אוניברסיטת תל אביב
בעקבות הלא נודע

Final Project- Report

Machine Learning

Ofri Nussel, Tomer Blumental

Part 1- Exploration & Visualization

At the start of the project, we conducted an initial examination of the data, gaining a broad understanding of its characteristics. This exploration yielded fundamental information that will be valuable for preprocessing purposes later.

Upon reviewing the provided data, we observed that it consists of a dataset with multiple features accompanied by a corresponding label column. Each column represents a specific feature, and we have access to explanations for each feature from the project guidelines.

Here are some key findings we gathered during our initial analysis:
Data Size: The DataFrame comprises 60,000 rows and 24 columns
Missing Values: Several columns contain missing values, which can be identified by the "Non-Null Count" for each column. Examples of such columns include vsize, imports, exports, has_debug, and others. Understanding the extent and patterns of missing data is crucial for data preprocessing and analysis, and we will address this in detail later in the project.
Numerical Features: Numerous columns, such as vsize, imports, exports, and more, contain numerical data
Categorical Features: Columns like file_type_trid store categorical data, representing different file types. Analyzing the distribution of these categories and exploring their relationship with the label column can offer informative insights for classification tasks.
During the exploration process, we computed basic statistics on our data, including measures such as mean, maximum, minimum, and standard deviation. These statistics can provide valuable insights into the distribution of our data and help us identify patterns or potential outliers. As we continue with the project, we will further investigate these outliers and explore their significance.
Based on the initial analysis, we discovered the presence of null values in the data. To gain further insights, we examined these values more closely and found that they constitute a relatively small percentage of the total observations, with the largest percentage being approximately 6%(appendix 1). Given this scenario, we can impute the missing values.
Furthermore, although **duplicates were not explicitly discussed in the course**, we proactively checked for their presence in the data. Duplicates in a machine learning project can have detrimental effects, such as introducing inaccuracies, biasing models, diminishing performance, and leading to misinterpreted data. By removing duplicates, we can enhance data quality, improve model accuracy, and increase computational efficiency. Upon examination, we determined that there are no duplicates present in the dataset. (the specific prosses of finding duplicates is in the notebook under the headline “Duplicates”)
We conducted data visualization to observe the distribution of each feature. Upon reviewing the distribution overview of each variable, it became apparent that the variable 'A' displayed a

distribution that closely resembled a normal distribution compared to the other variables. Furthermore, the variables 'size', 'vsize', 'imports', and 'exports' demonstrated similar ranges or magnitudes in their respective values. (Appendix 2)

We utilized two different methods, Spearman and Pearson, to calculate the correlation. Spearman measures the monotonic relationship between variables, whereas Pearson measures the linear relationship. A high correlation indicates a strong association between variables. Positive correlation suggests that the variables tend to increase together, negative correlation implies they change in opposite directions, and a correlation close to zero suggests no significant relationship. (Appendices 3-4).

From the correlation matrices provided, we can summarize the relationships between different variables in the dataset as follows:

Positive Correlations:

There is a positive correlation between "size" and "imports" (0.187422), suggesting that larger file sizes tend to have more imports.

"size" and "numstrings" exhibit a strong positive correlation (0.898811), indicating that larger files tend to have more strings.

There is a positive correlation between "size" and "MZ" (0.715902), suggesting that larger file sizes are associated with higher MZ header values.

"size" and "printables" show a positive correlation (0.412911), indicating that larger file sizes tend to have more printable characters.

Negative Correlations:

"has_debug" and "has_relocations" have a negative correlation (-0.337964), indicating that files without debug information are more likely to have relocations.

There is a weak negative correlation (-0.043614) between "size" and "label," suggesting a weak relationship between file size and the assigned label.

Weak Correlations:

Several variables, including "vsize," "exports," "has_resources," "has_signature," "has_tls," "symbols," "paths," "urls," "registry," "file_type_prob_trid," and "A," show weak correlations (close to 0) with the "size" variable.

It's important to note that correlation does not imply causation, and the strength and direction of the relationship between variables can vary. Correlation coefficients close to -1 or 1 indicate a strong linear relationship, while coefficients close to 0 suggest a weak or no linear relationship. Further analysis and consideration of the data context are necessary to draw meaningful conclusions from these correlations.

We observed that the variable 'A' has the weakest correlation with other variables, as indicated by both Pearson and Spearman correlations. Therefore, we made the decision to remove this feature from the dataset to reduce the dimensionality and potentially improve the performance of our analysis or model. (Appendix 5).

Part 2- Preprocessing

To prevent information leakage, we split the data into training and testing sets before preprocessing. This ensures that the preprocessing steps are applied independently to each set, maintaining the integrity of the evaluation and preventing any leakage of information from the testing set into the training process.

Based on the correlation matrix, we observed a strong correlation between the features "size" and "numstring." To reduce the dimensionality of the data, we made the decision to **create a new feature** by multiplying these two features together. Consequently, we **removed** the original "size" and "numstring" features from the dataset. Furthermore, as mentioned in the previous paragraph, we have also removed the 'A' feature from the dataset.
In addition, we are removing the identifier column because each file has its own unique name, and it does not provide any useful information for the prediction process.

During dataset exploration, it was discovered that the features "file_type_trid" and "c" were non-numerical. To simplify analysis and identify patterns, these features were **categorized**.
The category 'C' will not significantly increase the dimensionality because it has only 8 unique categories. However, we observed that the 'file_type_trid' category has 88 different categories, which can lead to a substantial increase in dimensionality.
Initially, our approach was to divide all the categories into 5 groups based on similar words. For instance, all categories containing the word 'Win' would be grouped together. We then applied One-Hot-Encoding to represent these groups. The main challenge was finding appropriate group names. Initially, we performed this task manually, but later attempted clustering techniques. However, in the end, we decided to opt for a simpler approach.
By simplifying the process, we avoided the complexity of grouping and clustering and instead focused on a more straightforward method.
Our chosen approach for categorization involves using one-hot encoding to convert categorical features into a binary representation. Subsequently, we aim to select the most frequent categories from the encoded data. This approach allows us to identify the categories that occur most frequently in the dataset and focus on them for further analysis or modeling.
We assumed that selecting 75% and 75% of the data would be sufficient to explain and interpret the patterns. We assume that keeping these variables only on 75% of the data is enough to give a good prediction.
After categorizing the data and dealing with **missing values**, we imputed the missing values using the median for numeric features. The median is a robust measure that is less affected by outliers, making it a suitable choice.
Before proceeding with normalization, we conducted an **outlier** analysis using the Isolation Forest algorithm. The Isolation Forest algorithm is an efficient and scalable method for

detecting anomalies by creating random splits in the data. It is especially effective in identifying outliers in numeric features while taking into account the associated labels or classes. This analysis helped us identify and isolate potential outliers in the dataset. (Appendix 6)

Based on the plots, it is evident that a significant number of outliers are labeled as 0. However, in this research phase, we have chosen not to remove these outliers to ensure that we do not lose any crucial information. Instead, we have opted to try an alternative approach. We will define lower and upper limits for the values in our data using box plots. Any data points that fall outside these limits will be considered outliers and will be replaced with the respective limit values. This method allows us to mitigate the impact of outliers while retaining the valuable information they may provide (Appendix 7).

After attempting to remove the outliers from the data, we observed a significant decrease in the auc scores. As a result, we made the decision **not to remove the outliers.**

Next, we preformed data **normalization** to scale the features from 0 to 1 for several reasons: to equalize feature scales, avoid bias, accelerate convergence of optimization algorithms, and improve model performance. Normalization ensures that all features contribute proportionally, prevents domination by larger values, and results in more accurate predictions.

Prior to performing dimensionality reduction, feature removal, and creation of new features, we evaluated the **AUC scores** of different models (results in the notebook- “First model try”).

Next, we attempted to improve the accuracy of the model by performing **dimensionality reduction.**

High-dimensional data has many features relative to the number of samples, presenting challenges such as sparsity, complexity, overfitting, and increased computational requirements. To address these issues, dimensionality reduction techniques can reduce the number of dimensions while retaining important information. Given the dataset's high dimensionality, we will apply dimensionality reduction methods.

