MATH 189 FINAL PROJECT: MACHINE VIRUS PROTECTION

A PREPRINT

Dukki Hong	Karl Rummler	Yiwen Li	John Diez	Nick Roberts
A98058412	A12405136	A13959913	A14751991	A11705541

March 26, 2019

1 Introduction

We are part of a competition to predict a Windows machine's probability of getting infected by various families of malware based on different properties of a machine. These properties can range from hardware specifications like type of model, how new it is, and its classification in terms of tier, to more internal properties such as softwares like anti-virus and default browsers. The data provided lists telemetry data containing more than 80 columns of these many properties and over a million rows and data point for each product. These data points were tied with information that microsoft's own defenses provided of if the product was infected by malware or not. Microsoft wants to know which one of these properties would be best at predicting the likelihood of a computer gaining an infection.

In this project, we start out with ground zero, explaining the foundations of what specifically a computer is, and list out important parts of it both hardware (physical properties) and also software (internal properties downloaded or are within the system). Once we've explained the role of these properties, we then explain the definition of malware and its unique ways in infecting the computer and how it affects these different properties of both software and hardware. From there, we develop research questions based on known information of the properties, and list out specific questions of where we think there is a higher or lower likelihood of predicting the likelihood of a malware attack. After we've set up these questions, we conducted individual analyses to see which columns had the highest likelihood of predicting malware, and ended it with a decision tree that could help predict which factors had the highest likelihood of helping predict malware infection. Additionally, we observe the performance of a decision tree classifier model and logistic regressor and describe the appropriateness of each model for the task. After observing and tuning these models we'll discuss our limitations and future works that can be done to improve our results and findings.

2 Background

Understanding Malware requires an understanding of the device in which it infects and destroys, and also the possible mechanisms of the device that it directly affects. Each malware that could infect a computer could vary greatly in its function: it could steal information stored and send it out online, destroy some internal function that could stop the computer from working, or even worse, weaken security systems to allow more viruses and negative malware to infect the system. Due to this variability, this also means that the mechanisms in the device (the PC) also matters as different parts of a car help in their own way to the overall function of the car, the computer is no different. Also as each part of the car is more vulnerable than the other, the computer system is the same exact way.

The structure of the background is to first describe the system of PC itself and some of the major components that we will be talking out throughout this paper. Once we lay the foundation of the computer itself, we'll be explaining the definition of malware and why it is important. Once those two groundworks have been stated, we'll be explaining the key programs within the computer that we suspect to be most vulnerable to malware to help develop our final analysis.

Personal Computer (PC)

The term computer has evolved greatly from what it was first originally developed. Created with a simple processor (something that follows specific sets of actions), it's original task was to do simple calculation and help ease the mechanical work in solving multiple problems (Treyz et al., 2002). Today, the computer has evolved to become the sum of even more stronger parts, with the ability to connect a user to the vast world wide web, allowing us to communicate and learn from each other around the world. How does a computer do that, and what are the main components that allow it to achieve this.

Physical Parts

Motherboard

The motherboard for a computer is the central hub of connection for the entire computer. Everything is connected to it, which allows communications throughout the many parts to flow efficiently and quickly. Descriptively, the motherboard is large (relatively) circuit board with multiple connections and slots for plugs to get in and out.

Power Supply

This supplies power to all other components to the machine and is usually the plug that you would use to charge or keep the device running.

Central Processing Unit (CPU)

The CPU is the computer's brain, and is the workhorse of the machines. This is what is able to do all the calculations and tasks that we ask it to do every day. Heat is created by the motherboard, which requires a fan to constantly cool it. A more powerful CPU is necessary for intense work such as analyzing and processing a lot of data or editing videos.

Random-Access Memory

Typically called RAM, this is temporary memory that allows the computer to hold information that a user has still running while doing something else. In a realistic sense, this allows a user to stream videos while doing another activity. This is typically more volatile and is temporary. Having more ram allows a computer to run more.

Hard Disk Drive/ Solid State Drive

Ram is temporary, but to store information into the computer permanently, a hard drive is required to store information. Solid state drives are just faster hard drives that allow for faster processing and actions.

Video Card

The final component is the Video card, a dedicated unit for handling the output of images to a display. This allows the computer to produce images clearly and well, and have their own RAM to perform this function. A higher graphics card is required to process extremely visual or high quality video, streams, or anything that we want to be displayed as clear. Gamers typically go for a higher graphics card to allow for smoother and clearer playing

Programs within the Computer

The last part that we will explain of the computer is the Internal software, or programs installed into the computer that malware typically infects. These internal programs typically communicate directly more with the hardware we have just detailed, and so if these programs get infected, then they might also lead to system failure.

Operating System

An operating system is a software that allows a user to run other application on any computing device (phone, desktop, laptop). Most applications or programs are written directly towards the O.S., for example, the most popular operating systems are Mac and Windows, and Safari is typically made only for Mac and explorer for Windows. Operating systems allow developers to take advantage of well known libraries to utilize in building the programs, which means some malware works easier for some O.S. than others. Other specific tools the OS and do is provide services to facilitate the efficient execution and management. Also it defines the User interface or style in which you're interacting with the device, main applications, and kernel or how it receives and sends data throughout different software and hardware.

Browser

To access the web, most PCs have their built in or downloaded browsers where they can access websites that yield information. Examples include internet explorer, safari, google chrome, firefox, and more. These allow users to explore the web, gather information, communicate easier online, and most importantly, download new software for their computer. These software could also potentially have the malware. Some browsers are typically deemed as better than others, and some malware thrives on different browsers.

Firewall and Anti-Malware/Anti-Virus Systems

The first defense for most PC computers (windows specifically) are natural built defenses called Firewall. Firewall typically keeps most services outside of a specific network from coming in and altering key programs necessary for normal functions. The next natural defense is Windows Defender, and this typically identifies well-known malware and prevents it from attacking the system. However, most computers have a quick and constantly defense system through anti-malware programs such as Malwarebytes, Avast, and more. Efficiencies for each different malware is another discussion, but not having any of these defense mechanisms working heightens the risk of malware.

Malware

Malware is a short term phrase of what we call malicious software, meaning software installed into the computer that can be utilized to compromise "computer functions, steal data, bypass access controls, or otherwise cause harm to the hose computer" (Veracode) Malware is a very broad term that refers to multiple types of malware that can affect and harm multiple software and hardware described in the previous section. Most of the major threats of the internet today is malware (Bayer et al. 20) Some of the forms malware can take place are: viruses, worms, botnets, rootkits, Trojan horses, and denial of service tools. It's typically spread and installed due to social engineering techniques such as a "required" click, or some ploy that tricks users into downloading it. If a person gets infected and/or downloads malware, the malware typically exploits software vulnerabilities in browsers and operating systems.

Typical defenses for malware are antiviruses/malware programs such as avast, malwarebytes, kaspersky, mcafee and more. For a more internal and basic defense, Windows Defender are typically the last line of defense for most Windows users. These programs utilize malware detectors which help them identify and contain malware before it can reach a system or network (Christodorescu Jha, 2003).

