Leeds School of Business Spring 2021

Microsoft Malware Prediction

By: Dria Fabrizio, Renee Gagne, Sam Al Qarzi

Table of Contents

Executive Summary

Malware, defined as software that is intentionally designed to harm various technology mediums, can exist in many forms including viruses, worms, and ransomware[1]. It is critical that businesses establish a foundational awareness of the types of malware, as well as the consequences and machine attributes associated with the risk of being overtaken by malicious software. Should a business fall victim to malware, the company can be subject to a variety of negative outcomes that can harm data security, data integrity, public perception, and company functions. With regards to company functions, ransomware can cause additional problems when a company is faced with the dilemma of paying the ransom. By paying the hackers, a company would lose a significant amount of money, which some companies may not be able to afford, but their data would be restored quickly. By not paying the ransom, a company would lose a significant amount of time restoring their data using their own backups. Either decision ultimately puts the company at risk of having sensitive data leaked or sold, and comprises the company's reputation.

Objective: The objective of this project is to predict the probability of Windows machines falling victim to infection by malware. The scope of this project will be valuable to companies using Microsoft machines and Microsoft itself by gaining insights as to what attributes increase or mitigate risk of malware infection.

Project Strategy: The overall structure of this project follows the CRISP-DM methodology, which is a sequential approach of data and business understanding, data preparation, modelling, evaluation, and deployment. The most important aspect of developing an understanding of both data and business is the risk associated with malware being present on a machine. Thus, the subsequent steps of analysis are motivated by building a model to predict malware presence based on machine attributes. The data preparation phase of the project entailed exploratory analysis, scaling down the dataset, variable matching between the train and test sets, and data type conversion. Once the data was cleansed and condensed, a random forest classifier model was constructed via a pipeline to train and test the model to produce key evaluation metrics.

Implications and Recommendations: Our best model, the random forest, was able to correctly predict the presence of malware 62.88% of the time, which we believe will be beneficial to businesses for protecting them against malware attacks. That being said, we recommend additional preventative measures being taken, such as having another detection program in place, and putting employees through digital security training so that companies do not have to rely as heavily on these prediction algorithms.

[1] Source: https://www.csoonline.com/article/2615925/security-your-quick-guide-to-malware-types.html

Business Understanding

Background: Microsoft is an international technology company known mainly for its software, as well as the devices it produces. Among machines around the world, Windows is the most installed operating system, present on approximately 77-87% of all machines[2]. Given the scope and popularity of Microsoft Windows, it is imperative that users are able to protect themselves from the harm of malware. For further context, it was found that in 2019, 71% of businesses experienced the spread of malware among employees' machines[3]. Though records indicate that this percentage is on the decline, it is clear that malware remains a prominent issue for copious businesses and users alike.

Data Goals: By creating a model to predict whether a machine contains malware, the ultimate goal is that the model performs such that businesses utilizing Windows machines can adopt insights from the model to reduce issues regarding infected machines.

Measuring Project Success: Accuracy and precision will be used as the guiding metrics of evaluating the model's performance, and thus, a baseline of understanding the model's implications for businesses using Windows machines. Additional metrics of evaluation will be sensitivity, specificity, and false positive rate seeing as accuracy and precision only give us a partial understanding of the output of the model.

[2] Source: https://en.wikipedia.org/wiki/Usage_share_of_operating_systems
[3] Source: https://www.comparitech.com/antivirus/malware-statistics-facts/

Data Understanding

Dataset Overview: The data used for this project originates from the dataset Microsoft Malware Prediction on Kaggle[4]. The dataset is 7.89 gigabytes in size, and consists of both train and test csv files. Each file amounts to approximately seven million records which resulted in some difficulty managing both datasets in Google Collaboratory, more commonly known as Google Colab. Therefore, upon reading the data in using Python's pandas library, we decided to set the number of rows to 1 million for each dataset. As for data features, the train set has 83 features (including the target variable 'HasDetections') of various data types (e.g categorical, floats, and integers). Similarly, the test set contains 82 columns, with the same variables except for the target variable 'HasDetections'. Most of the numerical columns actually represent machine identifiers as well as software and hardware version numbers.

