Marco Riva

Final Report:
Microsoft Malware Detection

Problem Statement

The majority of individuals in our society own at least one computing device, may this be a smartphone, notepad, laptop, computer desktop, etc. In the last 30 years these devices, together with the Internet, have drastically revolutionized social interaction and enabled technological progress in various fields. Especially during the COVID-19 pandemic more and more of us have transformed our usual routine to a remote setting, exclusively. We pay bills, do shopping, order food and groceries, rent cars, uber, lyft, file taxes, take classes, watch movies on streaming service platforms, and, let's not forget, even work from home, with our PC (or smartphone). However, this luxury comes with a great threat. Our computers and the internet know all there is to know about us: credit card number, bank accounts, social security number, emails, where we are and what we like - we post on instagram, don't we? Though it may not be too much of a threat to have some of our data available to reputable companies, businesses and persons, so we can get personalized ads, things are quite different when it comes to sensitive and personal information. And it is no secret that hacking attacks are on the rise. Internet criminals, or hackers, and their vicious attacks are a serious issue which affects the government, organizations, companies, businesses and every single one of us and our daily lives. Even though systematic prevention is performed on electronic computing devices through the use of sophisticated anti-virus software, malware constitutes a serious threat to the privacy of billions of people, particularly because of how dynamically these threats evolve.

In the era of big data it is not too difficult, at least for a tech giant like Microsoft, to collect millions of instances of computers which were infected by malware, together with all the data that characterizes them. To find the correlation between computer features and malware infection is, instead, a more complicated but fascinating task. As Microsoft takes their customer's security and privacy very seriously, they have provided an open source malware dataset that can be used to train machine learning algorithms to detect potential infections. In this project, I aim to leverage state-of-the-art machine learning techniques to develop a classification model which can predict the probability of a Windows machine getting infected by various families of malware, based on different properties of that machine. The metric of interest is the area under the curve (AUC) of the receiver operating characteristic (ROC) curve, with the particular goal of increasing the performance of the classifier model by 20-25% compared to random chance. The ROC AUC appears a valid metric for this case, as we also are interested in minimizing the false negative rate and maximizing the true positive rate.

Some of the constraints of this project include the sampling methodology chosen by Microsoft to generate the training and test dataset, since user privacy must be respected at all costs. Furthermore, being the malware problem intrinsically a time series issue, due to the fact that computer operative systems and installed software are updated constantly, machines go online and offline, new computers are introduced and older are removed, etc., with the aforementioned constraint on privacy, made the sampling technique rather complicated. Even if the datasets have been roughly split by time, a disagreement between cross validation scores and test set scores is likely to arise.

Data Wrangling

The raw dataset contains about 9M rows, each representing a Windows machine, and 83 features. The features were organized in 6 groups:

- Machine Specs and Operative System (OS)
- Microsoft Defender Specs
- AntiVirus & Security Settings
- Browser & Apps
- Geographical Information
- Device Census File Data

The target is a binary variable with 50% positives, i.e. malware detected, and 50% negatives, i.e. no malware present. Several rows contain a significant number of missing values, as shown in Figure 1 and Table 1. Features with more than 40% missing values were dropped whereas imputation will be performed for the remaining features. No duplicates were found in the dataset.