Some types of Malware:

- Trojan: software that contains or installs a malicious program with a harmful impact on a user's computer.

- Adware: software that automatically displays advertising material to the user resulting in an unpleasant user experience.
- Unknown/Obfuscated: A binary that has been obfuscated so that we could not determine its functionality

3 Investigative Questions

In this next part, we will be explaining some of the logic behind the analysis we will be doing throughout the paper. Now that we have built a foundation for specific hardware and software that Personal Computers (PC) contains, then we can further elaborate possible key points or data columns that would predict the likelihood for a person to have malware in their devices.

3.1 Do natural computer Defenses Matter?

As explained, computers specifically have natural defense systems either through a firewall or windows defender. Typically, some people have to turn off their firewall to allow access from other programs (non-malware) that they have to download or use. This can possibly increase the risk of a malware attack. Malware could come from the internet which would first have to go through the firewall server (protects from outside programs) which could them go into a personal server (Ackroyd 2007). We would suspect that a user who has this service turned off would be at a higher risk for infection. This goes equally as far for the next line of defenses such as Windows Defender.

After assessing the first natural defenses, the type of antivirus also matters. Many analyses have been done analyzing and testing different types of "Malware detectors" or anti-virus software. An analysis testing different types of malware by Christodorescu and Jha found that testing for the resilience of different types of scanners, they yielded poor results. They utilized commercial scanners and utilized different testing techniques of utilizing and modifying known malware and abusing the detector's weakness. So if a person has one downloaded, we would expect them to be at risk for malware, but not completely protected.

3.2 Does Browser Matter?

Web based malware is on the more prevalent ways an individual are able to get infected by malware. Internet transactions have the risk of exposing personal sensitive information such as bank and medical records, passwords, and or tools utilized for online transactions such as credit card information. Hackers or malware creators can insert small pieces of "HTML" in webpages which is a code that can automatically download malware when you click on a webpage (Provos

et al;). This suggests that the security implemented in the browser type matter, so we'd expect that if there's different rates of malware for different browsers, it's due to their security measures. Does having updated stuff/OS/Software influence the rates of Malware? OS developers constantly send updates to computers to keep them updating in new softwares and possible bug fixes. One of the updates typically is to its own windows defenders and firewall systems. This would mean that if there were any new threats that the OS might have patched, if the user has not updated, they might be more likely to get malware. So we predict that the older the OS system, the higher the likelihood for infection.

3.3 Does having a higher quality or more expensive processor influence malware?

One could expect higher quality machines to have a better chance at detecting and stopping malware. However, most malware can be downloaded off any site from any type of computer, so the entire relationship isn't as clear. Detecting novel malware that is typically unknown from simple defenses requires innovative or constantly updating anti-malware or malware-detecting devices (Rieck et al., 2008). People with higher quality systems might be more likely to purchase higher end virus systems.

3.4 Do Gamers have a higher chance or awareness of Malware?

Gamers typically buy higher end laptops or computers to keep up with more demanding video games. This requires them to typically have a higher quality mother board, graphics card, and more experience with using the technology themselves. However, gamers also have to download more software to be able to play the games they are interested in. We wonder if there's a relationship between the computer quality and gamer type, as well as whether or not having more experience or higher affinity to gaming computers leads to different rates of malware infection. Also, some OS are not specialized for gaming (like mac), which means there's a higher influx of Windows users.

4 Data

The dataset was large so 'read_csv' function was not able to load the dataset. Hence we specified data types to gain memory, i.e. change float64 to float32. We went ahead and switched 64 encodings into 32 or 16. Moreover, we switched binary values with missing values into float 16. In figure 1 below, we can see the counts for each variable type. From the features given, there's clearly an overwhelmingly large number of categorical variables over numerical. Due to this fact, we expect there to be difficulty in prediction given the data.

4.1 Data Exploration

The data sample does not only include samples of Microsoft customer's machines, but has been sampled to include much larger proportion of malware machines. Moreover, the dataset was sampled to meet some business constraints. Such business constraints include user privacy and time period during which the machine was functioning. Some machines come online and some go offline, some get updated with new features and some gets replaced by a different operating system. These complications have been included in the data, hence it is feasible that there may be imperfect agreements between cross validation.

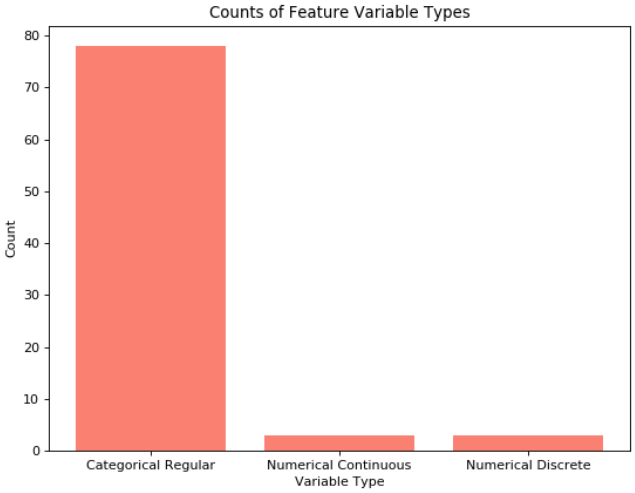

Figure 1: Count of Variables Types of Features

We intend to use the 'train.csv' dataset in our analysis. Each row in the dataset is labeled by the 'MachineIdentifier' column or feature. The dataset consists of 83 columns/features and 8,921,483 rows (8921483, 83). Since the data here is too large and the computational resources are not available, we have sampled down to 100000 rows. Before we begin with data cleaning and identifying different features in our dataset, we faced a situation where we were not able to load the dataset due to its sizeload ('read_csv' function could not load the data). Our approach to this problem was to specify data types to gain memory, i.e. switching float64 to float 32.

It is important to classify all the features according to different data types. We will mainly deal with two different data types: Categorical and Numerical Data. We have further divided the categories into 'Categorical Regular', 'Numerical

Continuous' and 'Numerical Discrete'. The following table is a brief summary of the data (Table 1).

Categorical Regular	**Numerical Continuous**	**Numerical Discrete**
EngineVersion	Census_DiagonalDisplaySizeInInches	Census_IntervalBatteryNumberOfCharges
AppVersion	Census_PrimaryDiskTotalCapacity	Census_ProcessorCoreCount
AvSigVersion	Census_SystemVolumeTotalCapacity	Cenus_TotalPhysicalRAM
OsBuild		
...		
Census_ProcessorClass		

Table 1: Sample of Dataset Features by Variable Type

Since our goal in this project is to determine whether Windows machine is infected by various families of malware based on different properties of the machine, it would be such that most of the features categorized above (Table 1) is important, i.e. some computers may be vulnerable to certain kinds of malwares while some may not be vulnerable, this would depend on the OS version, quality of the processor, different kinds of internet browsers and etc. However, for an investigation with a significant result, our goal is to opt out as much features as possible before we proceed with the investigation and analysis. In terms of the features that should be kept in our analysis, an example is 'HasDetections'. This feature will be the ground truth that indicates whether a malware was detected in a particular machine. As such, features like 'HasDetections' will be kept in our investigation.