Data Exploration: The process of data exploration involved an investigation of unique value counts, null value counts, and data visualizations. With regards to unique values, it was found that identifier-related variables contained comparatively higher proportions of unique values. Null values were found dispersed throughout the dataset, with multiple columns containing over 99% of the entries being null values. This was an important insight for the data cleansing process.

Distribution Visualizations: Basic data visualizations were created within Google Colab to gain an understanding of machine attributes (*See Appendix Figures 1-7*). Specifically, these visualizations investigated the distributions of machine attributes associated with malware presence, machine attributes, and location. Looking at the distribution of the target variable, 'HasDetections', it was found that approximately half of the Windows machines are detected as having malware present. The distribution of 'Census_OSInstallTypeName' revealed that the majority of users in both the training and test sets have either *IBS Clean* or *UUPUpgrade.* Both the binary variables 'hasFirewall' and 'isProtected' showed that the vast majority of users have an active firewall and an up to date anti-virus program. The distributions of country code revealed that documented machines are spread throughout the globe, without an apparent pattern.

Null Visualizations: To visually investigate the null values, heatmaps were used from the package *missingno.* The heatmaps reveal that the correlations between null values, for both the train and test sets, follow the same pattern overall. There appears to be strong positive correlations among variables correlated to themselves on the heatmap.

[4] Source: https://www.kaggle.com/c/microsoft-malware-prediction/data?select=test.csv

Data Preparation

Removing Variables with Many Unique Values: Variables with notably large counts of unique values, approximately 30,000 and greater were removed from the dataset. Specifically, the columns 'Census_FirmwareVersionIdentifier', 'MachineIdentifier', 'Census_OEMModelIdentifer', 'Census_SystemVolumeTotalCapacity', and 'cityIdentifier' were removed from the dataset. After this step, the train and test sets had 78 and 77 columns, respectively.

Removing Variables with Only One Unique Value: In order to remove redundant records in the dataset, a for loop was used to iterate through both the train and test datasets to remove columns with only one unique value. After this step, the train and test sets had 76 and 75 columns, respectively.

Removing Variables with More Than 5,000 Null Records: To further reduce the number of variables included in the datasets, a for loop was used to remove columns from the datasets that contained more than 5,000 null values. After this step, the train and test sets had 53 and 49 columns, respectively.

Duplicate Entries: It was found that there were not any duplicate records.

Equalizing Variable Counts: Because the data cleansing process above did not result in the train and test sets having the same number of variables, columns in the test set that were no longer present in the train set were removed.

Data Type Conversion: Numeric columns of types *Float64* and *Float32* were converted to object types. This conversion was specific to variables containing numeric identifiers that are categorical.

Modelling

Logistic Regression

Logistic regression models are known for their simplicity, timeliness of running, and easy-to-interpret results. Logistic regression models use one or more independent variables to predict one classification outcome.

Model Building: To further prepare the data for a logistic regression model, we converted all predictor variables to type 'object' and the target variable, 'HasDetections' to type 'integer' whose presented an issue for the model to read the binary labels with an 'object' type. We further investigated the count of unique values, and removed the variables 'Census_ AVProductStatesIdentifier', 'Census_AvSigVersion', 'Census_PowerPlatformRoleName', 'Census_MDC2FormFactor', and 'AppVersion' which were removed due to containing approximately 7000 different categories or more. After removal, the average number of unique values across all included variables was approximately 250. Afterward, the data was divided with an 80 to 20 split percentage for train and test; we set random state to 42 in this split. We then built a pipeline that calls on its first step of preprocessing the data with '*one hot encoder'*; for its next and last step is fitting a logistic regression model with all parameters set to default. After fitting the model, we validated the model with the test data which produced an accuracy score of 0.61. To evaluate model prediction and compare it to actual values, we used a confusion matrix and visualized the outputs (see figure 1). See the model section under 'Evaluation' for more in-depth calculations of the matrix rates.

Random Forest

Random forest models are known for both their simplicity and capability of running numerical and categorical data. Specifically, the algorithm is a supervised machine learning technique that builds a collection of decision trees and combines them for stronger predictions[5].