PCA- Our initial approach to reduce dimensionality is through PCA. We determined the number of components needed to explain 99% of the data, which resulted in 18 components (Appendix 8). The new results are in the notebook (“PCA”).

After applying PCA, we observed decrease in the AUC score. Despite reducing the dimensionality of the data and preserving the essential information, the decrease in dimensionality did not lead to any significant enhancement in the model's predictive performance. It seems that the reduced representation of the data resulted in the loss of important patterns and structures, which ultimately led to a decrease in the model's overall performance.

The Preprocessing function is used to perform the **preliminary data preprocessing on the test set.**

Part 3- Models

To find the **best hyperparameters** for each model and mitigate overfitting, we utilized either grid search or random search along with cross-validation using K-fold. (results in the notebook- “Part 3- Models”).
Explanation on the hyperparameters in Appendix 9.

Part 4- Model evaluation

In order to evaluate the models, we performed **K-fold cross-validation** and plotted the results for each fold and each model (Appendix 10).
Based on the graph, we can observe that the mean AUC scores for the selected hyperparameters are as follows: KNN: 0.86, Logistic Regression: 0.79, AdaBoost: 0.92, MLP: 0.88.
Based on the graph and the mean AUC scores, it appears that the AdaBoost and MLP models achieved the highest average performance among the evaluated models. However, based on these results, the decision is to select the MLP model as the preferred model for further analysis because we want to avoid overfitting. Additionally, when examining the performance of the different models, we observe that the **MLP model exhibits the smallest gaps between the training and validation results and had the highest score on the validation set.** This indicates that the MLP model generalizes well to unseen data and has a **relatively lower risk of overfitting compared to the other models in the pipeline**.
Moreover, we examined the results of the **confusion matrix** on both the training and validation datasets to check for the possibility of overfitting (Appendix 11A- confusion matrix on the validation, 11B- confusion matrix on the train, results and explanations in the notebook- “confusion matrix”).

Part 5- prediction

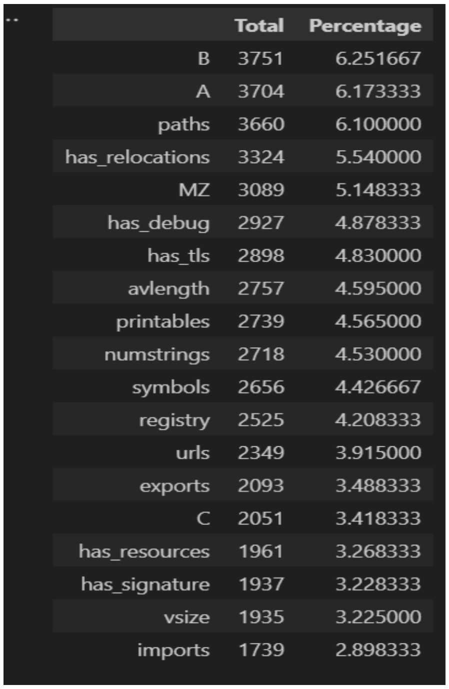
In this part, we read the data from the test file and make predictions using the trained model.

Part 6- Using tools not taught in the course

As mentioned earlier, we have decided to utilize the duplicates tool, which was not covered in the course.
*overall runtime in our computer 2372.88 seconds (at the very end of our notebook)

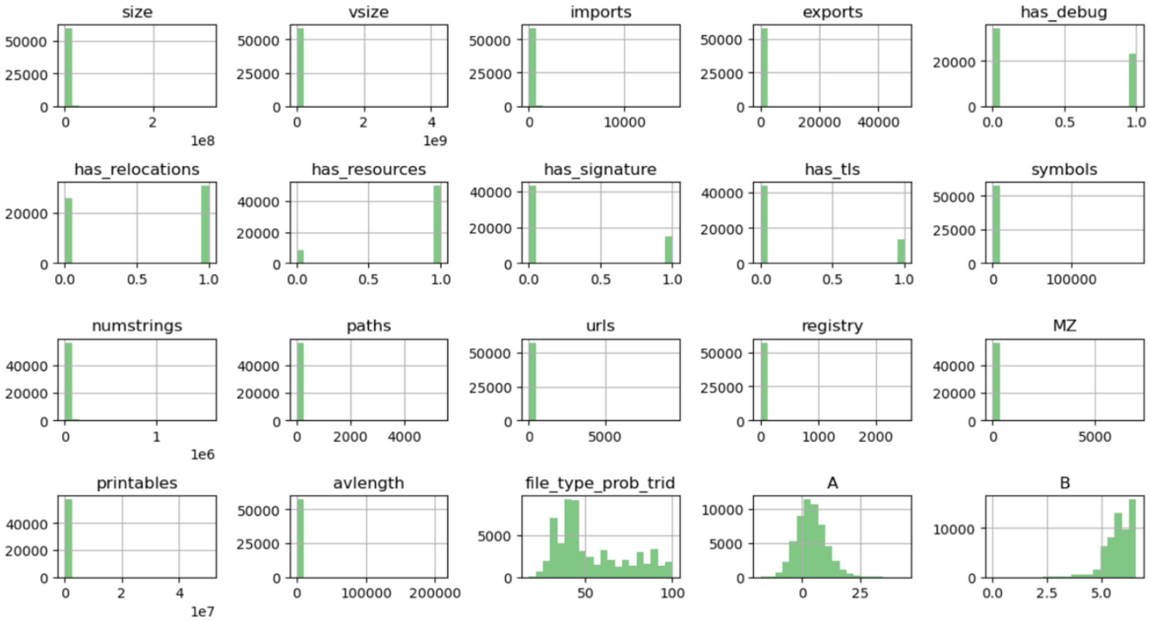
Appendices

1.