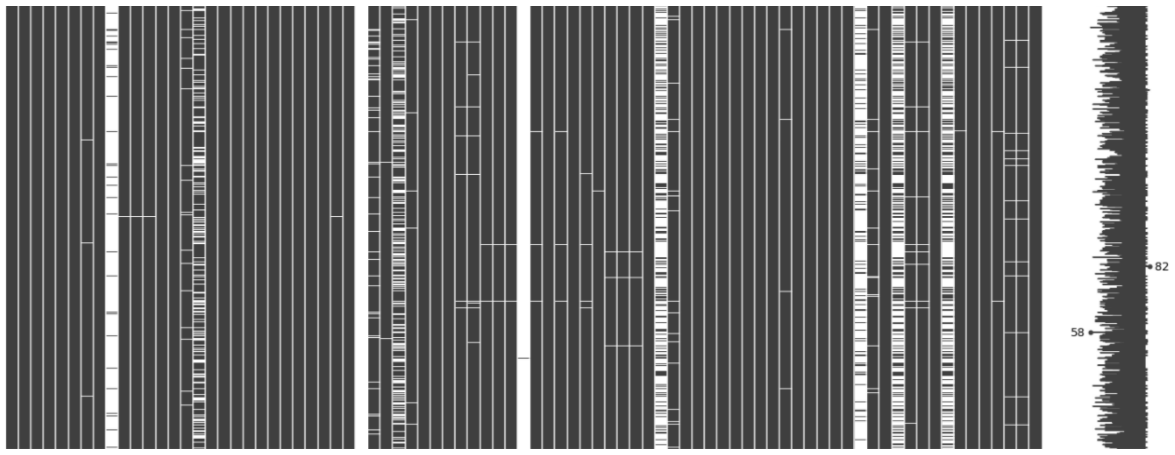

Figure 1: Graphical representation of the dataset: white represents a missing record.

	missing_count	%
PuaMode	8919174	99.97
Census_ProcessorClass	8884852	99.59
DefaultBrowsersIdentifier	8488045	95.14
Census_IsFlightingInternal	7408759	83.04
Census_InternalBatteryType	6338429	71.05
Census_ThresholdOptIn	5667325	63.52
Census_IsWIMBootEnabled	5659703	63.44
SmartScreen	3177011	35.61
OrganizationIdentifier	2751518	30.84
SMode	537759	6.03
CityIdentifier	325409	3.65
Wdft_IsGamer	303451	3.40
Wdft_RegionIdentifier	303451	3.40
Census_InternalBatteryNumberOfCharges	268755	3.01
Census_FirmwareManufacturerIdentifier	183257	2.05

Table: Top 15 features sorted by missing value count and fraction of the total.

Errors in the record and obvious outliers were converted to null values. As an example Figure 2 shows a box plot of the Primary Disk Capacity distribution with two obvious outliers which were converted to null values.

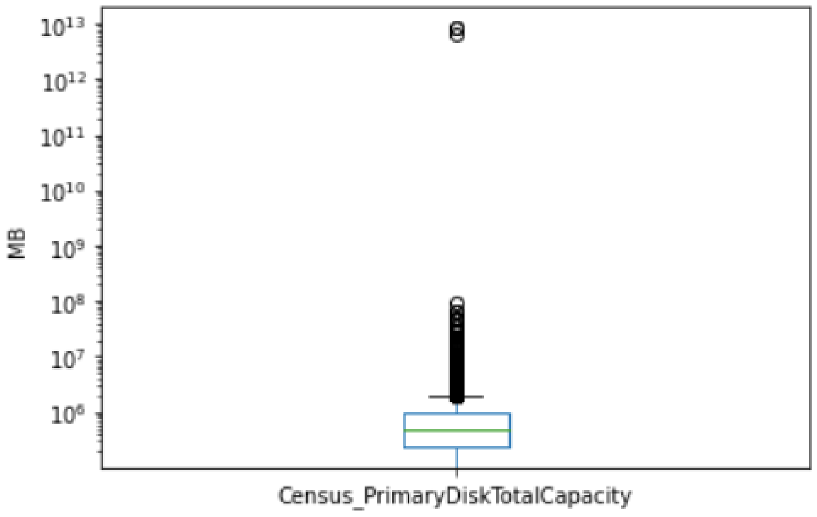

Figure 2: Primary Disk Capacity distribution before outlier removal.

Several categorical features present high cardinality, see Figure 3 showing the levels of categorical variables.