Data Reduction Note: While our logistic regression model ran smoothly using our full-sized train and test sets, we found upon building our initial random forest model that the Google Colab program was not operating correctly. Thus, the modeling data was reduced to 10 percent of the cleaned data and which was then split into 80 percent train and 20 percent test.

Model Building: Though the logistic regression model ran efficiently, we wanted to explore an alternative model to understand which would best predict whether a Windows Machine would be

[5] Source: https://builtin.com/data-science/random-forest-algorithm

detected as having malware. To build the random forest model, the target variable 'HasDetections' was isolated from the training dataset in order to create 'x_train' and 'y_train' variables. The 'x_train' variable was further broken down into categorical and numeric partitions. Like the logistic regression model, an 80/20 train test split was used, with a random state of 42. The numeric subset of training variables were scaled using *StandardScaler*, and the categorical variables were transformed via the *OneHotEncoder* from Python's sklearn library. We then built a pipeline using '*preprocessor*', the column transformer, and '*mod',* the Random Forest Classifier, as inputs. Note that this model's run time was comparatively much longer than that of the logistic regression model. The model was then fit using the data and model metrics were calculated (*See Table 1*).

Evaluation

Logistic Regression:

Our logistic regression model did not perform particularly well, however, this was expected because logistic regression models are less sophisticated than other methods of prediction, such as our random forest model. Regardless, building this initial regression model provided us with a deeper understanding of our data before moving on to our more complex random forest model. Below is the confusion matrix for our model (*See Figure 1*). From the confusion matrix, we determined that this model was 61.05% accurate, meaning that 61.05% of the predictions were correct. The sensitivity, or the true positive rate, was 65.14%. It is also worth noting that this model has a 43.05% false positive rate, which has great potential to lead to problems should the model undergo deployment. This metric suggests that a noteworthy proportion of machines predicted as having malware, in reality, do not.

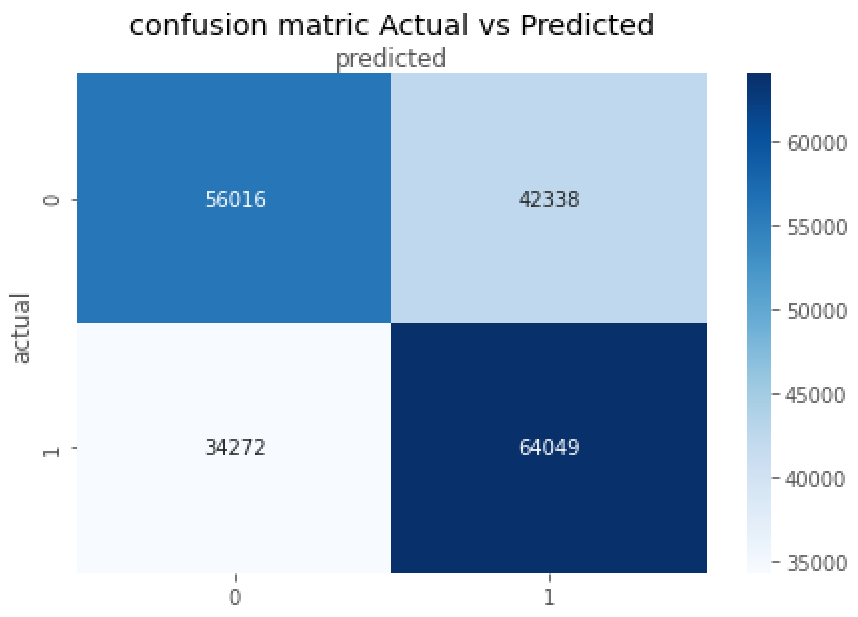

Figure 1: Confusion Matrix for Logistic Regression Model

Random Forest:

Compared to the performance of the logistic regression model, the random forest model performed as a slightly better predictor of whether machines are detected as having malware (*See Table 1*). However, the random forest model did not perform as well as we were expecting. We believe that this was due to our limited computing power hindering our ability to refine our model and run it with more data. Below is the confusion matrix for this model (*See Figure 2*). From this, we determined that the model was 62.88% accurate, which is a small improvement over the logistic regression model. However, the sensitivity of this model was 61.41%, which, while relatively small, is noticeably worse than the logistic regression model. Additionally, the false positive rate of this model was significantly improved in comparison to the logistic regression model with a value of 35.63%.

