MATH 189 Final Project

Microsoft Malware Competition

Yuxin Zou, A13996888
Yifei Li, A92082060
Bo Hu, A13805750
Bo Zhang, A13616488
Xuanyu Wu, A13569778

Introduction and Literature Review

Computer virus can bring various problems to the victims: from insignificant increase in outgoing traffic (if a computer is infected by a Trojan sending out spam) to the complete network breakdown or the loss of critical data, results in threat to exposure of security information and economic loss. Study shows that in 2016, computer viruses were estimated to have cost the global economy over 450 billion dollars(Graham). This means that around 0.5% of the world's economy is being lost due to computer viruses. Therefore, preventing malwares infecting the machine is essential for cyber security.

To lower the risk of infection to the controllable level, several methods had been used in the industry. Some methods include signature based antivirus applications with Aho-Corasick algorithm, which is a fast, simple-to-run approach that can match multiple patterns simultaneously to identify the potential virus by its signature(Cloonan). Behavior-based malware detection, which evaluates the virus's intended action before its actual execution using dynamic analysis(Cloonan). Recurrent neural networks can also predict whether an object is malicious from its behavioural data(Rhode). Some others use structured heterogeneous information network to build up a malware detection system(Hou). But what are the causes for a machine to be infected? Using the data given by Microsoft Malware Prediction Competition on Kaggle, we will establish several models to show which features are related to the infection of malware.

Data Description

The data we use is provided by Malware Prediction Competition on Kaggle. The data contains 48 measurements of each machine and a label which defines whether the machine is infected by malware or not. It contains bias data as well as missing values, which we need to take care of during data preprocessing. Here are some important features:

Column Name	Description
HasDetections	Whether the machine is infected by malware or not
MachineIdentifier	The machines' identifier number
AVProductsInstalled	Tells whether an antivirus software is installed in the device
Census_ChassisTypeName	Retrieves a numeric representation of what type of chassis the machine has
Census_InternalPrimaryDiagonalDisplaySizeInInches	Retrieves the physical diagonal length in inches of the primary display
Census_InternalPrimaryDisplayResolutionHorizontal	Retrieves the number of pixels in the horizontal direction of the internal display
Census_IsTouchEnabled	Whether the device is a touch device
Census_ProcessorCoreCount	Measures the number of core in the device's CPU

Exploratory Data Analysis

Scenario 1 - Software

In order to explore the behavior of malwares from the perspective of software, we investigate the effect of the number of antivirus product installed (AVProductsInstalled) on the malware detection rates.

Under the feature 'AVProductsInstalled', there are totally 8 unique integer values, i.e. 0, 1, 2, 3, 4, 5, 6, 7, indicating the number of antivirus products installed on the Window's machine. By plotting detection rates for each unique value of 'AVProductsInstalled' in *Figure 1.1*, we can see that there is a decreasing trend of detection rates as the number of antivirus products installed in the machine increases.

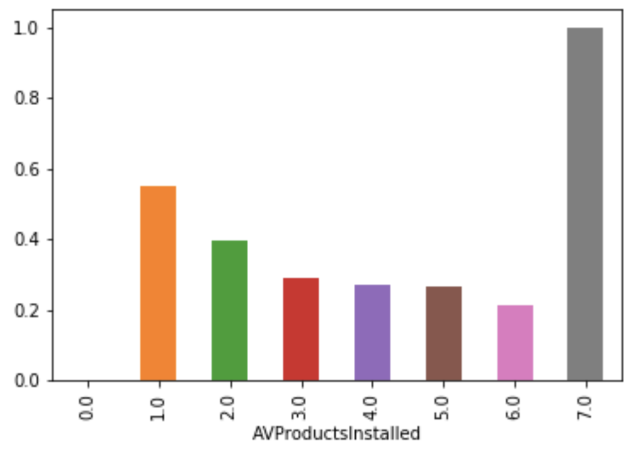
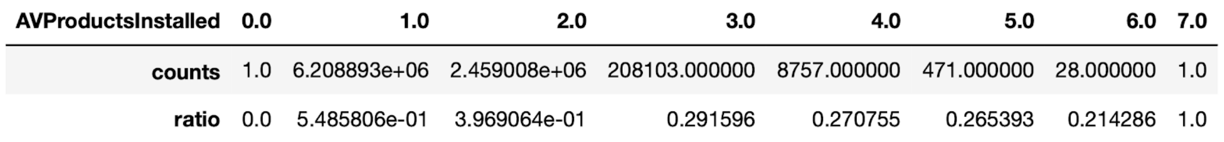


Figure 1.1: Rate of HasDetections Sorted by AVProductsInstalled

However, the gray bar corresponding to 7 installed antivirus products has an unexpected high rate of detection which is 1; machines with 0 installed antivirus products has an unexpected low rate of detection which is 0. This phenomenon can be explained by the incidence of each category from *Table 1.1,* which summarizes the total counts of machines and rates of detection for each category of number of antivirus products installed.



AVProductsInstalled	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
counts	1.0	6.208893e+06	2.459008e+06	208103.000000	8757.000000	471.000000	28.000000	1.0
ratio	0.0	5.485806e-01	3.969064e-01	0.291596	0.270755	0.265393	0.214286	1.0

Table 1.1: Counts of Machines & Rates of Detections w.r.t # of AVProductsInstalled

As the incidence frequency for 0 and 7 AVProductsInstalled are only 1, then their unexpected rates of detection can be explained by claiming that it happens by chance. Thus, by the graphical analysis, we can safely claim that there is a negative relationship between the detection rates and the number of antivirus products installed.

However, even though the negative relationship exists, we still cannot conclude the causal relationship between the number of antivirus products installed and detection rates. One of the reasons is the play of confounding factors. We notice that there is another feature 'AVProductsEnabled' which intuitively might have a monotonic relationship with 'AVProductsInstalled' and be responsible for the detection rates. Noticing that 'AVProductsEnabled' and 'AVProductsInstalled' are ordinal values, in *Table 1.2*, we summarizes the Spearman rank-order correlation.

	AVProductsInstalled	AVProductsEnabled	HasDetections
AVProductsInstalled	1.000000	0.238208	-0.149501
AVProductsEnabled	0.238208	1.000000	-0.042343
HasDetections	-0.149501	-0.042343	1.000000

Table 1.2: Spearman Rank-order Correlations.

From Table 1.3, we can see that both 'AVProductsInstalled' and 'AVProductsEnabled' are having a negative Spearman Correlation with 'HasDetections', which means that both are having a negative monotonic relationship with 'HasDections'. However, 'AVProductsInstalled' and 'AVProductsEnabled' are having a positive Spearman Correlation, meaning that they have a positive monotonic relationship. Thus, it is possible that 'AVProductsEnabled' are playing the role of confounding factors in the causal relationship between 'AVProductsInstalled' and 'HasDetections.'

Scenario 2 - Operating Systems

To investigate the effect of operating systems on malwares in a computer, we want to know if different platforms have different malware detection rates.

Platform column of the training set contains four unique values: Windows 10, Windows 8, Windows 7, and Windows 2016. Calculating total counts and the detection rate for each platform, from Figure 2.1, we can see that Windows 10, Windows 8, and Windows 7 have similar detection rates around 0.49. However, Windows 2016 has a lower detection rate of 0.35.

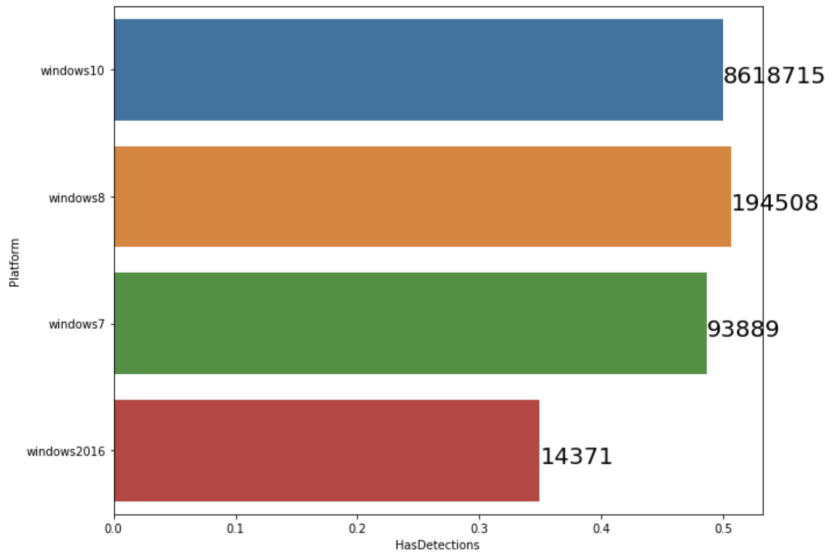


Figure 2.1: counts and detection rates for each platform

	Platform	Count	HasDetections
0	windows10	8618715	0.500032
1	windows8	194508	0.506720
2	windows7	93889	0.486511
3	windows2016	14371	0.349593

Table 2.1: counts and detection rates for each platform

To see if there exists a significant difference between detection rates across different platforms, we need to perform a Chi-square goodness-of-fit test. The null hypothesis is computers with different platforms have the same detection rate. The alternative is they have different detection rates. Under the null hypothesis, since each result for detection of virus (0 or 1) is from a Bernoulli trial, if detection rates (p) are the same for different platforms, each result is from the same Bernoulli distribution with parameter p. Using maximum likelihood estimator of p in a Bernoulli distribution, we have $\hat{p} = \frac{total\ successes}{total\ number} = \frac{4458892}{8921483} = 0.4998$. Therefore, the expected counts of detection of malware for each platform is total counts times $\hat{p}$, as shown in Table 2.2.