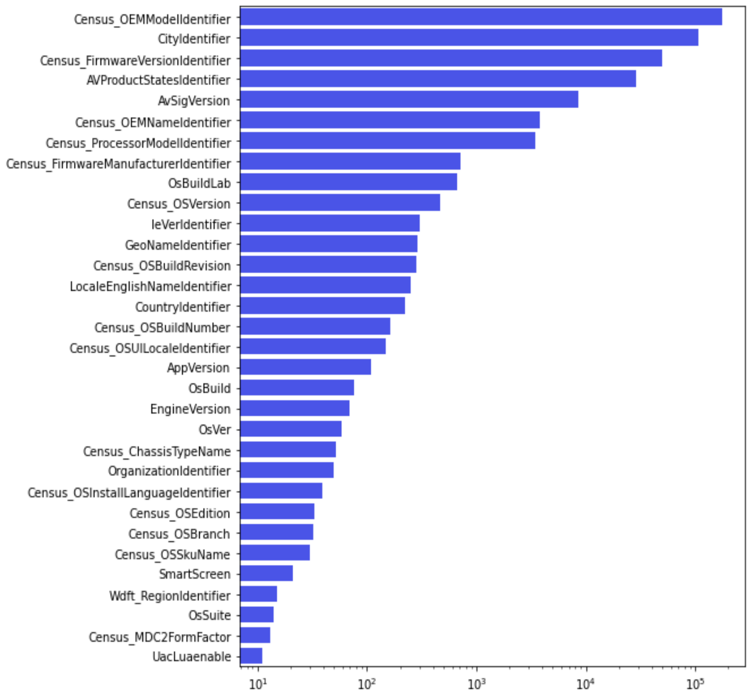

Figure 3: Number of levels in high cardinality features.

Some of these features are encoded numerically; in these cases the data type is converted to “category”. In most cases the cardinality was reduced by correcting inconsistencies and spelling typos, e.g. categories labeled as “OFF”, “Off”, “off”, and by grouping low frequency levels to the generic category “other”, see for example figure 4, and the data type conveniently converted to categorical. This allows me to perform One Hot Encoding later in the data pipeline without exploding the number of columns of the dataset and generating high sparsity. However, some other categories representing IDs showed thousands of levels with equal frequency. In these cases, the features were not modified and a different encoding, e.g. mean (target) encoding, chosen.

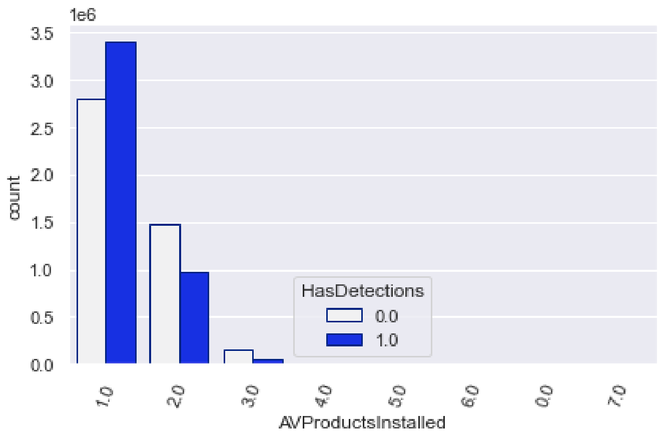

Figure 4: Example of feature with high cardinality with several infrequent levels (only the first 8 levels ordered by count are shown in the picture).

Exploratory Data Analysis

Numerical Data

A heatmap of Spearman's Rank correlation coefficient for numerical features is presented in Figure 5. The most correlated features are system and Disk Capacity are correlated, as one is a partition of the other, and horizontal and vertical resolution, since they together determine the screen resolution. Also, machines with more cores tend to have more RAM available. As seen in the boxplots of Figure 6, there are no significant differences in the distributions of numerical features hued by class, which suggests the features are likely not very predictive of the presence of malware.