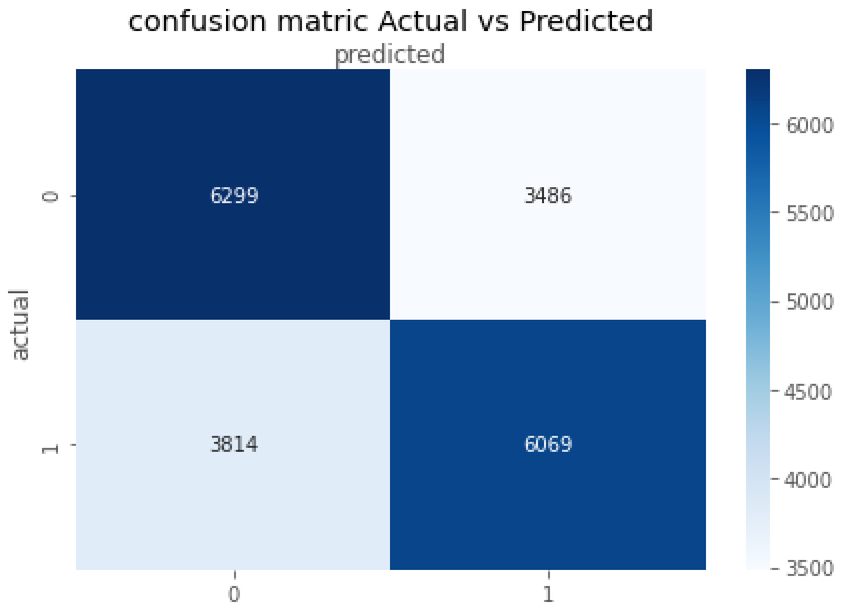

Figure 2: Confusion Matrix for Random Forest Model

Summary of Evaluation Metrics:

As mentioned above, the random forest model performed better in all metrics except for sensitivity. With this information and the lower false positive rate of the random forest model, this implies that the model is better at correctly identifying machines without malware than machines with malware. Because of this, we had to discuss the pros and cons of the random forest model before determining for sure that this was our best model, but we ultimately decided that the improvement in the other metrics outweighed our concerns regarding sensitivity. Below is a table containing all of the evaluation metrics calculated from each model's respective confusion matrix (*see table 1*).

	Accuracy	Precision	Sensitivity	Specificity	False Positive Rate
Logistic Regression	61.05%	60.20%	65.14%	56.95%	43.05%
Random Forest	62.88%	63.52%	61.41%	64.37%	35.63%

Table 1: Model metrics

Deployment

Recommendations:
We recognize that our models are by no means perfect for detection of malware, but we do believe that implementation of our random forest model may be able to significantly decrease a company's susceptibility to malware attacks. We also recognize, as with the case of the classic fairytale, *The Boy Who Cried Wolf*, that the more false positives there are in a system, the more likely people are to not take real detections seriously, which went into our decision for declaring the random forest model to be our best model.

That being said, we have additional recommendations for how a company can more effectively protect itself against malware. We believe that implementing a secondary detection program would benefit a company by potentially catching malware that slipped through the gaps of our model. Additionally, since no detection program is truly perfect, we recommend other preventative measures that can be taken by individuals working for the company. Specifically, we believe putting employees through digital security training would be highly beneficial. If employees are aware of how to identify suspicious emails or links by checking the email address or URL, respectively, the company will not have to worry about whether or not the detection program will work. Similarly, the company will have to rely on the detection program less if employees are aware of common sense security measures such as not doing personal stuff on machines for work, installing security updates rolled out by Microsoft, and choosing strong passwords.

Finally, it is important to acknowledge that as technology develops, methods of hacking, malware programs, and weaknesses in machines evolve as well. Thus, companies should adopt a philosophy of *risk mitigation* as opposed to the impossible objective of *risk elimination*. This implies that malware detection and prevention is ultimately a process that grows alongside technology development.