Platform	windows10	windows8	windows7	windows2016
expected	4307570.77	97213.68	46925.04	7182.52
observed	4309629	98561	45678	5024

Table 2.2: Expected and observed detections

Since each expected value is greater than 5, we can calculate the test statistic $\sum_i \frac{(Expected_i - Observed_i)^2}{Expected_i}$ =701.48. With degree of freedom 4-1-1=2, where the first -1 is from the theory and the second is from the fact that we estimated p, the p-value is 0, which is less than the significance level 0.05. Therefore, we reject the null hypothesis, concluding that different platforms have different detection rates.

Another useful feature is SkuEdition, the product type of a computer. Unique values in the training set include Home, Pro, Education, Enterprise, and etc.. Plotting counts of each type and corresponding detection rates, we have Figure 2.2. We can see that Server and Cloud have relatively lower detection rates, which deviate from detection rates of other product types.

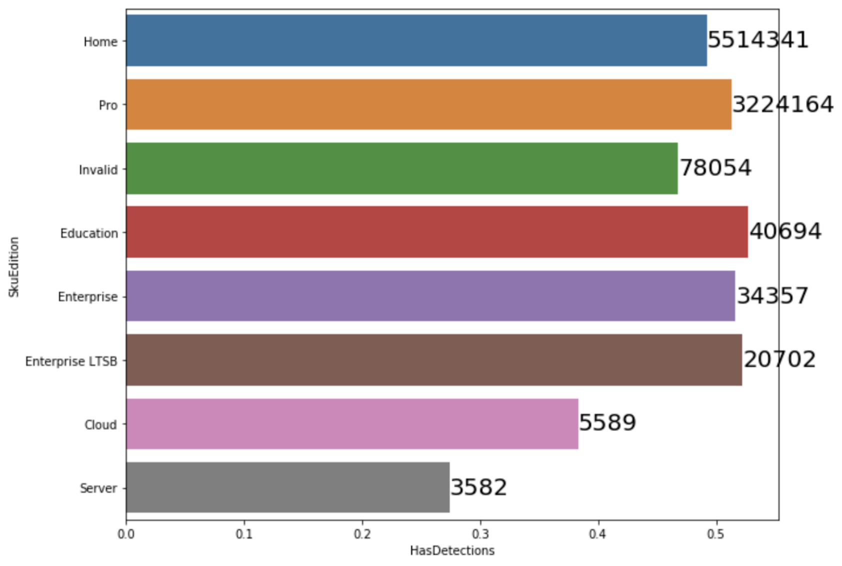
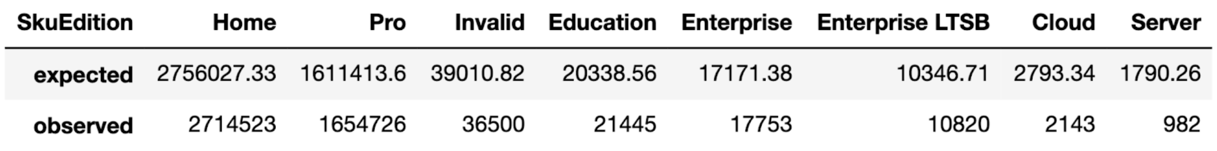


Figure 2.2: counts and detection rates for each SkuEdition

	SkuEdition	Count	HasDetections
0	Home	5514341	0.492266
1	Pro	3224164	0.513226
2	Invalid	78054	0.467625
3	Education	40694	0.526982
4	Enterprise	34357	0.516721
5	Enterprise LTSB	20702	0.522655
6	Cloud	5589	0.383432
7	Server	3582	0.274149

Table 2.3: counts and detection rates for each SkuEdition

Similarly, we want to know if their detection rates are significantly different. The null hypothesis is computers with different SkuEdition have the same detection rate. The alternative is they have different detection rates. Using MLE to estimate p, we have $\hat{p} = \frac{total\ successes}{total\ number} = \frac{4458892}{8921483} = 0.4998$. Table 2.4 summarizes the expected and observed detections by using counts multiplied by $\hat{p}$ and observing from the training set.



SkuEdition	Home	Pro	Invalid	Education	Enterprise	Enterprise LTSB	Cloud	Server
expected	2756027.33	1611413.6	39010.82	20338.56	17171.38	10346.71	2793.34	1790.26
observed	2714523	1654726	36500	21445	17753	10820	2143	982

Table 2.4: Expected and observed detections

Since each expected value is greater than 5, we can calculate the test statistic $\sum_i \frac{(Expected_i - Observed_i)^2}{Expected_i}$ =2568.67. With degree of freedom 4-1-1=2, where the first -1 is from the theory and the second is from the fact that we estimated p, the p-value is 0, which is less than the significance level 0.05. Therefore, we reject the null hypothesis, concluding that different product types have different detection rates.

Scenario 3 Hardware

In this section we discuss the relationship between a machine's hardware and its probability of being attacked by malware. We selected a few columns that attribute to a computer's hardware. We will focus on these specifications:

- Screen resolution (Census_InternalPrimaryDisplayResolutionHorizontal, Census_InternalPrimaryDisplayResolutionVertical)
- RAM size (Census_TotalPhysicalRAM)
- Hard drive capacity (Census_PrimaryDiskTotalCapacity)
- If the computer can be touch controlled (Census_IsTouchEnabled)

Horizontal screen resolution: 'Census_InternalPrimaryDisplayResolutionHorizontal':

First, we draw out the histogram grouped by the 'HasDetections' column. We can see through the histogram that there is no significant patterns in difference between devices that are detected and those that are not detected.

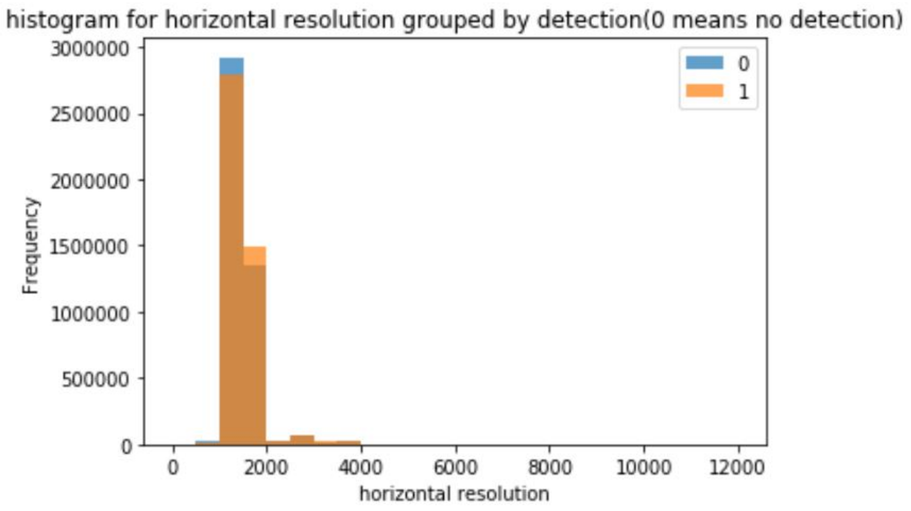


Figure 3.1: Histogram for Horizontal Resolution

If we take a closer look at where the most data is clustered, which is in the range 0 - 4000, we still cannot see a huge pattern to distinguish detected devices and non detected devices.

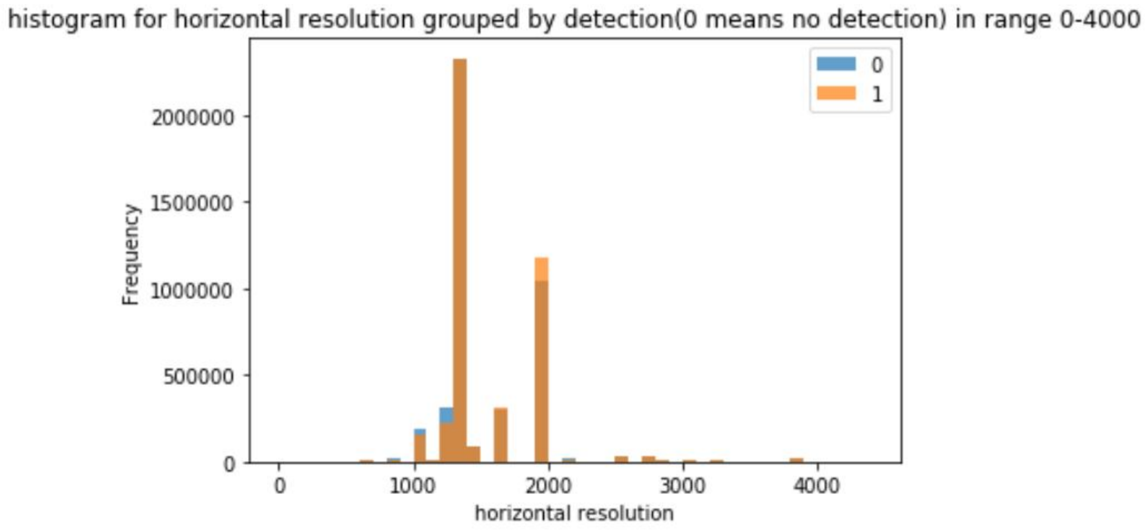


Figure 3.2: Histogram for Horizontal Resolution in Range 0-4000

To find more information through this column, we divide the bins into every 100 pixels. In that way, we want to find out for every increase in 100 pixels, what the detection rate would change.

