Microsoft Malware Detection

Joshua Castro, Kevin Elkin, Sneheil Saxena, Yuka Sakamaki, Tony Wu

INTRODUCTION

The malware industry is a well-organized, well-funded market dedicated to evading traditional security measures. Defined by AVG Security, malware is a shortened version of the phrase "malicious software". [Lemonnier]. Malware is an umbrella term of numerous pieces of software designed with the intent of harming data, products, and consumers. Different types of malicious software include the likes of virus, Trojans, Spyware, Worms, Ransomware, and more. Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways.

Different types of malware perform different actions on the computer. For instance, viruses usually infect consumers devices by attaching itself to a clean, usable file, spreading without bound and eventually deleting or corrupting the host's files. This would be different from the actions of Trojans, which enters the host under a disguise as trustable software, then acting discreetly by creating backdoors in the host's security systems in order to let in other types of malware.

Microsoft, being a major contributor in the advancements of technology, is deeply invested in improving the security of their products. As a method in improving their technology, Microsoft challenged the data science community with the task of developing techniques to predict whether or not a machine is being targeted by malware. By providing an unprecedented dataset regarding malware and Microsoft's products, data scientists are encouraged to work on finding techniques to effectively predict malware occurrences.

In this project, our team will analyze a data set from the Microsoft Malware Competition Kaggle Competition to answer specific questions regarding the malware detection status of machines following certain characteristics and provide a predictive model regarding the malware detection status of certain features from the dataset.

DATA

Our data is from a Kaggle competition entitled Microsoft Malware Competition [Microsoft Malware]. The goal of this competition is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine. The telemetry data containing these properties and the machine infections was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender.

We are given two data files, *train.csv* and *test.csv*. We shall use the *train.csv* file given to predict whether or not each machine may be targeted by malware. In the dataset, each row corresponds to a machine with a unique identifier, the *MachineIdentifier* observation. Once we conduct our investigations, we will be able to verify whether our predictions are correct as the *HasDetections* observation denotes whether or not the machine is in danger of malware infection.

From the *train.csv* file, we plan on focusing our predictive model study on the information from the *HasDetections*, *IsSxsPassiveMode*, *AVProductsInstalled*, *AVProductsEnabled*, *Platform*, *SkuEdition*, *IsProtected*, *FirewallI*, *Wdft_IsGamer* columns. Our reasoning for using the described columns is as follows:

- *HasDetections*
    - *HasDetections* allows us to validate whether or not our algorithm can be used to correctly predict if a machine is being targeted by malware. We will use this column to validate our work.
- *IsSxsPassiveMode*
    - *IsSxsPassiveMode* can be used to signal whether or not the machine is using the *Microsoft Defender* antivirus program actively or replacing this program with a third party antivirus program. We may be able to use this observation to check whether machines actively using Microsoft Defender are better protected from the threat of malware.
- *AVProductsInstalled* and *AVProductsEnabled*
    - *AVProductsInstalled* is used to describe the number of antivirus programs the machine has installed, whereas *AVProductsEnabled* is used to describe the number of antivirus programs the machine has enabled. We could use these observations to see if machines with more antivirus programs installed and enabled are better protected from the threat of malware.
- *Platform*
    - *Platform* describes the operating system of the machine. This could be used to see if machines with older operating systems could be more susceptible to malware infection.
- *SkuEdition*
    - *SkuEdition* denotes the product type of the machine. It is found that the machine could fall under the *Pro*, *Invalid*, *Education*, *Enterprise*, *Enterprise LTSB*, *Cloud*, or *Server* product types. With this, we could find which types of machines are more likely to be infected by malware.
- *IsProtected* and *Firewall*

    - *IsProtected* is used to signify whether a machine is using an active at least one active and up to date antivirus program. *Firewall* is used to denote whether or not the machine has firewall enabled. Similar to the *AVProductsInstalled* and *AVProductsEnabled* observations, we would like to see if having these programs installed will help protect the machine from antivirus.
- *Wdft_IsGamer*
    - *Wdft_IsGamer* indicates whether the machine is used to play video games. We wish to investigate if machines used to play video games can be more susceptible to malware

Using the described observations, we would like to see which combinations of characteristics from the list described above are more likely to be infected with malware, and possibly use such combinations to further predict the characteristics of machines being targeted by malware.