Business Implications:
Specific to the random forest model, the prevailing advantage of its deployment is that the model is capable of detecting malware on Windows machines based upon a plethora of machine attributes. However, this is balanced by the drawback of the model not having 100% accuracy, which implies that the model cannot predict all machines correctly. Despite this difference, an accuracy of 62.88% suggests that the model predicts more machines correctly than incorrectly. Therefore, deployment of the model can function as a foundational step towards further risk mitigation, as well as open up opportunities for additional research and model improvement.

Financially, use of the model is beneficial to companies in the sense that the likelihood of ransomware conundrums are reduced. Similarly, companies can improve their proactive rather than reactive protocols with regards to malware, which has room for company savings.

Alternatively, detection of malware among company machines may incentivize companies to budget for a creation or additions made to a digital security team. Liability wise, it is critical that companies acknowledge that this model is not a perfect predictor of malware on Windows machines. Therefore, businesses should not treat all model predictions as fact, but rather, pursue further investigations of machines to determine the condition of machines.

Future Directions:

For future research, we recommend that researchers further investigate feature importances. We foresee this as valuable for data cleansing and reduction of the number of variables prior to building and running a model. However, it is important to note that given the large size of the data, this can be a timely process for computers to run. For this project specifically, the data size was determined too large to run a feature importance investigation. Furthermore, we uphold that the creation of additional models would allow for a deeper analysis of which model performs best with the data. Lastly, we recommend that, for future analysis, an alternative program to Google Colab be used. The program did not function efficiently for the data, and often, crashed or did not save changes. With regards to the data, it would be interesting to compare malware presence on machines with different operating systems.

Appendix

Figure 1: Frequency of 'HasDetections' variable in the training set. Note that a '0' means no malware detected, and '1' means malware was detected.

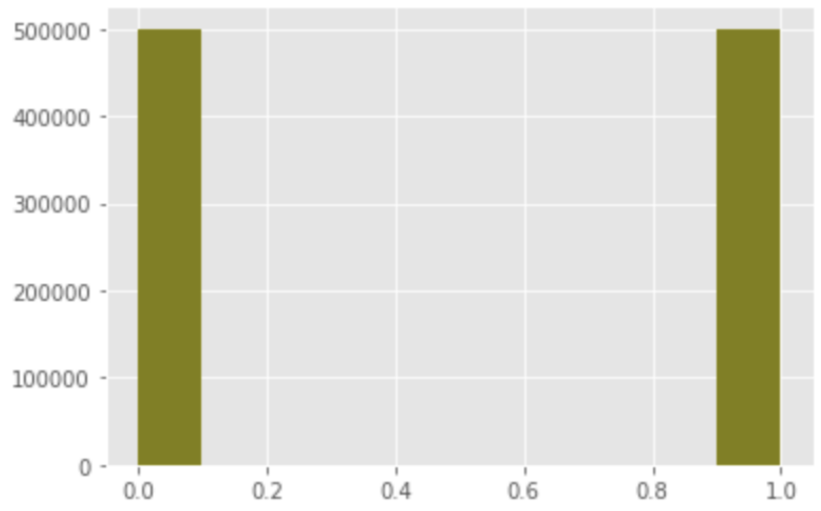

Figure 2: Frequency counts of 'Census_OSInstallTypeName' of train (left) and test (right) data