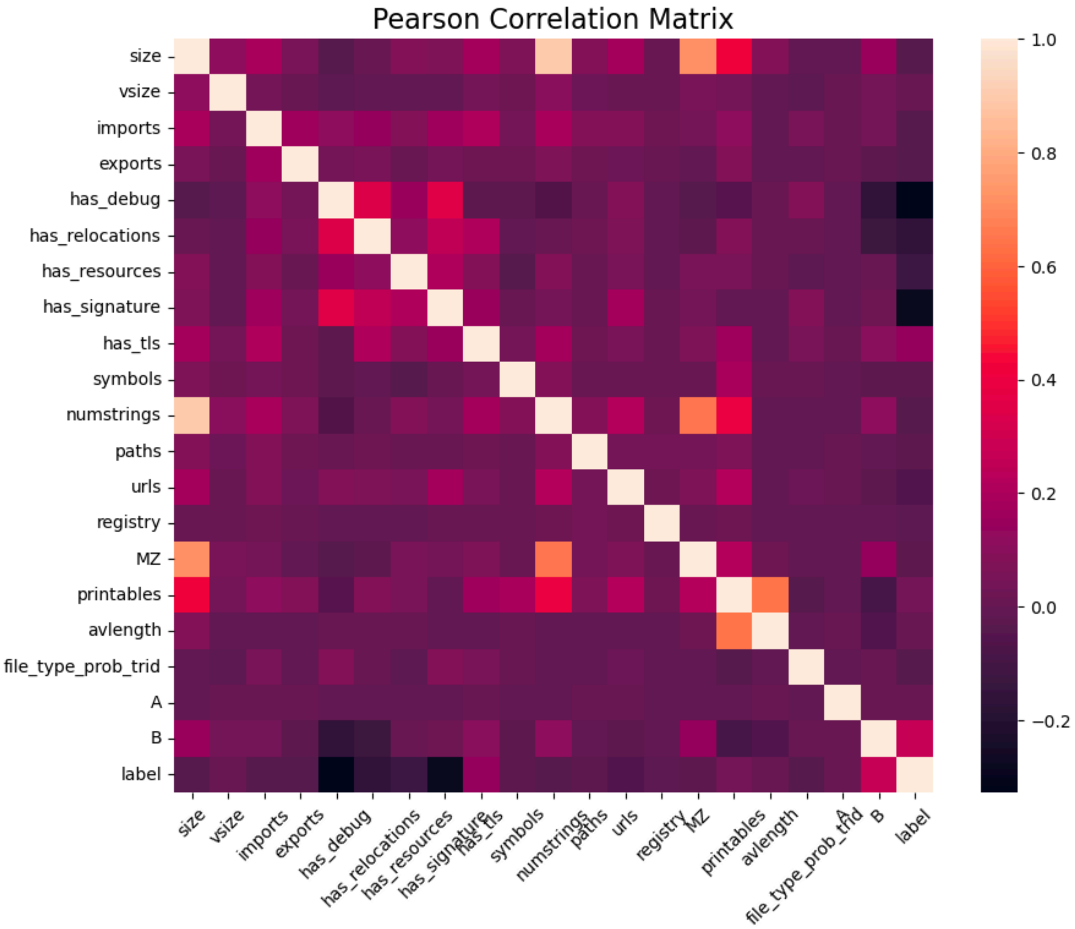


	Total	Percentage
B	3751	6.251667
A	3704	6.173333
paths	3660	6.100000
has_relocations	3324	5.540000
MZ	3089	5.148333
has_debug	2927	4.878333
has_tls	2898	4.830000
avlength	2757	4.595000
printables	2739	4.565000
numstrings	2718	4.530000
symbols	2656	4.426667
registry	2525	4.208333
urls	2349	3.915000
exports	2093	3.488333
C	2051	3.418333
has_resources	1961	3.268333
has_signature	1937	3.228333
vsize	1935	3.225000
imports	1739	2.898333

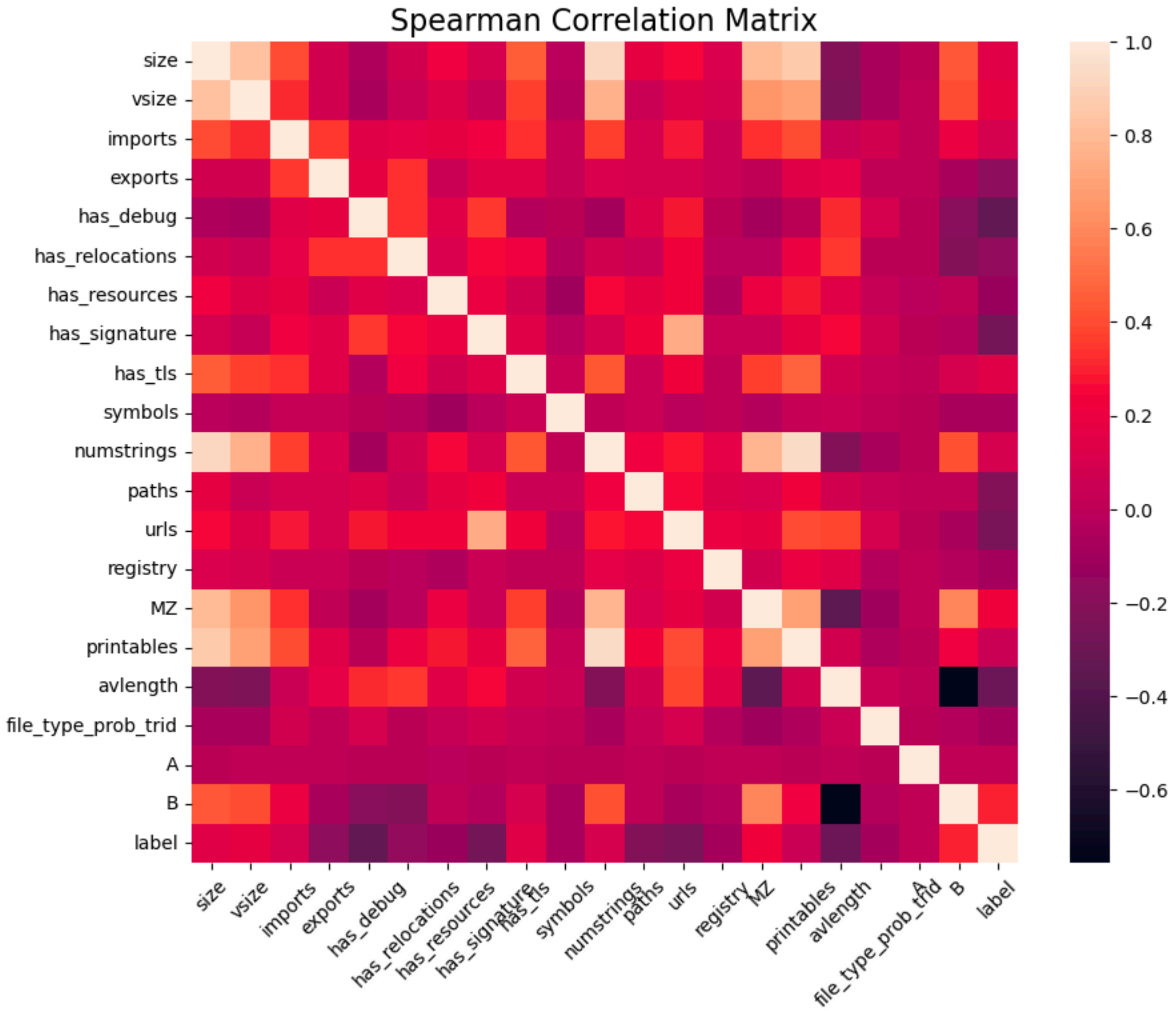
.2



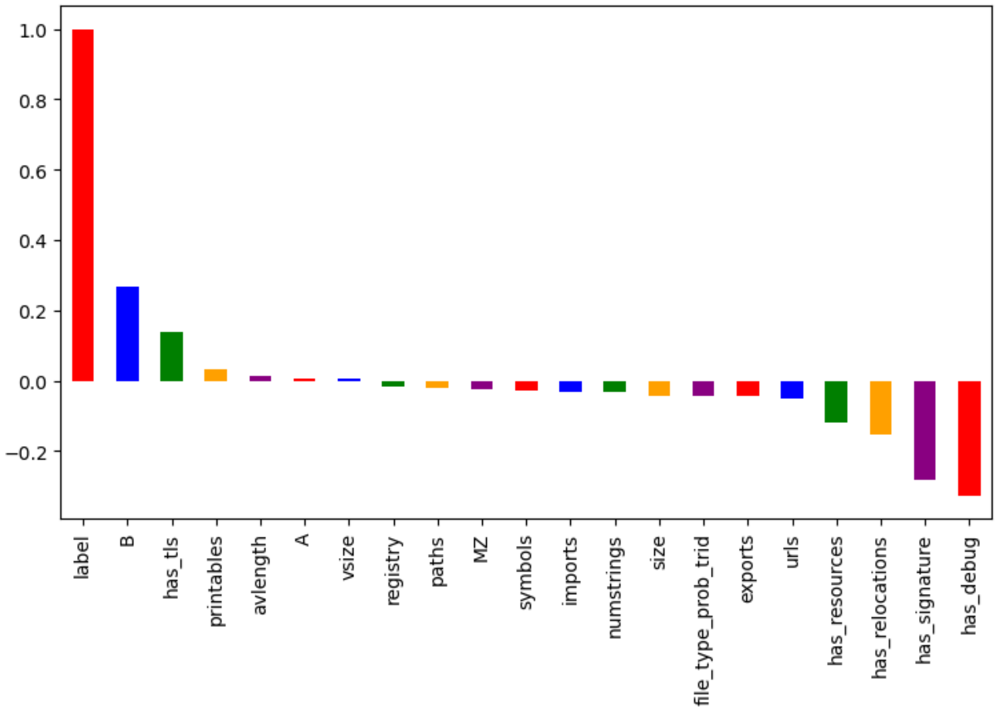
3.


Pearson Correlation Matrix
size
vsize
imports
exports
has_debug
has_relocations
has_resources
has_signature
has_tls
symbols
numstrings
paths
urls
registry
MZ
printables
avlength
file_type_prob_trid
A
B
label
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

4.