BACKGROUND

Malicious software is not a recent machine-complication-phenomena. The earliest recordings of malware include developers experimenting with the capabilities of the earliest computers. Such situations mostly involved developers experimenting with self-writing programs, as they used ideas from John von Neumann's "Theory and Organization of Complicated Automata" to model computer programs which reproduced themselves. Specifically, the developers at Bell Labs brought the ideas of von Neumann's paper to fruition in the 1950's as programmers developed software "organisms" used to compete for control of a computer in an activity known as the "Core Wars." [Rankin] These self-replicating programs and software "organisms" continued to evolve into the viruses we know them as today, as what began as an innocent game has transformed into a cybercrime giant with projected damages on the cyber industry amounting to $6 trillion anually by 2021. [CISION]

As malware continues to evolve, Microsoft maintains its status as an elite technological company by creating more efficient machine learning algorithms to better their client and cloud protection systems. Through their flagship protective software, Windows Defender Antivirus, Microsoft uses data science, machine learning, automation, and behavioural analysis techniques in order to better identify when a consumer's device is being targeted by malware. [Microsoft] Microsoft has adapted a protective software combining their data regarding current and past malware to block known viruses and machine learning techniques with their cloud protection in order to prevent future viruses from intruding on consumer privacy. This new technique allows for machines to effectively assess whether or not files contain known viruses or decipher if files may contain unique signals resembling the actions of past viruses implemented in updated ways to mask their impending damages. [Microsoft]

In a study conducted by a representative of the Product Security Incident Response Team of Adobe Systems, it has been found just as effective to detect malware infection from a "minimum feature set." [Raman] Using machine learning classifiers, this study compared their findings on malware infection using a minimum feature set, and found their false-positive rate and true-positive rate to be 5.68% and 98.56%, respectively. In other papers on using machine learning classifiers to detect malware but with larger features sets, this false-positive rate and true-positive rate is better than studies conducted with size-42 feature sets and size-89 feature sets. From this, we are inspired to use machine learning techniques with a size-8 feature set in order to predict, from the dataset, which machines are most likely to be targeted by malware.

SCENARIOS AND HYPOTHESIS

We wish to further investigate which characteristics and uses of Microsoft machines are more susceptible to malware infection. We formulate the following scenarios:

- *Scenario 1: Are gamer devices more susceptible to malware than other devices?*
  - We ask this since a person who uses their computer to partake in video games may download more files than non-gaming consumers, which could possibly make the machines of gamers more likely of being targeted by malicious software.
- *Scenario 2: Are machines with antivirus products installed and enabled better protected from malware than other devices?*
  - We ask this because we want to see if machines with antivirus products will truly better protect their machines, and if consumers find more protection for their devices with a higher number of installed antivirus programs.
- *Scenario 3: Can we provide a predictive model to indicate which of the listed features is the most significant in determining a machine's malware detection status?*
  - At this point, we have formed questions regarding the characteristics of machines in our dataset. However, we would like to use this information to provide some sort of predictive model, allowing consumers to understand which of the listed features is significant in determining the malware detection status of their machine.

With these scenarios, we formulate the following hypotheses:

1. *Are gamer devices more susceptible to malware than other devices?*

   Our hypothesis for Scenario 1 is machines used to partake in video games are more susceptible to malware infection.

