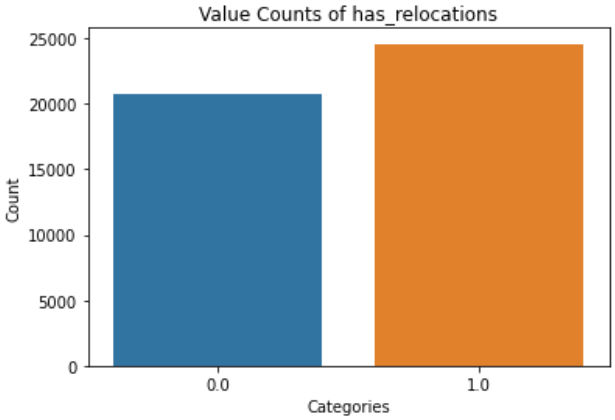


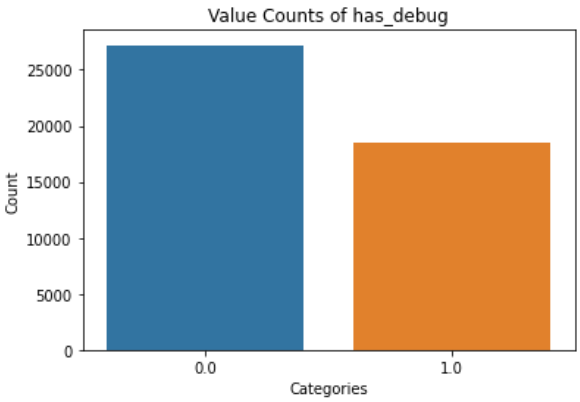


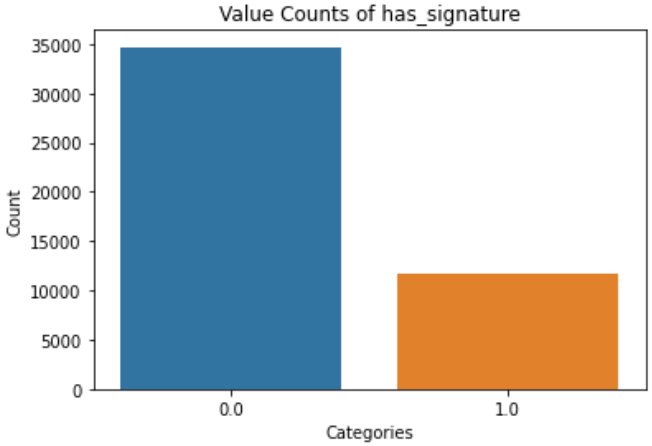
2.Correlation Map

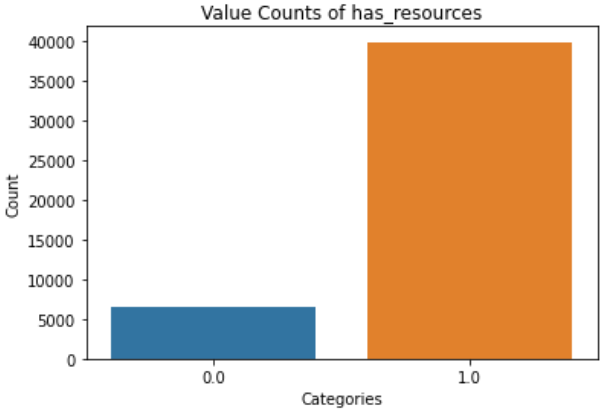


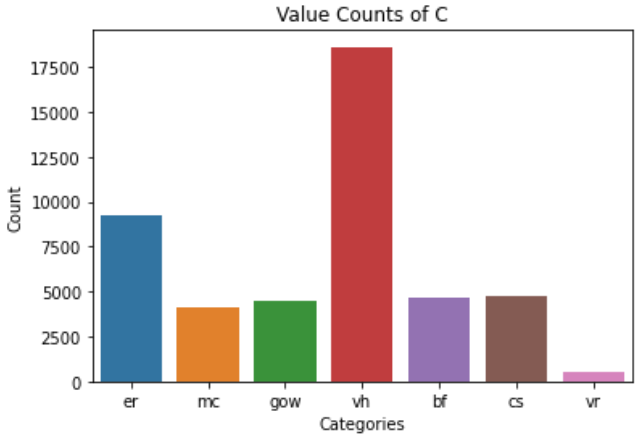
3.Bar plots for the categorial features:

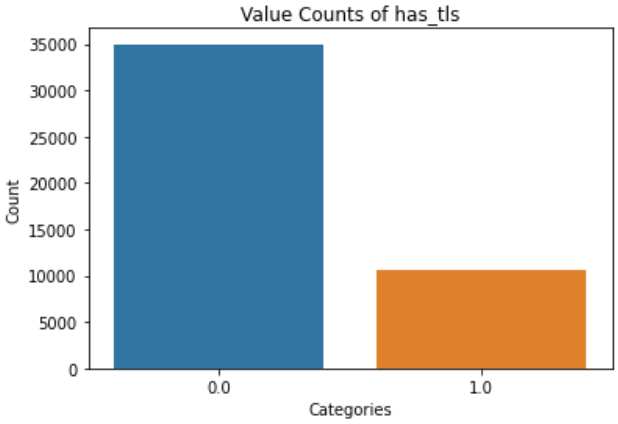


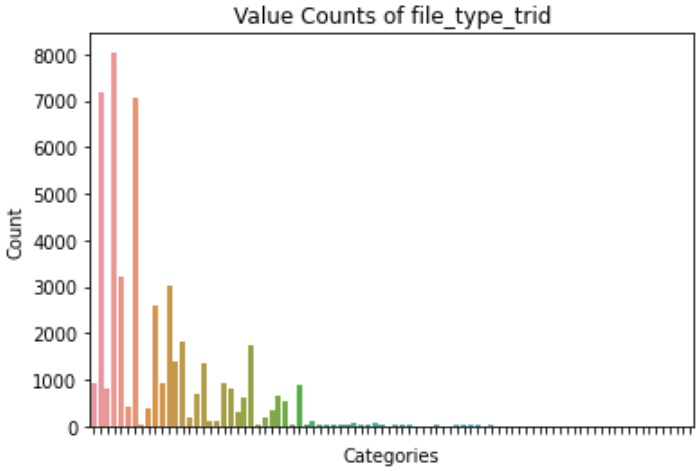




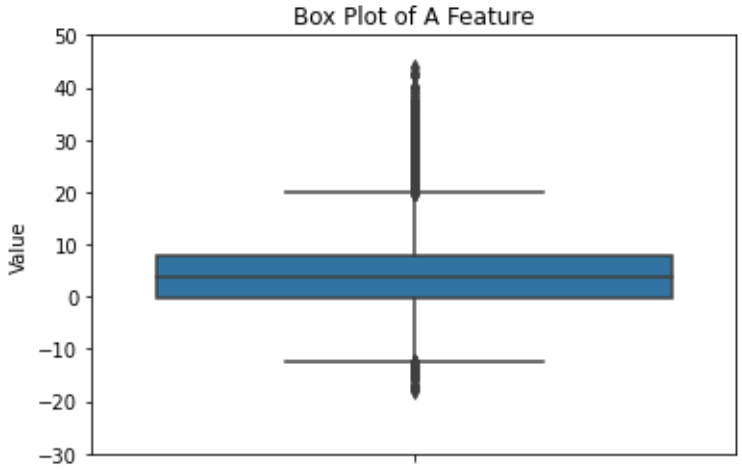








4.Box Plot for feature A:



PART 2-PREPROCESSING:

1. A detailed explanation of the parameters we chose and the features on which we performed the LOF algorithm:

We used the Local Outlier Factor (LOF) algorithm to identify outliers in specific features: size, imports, exports, symbols, urls, MZ, and printables.

These features were found to have outliers based on the visualization analysis. However, we decided not to remove outliers in the 'numstrings' feature due to its high correlation (0.89) with the 'size' feature, as indicated by the correlation map.

For the LOF algorithm, we set the n_neighbors parameter to 30 and the contamination parameter to 0.08. The n_neighbors parameter determines the number of neighboring points used to estimate local density, while the contamination parameter represents the expected proportion of outliers in the data. By choosing these values, we aimed to strike a balance between accurately estimating density and specifying the threshold for outlier classification.

The selection of these parameter values was based on experimentation to achieve a suitable balance between outlier detection accuracy and model performance.

