Malware Prediction

Maseerah Khatoon
Northeastern University
NUID: 002778147
khatoon.m@northeastern.edu

Christina Raymond
Northeastern University
NUID: 002746856
raymond.christ@northeastern.edu

Habeebuddin Mir
Northeastern University
NUID: 002713929
mir.h@northeastern.edu

Abstract

The malware industry continues to be a well-organized, well-funded market dedicated to evading traditional security measures. Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways. This seems like an exciting yet challenging issue that can be solved using machine learning models. We came across a Kaggle competition with data containing the machine properties and infections generated by combining heartbeat and threat reports collected by Windows Defender. So we built a machine learning model to predict the Windows machine's probability of getting infected by various families of malware.

1 Background

Malware is commonly used by cybercriminals as primary attack vectors, and malware proliferation is thus a significant challenge for security professionals to adapt and develop a matching defense mechanism. The prediction of malware attacks remains one of the most challenging problems for industry and academia. One of the big problems facing anti-malware applications today is the large volumes of data that need to be analyzed for possible malicious intent

1.1 Dataset

For this problem, we collected data i.e train.csv and test.csv directly from Kaggle where this competition was hosted.

Train.csv:- This file contains 8921483 entries where each entry corresponds to a machine that is uniquely identified by a MachineIdentifier and HasDetections is the ground truth and indicates that Malware was detected on the machine. And contains 83 columns including MachineIdentifier and HasDetections using which model is to be trained.

Test.csv:- This file contains 7853253 entries. And contains 82 columns including MachineIdentifier except for HasDetections.

1.2 Related Work

In (LIN, 2019), the authors described a method that used Na¨ıve Transfer Learning approach on Kaggle's Microsoft Malware Prediction dataset. The model used 20 of the most important features and achieved an accuracy of 63.7

2 Method

2.1 Data Cleaning and EDA

As part of data preprocessing, we did a null value analysis. We got 5 features containing null values greater than 70 percent, so we removed them. After this, we filled in missing values and cleaned categories of categorical features in order to reproducing cardinalities of categorical features. High cardinality is when a categorical feature contains highly unique values. Finally, we removed outliers

from the training dataset based on the Z-score method by selecting only the data points whose Z-score value is less than 3 because Z-score greater than 3 is considered an outlier. Z- score describes any data point by finding its relationship with the standard deviation and mean of the group of data points. And Z-score is calculated for features that contain continuous values.

The correlation of features with the target variable helps us to find out which feature will influence the model for predicting the ground truth or target variable in this case 'HasDetections' is the target variable. By default, the correlation technique used here was Pearson. From the results, AVProductStatesIdentifier, and CensusTotalPhysicalRAM are very highly correlated with the target variable in terms of the positive strength correlation technique. This technique can handle only linear relationships amongst the features and works very well only for numerical features, so we used a new correlation analyzer library called phik which outperforms the Pearson correlation coefficient technique, as it consistently works well for categorical, ordinal, and interval variables, this library also captures non-linear relationships amongst the variables. We found our dataset to be balanced too.

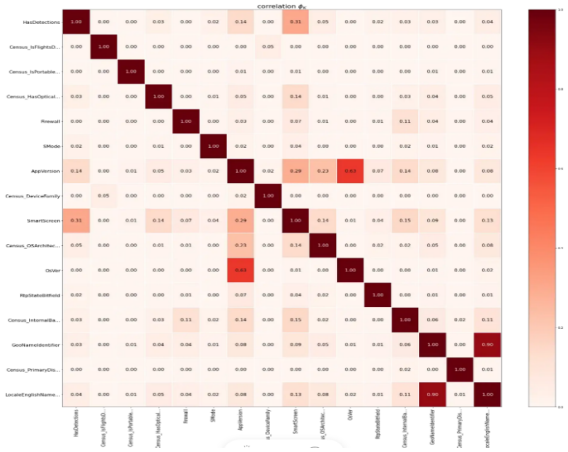

Figure 1: Correlation matrix using phi k library

2.2 Encoding Categorical Features

The dataset contains categorical features but the model performs well when numerical values are fed. We used frequency encoding for some features because it's not feasible to use the LabelEncoding technique when cardinalities of categories in categorical features are very high and for other features, we used the LabelEncoding technique.