2. *Are machines with antivirus products installed and enabled better protected from malware than other devices?*

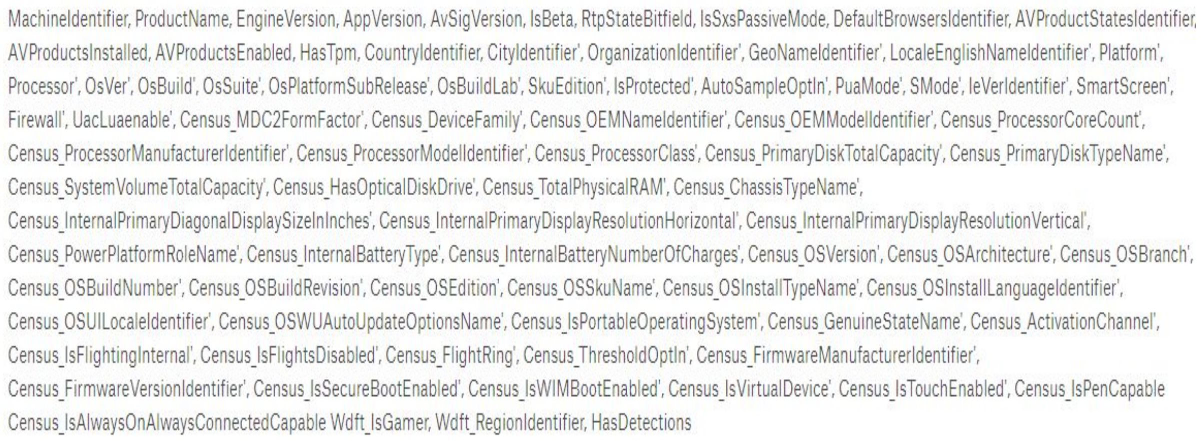
   Our hypothesis for Scenario 2 is machines with antivirus products installed and enabled are better protected from malware infection.

We refrain from formulating a hypothesis for our third scenario, as our findings for this scenario should provide enough information for readers to take away as is.

INVESTIGATIONS

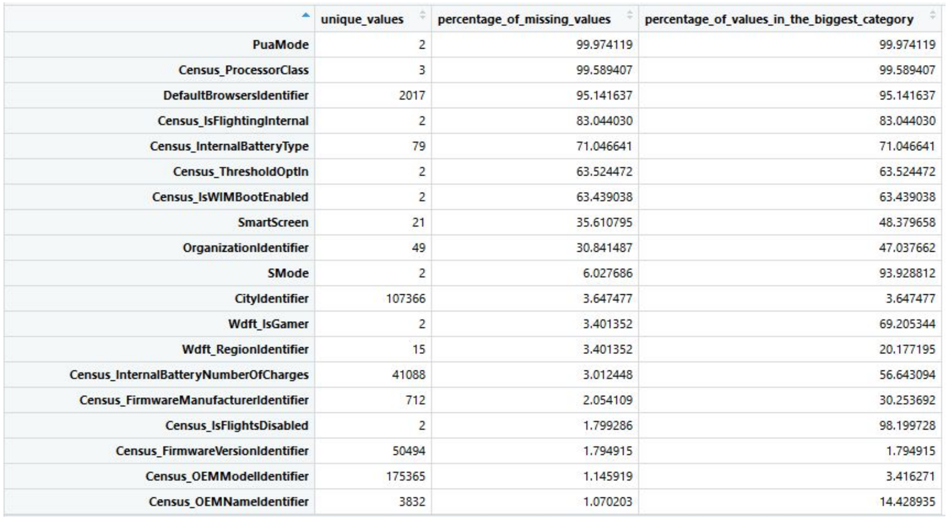
Data Preprocessing

Initially, we see there are 8,921,483 rows of data, each with 83 columns of features. Each data point is identified by a unique *MachineIdentifier* column with a list of different features. We notice that all the columns in the data set are categorical. Looking at the columns of features, we have:



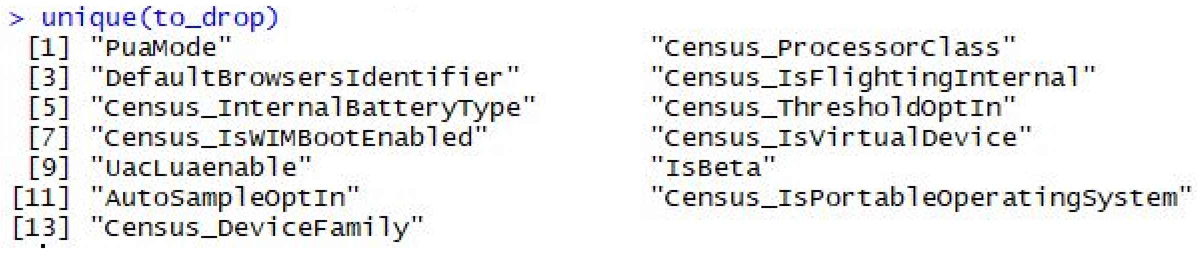
```
MachineIdentifier, ProductName, EngineVersion, AppVersion, AvSigVersion, IsBeta, RtpStateBitfield, IsSxsPassiveMode, DefaultBrowsersIdentifier, AVProductStatesIdentifier,
AVProductsInstalled, AVProductsEnabled, HasTpm, CountryIdentifier, CityIdentifier', OrganizationIdentifier', GeoNameIdentifier', LocaleEnglishNameIdentifier', Platform',
Processor', OsVer', OsBuild', OsSuite', OsPlatformSubRelease', OsBuildLab', SkuEdition', IsProtected', AutoSampleOptIn', PuaMode', SMode', IeVerIdentifier', SmartScreen',
Firewall', UacLuaenable', Census_MDC2FormFactor', Census_DeviceFamily', Census_OEMNameIdentifier', Census_OEMModelIdentifier', Census_ProcessorCoreCount',
Census_ProcessorManufacturerIdentifier', Census_ProcessorModelIdentifier', Census_ProcessorClass', Census_PrimaryDiskTotalCapacity', Census_PrimaryDiskTypeName',
Census_SystemVolumeTotalCapacity', Census_HasOpticalDiskDrive', Census_TotalPhysicalRAM', Census_ChassisTypeName',
Census_InternalPrimaryDiagonalDisplaySizeInInches', Census_InternalPrimaryDisplayResolutionHorizontal', Census_InternalPrimaryDisplayResolutionVertical',
Census_PowerPlatformRoleName', Census_InternalBatteryType', Census_InternalBatteryNumberOfCharges', Census_OSVersion', Census_OSArchitecture', Census_OSBranch',
Census_OSBuildNumber', Census_OSBuildRevision', Census_OSEdition', Census_OSSkuName', Census_OSInstallTypeName', Census_OSInstallLanguageIdentifier',
Census_OSUILocaleIdentifier', Census_OSWUAutoUpdateOptionsName', Census_IsPortableOperatingSystem', Census_GenuineStateName', Census_ActivationChannel',
Census_IsFlightingInternal', Census_IsFlightsDisabled', Census_FlightRing', Census_ThresholdOptIn', Census_FirmwareManufacturerIdentifier',
Census_FirmwareVersionIdentifier', Census_IsSecureBootEnabled', Census_IsWIMBootEnabled', Census_IsVirtualDevice', Census_IsTouchEnabled', Census_IsPenCapable
Census_IsAlwaysOnAlwaysConnectedCapable Wdft_IsGamer, Wdft_RegionIdentifier, HasDetections
```