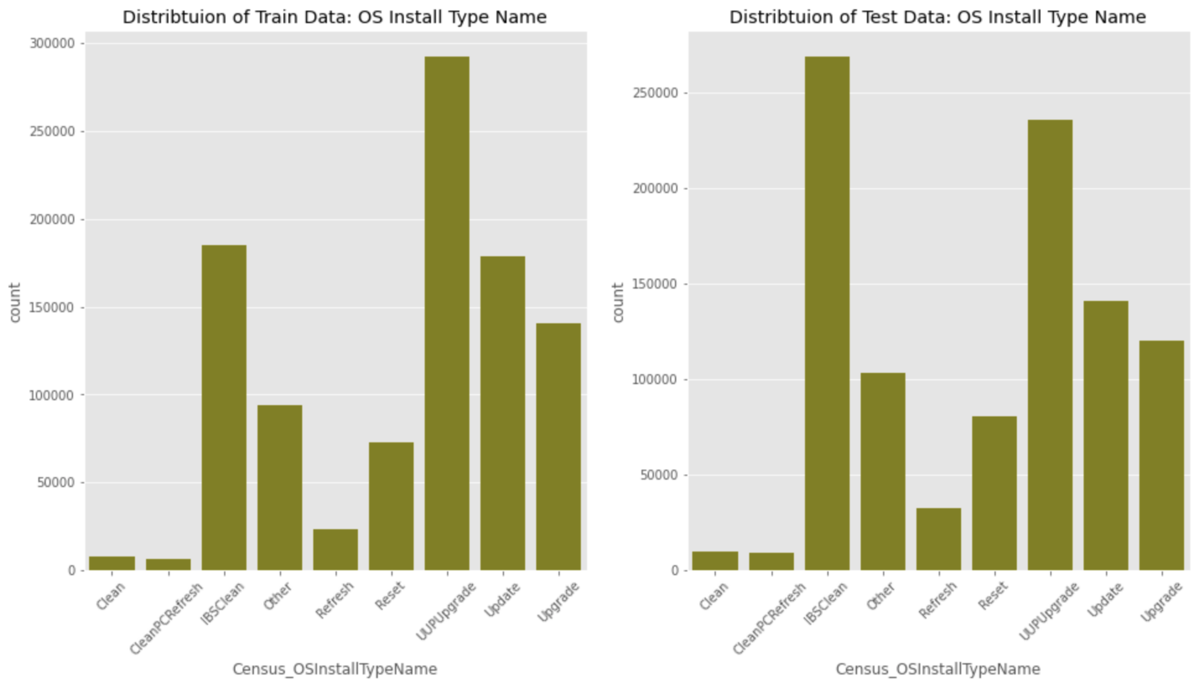

Figure 3: Frequency counts for the variable 'IsProtected' in train (left) and test (right) data sets

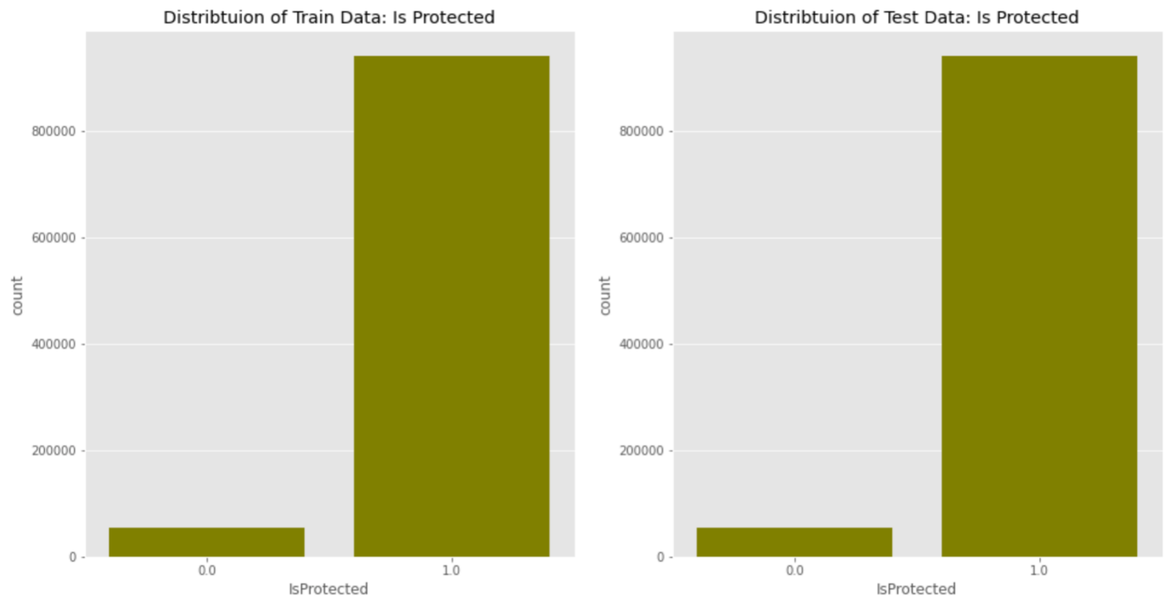

Figure 4: Distributions of 'Firewall' in train (left) and test (right) data sets. Note that a '0' suggests no active firewall.