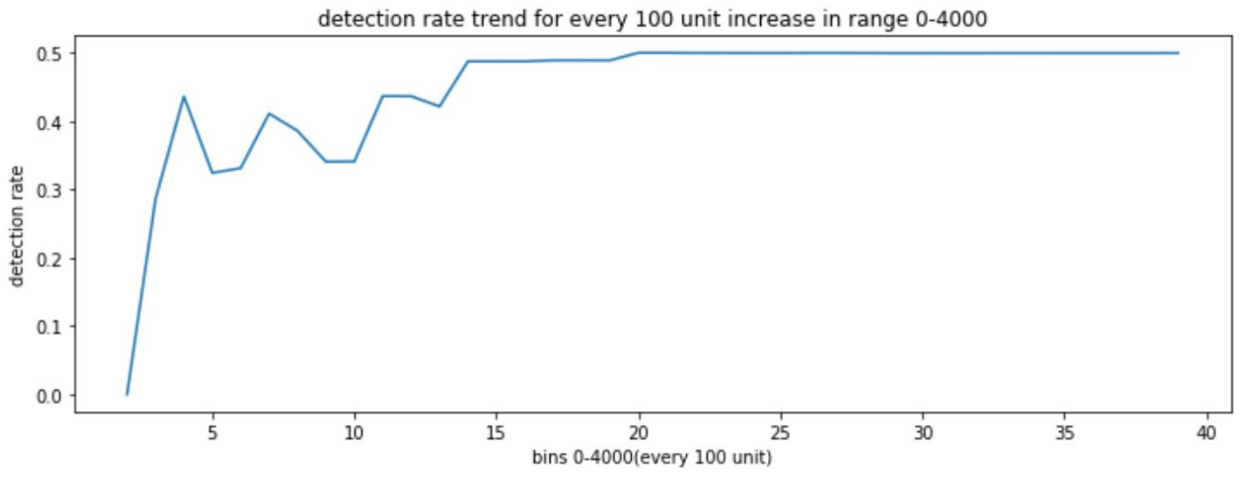


Figure 3.3: Detection Rate Trend in Range 0-4000

We could see that for the majority of the data in this column (in range 0-4000), as the horizontal resolution increases, the detection rate increases along the side. For the beginning part (0-1100 pixels), we could see about a 10 percent chance the device would not be reported as has detections. In order for our classifier model to have a higher accuracy, we could categorize the first few bins as "hard to be affected" and later part as "easy to be affected".

We then inspect the rest of the data, which is above 4000 pixels. We could see that most of the devices that have a higher resolution would tend to have less chance to be affected.

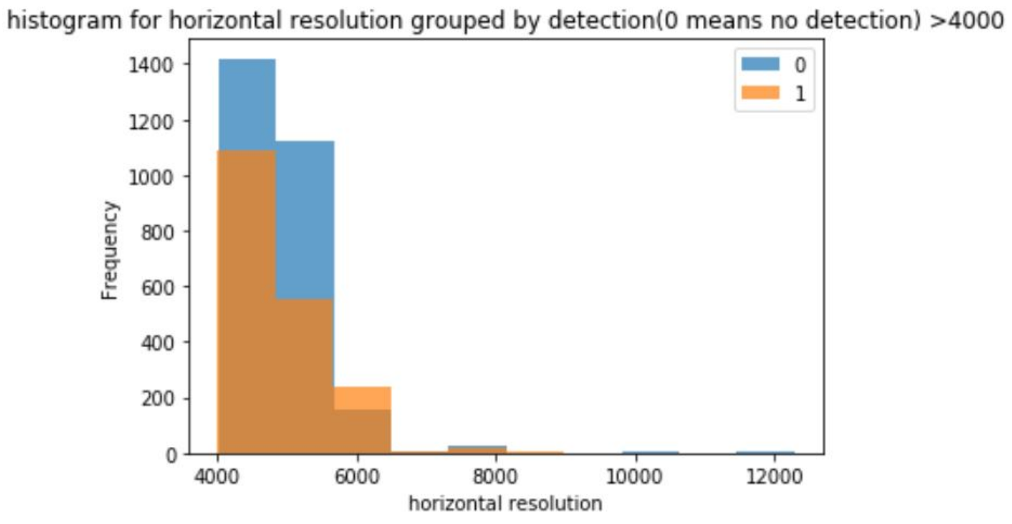


Figure 3.4: Histogram for Higher Resolution Devices

We then did the same thing for the higher resolution devices, which we check the detection rate for each 100 pixels increasing.

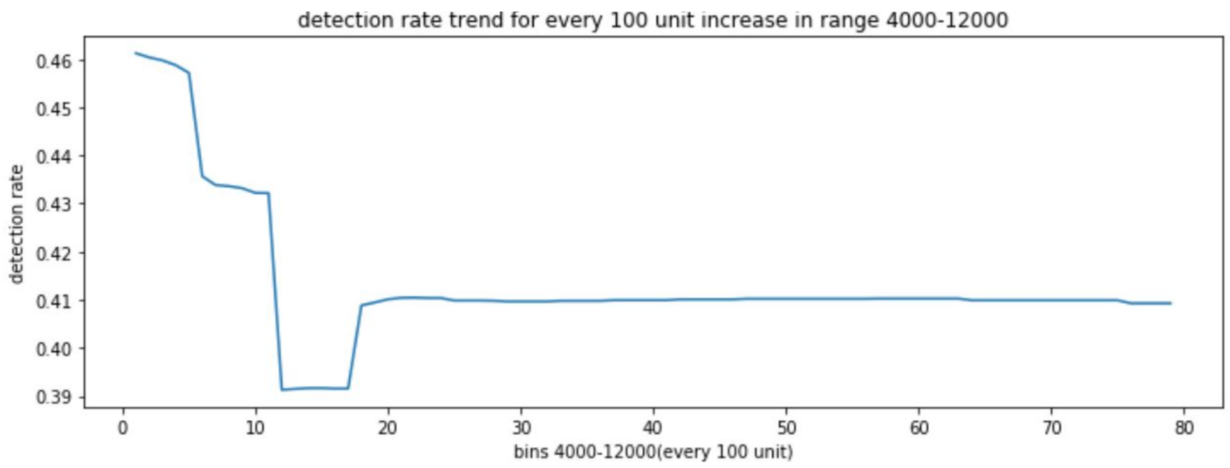


Figure 3.5: Detection Rate Trend for Higher Horizontal Resolution

This time, we could see that after the 5th bin, which is 4500 pixels, the chance of being affected drops to about 43 percent. We set this as the cutoff line to differentiate an “easy to be affected” device with a “hard to be affected device.

<u>Vertical screen resolution: 'Census_InternalPrimaryDisplayResolutionVertical'</u>

Here we see a similar pattern with the horizontal resolution. We could not see a huge difference between devices that are affected and those that are not affected.

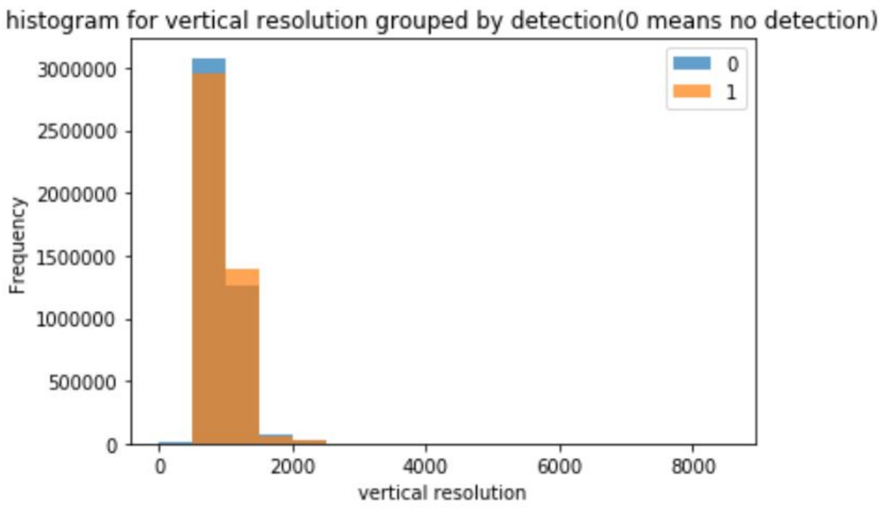


Figure 3.6: Histogram for Vertical Resolution

Similarly, we try to find out the trend that for every 100 pixels increase in the resolution, how the detection rate would change accordingly.