Before beginning our analysis, we will clean the data with mostly NULL value columns or if the column is too skewed, as those values will not tell us much about the data or if the machine has malware. Here, we calculate the percentage of NULL values and the percent of values of the biggest category:



	unique_values	percentage_of_missing_values	percentage_of_values_in_the_biggest_category
PuaMode	2	99.974119	99.974119
Census_ProcessorClass	3	99.589407	99.589407
DefaultBrowsersIdentifier	2017	95.141637	95.141637
Census_IsFlightingInternal	2	83.044030	83.044030
Census_InternalBatteryType	79	71.046641	71.046641
Census_ThresholdOptIn	2	63.524472	63.524472
Census_IsWIMBootEnabled	2	63.439038	63.439038
SmartScreen	21	35.610795	48.379658
OrganizationIdentifier	49	30.841487	47.037662
SMode	2	6.027686	93.928812
CityIdentifier	107366	3.647477	3.647477
Wdft_IsGamer	2	3.401352	69.205344
Wdft_RegionIdentifier	15	3.401352	20.177195
Census_InternalBatteryNumberOfCharges	41088	3.012448	56.643094
Census_FirmwareManufacturerIdentifier	712	2.054109	30.253692
Census_IsFlightsDisabled	2	1.799286	98.199728
Census_FirmwareVersionIdentifier	50494	1.794915	1.794915
Census_OEMModelIdentifier	175365	1.145919	3.416271
Census_OEMNameIdentifier	3832	1.070203	14.428935

We choose to drop values with more than 50% NULL values because we feel that there is not sufficient enough data to determine whether we can detect malware as we cannot use the column for a majority of the data. We also choose to drop values with more than 99% of values in the biggest category because we must note that we have a large number of samples, so we still have a lot of information of the data even if the column mainly has one value. Therefore, we see we are able to drop 13 columns, thus leaving us with 70 columns of features:

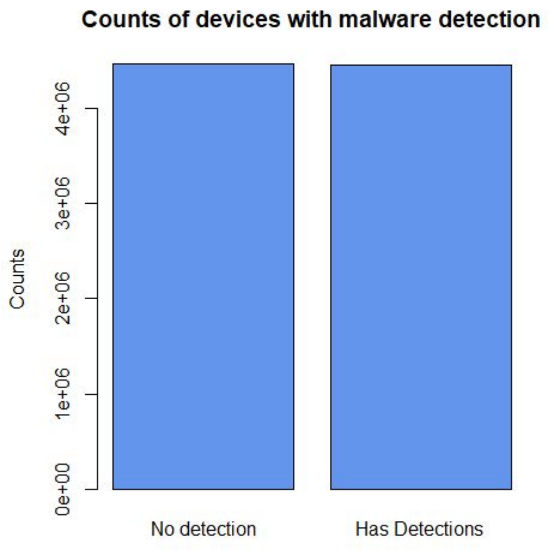


```
> unique(to_drop)
 [1] "PuaMode"                        "Census_ProcessorClass"
 [3] "DefaultBrowsersIdentifier"      "Census_IsFlightingInternal"
 [5] "Census_InternalBatteryType"     "Census_ThresholdOptIn"
 [7] "Census_IsWIMBootEnabled"        "Census_IsVirtualDevice"
 [9] "UacLuaenable"                   "IsBeta"
[11] "AutoSampleOptIn"                "Census_IsPortableOperatingSystem"
[13] "Census_DeviceFamily"
```

We see here that we are still able to use the columns we want to focus our study on. So, we will do our analysis using: *HasDetections*, *IsSxsPassiveMode*, *AVProductsInstalled*, *AVProductsEnabled*, *Platform*, *SkuEdition*, *IsProtected*, *FirewallI*, *Wdft_IsGamer* columns.

Observing the data set, we see that approximately 50% of the data have detection of malware, while the other 50% do not have any detection of malware (Fig. 1). This will be our baseline in our analysis whether a certain feature affects the rate of malware detection. An underlying assumption we are making is that each device is independent from each other which we will use throughout our analysis.

Fig. 1: A bar chart depicting the number of devices with and without malware in our dataset.



Scenario 1: Are gamer devices more susceptible to malware than other devices?

We want to know whether gamer devices are more susceptible to malware compared to other devices. We speculate that people with gamer devices are more likely to download more software that may lead to greater risk of malware, motivating our analysis.

We calculate that the proportion of gamer devices is approximately 0.273933, which is 2,443,889 devices (Fig. 2). We want to know if these devices are more likely to have malware compared to non-gamer devices. Thus, we will use these devices as our population.