Variable Name	Description
EngineVersion	Census_DiagonalDisplaySizeInInches
MachineIdentifier	Unique machine ID
ProductName	Defender state information e.g. win8defender
EngineVersion	Defender state information e.g. 1.1.12603.0
AppVersion	Defender state information e.g. 4.9.10586.0
AvSigVersion	Defender state information e.g. 1.217.1014.0
IsBeta	Defender state information e.g. false
GeoNameIdentifier	ID for the geographic region a machine is located in
LocaleEnglishNameIdentifier	English name of Locale ID of the current user
...	...
Census_DeviceFamily	Indicates the type of device that an edition of the OS is intended for.

Table 2: Dataset Feature Descriptions

4.2 Feature Engineering

Before we begin our analysis, we intend to opt out irrelevant and unnecessary features to help us increase model accuracy and quality.

4.3 Mostly Missing Feature Values

In order to opt out features, we first enumerated features according to the amount of missing information they had in decreasing order. Figure 2 shows a summary of features with missing values shown in fractions.

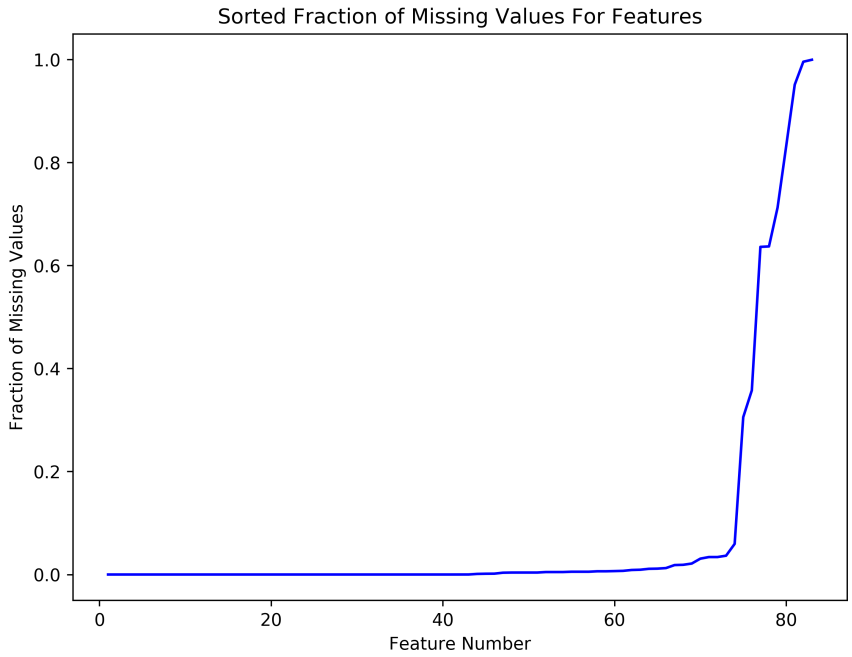

Figure 2: Fraction of Missing Values Across The Features

Variable Name	Fraction of Missing Values
PuaMode	0.9997
Census_ProcessorClass	0.9958
DefaultBrowsersIdentifier	0.9513
Census_IsFlightingInternal	0.8315
Census_InternalBatteryType	0.7126
Census_ThresholdOptIn	0.6372
Census_IsWIMBootEnabled	0.6363
SmartScreen	0.3576
OrganizationIdentifier	0.3056
SMode	0.0595
CityIdentifier	0.0365
Wdft_IsGamer	0.0339
Wdft_RegionIdentifier	0.0339
Census_InternalBatteryNumberOfCharges	0.0308
Census_FirmwareManufacturerIdentifier	0.0213
Census_FirmwareVersionIdentifier	0.0188
...	...

Table 3: Fraction of Missing Values From Highest to Lowest

4.4 Skewed Categorical Variables

We can further decide to remove features by checking skewness of the categorical data. If a categorical data is extremely skewed, then most of the data points belong to a particular category for that feature, i.e. when a feature is named 'HouseColor' but all the houses in a particular dataset is red. The following shows a table that includes features that have skewness of 1:

Decision Tree (Sqrt Features, Depth = 8)	Decision Tree (Log Features, Depth = 9)
AutoSampleOptIn	1.000
PuaMode	1.000
Census_IsFlightsDisabled	1.000
IsBeta	1.000
Census_IsWIMBootEnabled	1.000
Census_ThresholdOptIn	0.9998
SMode	0.9996
Census_IsPortableOperatingSystem	.9992
Census_DeviceFamily	.9987
...	...

Table 4: Table of Top Skewed Variables

We have removed 15 features so far: 'IsBeta', 'PuaMode', 'Census_ThresholdOptIn', 'ProductName','AutoSampleOptIn', 'UacLuaenable', 'Census_IsPortableOperatingSystem', 'SMode','Census_DeviceFamily', 'Census_IsVirtualDevice', 'Census_IsWIMBootEnabled', 'Census_ProcessorClass', 'Census_IsFlightsDisabled', 'Census_IsFlightingInternal', 'MachineIdentifier'

Note 'MachineIdentifier' was also removed from the dataset as it is not necessary in our investigation and only adds noise. It is merely a categorical data that provides nothing more than a numerical attribute to a particular row. Now we are only left with 68 features after irrelevant tables have been removed through checking missing features and skewness.

It can be seen in Figure 2 that the first two features have more than 99 percent (0.99) of their rows with missing values. Hence we will proceed and remove PuaMode and Census_ProcessorClass from our dataset.

4.5 Missing Data Values: Imputation

There were features that had missing values in some of the rows. Such missing values had empty rows. So, we modified the data values such that categorical variables filled in the missing values with the mode and we filled in missing

numerical values with the median.

4.6 Correlated Features

It is important to deal with numerical data types. In doing so, we will calculate the correlations between each of the features. Calculating the correlation between the features is significant because correlation can help in predicting one attribute to the other. This can be helpful in imputing missing values. Moreover, correlation can help determine the causal relationship between the features. Figure 3 shows correlations between the first 10 features of the dataset. Notice that the diagonal entries have correlation values of 1 since it is the correlation between the feature itself. We computed the correlation values between all of the features, however, we display the correlations in batches of 10 for legibility.