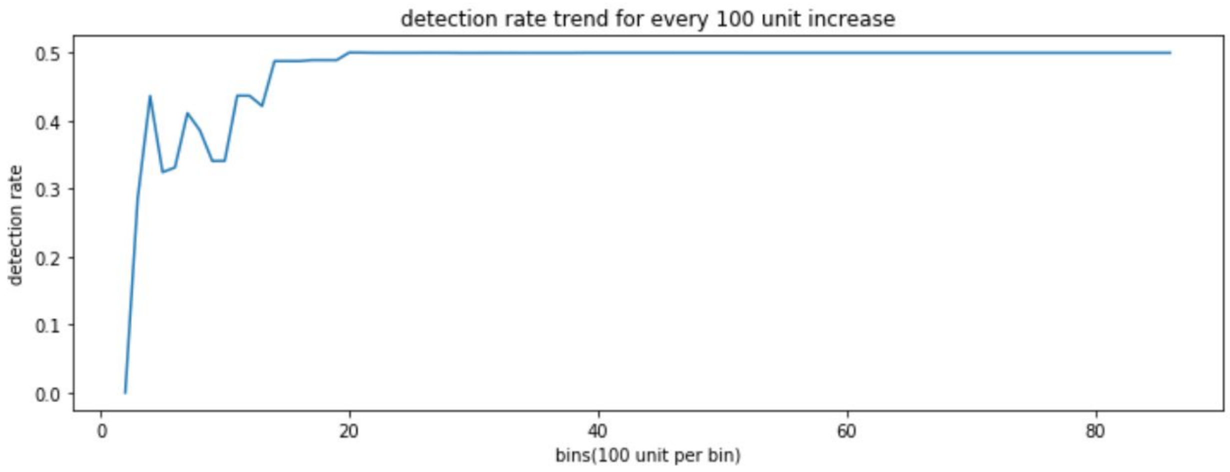
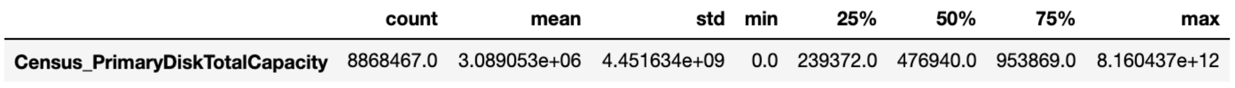


Figure 3.7: Detection Rate for Every 100 pixels increase

Looking at the graph, we set the cutoff line to be at the 11th bin, which is 1100 pixels. We would later clean this table as having two categories. “Hard to be affected devices” are having vertical resolution less than 1100 pixels, and others would consider to be potentially “easy to be affected”.

<u>Hard drive capacity: 'Census_PrimaryDiskTotalCapacity'</u>

This column refers to the amount of disk space on primary disk of the machine in Megabytes. Knowing that most modern personal computers have at least several hundred Gigabytes, the values in this column will be counted in thousands. Numerical analysis has the following output:



	count	mean	std	min	25%	50%	75%	max
Census_PrimaryDiskTotalCapacity	8868467.0	3.089053e+06	4.451634e+09	0.0	239372.0	476940.0	953869.0	8.160437e+12

Table 3.1: Disk Capacity Data Summary

We see that indeed the numbers are in the tens of thousands range. Plotting the raw distribution we get:

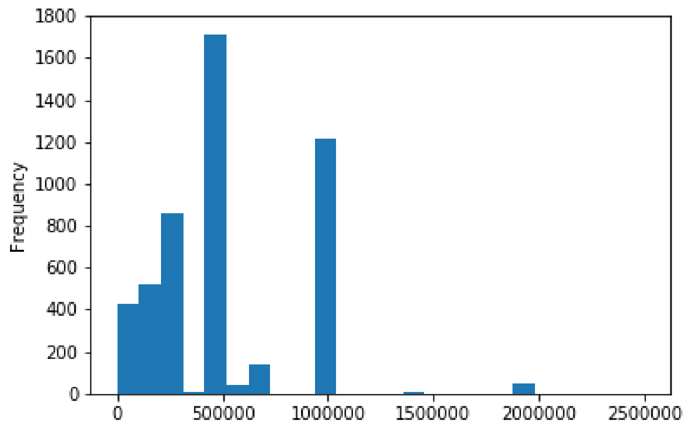


Figure 3.8: Histograms of disk capacity in normal and log scale.

It appears that most machines' disk capacities are centered around a few values. Most of the computer has a capacity of around 476 GB and around 953 GB.This can be explained by that most people choose the popular disk models or they are among the vendor default options. To include more values to analyze the full scope of the data, we apply a log function to scale this column. The reasoning behind this is because we want our data to be friendly toward our model, as some models struggles with large values and unstandardized data and outliers are mostly in the high range. The distribution after the transformation is shown below. You can clearly see that the distribution is more centered and ready to be fed into a model.

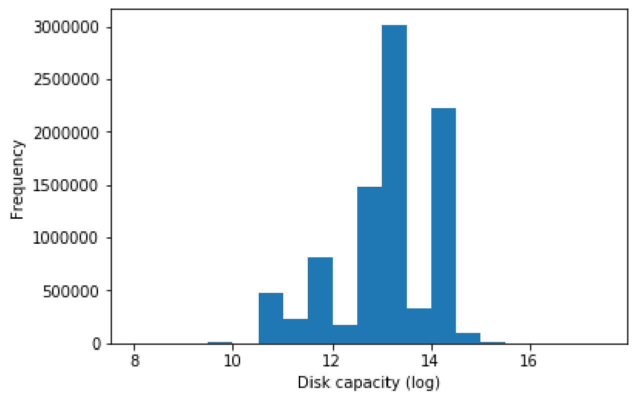


Figure 3.9: Histogram of logged Disk capacities

The next step is to do bivariate analysis to the column HasDetections. We rounded each logged disk capacity to the nearest 0.5 and calculate each group's detection rate.

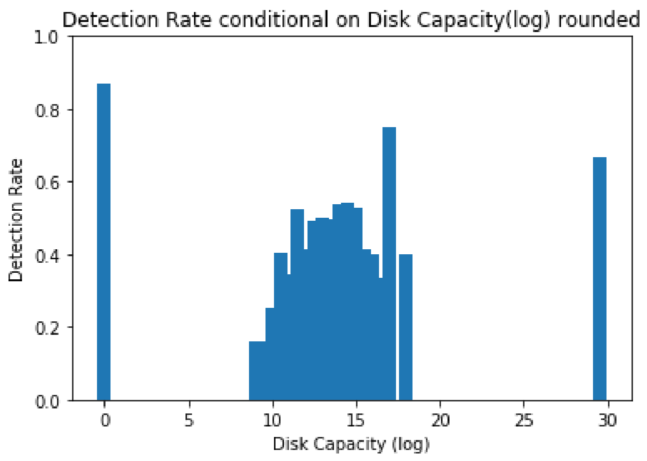


Figure 3.10: Detection rate distribution conditional on Disk Capacity (log and rounded)

Apart from the three obvious outliers in the detection rate, we can see that the rest of the points are followed by a polynomial function. We extracted the rounded logged capacity and its respective detection rate into a polynomial model and dropped those three points.

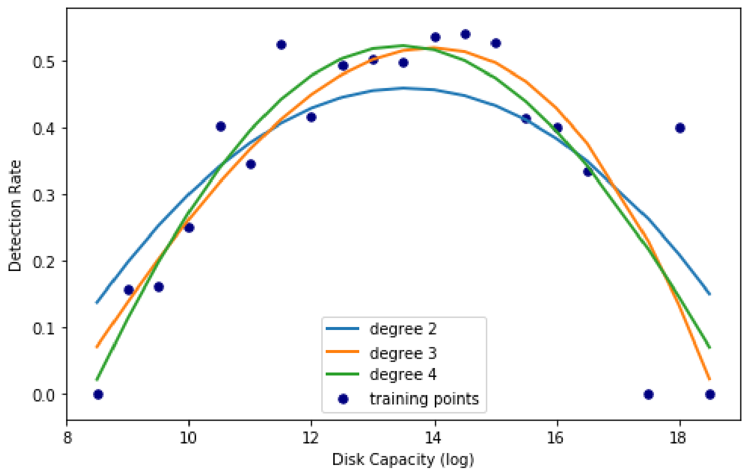


Figure 3.11: Polynomial feature training for disk capacity logged versus detection rate

We find that for a degree 2 polynomial, the correlation coefficient between the squared logged disk capacity and detection rate is 0.86876658, which suggests a strong linear relationship between those two values.