We observe that the proportion of gamer devices with malware is 0.54189 (Fig. 3). Calculating a confidence interval of the proportion of gamer devices with malware, we get (0.54127, 0.54252) with a standard error of 0.0003187. We notice that this proportion is higher than our baseline of devices with malware. Also, we see that the standard error is very small, due to the large number of values in the dataset. Thus, we will further verify this proportion to see if gamer devices are more susceptible to malware.

Fig 2. A bar chart depicting the number of gamer devices in the data set

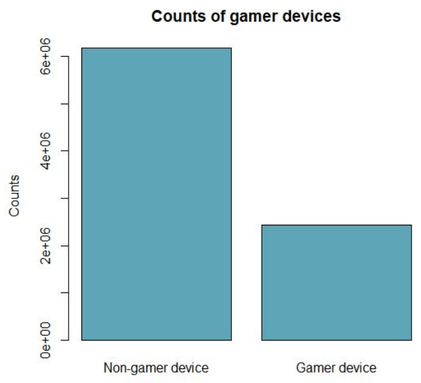
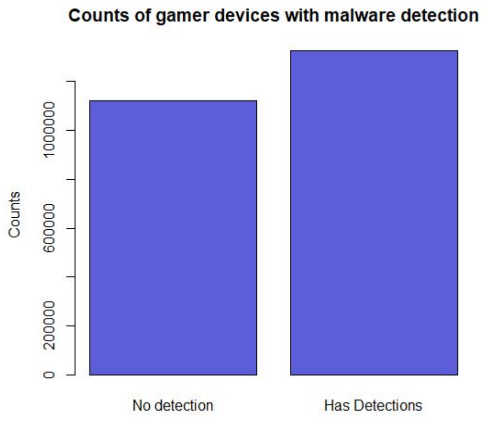


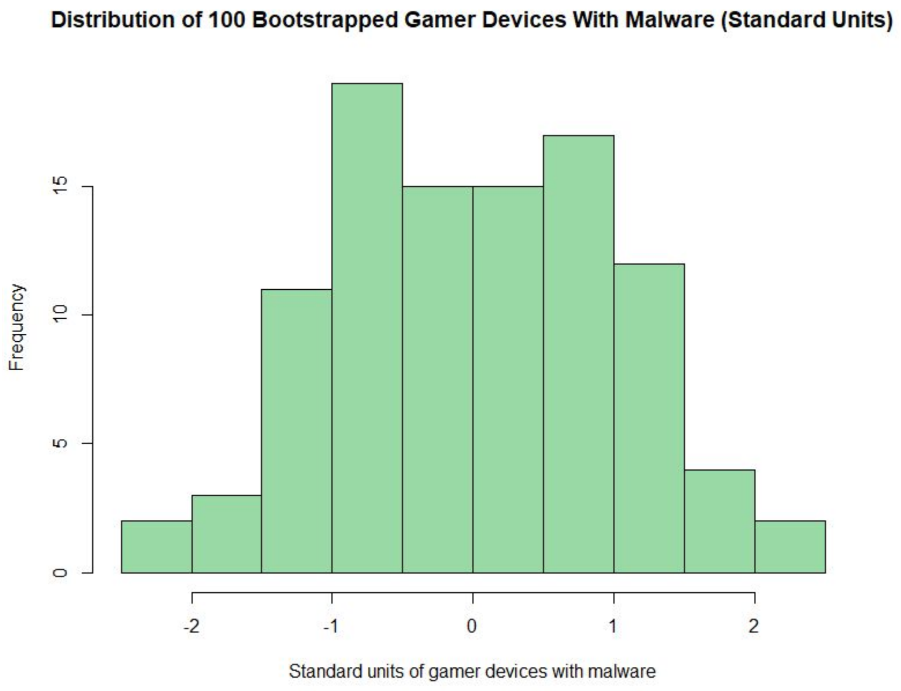
Fig. 3 A bar chart depicting the number of gamer devices with malware detection