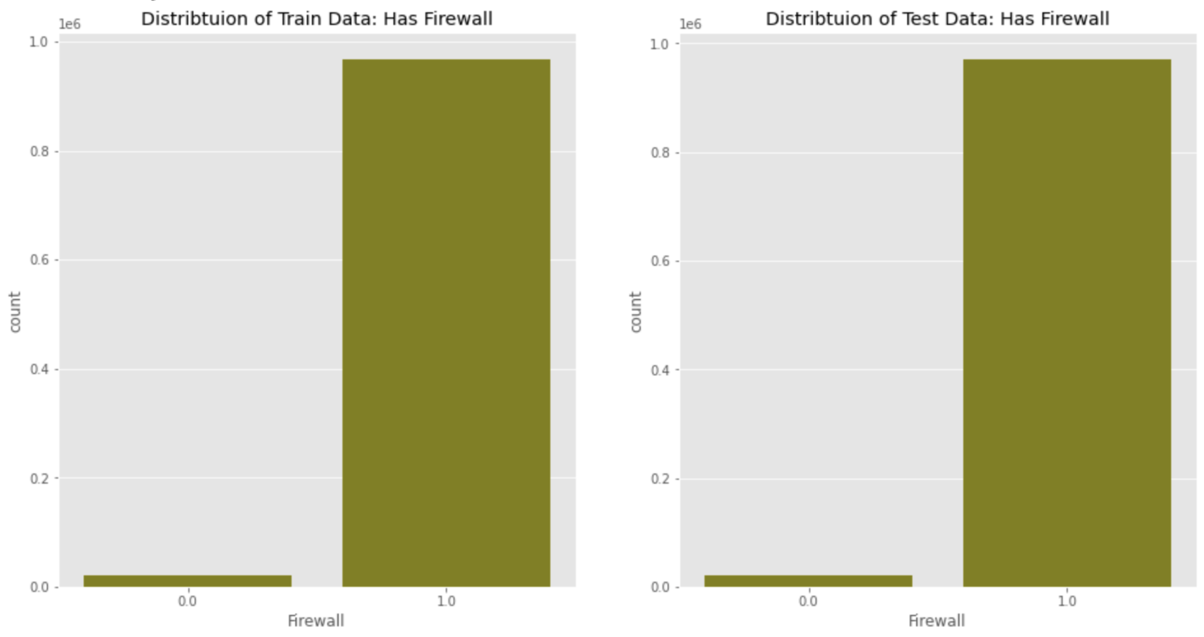

Figure 5: Distributions of country codes in train (left) and test (right) data sets