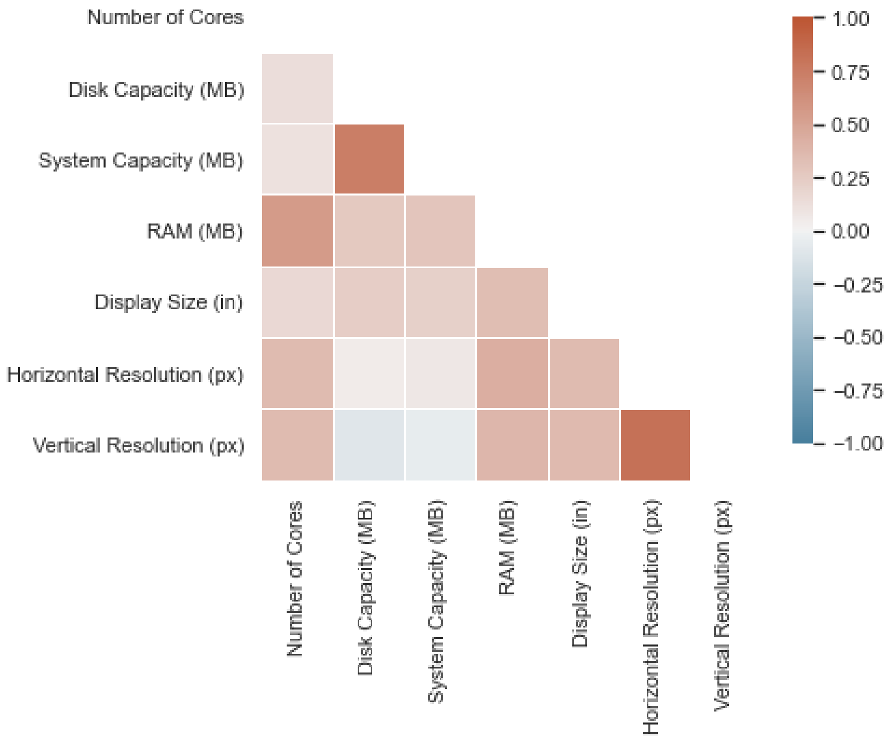

Figure 5: Heatmap of Spearman's Rank correlation coefficient for numerical features.

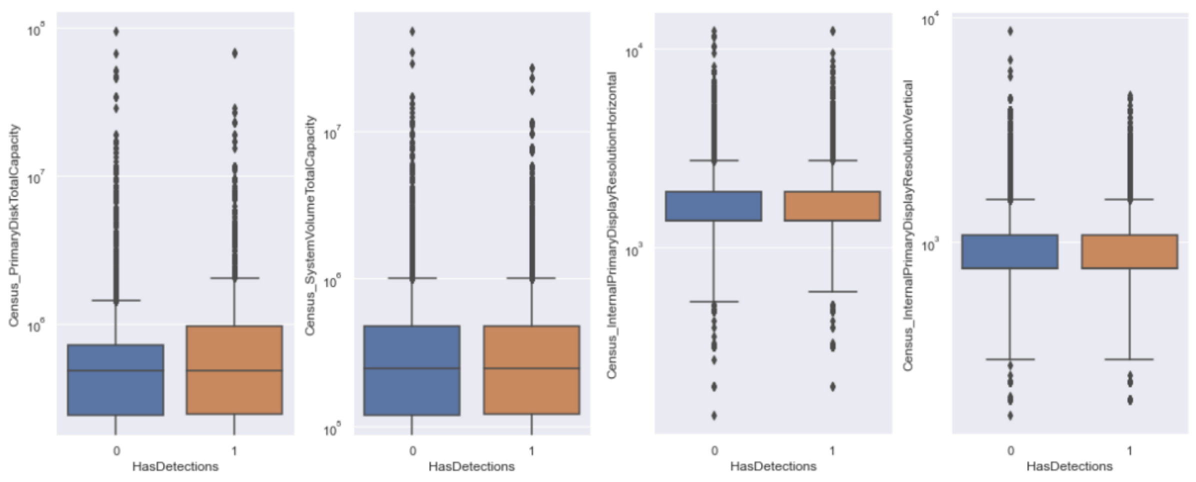

Figure 6: Box plots of numerical features by class.

Categorical Data

Some categorical variables show a significantly different churn rate for different categories. The most interesting findings are reported in Figures 7-10. Finally, Table 3 presents the results of Chi-squared test with test statistics and p-values for categorical features. From the figures and statistics test we notice:

- Smart Screen: when active under “RequireAdmin” malware presence is prevented whereas if “ExistsNotSet” lead to higher vulnerability to virus infection
- Malware rates vary significantly depending on what AntiVirus product is installed: Product State Identifier, Product and Version installed suggest some versions are more vulnerable than others, e.g. State Id 53447, Product1.0 and Version 273