<u>RAM size: Census_TotalPhysicalRAM</u>

RAM sizes are referring to the physical memory space that a computer can utilize in computing. A large RAM size can make computers run more applications at the same time. The numerical analysis is shown below:

	count	mean	std	min	25%	50%	75%	max
Census_TotalPhysicalRAM	8840950.0	6115.260794	5115.820685	255.0	4096.0	4096.0	8192.0	1572864.0

Table 3.2: Physical RAM Data Summary

We can see that most of the values are in measure of thousands. Plotting the distributions in normal scale and in log scale we get:

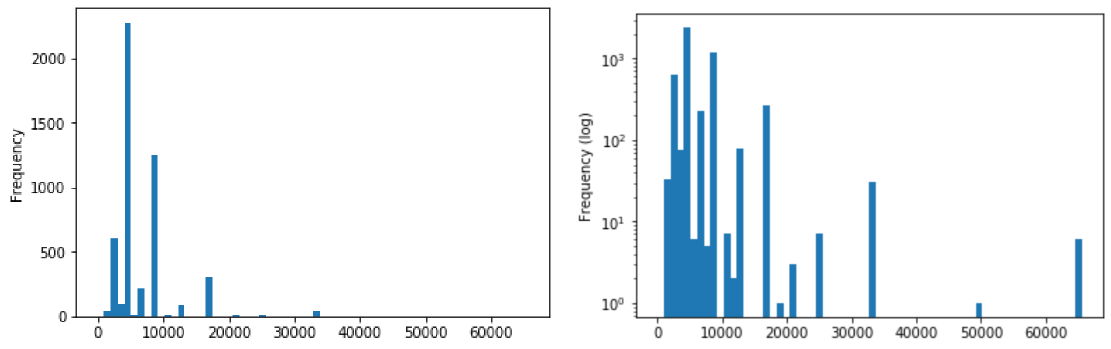


Figure 3.12: Histograms of RAM installed: normal scale and log scale

The distribution shows a similar nature: there are more values in the lower range and much less values in the higher range. So, we apply the log function here as well. The distributions after the transformation are shown below:

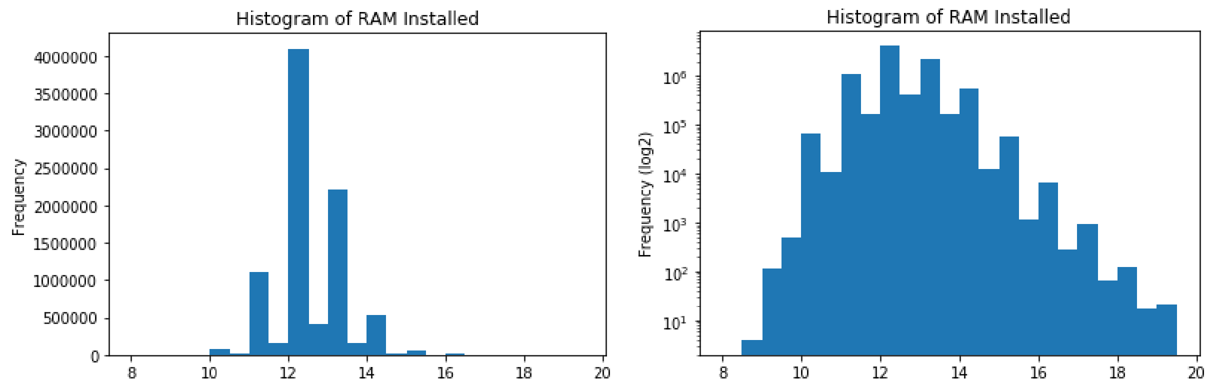


Figure 3.13: Histograms of logged RAM installed: normal scale and log scale

Next, we do bivariate analysis between the column and HasDetections.

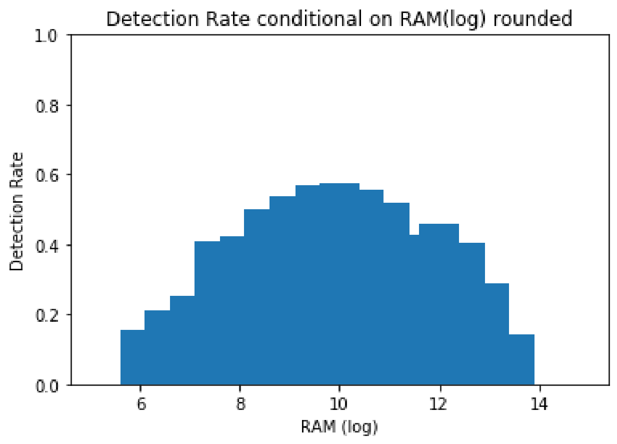


Figure 3.14: Detection rate distribution conditional on Disk Capacity (log and rounded)

Again, we rounded the logged RAM capacity to their nearest halfs. Grouping by these half values, we see that there is an unimodal and near symmetric distribution of probability for the RAM capacity (log) and the detection rate.

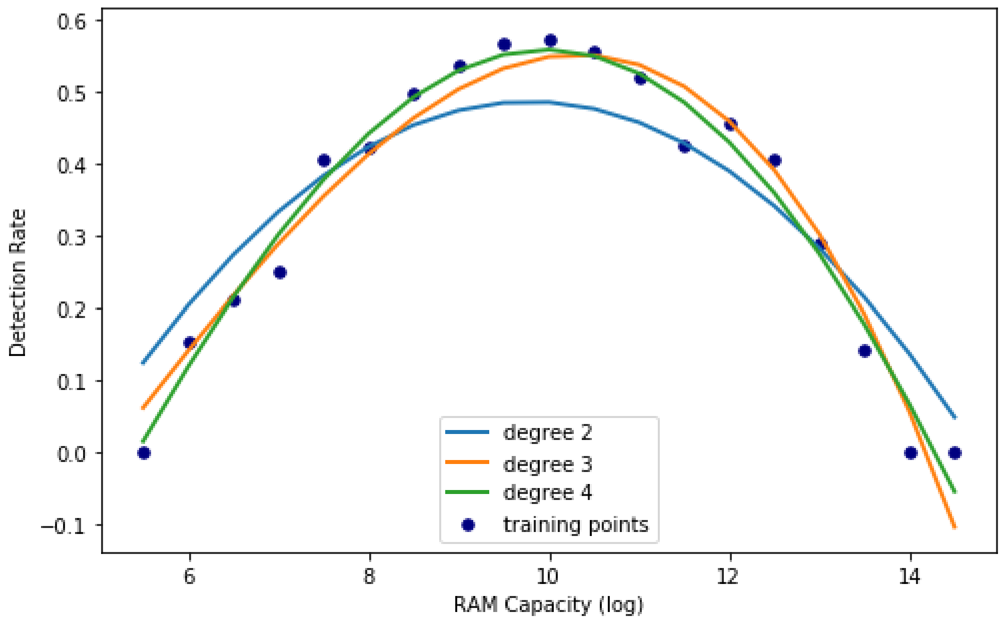
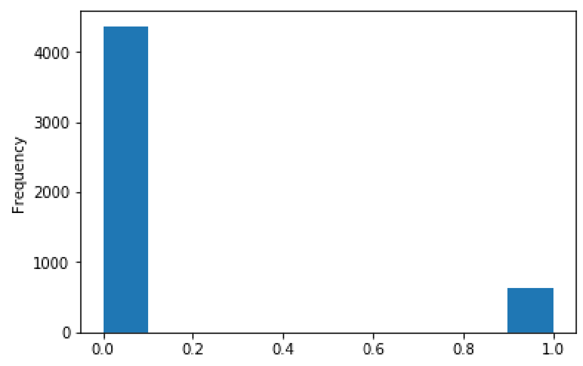


Figure 3.15: Polynomial feature training for RAM capacity logged versus detection rate

Under a degree 2 quadratic function, the absolute value of correlation coefficient is 0.9811, which means there is a very strong correlation between the log squared of RAM capacity and whether or not the machine has malware. In this context, it just means that if the RAM is closer to the middle of the data, there is a higher probability of malware's presence.

Touch-enabled devices: Census_IsTouchEnabled

Touch-enabled devices are machines with a touch screen attached as opposed to a traditional display. These devices typically are convertible, 2-in-1 devices that make human interaction easier. Some examples are Surface Pros, Chromebooks. This type of device accounts for a considerable proportion (12.55%) of the training data, so it's not logical to ignore this column.