2.3 Data Modelling

Given the difficulties of categorical features with high cardinality, and limited hardware resources (CPU, RAM, GPU). The best ML model capable of handling missing values, handling categorical features with minimal memory utilization produces a model quickly, and has high performance is a tree-based ensemble model, namely LightGBM, which has been open-sourced by Microsoft in 2017 and is a tree-based gradient boosting framework. For tuning the hyperparameters of lightgbm, we used the optuna library for selecting the best parameters which give good auc scores.

Next, we did feature selection based on the number of splits. We selected the top features which are used for splitting trees greater than or equal to 100 times. Aproducingture selection, we followed the same procedure of tuning for best parameters using optuna, trained the classifier, and got the train AUC score of 0.75 and validation AUC score of 0.73. After creating a data pipeline for the training dataset to evaluate metrics you can refer to code in the final.ipynb file achieved a final AUC score of 0.749.

2.4 Model Evaluation

For this problem, model performance was evaluated based on the area under the ROC score between the predicted probability and true label. The area under the ROC tells us how good a model is in distinguishing between true positives and false positives. Why AUC? With respect to a time-series

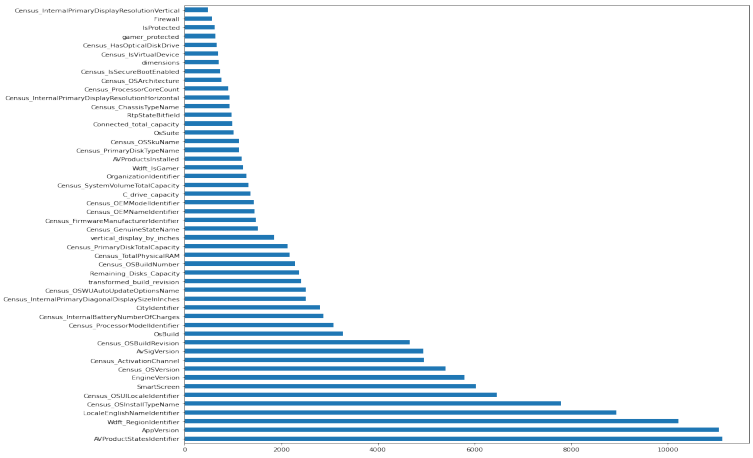

Figure 2: Feature importance

problem such as malware detection rates over time, Microsoft and security researchers may be able to see specifically which computers are staying infected over time, and which computers' malware detection status has changed over time. They may very well have a decreasing detection rate over time, however, in order to understand the effectiveness of their security improvements, they also need to understand if the same exact computers that were previously detected with malware are also being cleaned with regular Anti-virus definition updates. Consider the case where detection probabilities are decreasing with respect to time. It would be prudent to have a metric that can measure prediction ordering since time series is a critical factor.

We also used Stratified K-Fold Cross-validation to validate the evaluation metric by dividing the data into folds and then each fold is used as a testing set. For this problem statement, we achieved a final AUC Score of 0.749.

3 Experiment

3.1 Feature Engineering

As a part of feature engineering, we created 12 new features from the given set of features after preprocessing using our domain knowledge about which combinations of features may help in predicting machines getting infected with malware.

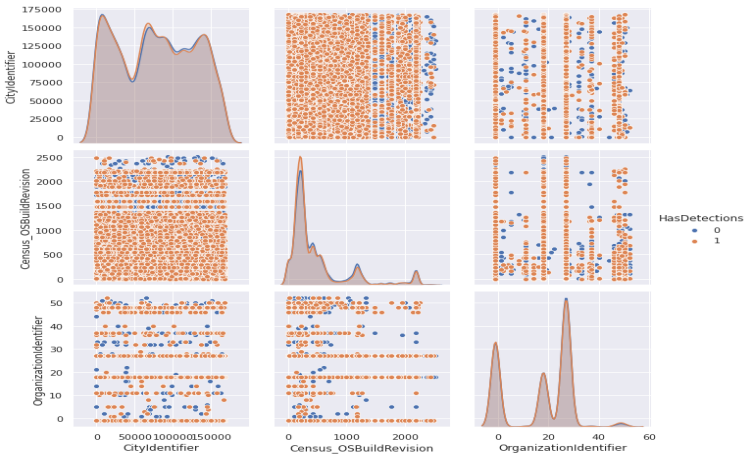

Figure 3: Feature Analysis plot

For example, in the given dataset we had features like wdftisgamer and firewall, so we created new features gamerwithfirewall, using these two features because Gaming machines are more prone to

getting infected with malware and when the firewall is not turned on then there are higher chances of getting infected. So somehow this feature may help in prediction. Similarly, we created new feature Dimensions using CensusInternalPrimaryDisplayResolutionHorizontal and CensusInternalPrimaryDisplayResolutionVertical.