After applying the LOF algorithm, we obtained outlier labels and scores for each data point. To ensure that the model is not influenced by potentially anomalous data points, we removed the corresponding rows from the training set (X_train and y_train). By removing these outliers, we improve the model's ability to learn patterns and make accurate predictions, as it can now focus on the majority of the data that represents the underlying patterns and relationships.

2. The new features names and the rationale behind them (Features Engineering):

* Difference between Size and VSize:

Rationale: This new feature aims to capture the difference between the "Size" and "VSize" variables. By examining this difference, we can potentially identify the presence of compression or padding techniques employed in the file. These techniques are often indicative of malicious behavior and can provide valuable insights for our analysis.

* Ratio between Size and VSize:

Rationale: The ratio between the "Size" and "VSize" variables offers information about the packing or compression status of the file. By calculating this ratio, we can determine if the file has been packed or compressed, which is another relevant characteristic to consider when identifying malicious files.

*Imports minus Exports:

Rationale: This newly created feature represents the difference between the number of imports and exports in the file. By evaluating the balance between incoming and outgoing function calls, we can potentially uncover patterns associated with malicious files. This feature provides insights into the behavior and interactions of the file with other components, contributing to our understanding of its malicious nature.

* Ratio of avlength divided by size:

Rationale: The ratio of avlength (average string length) divided by size" introduces a measure of the average string length relative to the file size. This feature provides insights into the compactness or density of malicious files. We specifically chose to create this feature using "avlength" because its histogram predominantly contains values associated with malicious files. By incorporating this new feature, we aim to enhance the discriminatory power of our model and improve its performance.

* We attempted to create the one more feature called- Ratio of Imports to Exports.

Rational: capture the dependency or external interaction of the file, which could indicate its maliciousness.

However, after calculating the ratio, we discovered that a large number of values resulted in 'inf' (infinity) due to division by zero. With 31,776 'inf' values and a high prevalence of zeros, we decided to drop this feature from our workflow. Despite its failure, we kept this attempt as part of our analysis and experimentation process.

3. A broader explanation of each of the reasons it is important to normalize the data:

* Eliminating scale differences: Different features in the dataset may have different scales. Normalizing the data ensures that all features are on a similar scale.

* Improving convergence of optimization algorithms: Many machine learning algorithms rely on optimization techniques to find the optimal model parameters. Normalizing the data can help improve the convergence of these optimization algorithms by providing a well-conditioned problem.

* Facilitating comparison of features: Normalization enables fair comparison and evaluation of different features. It ensures that each feature contributes proportionally to the learning process and avoids bias towards features with larger magnitudes.

* Handling distance-based algorithms: Distance-based algorithms, such as KNN, are sensitive to the scale of the features. Normalizing the data allows these algorithms to correctly measure the distances and similarities between samples.

* Ensuring robustness to outliers: Normalization can help reduce the impact of outliers by bringing the data within a smaller range. Outliers can have a significant effect on some algorithms, and normalization helps to mitigate this effect.

In conclusion, normalizing the data is important in our problem as it helps address the issues mentioned above and can improve the performance and reliability of our machine learning models.

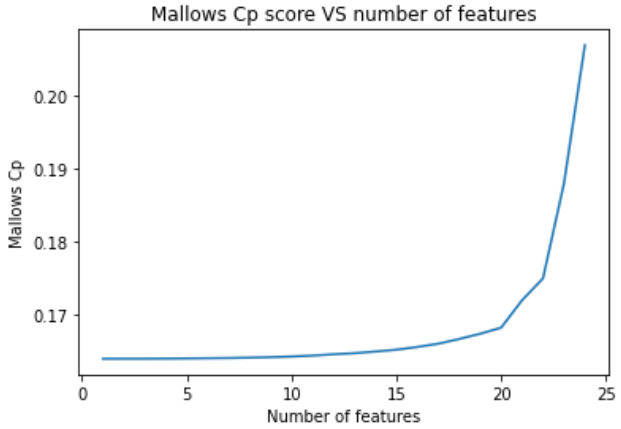
4. A broader explanation of each of the benefits the reduction in dimensionality can bring to the model:

- Improved generalization: By removing irrelevant or redundant features, the model becomes more focused on the most informative attributes, reducing overfitting and improving its ability to generalize well to unseen data.