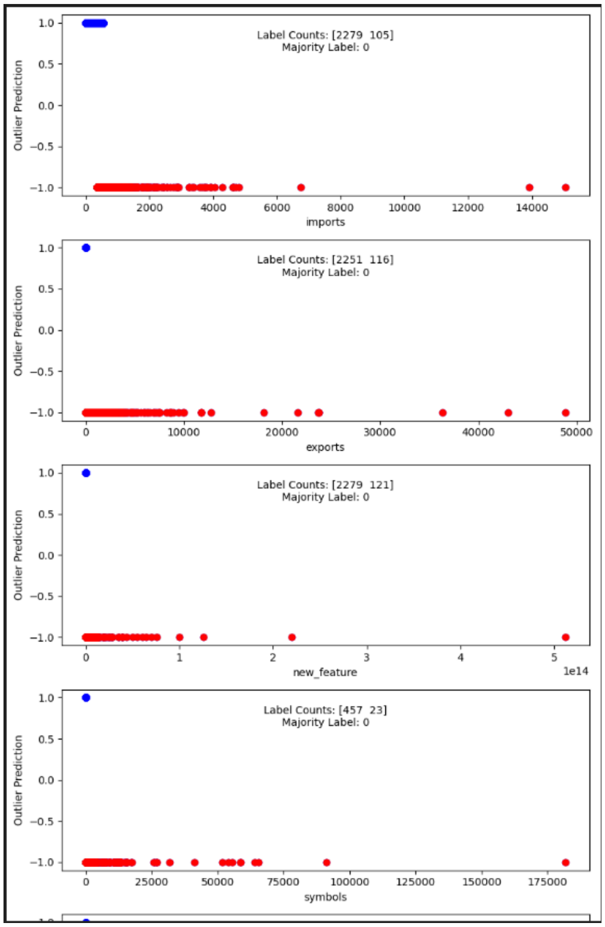

Spearman Correlation Matrix
size
vsize
imports
exports
has_debug
has_relocations
has_resources
has_signature
has_tls
symbols
numstrings
paths
urls
registry
MZ
printables
avlength
file_type_prob_trid
A
B
label
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

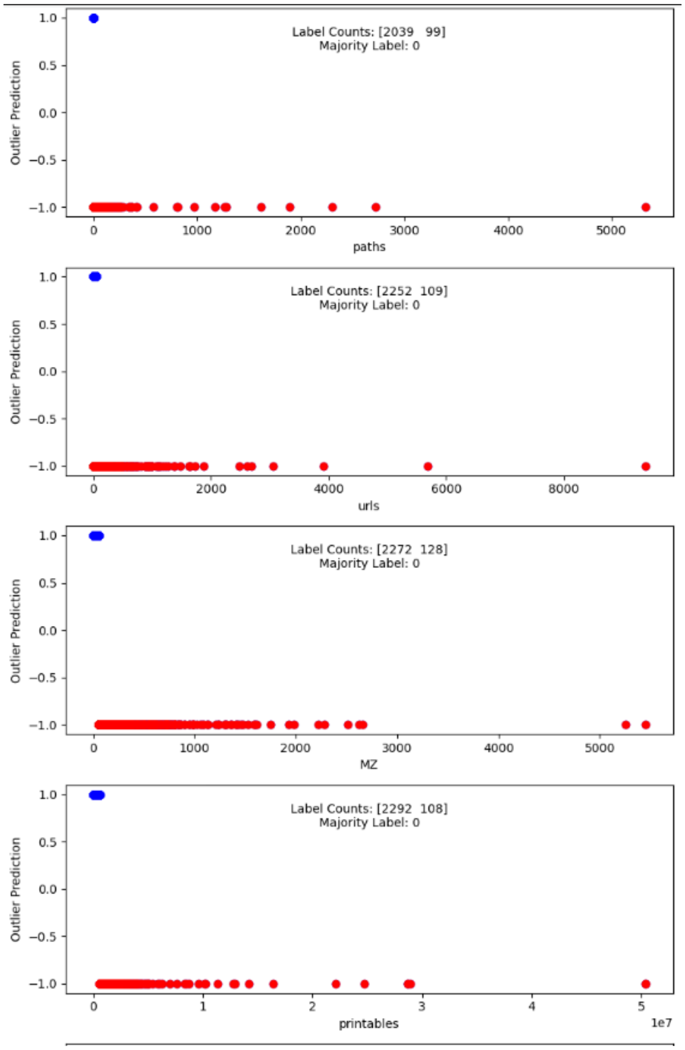
5.

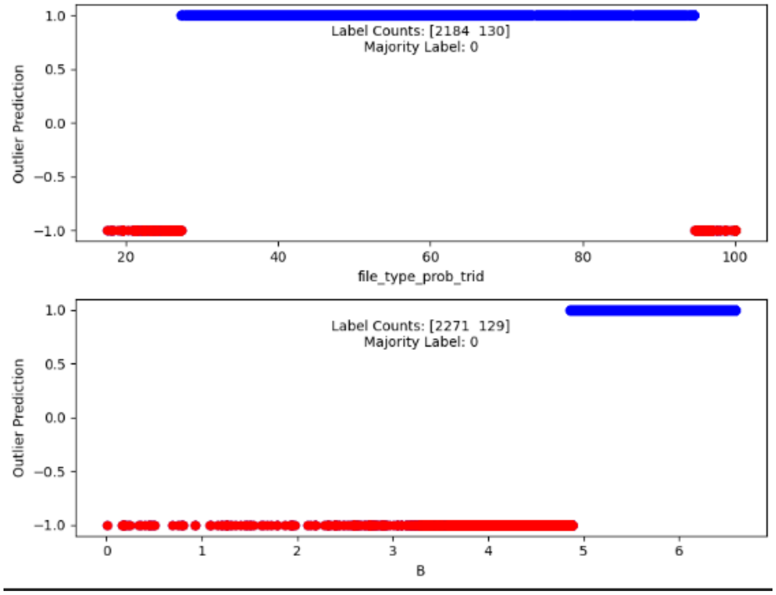


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
label
B
has_tls
printables
avlength
A
vsize
registry
paths
MZ
symbols
imports
numstrings
size
file_type_prob_trid
exports
urls
has_resources
has_relocations
has_signature
has_debug

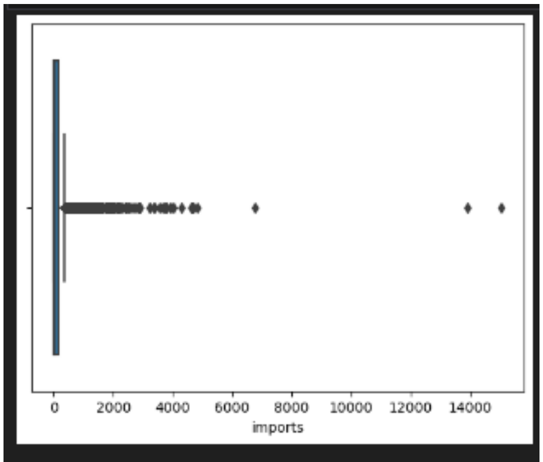
6.

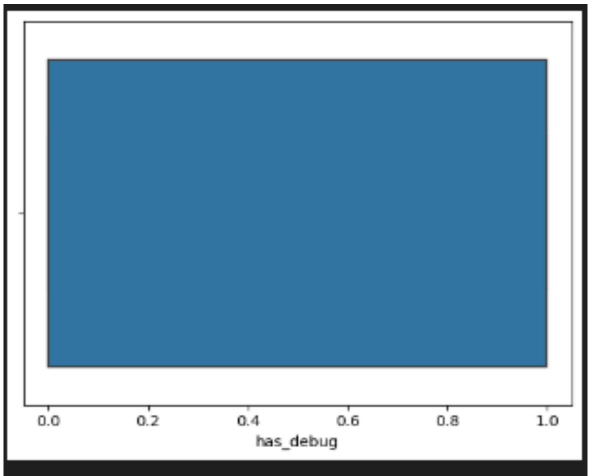

Outlier Prediction
1.0
0.5
0.0
−0.5
−1.0
Label Counts: [2279 105]
Majority Label: 0
0
2000
4000
6000
8000
10000
12000
14000
imports
Outlier Prediction
Label Counts: [2251 116]
Majority Label: 0
0
10000
20000
30000
40000
50000
exports
Outlier Prediction
Label Counts: [2279 121]
Majority Label: 0
0
1
2
3
4
5
1e14
new_feature
Outlier Prediction
Label Counts: [457 23]
Majority Label: 0
0
25000
50000
75000
100000
125000
150000
175000
symbols