We bootstrapped 100 gamer devices sampling with replacement of the population of gamer devices. In our bootstrap, we notice that many of the proportion of gamer devices with malware were around 54%. Plotting the bootstrapped proportions, we see that the histogram is fairly normal (Fig.4). The skewness and kurtosis of the bootstrapped sample are 0.003977 and 2.426148, respectively. Although skewness is close to zero, it is unclear if our bootstrapped sample follows a normal distribution. So, we will conduct a Kolmogorov-Smirfnov test against the normal distribution to test if our bootstrapped sample follows a normal distribution. We calculate that our test statistic D = 0.052432 and p-value = 0.9462. Against an $\alpha = 0.05$, we see our p-value is greater than the significance level 0.05, so we fail to reject that the bootstrapped sample follows a normal distribution. Thus, we will be able to use this information to further guide our analysis.

Taking a 95% confidence interval of the bootstrapped proportions, we observe the mean proportion of gamer devices is 0.541859, with an interval (0.541801, 0.541916) and standard error 0.00002900769. This is very similar to our observed sample confidence interval, giving us evidence that there is a higher likelihood of gamer devices having malware. Also, note that the standard error of the bootstrapped samples is smaller than the original data set's standard error by about a factor of 10, giving us more accurate results.

Fig. 4 A histogram of the proportion of 100 bootstrapped gamer devices with malware, in standard units.



We have the underlying assumptions to conduct a Student's t-test to compare if the proportion of gamer devices with malware differs from the baseline of 50 percent. We calculate our test statistic t = 131.45 and p-value = 2.2e-16. Since we see our p-value is less than a significance level of 0.05, we reject that the proportion of gamer devices with malware is equal to 50 percent. Thus, we have sufficient evidence that gamer devices are more susceptible to malware.

Scenario 2: Are machines with antivirus products installed and enabled better protected from malware than other devices?

Antivirus software are heavily emphasized in keeping devices safe from malware. We want to explore whether or not antivirus products actually protect devices from malware and if the number of antivirus software has an effect on whether or not a device has malware. We speculate that there should be less detection of malware with the number of antivirus products on a device.

Looking at Fig. 6, we see that a majority of devices only have one antivirus product installed. We observe that the proportion malware detection rate goes down as the number of antivirus products installed increases, but the count is very small to analyze. Also, note in Fig. 5 that there are more malware detections when one antivirus product is installed compared to the other counts.

Fig. 5 Bar plot of the number of devices with malware detected based on the amount of anti-virus products installed

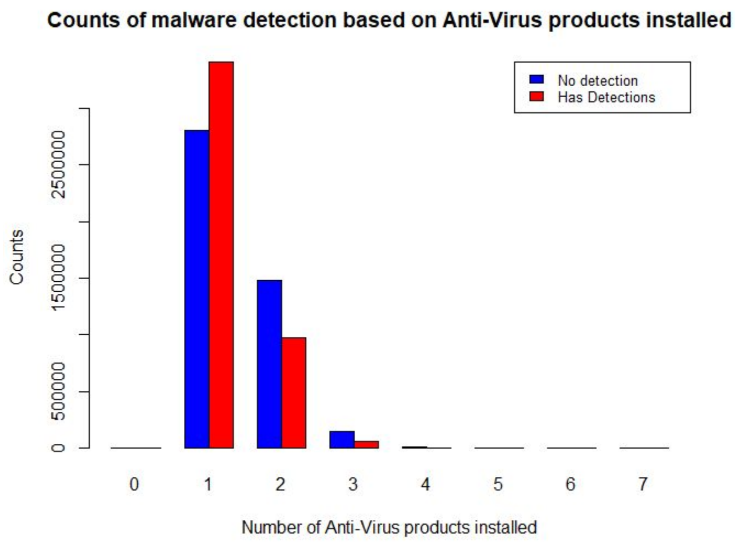
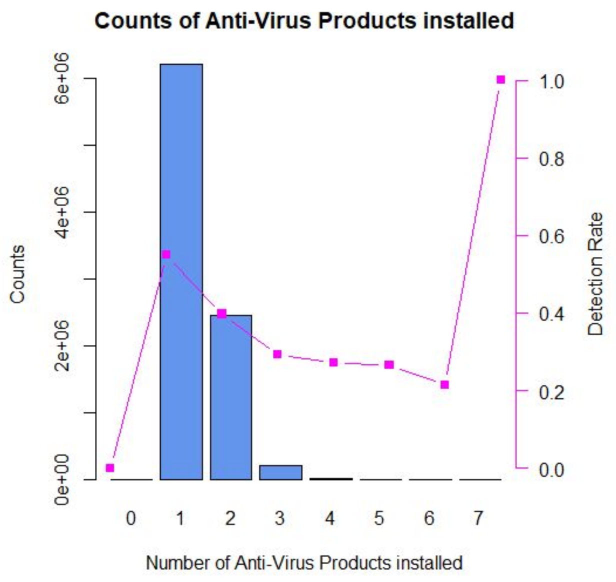