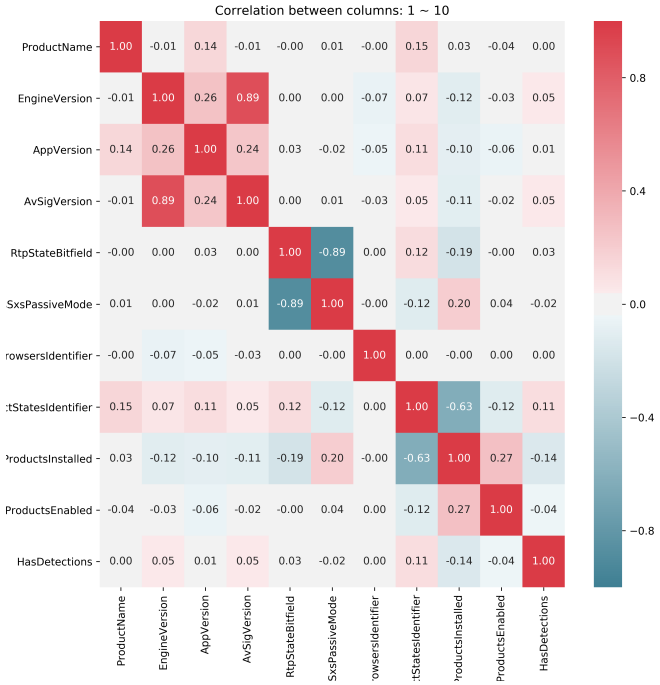

Figure 3: Correlation Between Features 1 Through 10

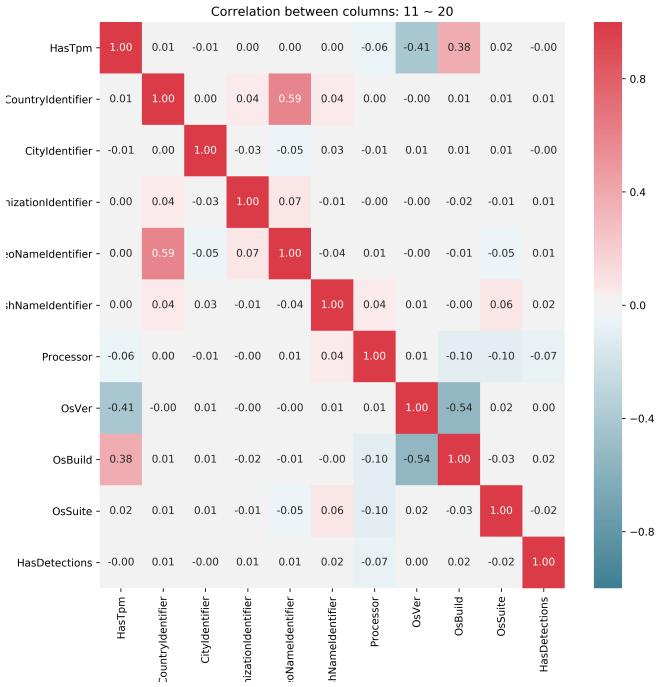

Figure 4: Correlation Between Features 11 Through 20

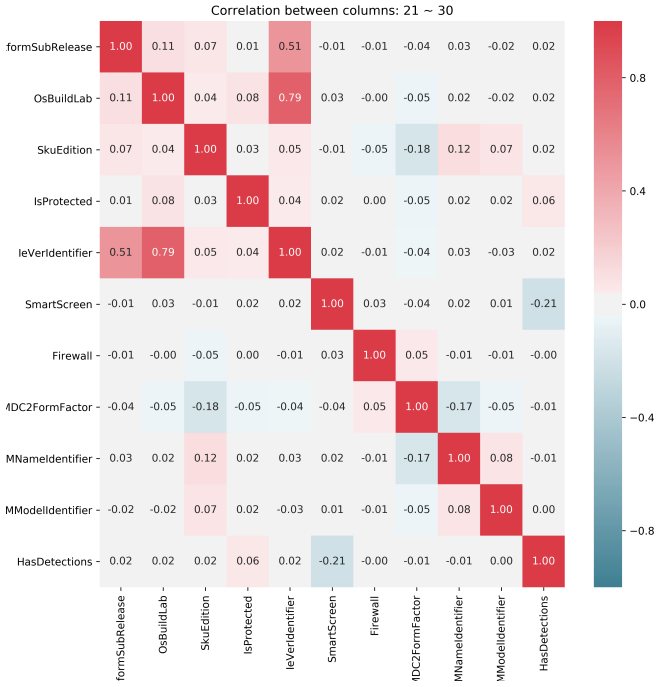

Figure 5: Correlation Between Features 21 Through 30

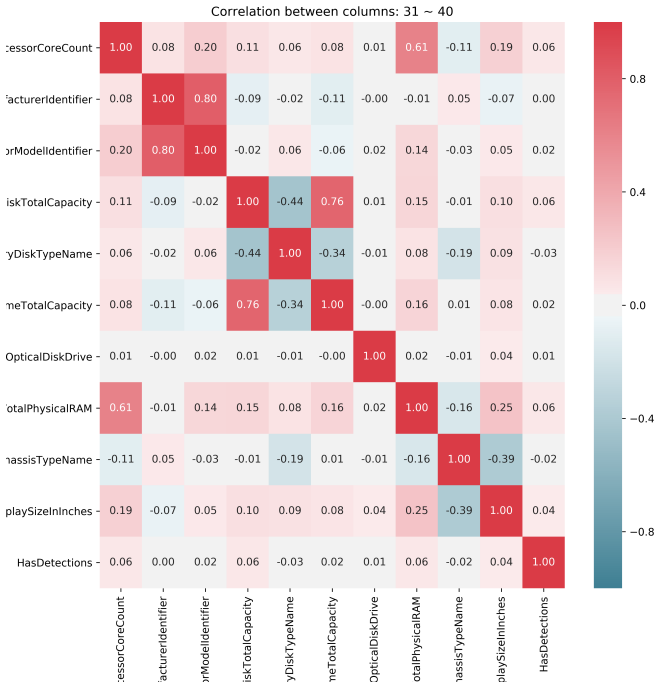

Figure 6: Correlation Between Features 31 Through 40

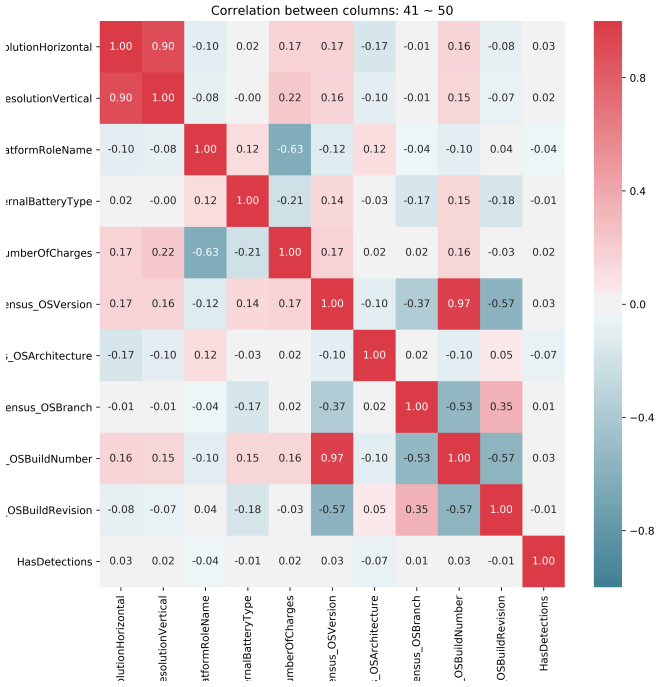

Figure 7: Correlation Between Features 41 Through 50

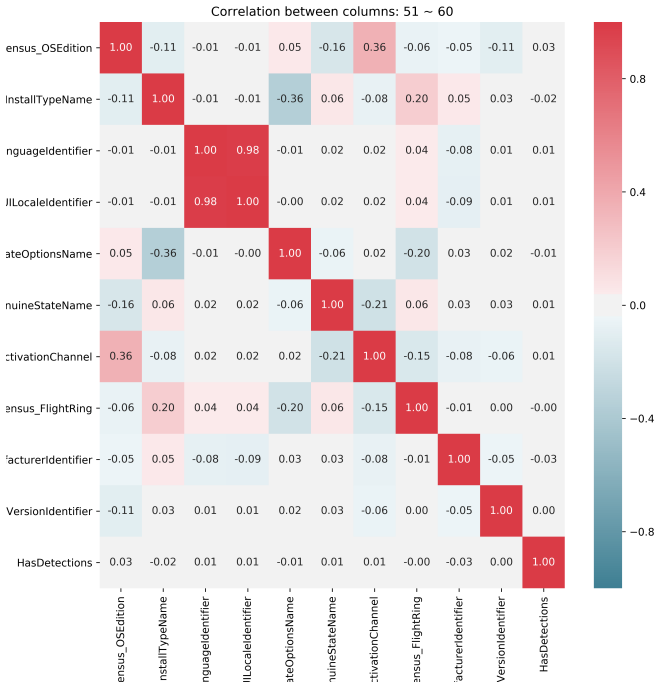

Figure 8: Correlation Between Features 51 Through 60

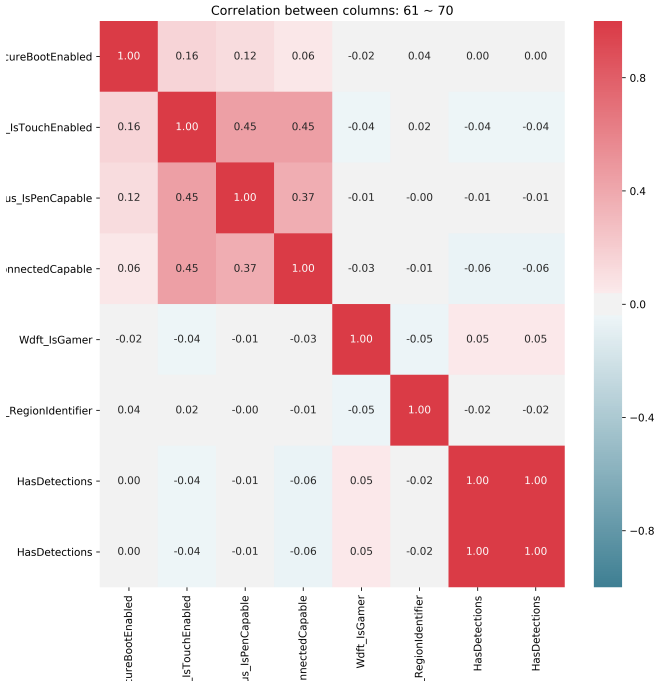

Figure 9: Correlation Between Features 61 Through 68

We are only left with 66 features to begin our analysis. Had it been such that we begin our analysis without any feature engineering, it would be such that the dataset will create more noise as the dataset itself is already high dimensional. Take for example, if we were to create decision trees with 83 features, we will be fitting even the data that are irrelevant to the decision making process. This might lead to overfitting and create noise. So it is important that we remove irrelevant features before starting our analysis.

5 Methods

In this project, we are trying to do a binary prediction on whether a computer will be hit with a virus or not. However, the data provides us with 82 features to work with. Utilizing the features without any modifications would be very difficult, since the data is noisy with missing data points and a large number of features which could lead to overfitting. So, a solution to this is use some feature engineering techniques and reduce the number of variables and clean up any missing data.

5.1 Feature Engineering: Purification

For the first part of the feature engineering, we remove features whose values are nearly all missing. Our removal threshold was at 99%. so if the feature has over 99% of its values missing, then we remove the feature. This removal will cause only a small loss of information, (for which the feature wasn't missing), but helps to reduce our models overall complexity.

Next, we looked to remove highly skewed categorical variables. We again used a threshold of 99%. So, if a categorical feature was the same category 99% of the time, we'd remove that feature. The reasoning is that if nearly all data points belong to a particular category then we can assume all data points are of that category to reduce the model complexity with only a small loss of information.

Next, we looked to solve the remaining missing data problem via imputation. For this portion, we opt to fill in numerical data with the median and categorical data with the mode. We cannot simply remove all the rows with a missing feature because this would account for removing 99.03% of the data. Additionally, since the dataset is over 3GB it was computationally sound to do a simple imputation on our data.