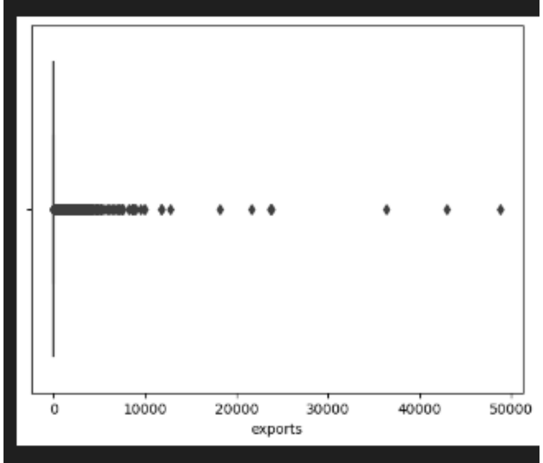

Outlier Prediction
1.0
0.5
0.0
−0.5
−1.0
Label Counts: [2039 99]
Majority Label: 0
0
1000
2000
3000
4000
5000
paths
Outlier Prediction
Label Counts: [2252 109]
Majority Label: 0
0
2000
4000
6000
8000
urls
Outlier Prediction
Label Counts: [2272 128]
Majority Label: 0
0
1000
2000
3000
4000
5000
MZ
Outlier Prediction
Label Counts: [2292 108]
Majority Label: 0
0
1
2
3
4
5
1e7
printables

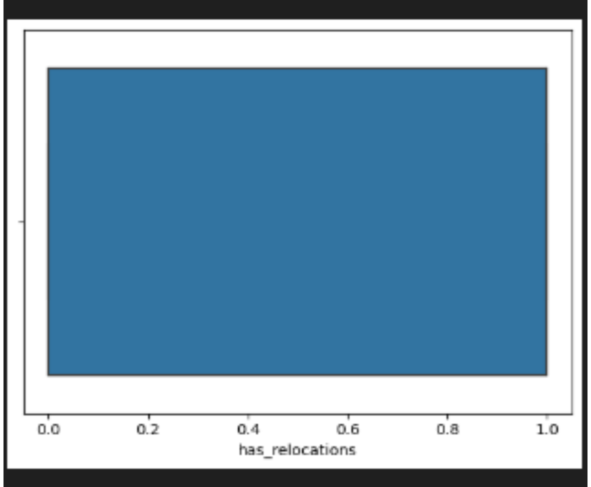

Outlier Prediction
1.0
0.5
0.0
−0.5
−1.0
Label Counts: [2184 130]
Majority Label: 0
20
40
60
80
100
file_type_prob_trid
Outlier Prediction
Label Counts: [2271 129]
Majority Label: 0
0
1
2
3
4
5
6
B

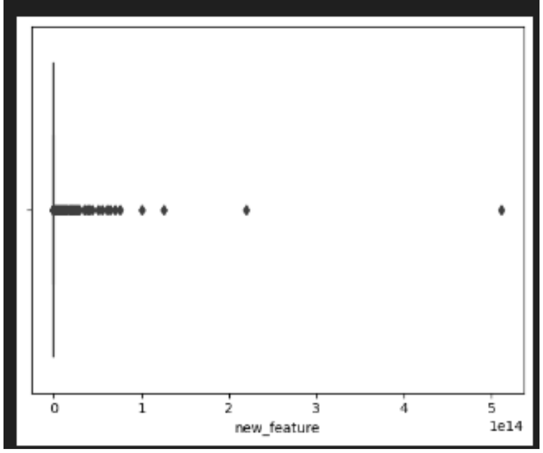
7.