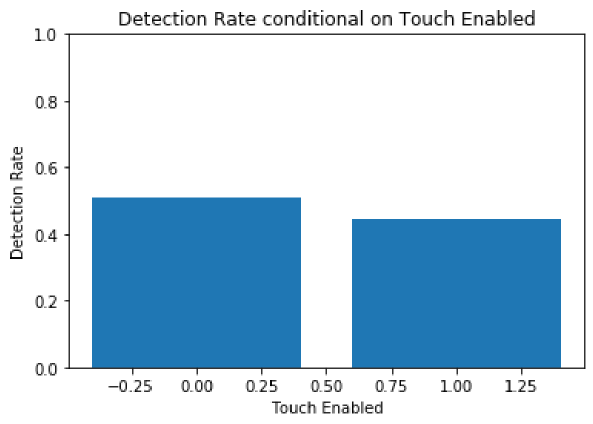
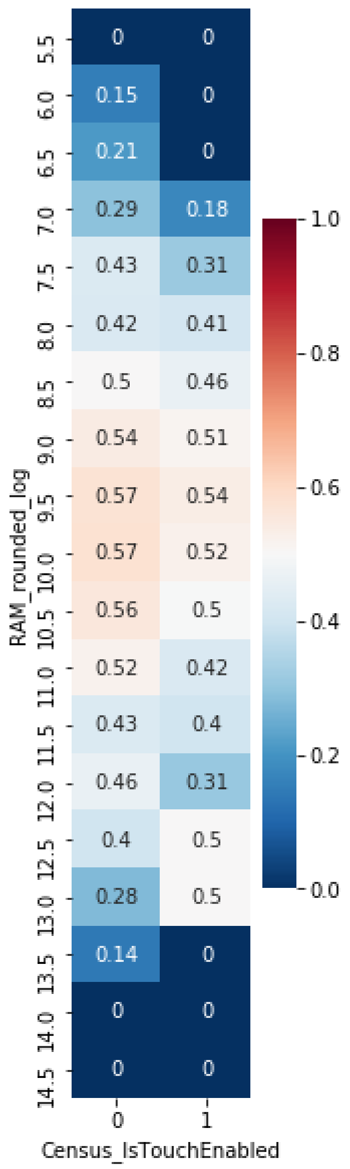


Figure 3.16: Distribution of Touch-Enabled devices and their respective detection rates

Here, just from the graphical analysis, it seems like there is no considerable disproportion of detection rates for different types of devices grouped by their touch-screen availability. But to know if proportions are significantly different, we need to to a two-proportion z test. Let p_1 be detection rate for not touch-enabled devices and p_2 be that for touch-enabled devices. The null hypothesis is $p_1 = p_2$ and the alternative is $p_1 \neq p_2$. In this case, $\hat{p}_1 = 0.507$, $\hat{p}_2 = 0.446$. To calculate the test statistic, we calculate $z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}_{pooled}(1-\hat{p}_{pooled})}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$, where $\hat{p}_{pooled} = \frac{\hat{p}_1 n_1 + \hat{p}_2 n_2}{n_1 + n_2}$ and $n_1 = 7801452, n_2 = 1120031$. The z-score we get is 120.7, and the corresponding p-value is 0, which is less than the significance level 0.05. Therefore, we reject the null hypothesis, saying that there is a significant difference between two proportions.

To understand the distribution of HasDetection more, we plot several heatmaps indexed by the various columns:



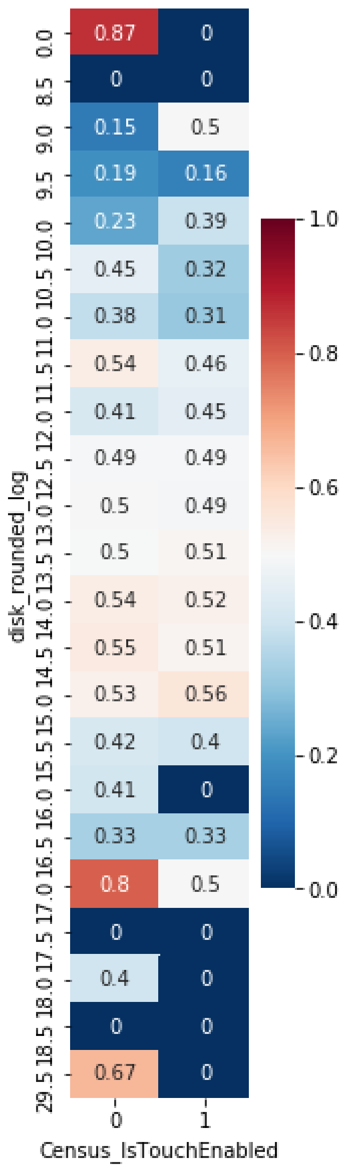


Figure 3.17 left: Heatmap of detection rate for different RAM sizes and whether touch-enabled
Figure 3.17 right: Heatmap of detection rate for different disk sizes and whether touch-enabled

From these two heatmaps, we see that devices that are not touch-enabled and are around the center of either RAM size or disk size's distributions are more likely to be affected by malwares.

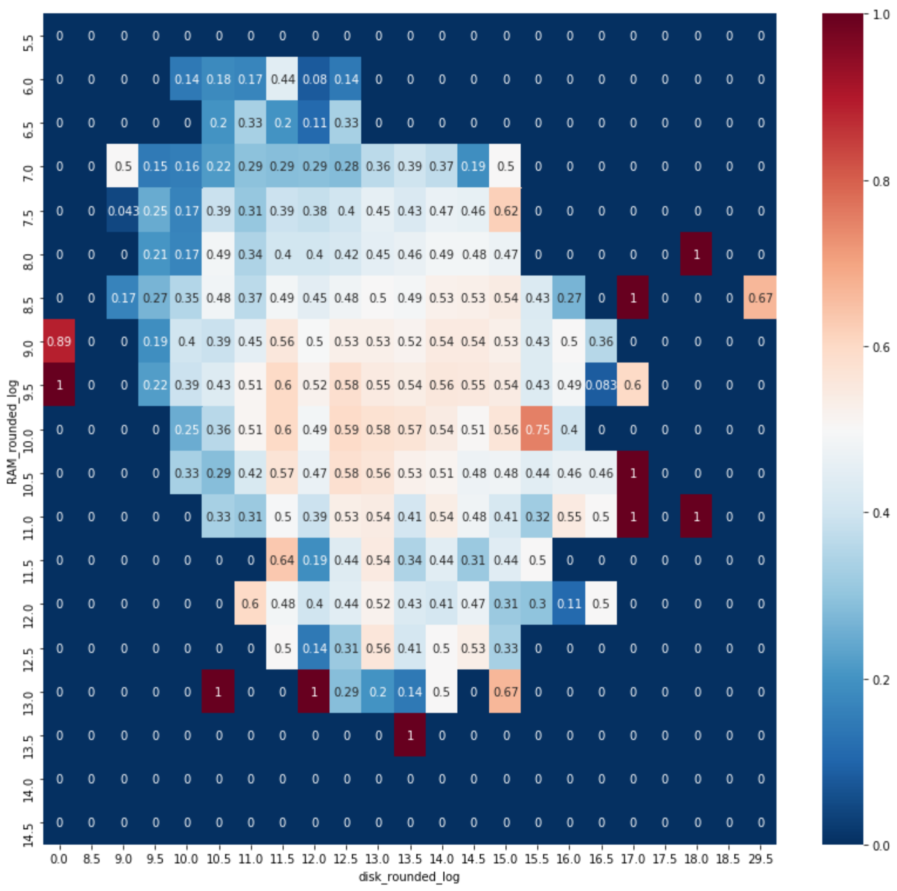
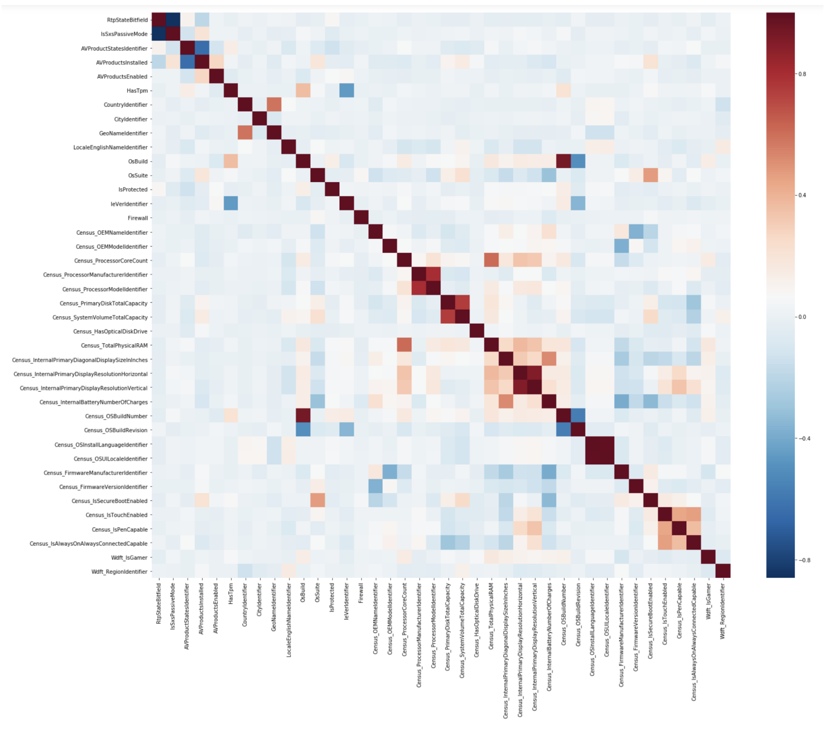


Figure 3.18: Heatmap of detection rate for different Disk sizes and RAM sizes

For this heatmap, apart from the few heat points of in the outer areas of the grid, machines with popular disk sizes and RAM sizes are likely to be affected by malwares. This seems to suggest that people who do not have much literacy around computers, which in turn choose the default configurations for their machine are more likely to be affected by malwares.