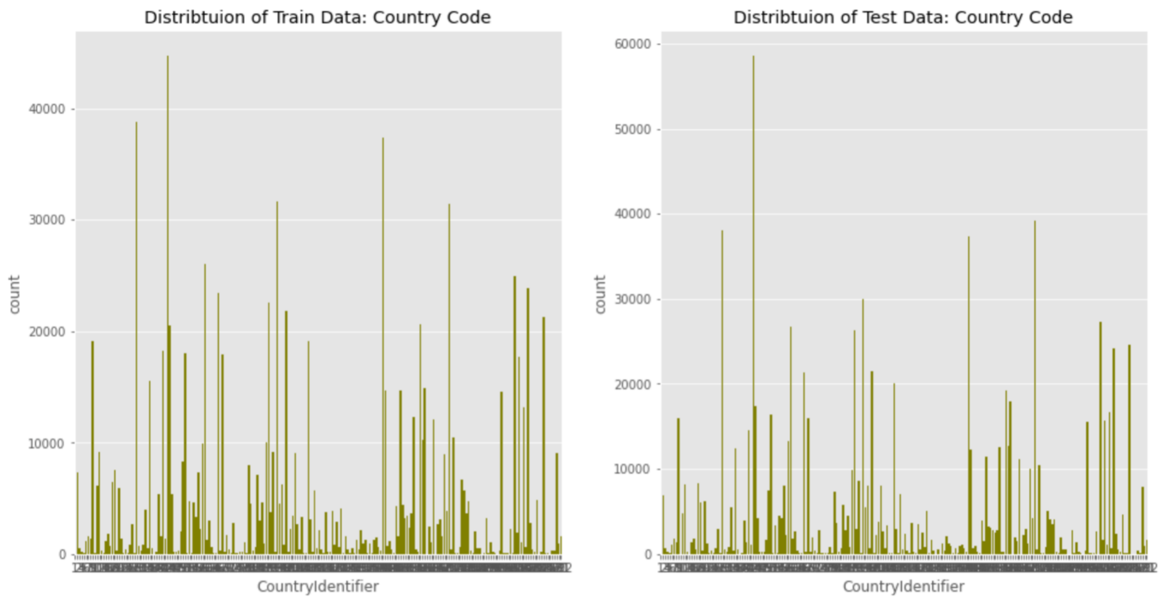

Figure 6: Null correlation plot for the training dataset

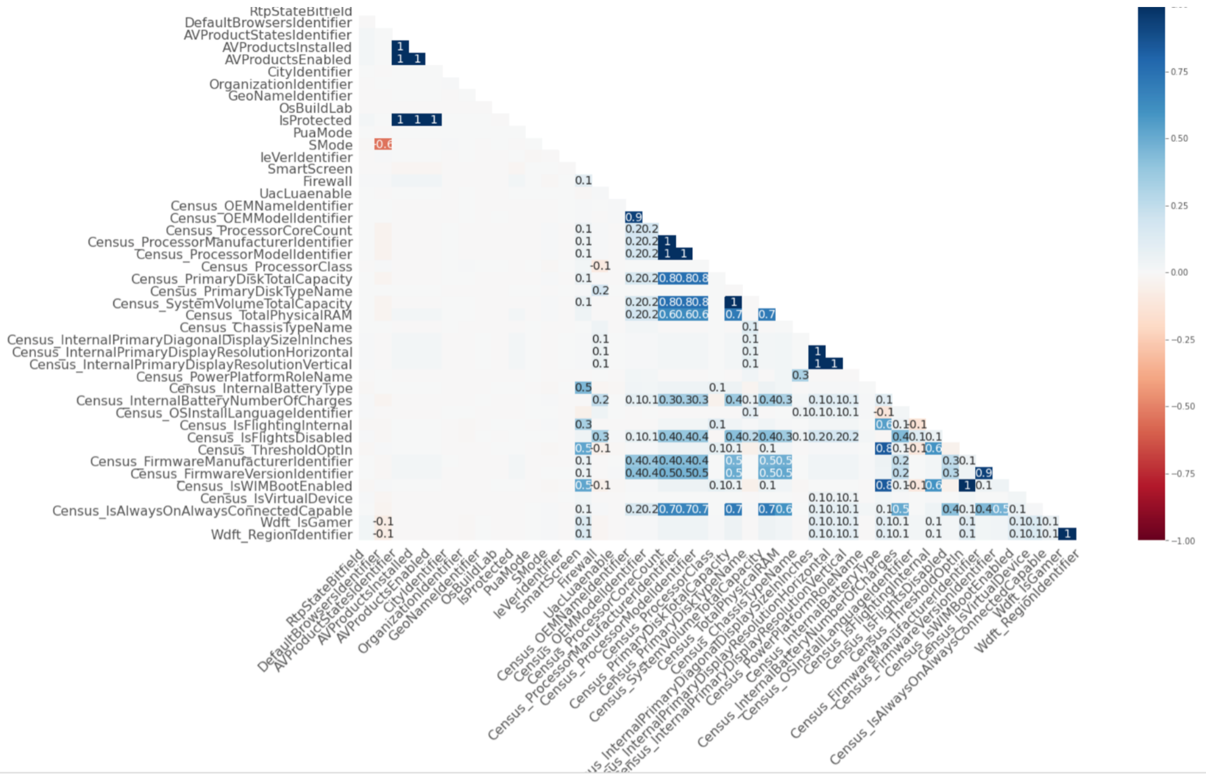

Figure 7: Null correlation plot for the test dataset