- Increased interpretability: With fewer features, it becomes easier to interpret the model and understand the relationships between the selected features and the target variable.

- Enhanced computational efficiency: Having fewer features reduces the computational burden during training and prediction, allowing for faster model development and inference.

In summary, the backward selection process effectively reduced the dimensionality from 25 to 24 features, resulting in improved model performance and potential benefits in terms of generalization, interpretability, and computational efficiency.



PART 3-MODELING:

1. The meaning of the hyperparameters in each of the models and how they affect the model in terms of variance and bias:

The meaning of the hyper parameters in the KNN model and how they affect the model in terms of variance and bias:

The hyperparameter we chose to change for the KNN (K-Nearest Neighbors) model is n_neighbors. This parameter determines the number of neighbors considered for classification. By varying this parameter, we can control the complexity of the model and its bias-variance trade-off.

When n_neighbors is set to a smaller value, such as 5, the model becomes more flexible and can capture intricate patterns in the data. This reduces bias as the decision boundaries are more closely fitted to the training data. However, a lower value also increases the variance because the model becomes more sensitive to noise and outliers in the data. This can lead to overfitting and decreased generalization on unseen data.

On the other hand, when n_neighbors is set to a larger value, such as 13, the model becomes less flexible and the decision boundaries become smoother. This increases bias as the model assumes a more generalized representation of the data. By considering more neighbors, the model can make more robust predictions and reduce the influence of individual data points. However, setting a higher value also reduces variance as the model becomes less sensitive to noise and outliers.

During the grid search cross-validation, we tested different values of n_neighbors (5, 8, 9, 11, and 13) to find the best-performing hyperparameter value. The grid search evaluated the model's performance on different

folds of the training data, allowing us to identify the optimal value of n_neighbors that balances bias and variance.

The best hyperparameter value (n_neighbors) that selected through the grid search is 5. It represents the configuration that achieved the highest performance based on the chosen evaluation metric. By choosing the best hyperparameter value, we aimed to strike a balance between bias and variance, leading to a well-performing KNN model that can generalize well to unseen data.

We used the default value for the 'weights' parameter which is 'uniform'.

When 'uniform' is set as the value for weights, all points in each neighborhood are weighted equally, and the majority class of the nearest neighbors is used for classification. This means that each neighbor contributes equally to the decision-making process.

The meaning of the hyper parameters in the Logistic Regression model and how they affect the model in terms of variance and bias:

In logistic regression, we focused on changing the hyperparameter 'C', which represents the inverse of the regularization strength. 'C' controls the amount of regularization applied to the model. A smaller 'C' value increases regularization, reducing overfitting and decreasing variance but potentially increasing bias. Conversely, a larger 'C' value decreases regularization, allowing the model to fit the training data more closely, potentially increasing variance.

Through grid search cross-validation, we tested different 'C' values ([0.01, 0.1, 1]) to find the optimal balance between bias and variance. The best hyperparameters, determined by grid search, strike a balance between model complexity and generalization performance. Additionally, logistic regression employs a penalty term to the loss function, with the 'liblinear' solver being used in this case.

The best hyperparameter value (C) that selected through the grid search is 1.

The meaning of the hyper parameters in the AdaBoost model and how they affect the model in terms of variance and bias:

The hyperparameter grid specified for AdaBoost consists of two parameters: n_estimators and learning_rate.

n_estimators refer to the number of weak classifiers used in the boosting process. In the grid search, the values tested were 120, 180, 200, 250 and 300. Increasing the number of estimators can capture more complex patterns in the data, but it also increases the risk of overfitting.

learning_rate represents the contribution of each weak classifier to the final ensemble. The values tested in the grid search were 0.5, 1.0,1.5 and 1.8. A smaller learning rate emphasizes the correction of misclassifications, leading to improved robustness, but it requires more estimators to achieve comparable performance.

These hyperparameters play a role in the bias-variance trade-off. A higher n_estimators value can reduce bias but may increase variance and overfitting. On the other hand, a smaller learning_rate improves generalization but necessitates more estimators for convergence.