5.2 Feature Engineering: Correlation

Lastly, we found features that were highly correlated with other features. By creating a correlation table and heatmap, we were able to indicate which variables should be removed. However, when it came to two variables being highly correlated with one another, we chose to remove the data point with the least information. For example, the features EngineVersion and AvSigVersion are highly correlated. However, AvSigVersion contains more information than EngineVersion based on the meaning and the number of categories that AvSigVersion has. Thus, we opt to remove EngineVersion.

5.3 Modeling: Decision Trees

Decision trees are strong predictors for large data sets with possibly redundant variables (Hastie, Trevor). Since our data set consists of 8 million rows and 82 features, decision trees seem fit for use. We chose to use information based decision trees. These trees are deterministic in nature as they create node splits based on which node split would yield the most information (information gain) across the data its working with. Information Gain (IG) is defined as:

$$IG(T, a) = Entropy(T) - Entropy(T \mid a), \tag{1}$$

where T represents the target variable and a represents the value of the target variable. Informally, we see that IG is high if the resulting entropy of splitting on T is very low, indicating a large increase of information.

With our decision tree, we have to tune on two main parameters: max feature length and max tree length. For the first we used a max feature length of $sqrt(\#ofFeatures)$ and $log_2(\#ofFeatures)$, which are known values that perform well with decision trees. Then for the max tree length we do a linear search across values in $[0.001, 1000]$. using cross-validation.

5.4 Modeling: Logistic Regression

Since we are predicting on a binary variable, linear models will not work here. So we look to generalized linear models, specifically logistic regression, to model our data. Logistic regression is optimal for predicting binomial data, as it finds $E[Y \mid X]$ where $Y \in \{0, 1\}$. Logistic regression utilizes a logit function to map the unit interval to the real line, and uses the inverse logit function (expit function) to map the real line to the unit interval. Where the logit function defined as:

$$logit(x) = log\left(\frac{x}{1-x}\right), \tag{2}$$

and the expit function is defined as:

$$expit(x) = logit^{-1}(x) = \frac{e^x}{1 + e^x} \tag{3}$$

For our logistic regression models, we opted to separately use $L1$ and $L2$ regularization and perform a linear search on the regularization constant, (typically λ), using cross-validation.