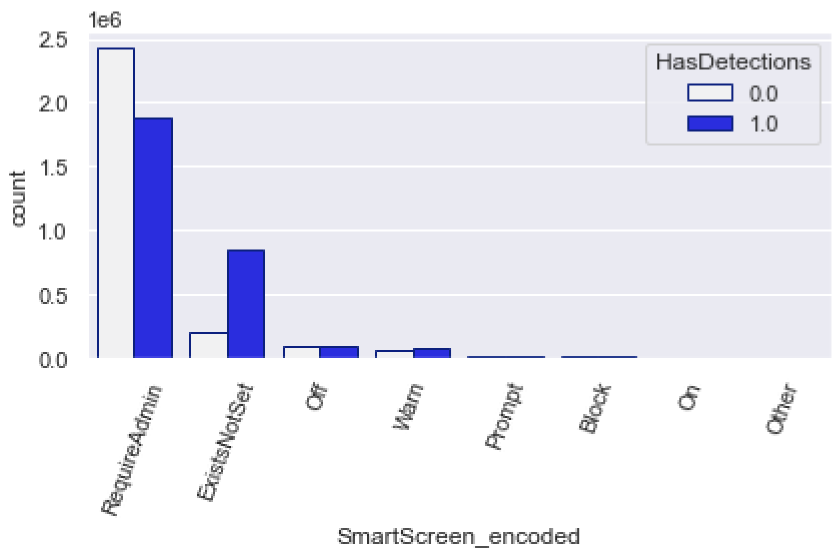

Figure 7: Barplot of categorical variable SmartScreen hued by target class.

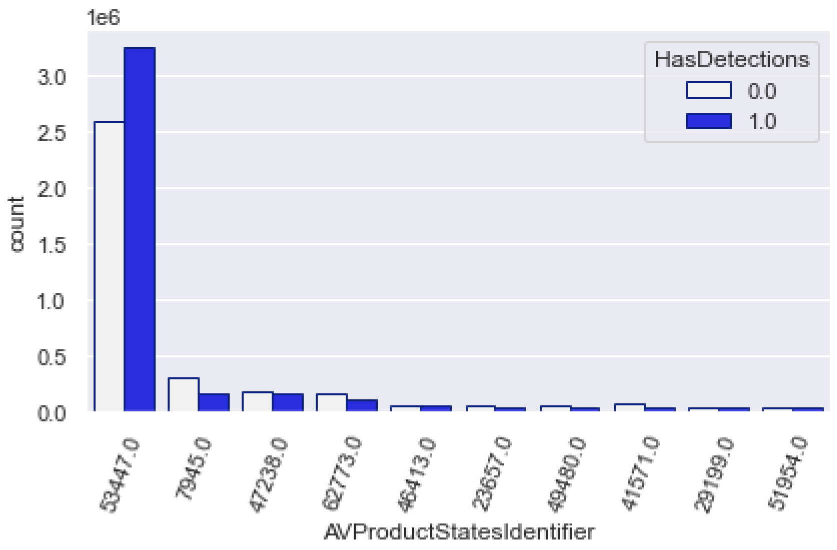

Figure 8: Barplot of categorical variable AntiVirus Product States Identifier hued by target class.

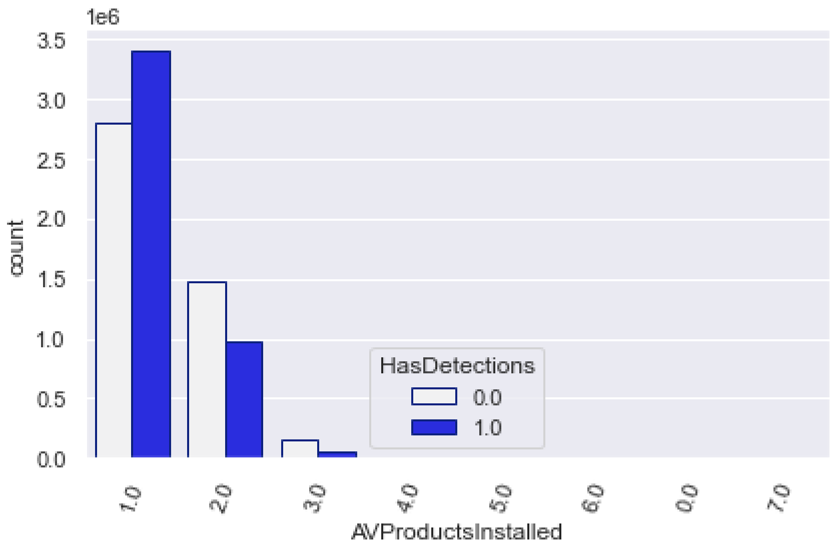

Figure 9: Barplot of categorical variable AntiVirus Product Installed hued by target class.