Data Preprocessing & Feature Selection

The data from this Malware Detection Competition has high dimension, where each row entry has 82 features. Thus it is computationally heavy to train the model on such a high dimensional file. Thus, we need to perform feature selection ahead to both reduce the dimension of this dataset and extract columns that are most representative of the data. First of all, we remove the column 'MachineIdentifier' which is distinctive to all row entries as the distinctiveness will contain no pattern, which is redundant information in predicting the label. Also, we notice that some of the features have high proportions of missing values. If proportions of missing values are too high, we might encounter those features as missing in testing data, which provides no information in predicting the label. Thus, we drop columns with more than 30% missing values. Now there are 72 columns left. Furthermore, highly skewed categorical columns (i.e. one single category with more than 99% appearances) are dropped as when skewness being removed, the dataset approximates to the Gaussian distribution better, and thus has more robust application with most models. Now there are still 64 columns left. Thus the next step is to drop one within the column pair of high linear correlation (>0.9). If the correlation is very high, it guarantees that two columns are strongly linearly correlated. The heat map below summarizes the linear correlation between each pair of features.



Since each column has more than 90% of non missing values, each column size is large enough is conclude that the result from Pearson correlation is statistically significant. In each pair of columns of high correlations, we drop the column with more missing values. Now there are 62 columns left. Even though columns of large proportion of missing values are dropped, most of the remaining columns still has some small proportions of missing values. Thus we perform some simple and naive imputations at first to make the data feddable into the feature selection model. We replace missing values in categorical features with the string “unknown” to make it another category and replace missing values in numerical features by sampling from its own distribution. By sampling from the distribution, we can retain the variance and bias within the column. Furthermore, we perform OrdinalEncoding to categorical features and standardizes the numerical features. Now, the data contains all numerical values which are ready to be fed into any models. To further reduce the dimension of the dataset, we perform Recursive Feature Elimination with Cross Validation (RFECV) to select the best number of features with Logistic Regression to assign weights to each feature (i.e. the coefficient of the features in the decision function). Afterall , there are 34 features left to best represent the dataset. Since imputations of each column in the process above are mostly naive and straightforward with the aim of reducing data dimension, we need to perform further detailed analysis on each column to decide what kind of imputation should be performed.

To summarize, we clean the data in advance with one hot encoding, ordinal encoding, and additive smoothing for most of them. For the rest of the data, we choose to either passthrough if it’s numerical column or we clean those columns by our exploratory data analysis’ result (e.g. horizontal resolution and vertical resolution becomes categorical).

For most of the nominal columns, we decide to one hot encode them in order to pass into the classifier model. If there are lots of categories in that column and it is very balanced in the sense that each category has a similar amount of data, then it would be appropriate to be one hot encoded. If the column’s categories are imbalanced, we use additive smoothing, which will be discussed immediately. Through one hot encoding, we could binarize the column and the classifier might be able to work on those binarized columns more easily instead dealing with plain text. Moreover, through binarizing the column, it prevents machine learning algorithm from calculating non-sense distance related information. It completely cuts off the correlation between all the categories in the same column, which is desirable since the column is nominal.

For those columns that we could identify an order among the categories, we decide to perform ordinal encoding. In that way, we can convert the values into numerical ones while preserving the ordinal relationship. As a result, the model could have a better use on that information.

Another technique that we use on processing the data is smoothing, especially additive smoothing conditioned on the column “HasDetections”. The purpose of additive smoothing is to reduce the noise on the tail of the distribution. If a category has high occurrence, then the value we observed would be closer to the true occurrence, but if one of the categories in the column has really low occurrence, then the value might not be that accurate if we have a limited amount of data. In other words, this category is more likely to be affected by the noise. For those columns that have this problem, we try to use additive smoothing technique.

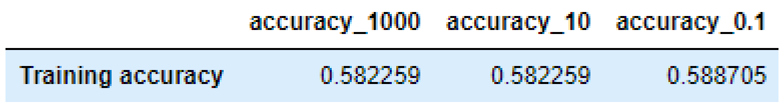
Given a dataset of observations x of size N, with d categories. We have the occurrence $p_i = x_i/N$ for empirical distribution. In additive smoothing, we introduce the coefficient alpha. The equation after additive smoothing would become: $p_i = x_i + \alpha/(N + \alpha * d)$. we could see once we have a larger alpha, then $p_i \approx \alpha/\alpha * d = 1/d$. In that way, all the categories would have the real equal likelihood to present.

Additionally, since this competition is about finding out the detected devices. We could use conditional additive smoothing that conditioned on the “HasDetections” column. We introduce another variable b, which is the occurrence in each category x. The equation becomes: $p_i = b_i + \alpha * (b/N)/(x_i + \alpha)$. b/N is the overall incidence, which in this setup would be the overall detection rate. The empirical distribution is b_i/x_i, which is each category’s detection rate. By conditional smoothing, if alpha is large, then each category’s detection rate would converge to the overall detection rate. Therefore, smoothing has a larger effect on smaller values of detection rates and less effect on larger rates. Through this process, we could reduce the variance of the categories with few values, but the bias would increase. There would be a trade off for the variance and bias. As long as the variance reduced is greater than the bias increased, we could accept that because the overall performance would improve.

Classification

Logistic Regression Classifier

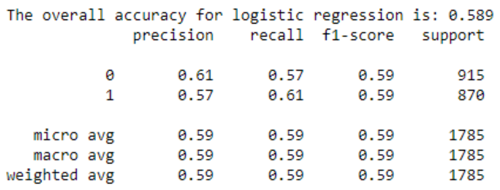
Since the target we predict is binary, it is sound to use logistic regression for our prediction as logistic regression is good for binary classification problems. We first use the training data to estimate the coefficients (Beta values b) of the logistic regression algorithm. This is done using maximum-likelihood estimation, which is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data. Similar to the process in LinearSVC, we used the LogisticRegression API with 5 fold cross validation to find the single best parameter set that optimizes the performance of the model. From the table below, we find the training accuracy is around 58.9%.



	accuracy_1000	accuracy_10	accuracy_0.1
Training accuracy	0.582259	0.582259	0.588705

Table 4.1: Training accuracy for logistic regression

Now we move on to the test set. The result of our prediction using logistic regression is shown in the table below. We have an overall accuracy of 58.9%. Which is worse than the random forest model and SVM below.



```
The overall accuracy for logistic regression is: 0.589
              precision    recall  f1-score   support

           0       0.61      0.57      0.59       915
           1       0.57      0.61      0.59       870

   micro avg       0.59      0.59      0.59      1785
   macro avg       0.59      0.59      0.59      1785
weighted avg       0.59      0.59      0.59      1785
```

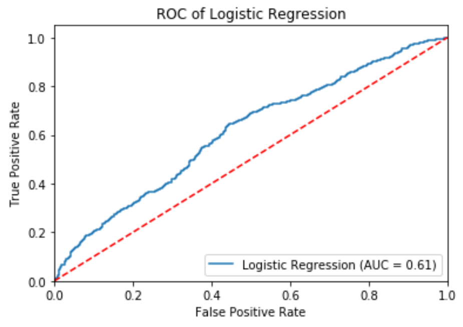


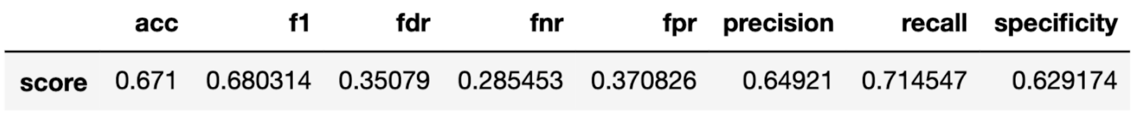
Figure 4.1 up: Training accuracies for logistic regression
Figure 4.1 down: ROC curve for logistic regression

Random Forest Classifier

The second classifier to use is Random Forest Classifier, which is one of the most commonly used classifiers. It constructs a multitude of decision trees and outputs the class that is the mode of the classes during classification.

First of all, we need to tune our model to find the best parameters that give us the highest accuracy. A useful tool is grid search. Passing in a collection of parameters, grid search gives us the combination that provides the best score using cross-validation. In this case, it turns out that the classifier performs best when there are 50 trees in the forest, trees have max depth of 7, The minimum number of samples required to split an internal node is 4, the minimum number of samples required to be at a leaf node is 3, and trees only consider $\sqrt{n}$ features, where n is the number of total features, when looking for the best fit. Using this set of parameters, after splitting our data into training data and test data and fitting the model onto the training one, the model achieves an accuracy around 67% on the test data. This means for new data that is similar to our dataset, we would predict correctly 67% of the outcomes.