Logistic regression with $L2$ regularization is equivalent to the optimization problem:

$$\min_{\theta} \sum_{i=1}^{M} -log\ p\left(y^{(i)} \mid x^{(i)}; \theta\right), \text{ subject to } \sum_{j} \theta_j^2 \leq C. \tag{4}$$

Here we expect $L1$, also known as lasso regularization, to work well since $L1$ encourages sparsity in the learned parameters. This fairs well with our data as we have many features and it will simplify our model making it more generalized (Hastie, T). Logistic regression with $L1$ regularization equivalent to the optimization problem:

$$\min_{\theta} \sum_{i=1}^{M} -log\ p\left(y^{(i)} \mid x^{(i)}; \theta\right), \text{ subject to } ||\theta||_1 \leq C. \tag{5}$$

6 Theory

6.1 Skewness

Skewness is the measure of asymmetry of the probability distribution of a real-valued random variable around its mean. The skewness can be negative, positive or undefined. A positive skew means that the tail is on the right and more data can be found on the left. On the other hand, negative skew means that the tail is on the left and more data can be found on the right.When it comes to categorical data, skewness will tell you if categorical variable is mostly the same values.

6.2 Data Correlation

A way to understand the relationship between different variables and features in the dataset. Correlation can help us in gaining insights such as the following:

- One of multiple features depend on another feature or lead to another feature.
- One or more features are associated with other features

A positive correlation value means that either feature A increases with feature B or decreases with feature B (Linear relationship between features). On the other hand, a negative, correlation value means that if feature A increases, then feature B decreases and vice-versa. No correlation (value of 0) means that the features are not related.

6.3 Linear Model and Nonlinear Model

The Key properties of a linear model are such that $E(Y \mid X) = \beta^{'} X$ and $Var(Y \mid X) \propto I$. In some cases where these conditions are not satisfied, we transform Y such that the linear model assumptions are approximately satisfied. It is difficult to find a transformation that simultaneously linearizes the mean and gives a constant value of variance. If Y is in a restricted domain, i.e. $Y = 0$ or 1, parametrizing $E(Y \mid X)$as a linear function of X violates the domain restriction.

6.4 Generalized Linear Model (GLM)

Generalized Linear Models are a class of nonlinear regression models that can be used in certain cases where linear models are not appropriate. Logistic regression is one such of these models under this class.

6.5 Logistic Regression

Logistic Regression is a specific type of GLM. Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous(binary, when $Y = 0$ or 1). Similarly to other regression analyses, logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, interval or ratio-level independent variables. The following is the probability mass function of Logistic Regression:
$P(Y = 1 \mid X = x) = \frac{exp(\beta' x)}{1+exp(')} = \frac{1}{1+exp(-\beta' x}$) which implies that $P(Y = 0 \mid X = x) = 1 - P(Y = 1 \mid X = x) = \frac{1}{1+exp(\beta' x)}$. where x_0=1, so β_0 is the intercept.

The following is the logit function: $logit(x) = log\left(\frac{x}{1-x}\right)$, , which maps the unit interval onto the real line.

The following is the inverse logit function (expit function): $expit(x) = logit^{-1}(x) = \frac{e^x}{1+e^x}$, which maps the real line onto the unit interval. In logistic regression, the logit function is used to map the linear predictor $\beta' X$ to a probability.

6.6 Conditional Log Odds

The linear predictor in logistic regression is the conditional log odds, which is the following: $log\left(\frac{[P(Y=1|X)}{P(Y=0|X)]}\right) = \beta' X$
One way to interpret logistic regression model is that a one unit increase in X_j results in a change of β_j in the conditional log odds. Or, a unit increase in X_j results in a multiplicative change of $exp(\beta_j)$ in the conditional odds.

6.7 Random Forest Trees

Random forest is a supervised learning algorithm. It builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is, that it can be used for both classification and regression problems. A great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Random forest is considered as a compare easy to use algorithm, because it's default hyperparameters often produce a good prediction result. Also, overfitting is not a big issue for random forest classifier. That's because if there are enough trees in the forest, the classifier won't overfit the model.

6.8 Decision Tree

The random forest decision tree that we are using are the decision (information) based trees. While these trees are being trained on the data, will create nodes based on splits that will maximize the Informatoin Gain.

A decision tree is a decision support tool that uses a tree-like model of decision and possible consequences which include chance event outcomes, resource costs and utility. In a decision tree each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values of competing alternatives calculated.

6.9 Feature Selection/Engineering

The process of selecting the features that can make the predicted variable more accurate or eliminating those features that are irrelevant and can decrease the model accuracy and quality. These incorporates a wide array of topics such as imputation, data skipping, and creating modified algorithms for learning. Some of the well known feature selectors are lasso and ridge regression. Additionally a decision tree could be built and features could be selected based on the nodes the tree used.

6.10 L1 Regularization (Lasso Regularization)

A regression model that uses L1 Regularization is Lasso Regression. The outcome variables are y_i for $i = 1, 2, ..., n$ and features $x_{ij}, j = 1, 2, ..., p$. We would like to minimize the following:

$$\sum_{i}^{n} \left(y_i - \sum_{j} x_{ij}\beta_j \right)^2 + \lambda \sum_{j}^{p} \mid \beta_j \mid \tag{6}$$

This is equivalent to minimizing sum of squares with the constraint $\sum \mid \beta_j \mid \leq s$. L1 Regularization is similar to the ridge regression, which has a constraint $\sum \beta_j^2 \leq t$. L1 Regularization is used for variable selection and to shrink the variables. On the other hand, ridge regression only shrinks the variables.

6.11 L2 Regularization (Ridge Regularization)

Ridge Regression adds "squared magnitude" of coefficient as penalty term to $\lambda \sum_j^p \beta_j^2$.Note$\lambda \sum_j^p \beta_j^2$ is also called the cost function. If $\lambda = 0$, then we will get back the Least Squares Linear Regression formula. On the other hand, if λ is very large, then it will add too much weight and lead to underfitting. The key difference between L1 Regularization and L2 Regularization is that L1 shrinks the less important feature's coefficient to zero, thereby removing some feature altogether. Hence, this works well when in case there are huge number of features.

6.12 Cross Validation

Cross Validation aims to provide stronger guarantee of generality for a given statistical or machine learning model. In practice, this is often done by partitioning data into training, validation, and testing sets. The training data is used to fit the model, while the validation set is used to tune additional parameters of the model called hyperparameters. The test set is used to evaluate the final model, and no decisions regarding the parameters of the model should be made on the basis of performance on the test set. Cross validation refers specifically to the training and validation sets, however, in cases in which the data are more scarce, k-fold cross validation may be used. K-fold cross validation splits data into K-partitions, and k different models are fit to the data.

6.13 Entropy

Entropy is the average rate at which information is produced by a stochastic source of data. The measure of information entropy associated with each possible data value is the negative log of the probability mass function: $S = -\sum_i P_i log(P_i)$ When the data source provides probability value that is low, i.e. low probability of an event occurring, the event carries more information than when the source data produces high-probability value.