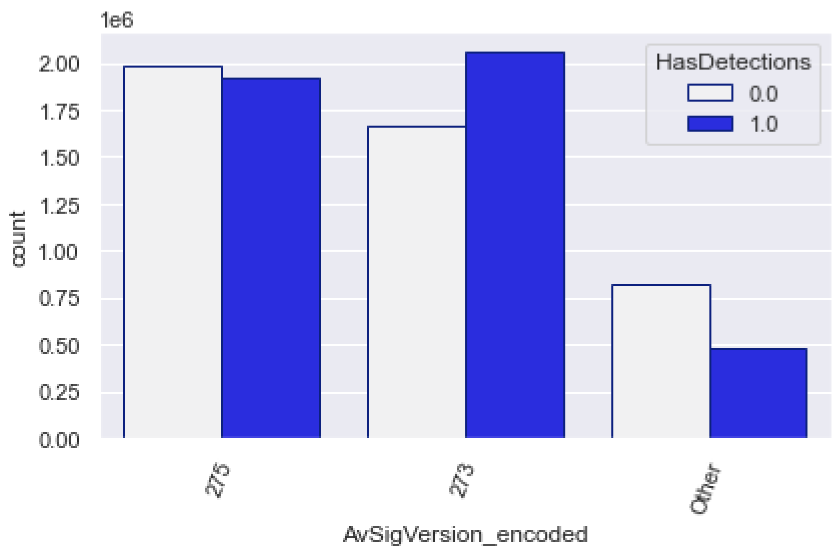

Figure 10: Barplot of categorical variable AntiVirus Version hued by target class.

	Chi2-statistics	p-value
SmartScreen_encoded_ExistsNotSet	2.214759e+05	0.000000e+00
SmartScreen_encoded_RequireAdmin	5.698300e+04	0.000000e+00
AvSigVersion_encoded_Other	4.569324e+04	0.000000e+00
AVProductsInstalled_2.0	4.478073e+04	0.000000e+00
AVProductsInstalled_1.0	2.998999e+04	0.000000e+00
Processor_x86	2.574212e+04	0.000000e+00
Census_OSArchitecture_x86	2.557563e+04	0.000000e+00
AvSigVersion_encoded_273	2.183277e+04	0.000000e+00
EngineVersion_encoded_15100	2.160499e+04	0.000000e+00
EngineVersion_encoded_14901	1.711393e+04	0.000000e+00
Census_PowerPlatformRoleName_Slate	1.686514e+04	0.000000e+00
AppVersion_encoded_14	1.560553e+04	0.000000e+00
AVProductsInstalled_3.0	1.482473e+04	0.000000e+00
IsProtected_1.0	1.297864e+04	0.000000e+00
AppVersion_encoded_16	1.270238e+04	0.000000e+00
EngineVersion_encoded_15000	1.224884e+04	0.000000e+00
AppVersion_encoded_18	1.111389e+04	0.000000e+00
EngineVersion_encoded_14800	1.027182e+04	0.000000e+00
AVProductsEnabled_2.0	9.696989e+03	0.000000e+00
Census_MDC2FormFactor_encoded_Detachable	8.591842e+03	0.000000e+00

Table 2: Summary of Chi-squared statistics and p-values for categorical features

Feature Engineering and Modeling

The dataset was splitted into a train and test set with an 80%-20% partition. The feature engineering and modeling process is performed in the Scikit-learn environment. In particular a data preprocessing pipeline was implemented by combining Simple Imputers for missing values imputation and column transformers to encode the categorical features with either one-hot-encoding, when the cardinality of the features is below 15 levels, or target encoding for high cardinality features, e.g. ID columns with hundreds or thousands of levels. After the encoding the dataset contains 511 features. This preprocessing step is then included in the overall pipeline where feature selection is performed by using a variance threshold to eliminate features with constant variance and a select k-best method. Finally, the classifier is added as Shown in figure 11.