To summarize, the values specified for n_estimators and learning_rate in the grid search allow for exploration of different levels of model complexity and regularization. The optimal combination of these hyperparameters depends on the specific dataset and the desired trade-off between bias and variance. The best hyperparameters we got: n_estimator=250 and learning_rate=1.5.

in addition to the n_estimators and learning_rate, the code does not specify any other hyperparameters explicitly, which means the default values for those hyperparameters will be used. The default hyperparameter values for AdaBoostClassifier in scikit-learn are as follows:

- base_estimator: The default base estimator for classification tasks is a decision tree with a maximum depth of 1 (DecisionTreeClassifier(max_depth=1)).

- algorithm: The default algorithm for boosting is the SAMME.R algorithm, which is an improved variant of SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function).

- random_state: The default value is None, which means the random number generator used by the algorithm is the RandomState instance used by np.random.

- loss: The default loss function for classification is the exponential loss ('exponential'), which is commonly used in AdaBoost.

The meaning of the hyper parameters in the SVM model and how they affect the model in terms of variance and bias:

In the SVM (Support Vector Machine) model, we focused on changing the hyperparameter 'C'.

'C' is the regularization parameter in SVM, also known as the penalty parameter. It controls the trade-off between achieving a low training error (low bias) and allowing more margin violations (high variance).

A smaller value of 'C' leads to a larger margin and a simpler decision boundary, which may result in higher bias but lower variance. Conversely, a larger value of 'C' allows for a smaller margin and a more complex decision boundary, potentially reducing bias but increasing variance.

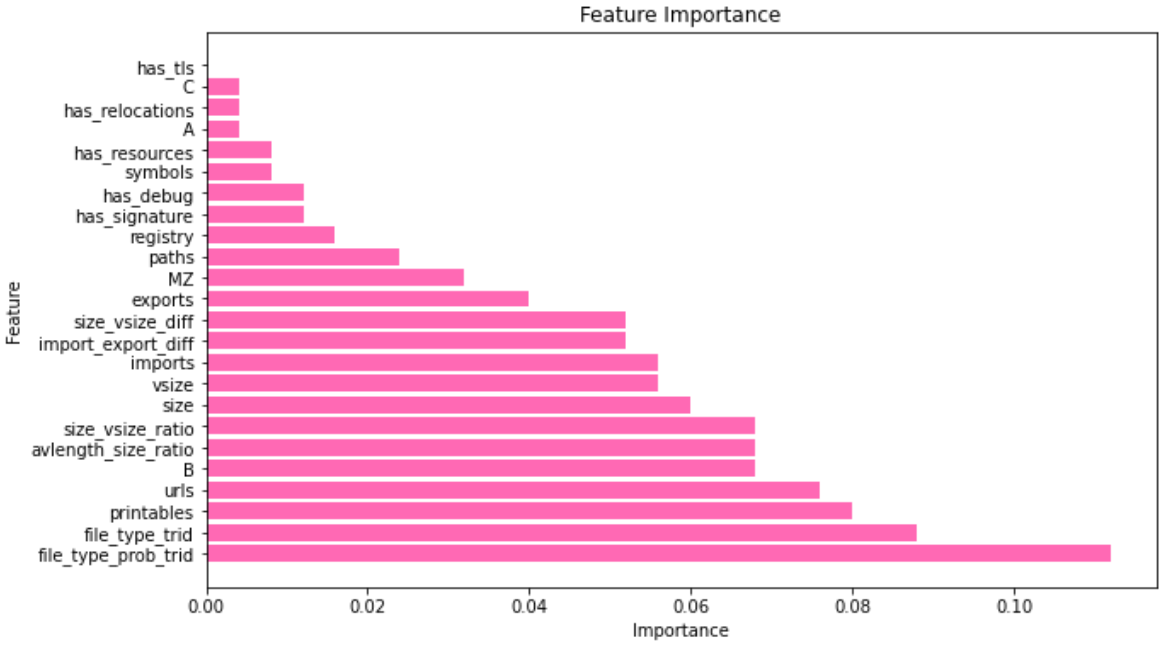
By specifying different values for 'C' ([1, 10, 15]), we performed grid search cross-validation to find the optimal value of 'C' that minimizes the error and achieves a good balance between bias and variance.

The best hyperparameter value (C) that selected through the grid search is 15.

We chose the default value for the 'kernel' hyperparameter in the SVM's which is 'rbf'.