We also created transformedbuildrevision by applying log transformation on the CensusOSBuildRevision feature because we saw in EDA that CensusOSBuildRevision follows right-skewed distribution so applying log transformation to make the data follow a normal distribution and somehow it can be useful for data modeling.

3.2 Results

After applying stratified 3- fold cross-validation, the model achieved a training AUC score of 0.724 and a validation AUC score of 0.723 by taking the mean AUC score of all the folds.

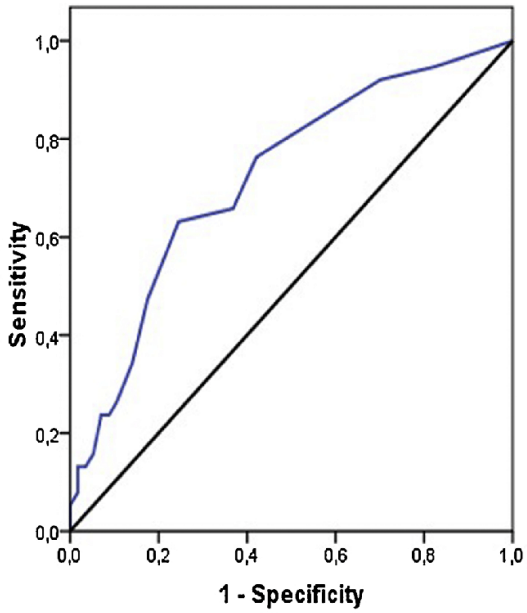

Figure 4: AUC curve for test data

After evaluating metrics, using our lightgbm model we obtained final predicted probabilities by passing test data to the trained lightgbm model.We also observed that the feature AVProductStatesIdentifier was very much useful in predicting the target variable. Followed by AppVersion, WdftRegionIdentifier, and transformedbuildevision which we engineered.

	MachineIdentifier	Probabilities
47136	a5ec1e6ea1b8e74606f5e6a0389bea2c	0.679396
74166	8950c67ce6dde8e07edf74c599ec6f2c	0.640029
24990	c708f601c99efdc1a40609d4f2d5aad1	0.291990
106489	96fe94db4ff0e9b66f05629ea2267295	0.334789
119496	0c55e728894a36f79a480766096e86b8	0.186865

Figure 5: Final prediction

The algorithm took around 24262.4 seconds which is around 6 hours for 6 folds each with 100 rounds each round being 100 iterations.

4 Conclusion

As data science enthusiasts, this was a fantastic opportunity with a substantial amount of data to try and ask questions like: "Why are computers infected?", "What are the biggest contributors?" "How is patching going?" Unfortunately, the competition answers more questions about machine learning rather than malware prevention or patching best practices. We still were able to gain some valuable insights. For example, a gaming desktop is more likely to be infected than when it is not a gaming PC. These data clearly show that someone who has a desktop and is not a gamer should not be a hot target of an attack. These data gave us a clue that having a desktop with Windows 10 made us a more susceptible candidate to an attack.

The next major lesson we learned was that there is always a tradeoff when we choose different machine-learning algorithms in order to carry out our training and testing. The main challenges or the so-called concerns are memory usage, the time it takes to run, and the accuracy that it gives. The results of this project also witness that. This is because when the algorithm tries to prioritize memory, the run time has become greater.

4.1 Future Work

In the future, we would love to dig deeper into malware predictions using other datasets. In lightgbm, there is a parameter of the device where we used the default i.e CPU for training, this can be extended to GPU as future work to reduce the training time and if we have sufficient computational power. Next, for encoding categorical features, we can explore various different encoding techniques. As a part of future work, one can even try another promising machine learning algorithm that uses gradient boosting on decision trees known as catboost.

References

[1] LIN, C. (2019). Naive Transfer Learning Approaches for Suspicious Event Prediction. In the *2019 IEEE International Conference on Big Data (Big Data)*, pages 5897–5901.

[2] https://www.kaggle.com/c/microsoft-malware-prediction

[3] M. Baaka, R. Koopmana, H. Snoekb, S. Klousa (2019). A new correlation coefficient between categorical, ordinal, and interval variables with Pearson characteristics(https://arxiv.org/pdf/1811.11440.pdf)

[4] https://towardsdatascience.com/how-to-make-your-model-awesome-with-optuna-b56d490368af

[5]https://phik.readthedocs.io/en/latest/

https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e