6.14 Information Gain (Decision Trees)

Information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain, i.e. the most heterogeneous branches.

6.15 Imputation

Imputation is the process of filling in missing values with substituted values. Typically these values are either predicted for using the other features, or they are filled in. In the case that the missing value is filled in, if the variable is categorical it is typically filled in with the mode of the feature variable. If the variable is numerical it is either filled in with the median or the mean.

6.16 Decision Tree Feature Importance

The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance. This is computed by the difference in impurity times the probability crossing that node in the decision path. The probability is computed by the number of samples hit that node divided by the total random sample.

6.17 Classification Trees

Two-class Classification: When observations are classified into two or more classes, coded by a response variable Y taking values from $1, 2, ..., K$. There is a feature vector $X = (X_1, X_2, ..., X_p)$, and we hope to build a classification

rule $C(X)$ to assign a class label to an individual with feature X. Sample pairs are denoted as (y_i, x_i) where $i = 1, ..., N$. It is important to note that $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$as vectors.

Classification trees are represented by a series of binary splits. Each internal node represents a value query on one of the variables, i.e. "Is $X_3 > 0.4$". If the answer is "yes", move to the right, else go to the left.

7 Models

We test two types of models: decision tree classifiers and logistic regression models. We will run these models on the dataset with and without the pre-processing being used to demonstrate the effectiveness of our pre-processing in the beginning. Additionally, we will use cross-validation to hyper-tune our models and pick out the best model.

7.1 Pre-processing the Data

We first set up our training, validation and testing sets by taking 100,000 data points and pre-processing and shuffling the data. Then we partitioned the data into sizes 60000, 20000 and 20000 for training, validation and testing, respectively. After pre-processing our data was left with 66 features.

7.2 Decision Tree Classifier

Once the data was set up we began training models over two hyper-parameters: tree depth and max features per tree. For the max features per tree, we used the set of values $\{\sqrt{(p)}, log_2(p)\}$, where p is the total number of features after pre-processing. For tree depth we iterated over the valeus $\{5, 10, 15, 20, 25, 30, 35, 40\}$.

For the decision tree with max features $= log_2(p)$ we see in figure 10 that after performing cross-validation its peaks with its highest validation accuracy at depth 9. Specifically it has a training accuracy of 0.633 and validation accuracy of 0.607.

As for the decision tree with max features $= \sqrt{(p)}$ we see in figure 11 that after performing cross-validation its peaks with its highest validation accuracy at depth 8. Specifically it has a training accuracy of 0.634 and validation accuracy of 0.610.

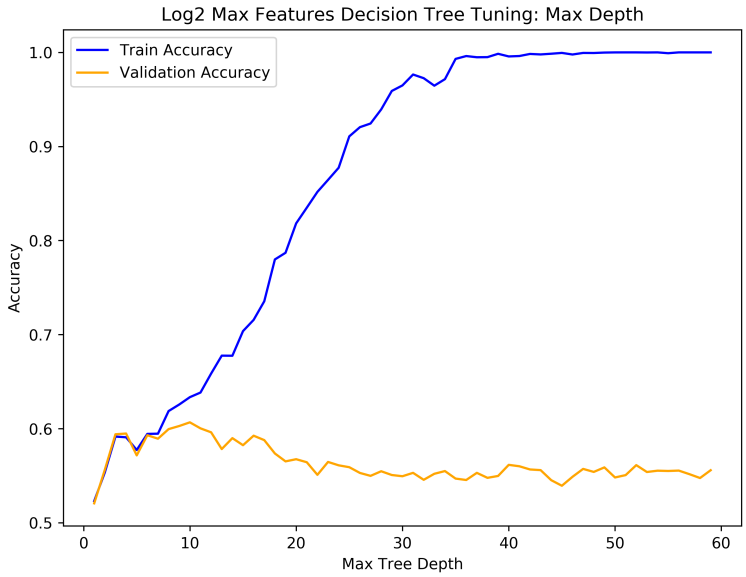

Figure 10: Log2 Decision Tree: Iterative Accuracy Over Max Depth

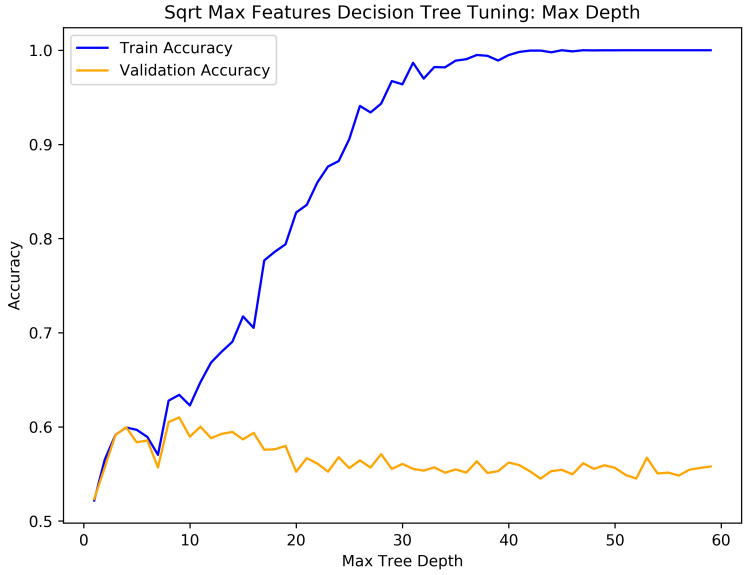

Figure 11: Sqrt Decision Tree: Iterative Accuracy Over Max Depth

	Decision Tree (Sqrt Features, Depth = 8)	Decision Tree (Log Features, Depth = 9)
Training Accuracy	0.626	0.6045
Validation Accuracy	0.608	0.593

Table 5: Training, Validation and Test Errors of the Best Decision Tree Models With Max Features: Sqrt vs Log

7.3 Logistic Regression

Now we move onto the logistic regression model. For this we chose to go with logistic regression since its the optimal for choosing our parameter. For hyper-tuning parameters, we used cross validation over the regularization constant $C = \frac{1}{\lambda}$ seen in equations 4 and 5. We did this for both a lasso logistic regression model and a ridge logistic regression model. In figure 12, we see the accuracy difference as we modify our regularization constant values and pick out the best model according to the highest validation accuracy.

From this we get that for the lasso logistic regression model, it is best with $C = 100$ with training accuracy of 0.621 and validation accuracy 0.615. As for the ridge logistic regression model, we saw very little improvement with accuracy as we modified the regularization constant. The model was best with $C = 1$. Obtaining a training accuracy of 0.531 and validation accuracy of 0.527. This supports our position that L1 regularization would perform better due to the large number of categorical variables and the overall feature size. Since L1 encourage sparsity we can simply our model, allowing it to become more generalized.