The 'rbf' kernel is one of the most commonly used kernels in SVM models. It is a popular choice because it can effectively capture complex, non-linear relationships between the data points. The 'kernel' hyperparameter determines the type of kernel function used in the SVM model.

2. Feature Importance:



* In the code the accurate scores are printed.

Based on the feature importance scores, the top features contributing to the success of the model are as follows:

1. file_type_prob_trid (0.112): This feature represents the file type with the highest probability, as determined by the TRID algorithm. The high importance suggests that certain file types have a strong association with either malicious or benign files. Therefore, the identification of file types is a crucial factor in determining the nature of the file.

2. file_type_trid (0.088): Although slightly less influential than file_type_prob_trid, it remains a significant aspect in our model. The file type provides critical contextual information, aiding in the distinction between malicious and benign files.

3. printables (0.080): The number of printable characters in a file is an informative feature. Malicious files often exhibit distinct patterns in terms of printable characters. Hence, this feature plays an essential role in our model, capturing the variations in printable character frequencies between the two classes.

4. urls (0.076): The occurrence of URLs is a strong indicator of potentially malicious behavior. This feature allows the model to identify files that may be involved in malicious activities through the detection of URLs within them.

5. B (0.068): The specific details of feature B are not provided, but it demonstrates significant importance in our model. Its contribution suggests that it captures essential information related to the distinction between malicious and benign files.

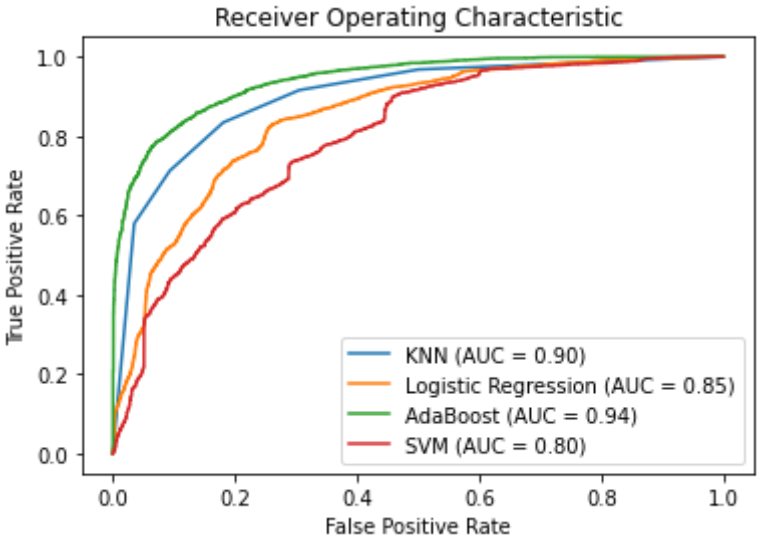
6. avlength_size_ratio (0.068): This is an engineered feature we created which has a notable importance score. We decieded to create it due to the fact that the avlength feature's histogram in the exploration part exclusively contains values associated with malicious files. This ratio suggests that the relative proportion of string lengths within the file is relevant to differentiating between malicious and benign files. It is providing insights into the compactness or density of malicious files.

7. size_vsize_ratio (0.068): a feature which we created as well, represents the ratio of the file size to the virtual size. This feature captures the relationship between the physical size of the file and its virtual size

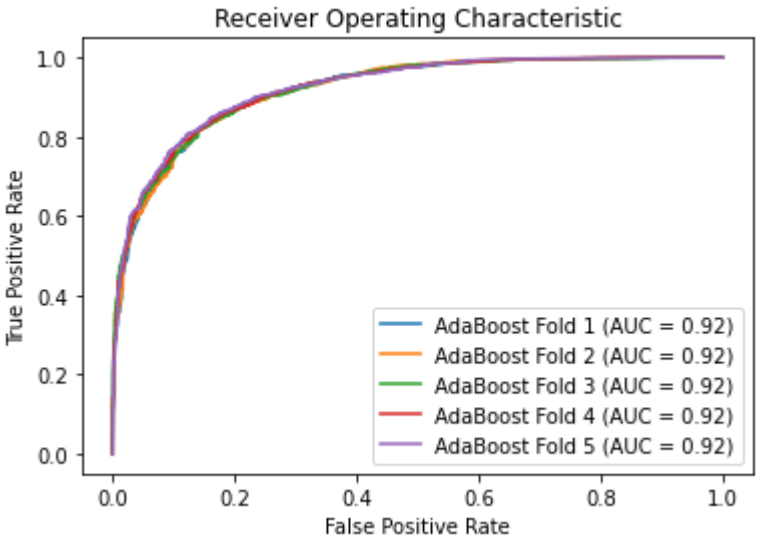
when loaded into memory. Different ratios can indicate variations in the file's structure or compression techniques, contributing to the classification task.