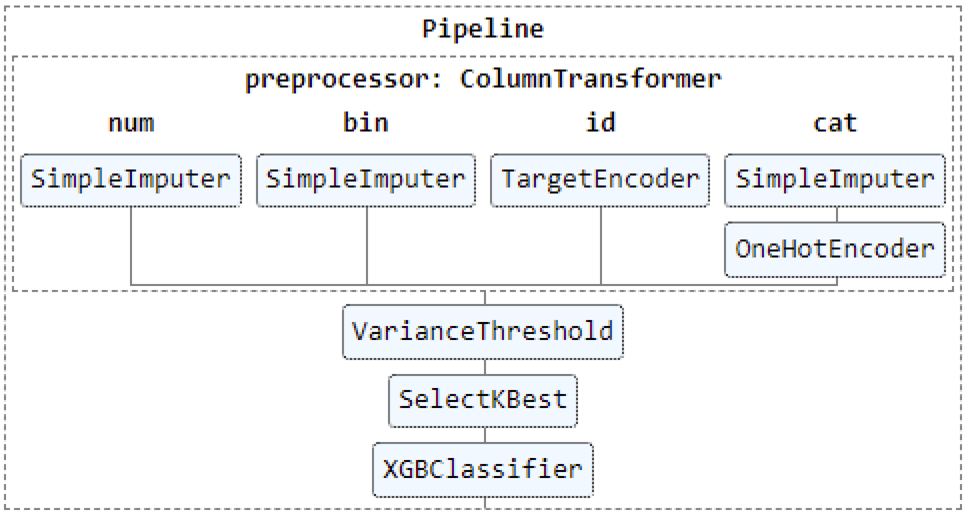

Figure 11: Schematic of overall pipeline including preprocessing pipeline, feature selection and machine learning classifier

The trained classifiers are:

- Logistic Regression with LASSO Regularization
- Random Forest
- XGBoost
- LightGBM

Even though bagging and boosting models are less interpretable than a simpler logistic regression, they still offer feature importances which shed some light on what choices the classifiers made and give insights into what features are most important. The metric of interest is the area under the curve (AUC) of the Receiver Operating Characteristic (ROC) curve with baseline performance of dummy classifier (random chance) of 50%.

Tuning and overfitting were addressed by searching the best parameters through a randomized search with cross validation. Due to limitations in computational power and time constraints, only 3 folds were used for cross-validation and 15 models were generated for each classifier. Although train and validation scores were consistent for the best parameters and no significant overfitting was found, i.e. train mean ROC AUC of 0.75 and validation mean ROC AUC of 0.72, it is advisable to increase the number of folds to 5 and generate more models, e.g. by setting n_iter to 60, to explore a larger space of the loss function and find better minima and performance.

The ROC AUC results of the trained classifiers on the test set are shown in Figure 12. As seen, bagging and boosting models only slightly improved the 0.70 AUC obtained by Logistic Regression to scores of 0.71 for random forests and 0.72 for both boosting algorithms. A stacking ensemble of the four models offered 0.73 ROC AUC performance. The coefficients assigned to each base learner by the meta-model of the stacking ensemble are shown in Table 3, random forests and XGB are given higher coefficients. Given the relatively low performance increase at the expense of a much higher complexity, the stacking model does not seem to be as useful as hoped. Therefore, the final model is the tuned LightGBM, which offers good performance, reasonable computational time or training and scalability.