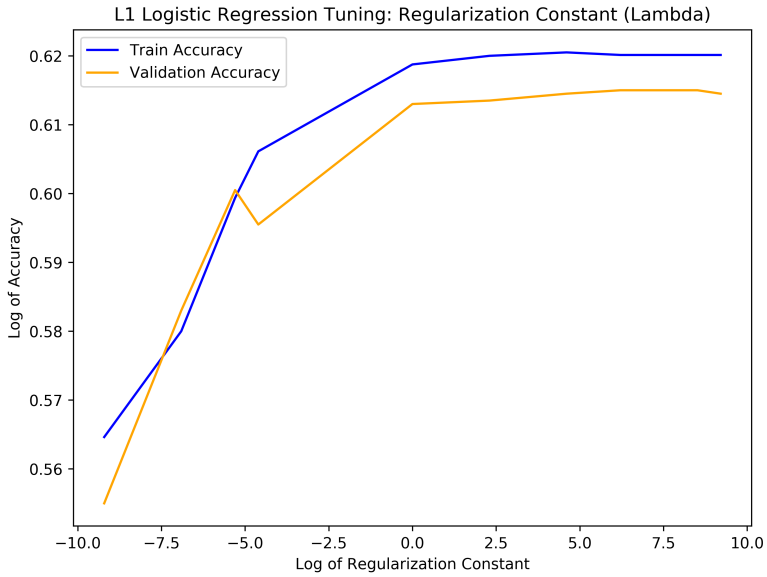

Figure 12: Lasso Logistic Regression: Hyper-tuning on Regularization Constant

	Logistic Ridge Regression	**Logistic Lasso Regression**
Training Accuracy	0.531	0.621
Validation Accuracy	0.527	0.615

Table 6: Training, Validation and Test Errors of the Logistic Regression Models

8 Conclusion

8.1 Investigative Conclusions

When looking at the natural defenses on the computer, we found several columns or features that yielded a significant influence in how we would predict malware results. Specifically we found that these factors led to strong predictions in malware: The version number of the anti-virus program downloaded in the system. Firewall wasn't included, but its closer counterpart of smart-screen was also a significant predictor for malware, where if they had this off, they would be more susceptible to infection. Finally, having a "secure boot" option on was important for developing a more

protected systems. This makes sense as some malware inhibit the ability for the computer to turn on securely. We would recommend microsoft to start defaulting secure boot for the computer.

The other significant predictor for malware was having an updated OS. We found that a negative relationship with malware infection and updated OS, where the older the update, the higher chance they were updated. Version wasn't specifically important for OS either, it was just as important for antivirus software, yielding the same pattern of results. We would recommend future users and microsoft to push more for their updates and hen we analyzed the columns related to quality, we found multiple significant properties that helped predict malware infection. Disk capacity, computer inch size/display, and OS architecture were significant in predicting for malware. For disk capacity, the larger the disk size, the higher the chance for malware. Higher computer size led to less We would recommend to keep in mind higher value computers, but know that user-error is just as an important factor for getting malware.

Feature Name	**Importance**
SmartScreen	0.121
'AVProductsInstalled'	0.111
'EngineVersion'	0.098
'Census_PrimaryDiskTotalCapacity'	0.065
'GeoNameIdentifier'	0.048
'Census_ProcessorModelIdentifier'	0.048
'CityIdentifier'	0.033

Table 7: Feature Importance for Decision Tree (Max Features = Sqrt, Max Depth = 8

Finally, we didn't plan for this in the original analyses, but post hoc tests found that geographic location and city were significant predictors for our analysis. From a post-analysis, this would make sense and computer literacy may not be standardized in multiple areas in the world. Income and world-wealth status could greatly affect the information that could be taught to people who use the internet. People with lower internet literacy could be more easily influenced or tricked by hackers or malware creators to download their software and be infected. We would recommend microsoft to provide more teaching tools that are easy to understand to less informed populations if they send out computers to those specific areas.

There are many limitations to these conclusions. First, malware was identified by the systems own virus systems, compared to analyzing the data from many popular antivirus software. Microsoft doesn't have a full directory or super

active self-defense system for their computer, but commercial ones are constantly being updated and analyzing new threats, which means that some of the computers may have had malware than originally expected. Then the data only says if they have malware or not, and it doesn't say how many malware there is. This leads to only having a binary input of yes or no, compared to having a full description of information or range that could help us determine if one property could yield larger results.

The variables that did not yield a significant predictor for malware were browser type or if they were a gamer.

8.2 Model Accuracy and Future Works

From the best decision tree and logistic regression models we get a testing accuracy of 0.601 and 0.606 respectively. Although these are relatively low accuracies to the .5 accuracy of a completely naive model that randomly guesses virus vs not virus, this was expected due to the nature of the dataset. In comparison to other work seen in Kaggle, this fairs decently well. Where the best scores reach towards 0.7 accuracy (Kaggle).

	Decision Tree (Sqrt Features, Max Depth $= 8$**)**	**Logistic Lasso Regression (**$C = 100$**)**
Test Accuracy	0.601	0.606

Table 8: Testing Error of the Best Models

However, our models faced many computational limitations and there is additional work that can be done to improve the models. Additional models such as Light GBM: A lightweight gradient boosted decision tree model are more complicated should be investigated and additional work, neural networks, and time-series analysis work could be relevant and done. In addition to utilizing different models, finding better ways to fill in missing data by creating a modified learner could improve the model as this would be better than filling in data with the mean, median or mode.

References

[1] Hastie, T., Tibshirani, R., Friedman, J. (2001). The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc..

[2] Scalable, behavior-based malware clustering. Bayer, Ulrich, et al. NDSS. Vol. 9. 2009.

[3] 8 Standard Computer Components and What They Do. Houk Consulting, Https://Www.houkconsulting.comHouk Consulting, 2 Mar. 2017, www.houkconsulting.com/2017/03/standard-computer-components/.

[4] Common Malware Types: Cybersecurity 101 Lord, Nate. https://www. veracode. com/blog/2012/10/common-malware-types-cybersecurity-101 (2012).

[5] United States Patent Ackroyd, Robert John. United States Patent, 11 Sept. 2007, patentimages.storage.googleapis.com/06/13/f0/a7ef10795b3674/US7269851B2.pdf.

[6] Malware analysis via hardware virtualisation extensions Dinaburg, Artem, et al. technical report, georgia institute of technology. 2008. 1-3.

[7] The Ghost In The Browser Provos, Niels, et al. First Workshop on Hot Topics in Understanding Botnets (HotBots). 2007.

[8] Learning and classification of malware behavior Rieck, Konrad, et al. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Berlin, Heidelberg, 2008.

[9] Testing malware detectors Christodorescu, Mihai, and Somesh Jha. ACM SIGSOFT Software Engineering Notes 29.4 (2004): 34-44.

[10] Microsoft Malware Prediction Leaderboard Kaggle. https://www.kaggle.com/c/microsoft-malware-prediction/leaderboard

[11] Boosting Hastie, Trevor https://math189.edublogs.org/files/2019/03/10778-boost-tjjkey-1pzf2h7.pdf