PART 4-MODELS EVALUATION:

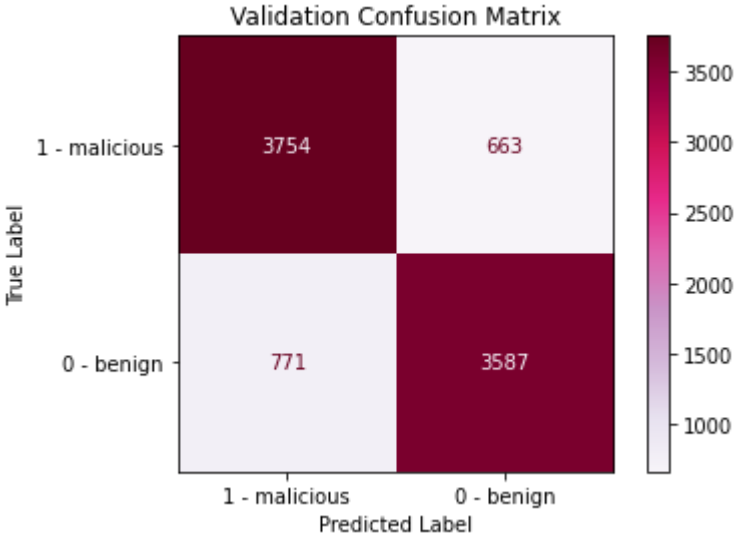
1. A plot displays the comparative performance of different models by showcasing the average Receiver Operating Characteristic (ROC) across multiple folds:



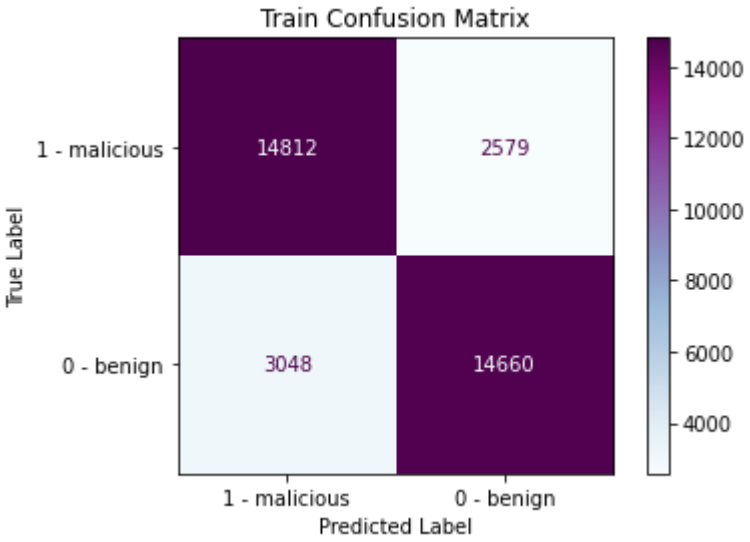
2. A plot showing each fold for AdaBoost Model:



3. Validation Confusion Matrix for AdaBoost:



4. Train Confusion Matrix for AdaBoost:



- **<u>Each partner's responsibility and contribution to the work:</u>**

Each partner actively participated and collaborated in the implementation of the project, working together in full cooperation. We maintained regular communication and held numerous meetings, both in-person and via Zoom, to ensure synchronized progress and facilitate efficient collaboration.

During these meetings, we allocated tasks and responsibilities, dividing the work among ourselves. Each partner took ownership of their assigned tasks and strived to optimize their performance. We shared ideas, insights, and feedback, leveraging our individual strengths and expertise to contribute to the success of the project.

Working together was a pleasure, as our shared commitment and dedication to the project drove us to achieve the best possible outcomes. Our collaboration facilitated the exchange of knowledge and ideas, enabling us to overcome challenges and reach our objectives successfully.