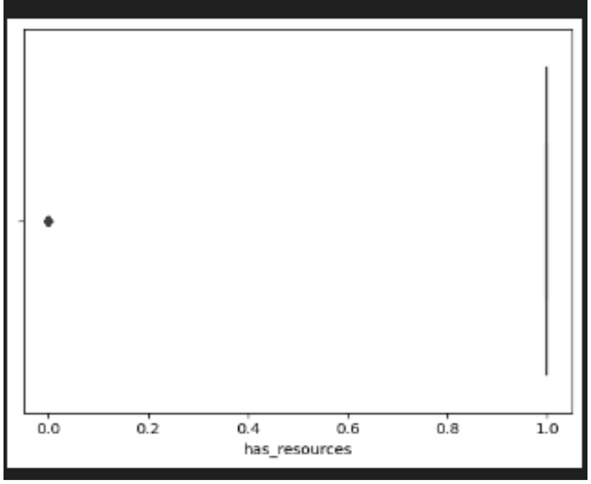

0
2000
4000
6000
8000
10000
12000
14000
imports

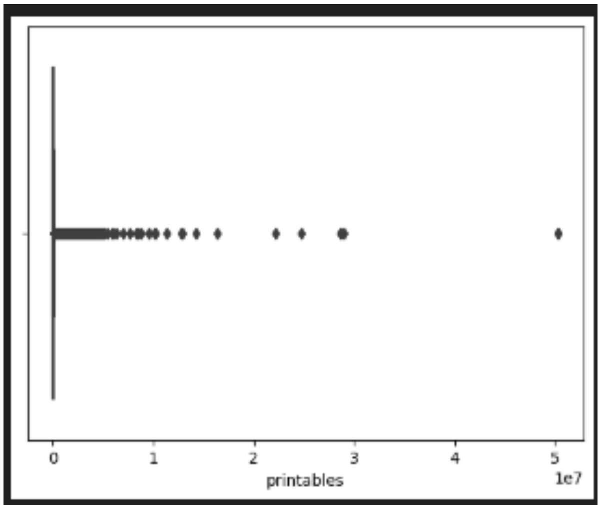

0.0
0.2
0.4
0.6
0.8
1.0
has_debug

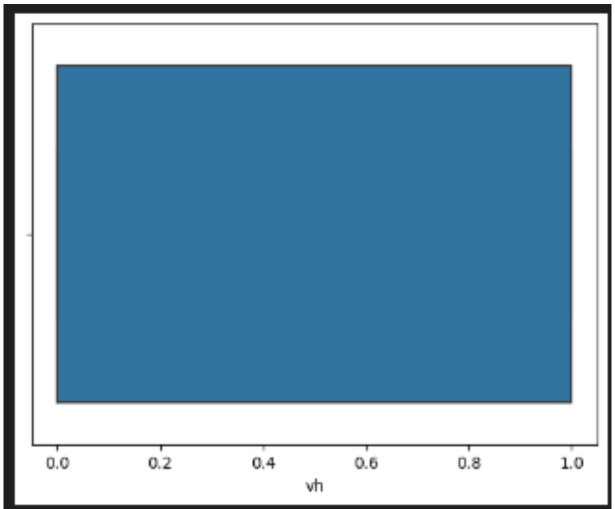

0
10000
20000
30000
40000
50000
exports

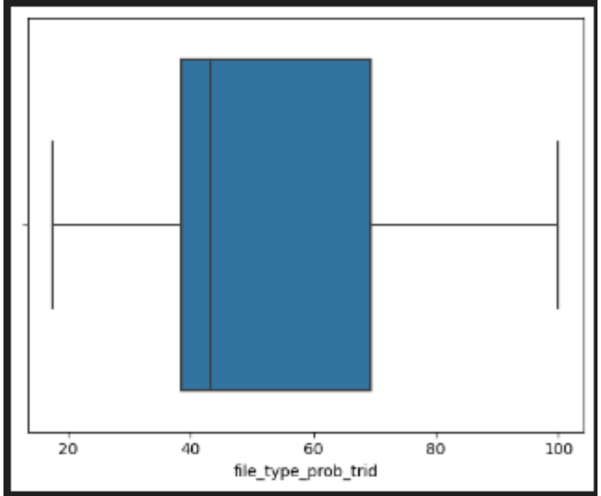

0.0
0.2
0.4
0.6
0.8
1.0
has_relocations

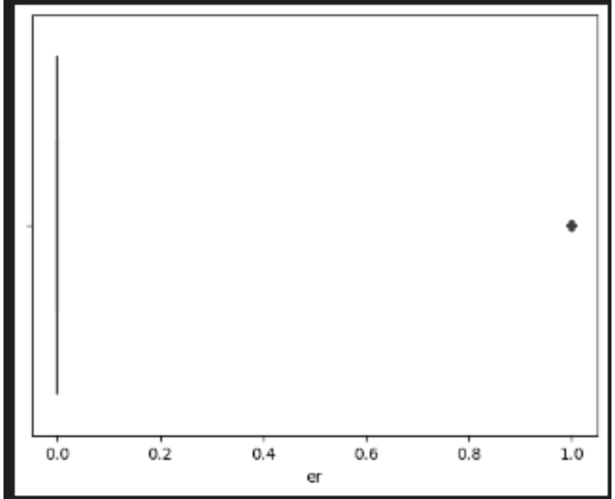

0
1
2
3
4
5
1e14
new_feature

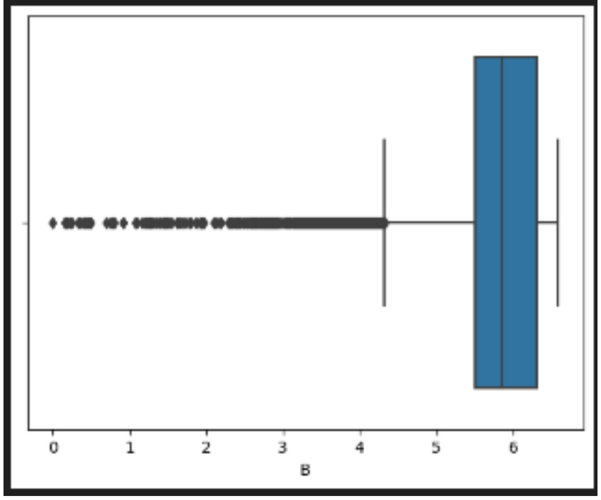

0.0
0.2
0.4
0.6
0.8
1.0
has_resources

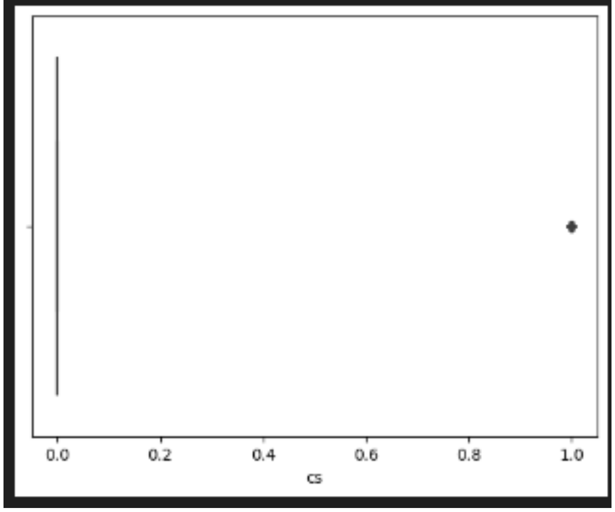

0
1
2
3
4
5
1e7
printables

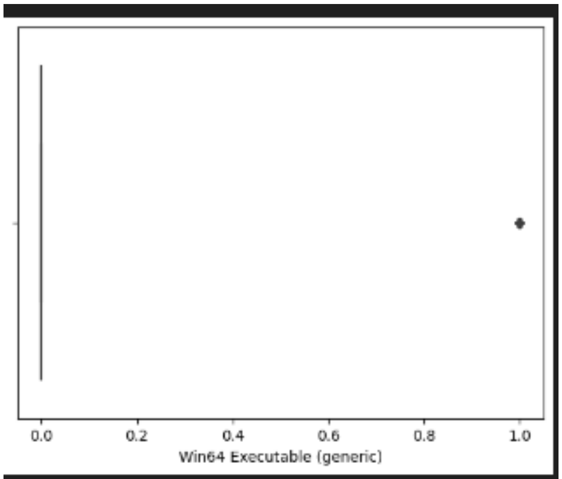

0.0
0.2
0.4
0.6
0.8
1.0
vh

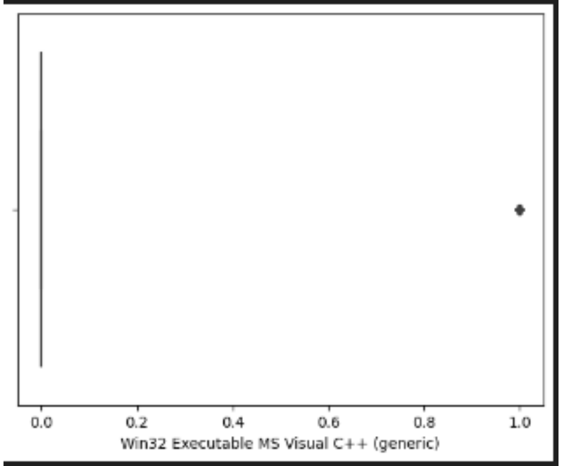

20
40
60
80
100
file_type_prob_trid

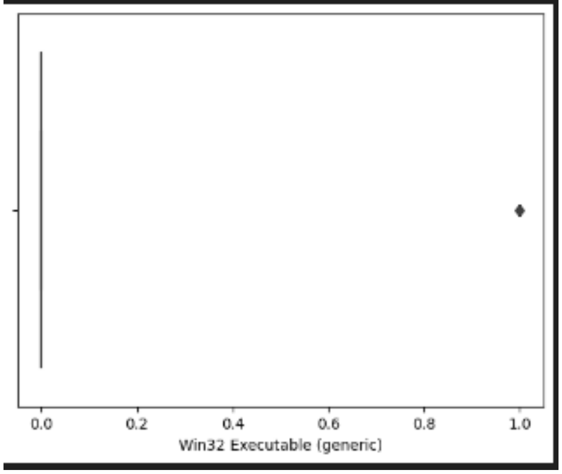

0.0
0.2
0.4
0.6
0.8
1.0
er

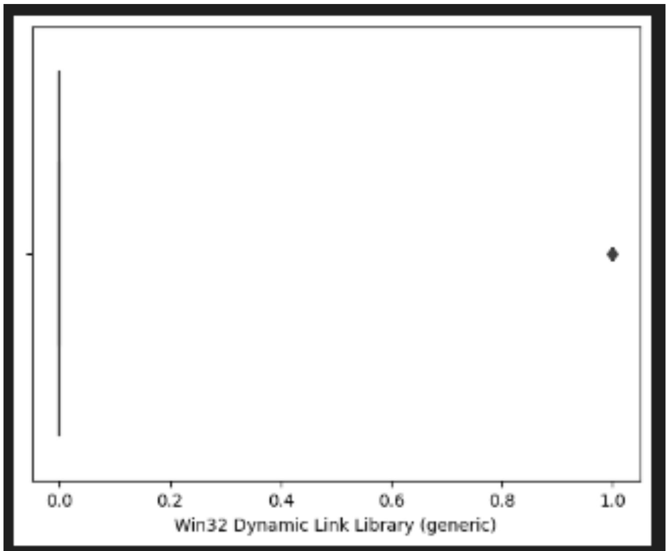

0
1
2
3
4
5
6
B

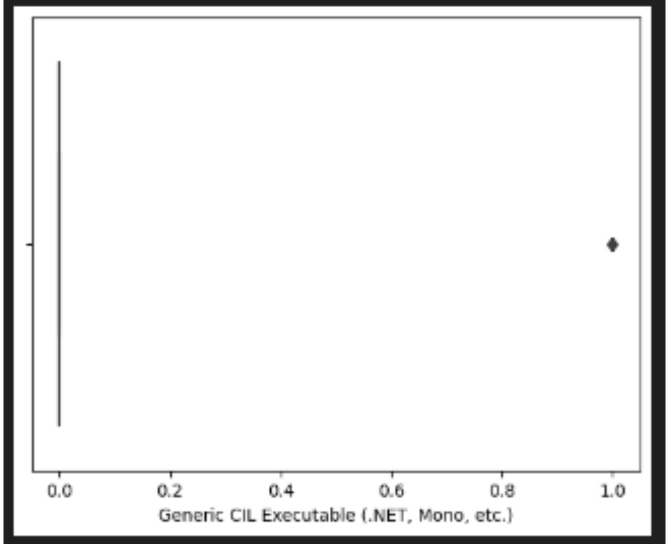

0.0
0.2
0.4
0.6
0.8
1.0
cs

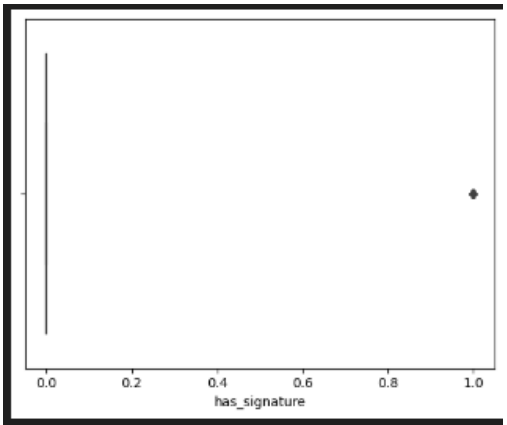

0.0
0.2
0.4
0.6
0.8
1.0
Win64 Executable (generic)

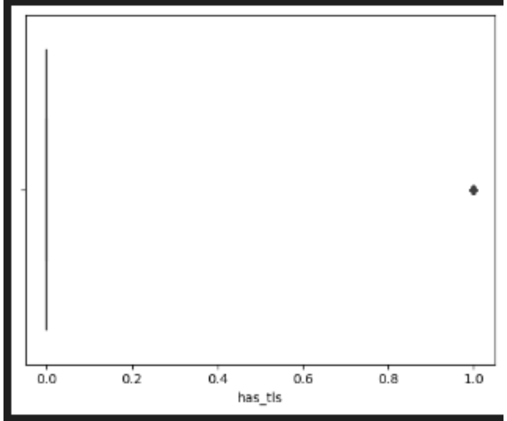

0.0
0.2
0.4
0.6
0.8
1.0
Win32 Executable MS Visual C++ (generic)

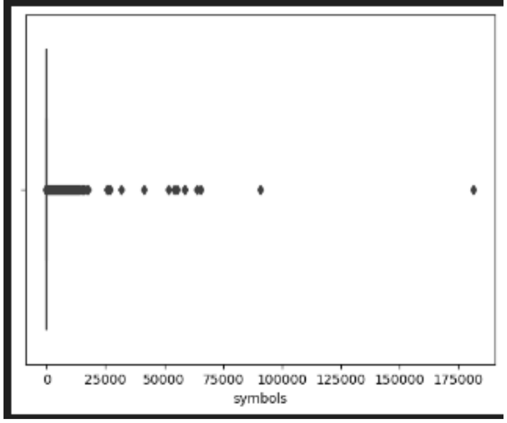

0.0
0.2
0.4
0.6
0.8
1.0