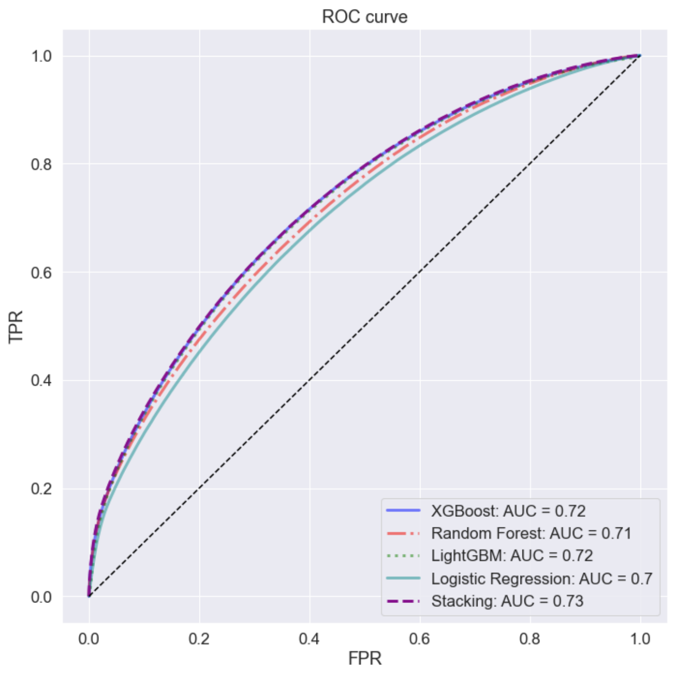

Figure 12: ROC curve and AUC results for the trained models.

	Coefficient
Model	
XGBoost	2.169014
Random Forest	3.303607
LightGBM	1.313139
Logistic Regression	-1.748189

Table 3: Coefficients of each base-model determined by Stacking meta-learner.

Feature importances for the top 20 features of the final model are shown in Figure 13. It is found that Smart Screen is the most dominant feature in determining whether a malware will infect the machine, in line with the exploratory data analysis and chi-squared test results. The remaining features are significantly less important, with relative importances less than 5%. Among these, we notice the Anti Virus Product ID and Version as well as some OS and machine characteristics, e.g. OS platform sub-release and processor kind.

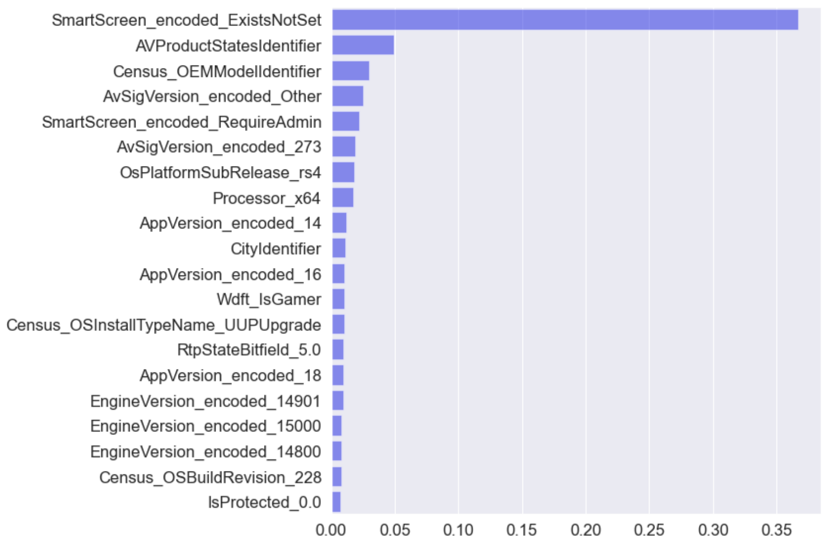

Figure 13: relative feature importances for best XGBoost model.

Key Findings and Future Research

First, the most predictive feature is the Windows SmartScreen, i.e. a cloud-based anti-phishing and anti-malware component included in several Microsoft products which checks downloaded files from the web against a list of reported malicious software sites. In particular, a significant decrease in malware infections is seen when the SmartScreen is active. Second, the third party antivirus plays a noticeable role in reducing malware. In fact, it is seen that different versions of antiviruses lead to different protection from malwares. This also suggests that the problem of malware is intrinsically a time series problem likely dependent on whether the antivirus, Windows Defender and SmartScreen are up-to-date when a virus is detected. Third, some OS and machine models seem to be more prone to being infected by viruses, likely due to the use of the machine rather than the intrinsic machine specs, however further research should be performed on this topic.

Future research should include more descriptive information about the computer status when a malware is detected and aim to generate features which can account for the temporal aspects of malware infections, e.g. by deriving what components are out-of-date and what are up-to-date. OS, antiviruses, web browsers, applications evolve fast with several updates released on a weekly basis. Our data should therefore capture the entire history of a machine if we want to gain real insight into what is most important to prevent malwares and preserve customer’s privacy.