However, accuracy might not be an indicator of a good model because it treats false positives and false negatives in the same way. It might occur that a model predicts well for positives but does poor for negatives. Therefore, it is necessary that we look at other metrics as well. Table 1 gives us values of accuracy, recall, specificity, precision, false-negative rate, false-positive rate, false discovery rate, and F1-score.



	acc	f1	fdr	fnr	fpr	precision	recall	specificity
score	0.671	0.680314	0.35079	0.285453	0.370826	0.64921	0.714547	0.629174

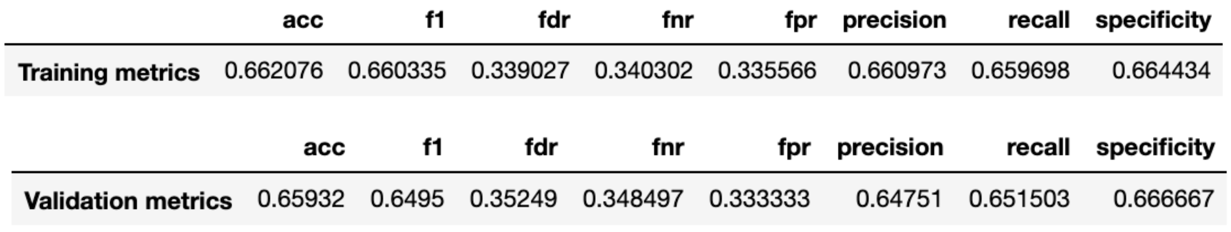
Table 5.1: Scores of other metrics

From Table 1, we can see that recall, the ratio of correctly predicted computers with malwares, and specificity, the ratio of correctly predicted computers without malwares are 0.71 and 0.63 respectively. Therefore, the model performs better when it is given a computer that has detection of malwares. Also, the gap between two scores are not large so it might be a good model overall.

Support Vector Machine Classifier

We now move to a more sophisticated classifier named support vector machine. This type of regression model is generally used to analyze data for a binary classification problem, which is the exact intent of this project. Every individual data point is measured in a high dimensional space, where each feature attributes to one axis. The model will try to find a hyperplane that best divides the two distributions of different classification outcomes. As a result of this model's nature, it will find a nonlinear relationship between the features and the target classification. Since there are so many features that are inter-related and have a tree-like decision process, we expect this model will have a better performance than a pure-linear one, such as logistic regression.

First, we try the LinearSVC model from scikit-learn. Despite the fact that this is a pure linear model and contradicts the notion that we are applying a non-linear model here, it is important to set up a baseline model to systematically evaluate the performance of the non-linear SVM model. We used the data straight from preprocessing and feed into the model using a random training-testing split. Because of the overabundance of features in this classification, we upped the number of iterations for convergence to 50000 times, making the model to fit quite well before we return the model weights. Again, we used the GridSearchCV API to find the best parameters and find that using an L2 penalty with a parameter of 100, the model performs best in accuracy. The training and validation metrics of this best model are shown below. It has a ROCAUC value of 0.7151 on the test set.



	acc	f1	fdr	fnr	fpr	precision	recall	specificity
Training metrics	0.662076	0.660335	0.339027	0.340302	0.335566	0.660973	0.659698	0.664434

	acc	f1	fdr	fnr	fpr	precision	recall	specificity
Validation metrics	0.65932	0.6495	0.35249	0.348497	0.333333	0.64751	0.651503	0.666667

Table 6.1: Training and test metrics of the Linear SVC model

With the performance of a simple baseline model in hand, we delve into a more complex version of SVM (Support Vector Machine), which is a SVM with RBF (Radial Basis Function) kernel. Similar to the process in LinearSVC, we used the GridSearchCV API with 5 fold cross validation to find the single best parameter set that optimizes the performance of the model. The training and validation metrics of the single best model of RBFsvm is given below. (the single best model has σ = 1e-05 and c = 5000)

	acc	recall	specificity	precision	fnr	fpr	fdr	f1
Training metrics	0.634689	0.62882	0.64051	0.634328	0.37118	0.35949	0.365672	0.631562

	acc	recall	specificity	precision	fnr	fpr	fdr	f1
Testing metrics	0.54576	0.54973	0.542029	0.530112	0.45027	0.457971	0.469888	0.539743

Table 6.2: Training and test metrics of the SVC model using a RBF kernel

As we can see from the table above, SVM with RBF kernel is performing much better in training data than in testing data, which might be caused by overfitting due to the complicatedness in RBF kernel.

Further Discussion and Conclusion

Nowadays, malware attacks have become more serious and more damaging to individuals and companies. Finding a way to detect those attacks is imminent. Through our exploration on this set of data, by doing parametric prediction with logistic regression, non-parametric prediction with random forest regression, and a higher dimension's analysis on lots of variables through SVM, we could get decent precision and recall score for detecting those attacks. If using more advanced techniques like training neural networks, we might improve our model to another level. For now, the comprehensive analysis on this data indicates that random forest classifier could better achieve the goal for detecting malware attacks partially because many of our features are categorical and random forest is immune to those categorical features. In our test data after train-test split, the accuracy we get is 0.67, meaning that we've correctly predicted 67% of the computers in terms of the affection of malwares.

Due to hardware limitations, we are not able to run those models on the whole training data provided by Kaggle. If we could have a better hardware support, our models might work better compared with the current one, which is only 0.1% simple random sample of the whole data set and might not be representative. Because of the same reason, we were not able to make inferences with our models on the official dataset provided by Kaggle, so there is no way to get feedback on our model on the actual prediction dataset that were centered around a few months after the training data timeframe. Instead, we were only able to validate our model by performing train-test splitting on the training data.

Method

Logistic Regression:

Logistic Regression is a predictive analysis that describe data and to indicate the relationship between one dependent binary variable and one or multiple independent variables. Logistic Regression is defined as follows:

$$\log it(p) = \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Random Forest Classifier:

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees.

Support Vector Machine:

The Support Vector Machine is trying to find a hyperplane that best divides the two distributions of different classification outcomes. It will look for a nonlinear relationship between features and the target classification. The radial basis function from linear kernel is making the hyperplane decision boundary in classes. In mapping the original dataset into a higher separable dimensional space, RBF kernel calculates the distance by $K\ (x_i,\ x_j)\ = exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2})$ where σ(gamma) is a free parameter.

Recursive Feature Elimination with Cross Validation (RFECV):

Given an external estimator that assigns weights to features (e.g. the coefficients of logistic regression), recursive feature elimination is to select features by recursively considering smaller and smaller sets of features. First the estimator is trained on the initial set of features and obtains the importance attribute of each feature. Then the least important features are pruned from current set of features. The procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached. RFECV performs the above process of RFE in a cross-validation loop to find the optimal number of features.

Theory

Chi-square Goodness of Fit Test:

Chi-squared goodness of fit test Chi-squared goodness of fit test is used to determine whether the sample data matches the hypothesized distribution. In scenario 2, we expected the distribution of detection rates follows a uniform distribution. To prove our judgment, we use we use Chi-square goodness of fit test to find out if there is a significant difference between the distribution under the null hypothesis and what we've observed.

Two-Proportion Z test:

Two-Proportion Z test is used to determine if there is a statistically significant difference between two proportions. The null hypothesis is two proportions are the same while the alternative can be two-sided (they are not equal) or one-sided (one is greater than the other). In Senario 3, we use this test to know if the detection rate of touch-enabled devices and that of not touch-enabled devices are significantly different.

Recall and Specificity:

Recall measures the fraction of positives that are successfully predicted, while specificity measures the fraction of negatives that are successfully predicted. In this case, recall is the percentage of computers with malware detections that are predicted correctly and specificity is the percentage of computers without malware detections that are predicted correctly.

Pearson Correlation:

In statistics, the Pearson correlation coefficient is a measure of the linear correlation between two variables X and Y. It has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.

Spearman's Rank-Order Correlation:

The Spearman's rank-order correlation is the nonparametric version of the Pearson product-moment correlation. Spearman's correlation coefficient measures the strength and direction of association between two ranked variables.

Train-test split:

In machine learning, a common practice to validate the performance of a model is to test on a validation set. Because calculating the performance of the model requires the ground truth of the target labels, the only way for us to validate is from the training set. To do this, we sample at random from the training set into two sets of data where one is exclusively for training and the other is exclusively for testing. The ratio for testing is often set to be around 0.25, where you have enough data to generalize the entire dataset and enough data to validate at the same time.

Cross-validation:

In machine learning, it is also important to verify the performance of a model multiple times to ensure a model is not overfitted on a specific subset of the training data. In the GridSearchCV function from scikit-learn, the algorithm apply the train-test split method for a number of times and fit the model on the training set, and validate on the testing set. Then, the algorithm finds the best model across the multiple times it is cross-validated based on some metrics.

Works Cited

L. Graham, “Cybercrime costs the global economy $450 billion: CEO,” CNBC, 07 Feb 2017. http://www.cnbc.com/2017/02/07/cybercrime-costs-theglobal-economy-450-billion-ceo.html.

Cloonan, John, “Advanced Malware Detection - Signatures vs. Behavior Analysis,” infosecurity, 11 April 2017. https://www.infosecurity-magazine.com/opinions/malware-detection-signatures.

Rhode, Matilda, “Early-stage malware prediction using recurrent neural networks,” ScienceDirect, August 2018. https://www.sciencedirect.com/science/article/pii/S0167404818305546

Hou Shifu, “HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network,” KDD2017, August 2017. https://dl.acm.org/citation.cfm?id=3098026