Win32 Executable (generic)

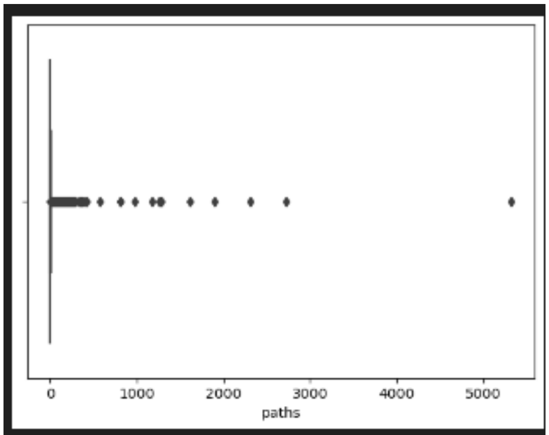

0.0
0.2
0.4
0.6
0.8
1.0
Win32 Dynamic Link Library (generic)

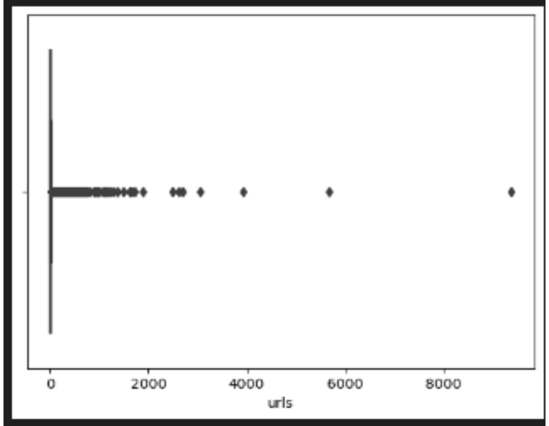

0.0
0.2
0.4
0.6
0.8
1.0
Generic CIL Executable (.NET, Mono, etc.)

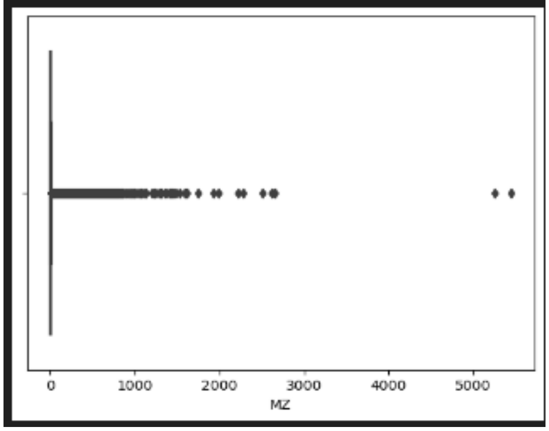

0.0
0.2
0.4
0.6
0.8
1.0
has_signature


0.0
0.2
0.4
0.6
0.8
1.0
has_tls


0
25000
50000
75000
100000
125000
150000
175000
symbols


0
1000
2000
3000
4000
5000
paths


0
2000
4000
6000
8000
urls


0
1000
2000
3000
4000
5000
MZ

8.

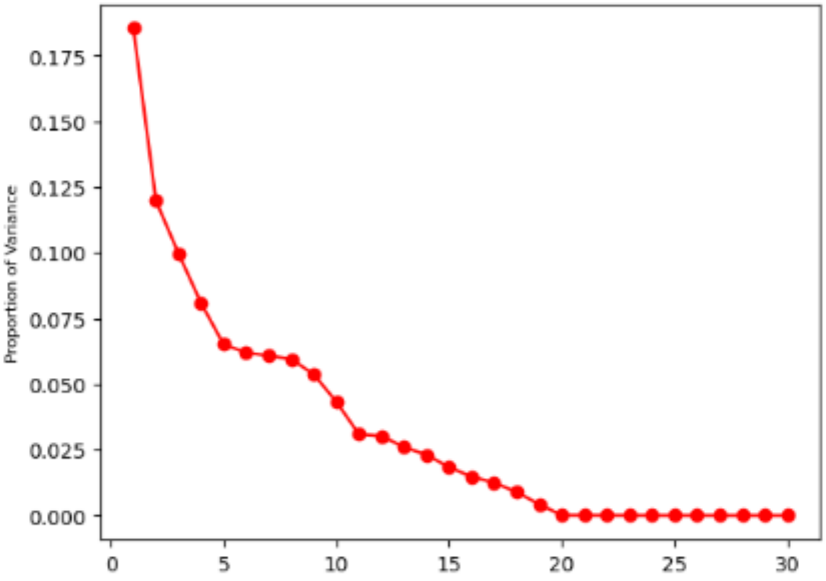


Figure 1: Scree Plot

9. The hyperparameters of a model play a crucial role in balancing the trade-off between bias and variance.

1. Logistic Regression: C=10.0, max_iter=50, penalty='l1', solver='liblinear'
- max_iter: The maximum number of iterations for the solver to converge. Increasing max_iter can reduce bias but may increase variance if the model overfits the data.
- penalty: The regularization term to prevent overfitting. The choice of penalty (e.g., L1 or L2) affects the bias-variance trade-off.
- solver: The optimization algorithm used to fit the model. Different solvers can lead to variations in bias and variance.

2. K-Nearest Neighbors (KNN) Classifier: The optimal K: 7
- n_neighbors: The number of neighbors considered for classification. Higher values of n_neighbors can decrease variance but increase bias.

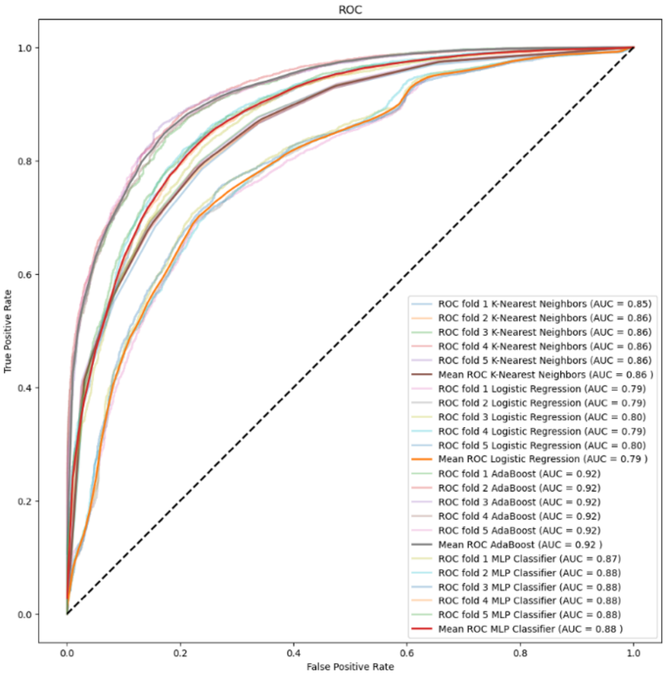
3. AdaBoost Classifier:
base_estimator=DecisionTreeClassifier(max_depth=1),learning_rate=0.6825806912012297, n_estimators=96, random_state=0)
- base_estimator: The base estimator used for boosting. The choice of base estimator, such as Decision Tree Classifier, can impact bias and variance.
- learning_rate: Controls the contribution of each base estimator. A higher learning rate can increase variance and reduce bias.
- n_estimators: The number of base estimators used. Increasing n_estimators can reduce bias but may lead to higher variance.

4. MLP Classifier: alpha=0.001, random_state=0
- alpha: The regularization parameter that controls the complexity of the neural network. Higher values of alpha increase bias but decrease variance.
- random_state: The random seed for reproducibility. It does not directly affect the bias-variance trade-off.

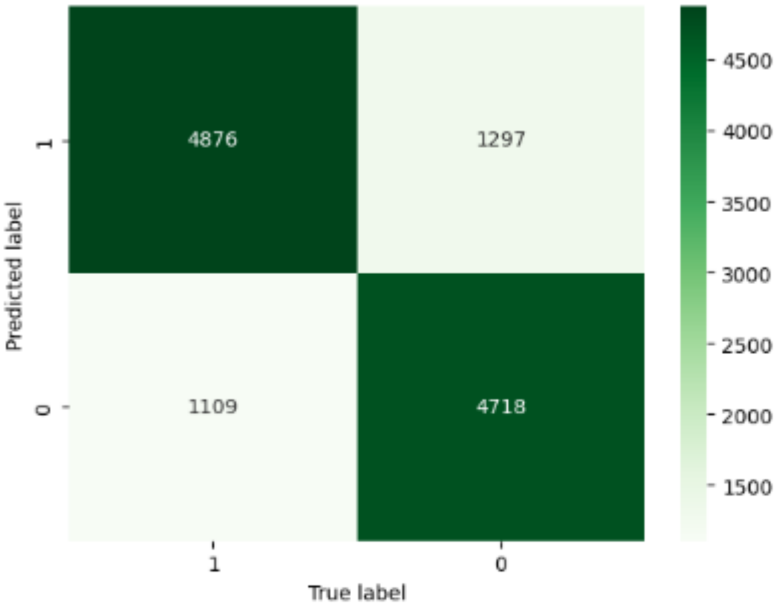
*The unspecified hyperparameters are the default values:
MLP (Multilayer Perceptron):
activation: "relu" (Rectified Linear Unit)
learning_rate: "constant"
learning_rate_init: 0.001
max_iter: 200
hidden_layer_sizes: (100,)

KNN (K-Nearest Neighbors):
weights: "uniform"
algorithm: "auto"
p: 2
*We chose to use Randomized in MLP and AdaBoost in order to reduce runtime.

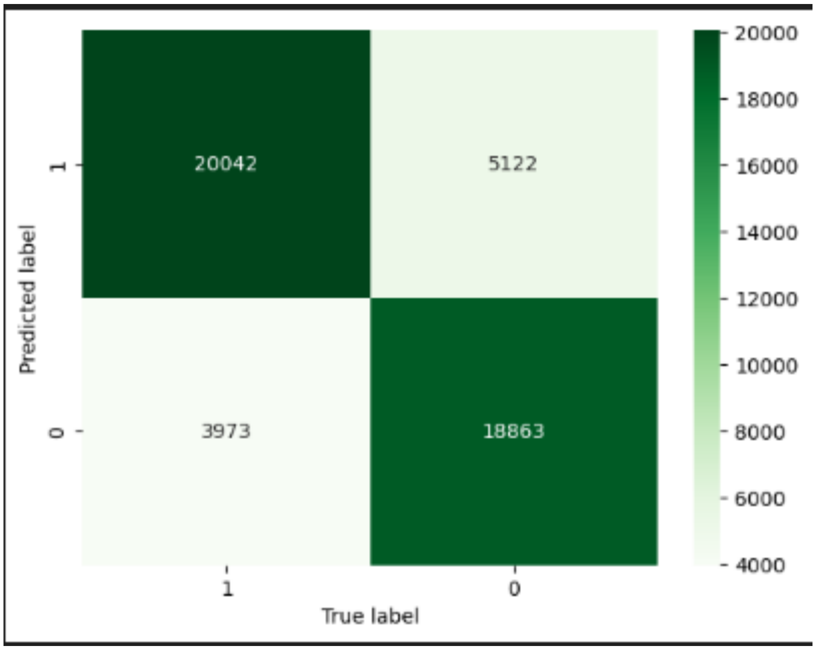
10.


ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC fold 1 K-Nearest Neighbors (AUC = 0.85)
ROC fold 2 K-Nearest Neighbors (AUC = 0.86)
ROC fold 3 K-Nearest Neighbors (AUC = 0.86)
ROC fold 4 K-Nearest Neighbors (AUC = 0.86)
ROC fold 5 K-Nearest Neighbors (AUC = 0.86)
Mean ROC K-Nearest Neighbors (AUC = 0.86)
ROC fold 1 Logistic Regression (AUC = 0.79)
ROC fold 2 Logistic Regression (AUC = 0.79)
ROC fold 3 Logistic Regression (AUC = 0.80)
ROC fold 4 Logistic Regression (AUC = 0.79)
ROC fold 5 Logistic Regression (AUC = 0.80)
Mean ROC Logistic Regression (AUC = 0.79)
ROC fold 1 AdaBoost (AUC = 0.92)
ROC fold 2 AdaBoost (AUC = 0.92)
ROC fold 3 AdaBoost (AUC = 0.92)
ROC fold 4 AdaBoost (AUC = 0.92)
ROC fold 5 AdaBoost (AUC = 0.92)
Mean ROC AdaBoost (AUC = 0.92)
ROC fold 1 MLP Classifier (AUC = 0.87)
ROC fold 2 MLP Classifier (AUC = 0.88)
ROC fold 3 MLP Classifier (AUC = 0.88)
ROC fold 4 MLP Classifier (AUC = 0.88)
ROC fold 5 MLP Classifier (AUC = 0.88)
Mean ROC MLP Classifier (AUC = 0.88)

.11A


4876
1297
1109
4718
Predicted label
1
0
True label
1
0
4500
4000
3500
3000
2500
2000
1500

11B



<u>The work division</u>

Part 1- Exploration- Ofri, Visualization-Tomer.

Part 2- Together.

Part 3- KNN& Logistic Regression – Ofri, MLP& AdaBoost- Tomer.

Part 4-6- Together.

Report- corresponding to the notebook work division.