Fig. 6 Bar plot of the counts of the number of anti-virus products installed with the malware detection rate.



Taking a 95% confidence interval, we find that there is an average of 1.326779 antivirus products installed with interval (1.326435, 1.327122) and standard error 0.0001754309. So, we see that we would expect devices to have at least one antivirus product installed. In Table I, we see the counts of devices with antivirus products installed as well as the number of devices with malware detection.

Table I: Number of devices with and without malware detections according to the number of antivirus products installed.

Number of Antivirus Products Installed								
Count	0	1	2	3	4	5	6	7
No detection	1	2802815	1483012	147421	6386	346	22	0
Has Detections	0	3406078	975996	60682	2371	125	6	1
Total Count	1	6208893	2459008	208103	8757	471	28	1
Proportion with malware	0	0.5486	0.3969	0.2915	0.2707	0.2653	0.2142	1

We want to see if the proportion of malware detections vary. Visually, we clearly see that the proportion goes down as the number of antivirus products installed increases. Using a chi-square goodness of fit test, we test whether or not these proportions really do differ. Calculating the expected counts, we see that we cannot use counts of 0 or 7 because their expected counts are less than 5, so we will drop them for those column values. We calculate our test statistic to be $\chi^2 = 201258$, giving us a p-value of 0, giving us strong evidence to reject the hypothesis that the proportion of malware based on count of antivirus products installed do vary. Therefore, we will consider separate cases for each of the counts.

We have the underlying assumptions to conduct a Student's t-test to compare if the proportion of devices with malware based on count is less than the baseline of 50 percent. Our results shown in Table II describe a variety of different results. We see that if only one antivirus product is installed, there is actually an increased chance that device has malware. If a device has more than one antivirus product installed, however, we see our p-value is less than a significance level of 0.05, thus we have sufficient evidence that the proportion of devices with malware are lower than 50 percent, with more antivirus products installed.

Table II: Results of the Student's t-test on proportion malware based on count of antivirus products installed

Student's t-test Results						
Count	1	2	3	4	5	6
Proportion	0.548581	0.396906	0.291596	0.270755	0.265393	0.214286
Test Statistic	243.25	-330.43	-209.18	-48.276	-11.519	-3.6181
p-value	1	2.2e-16	2.2e-16	2.2e-16	2.2e-16	0.000602

Although users may install antivirus products on their computer, there may be a chance that it may not be in use or enabled. In Fig. 7 and Fig. 8, we see that a vast majority of users only have one antivirus product enabled, despite the fact the approximately three fourths of people have more than one antivirus product installed. So, we wish to explore this phenomenon that although people may have more antivirus products installed, it is likely they only have one antivirus product enabled.

Fig. 7 Bar plot of the number of devices with malware detected based on the amount of anti-virus products enabled

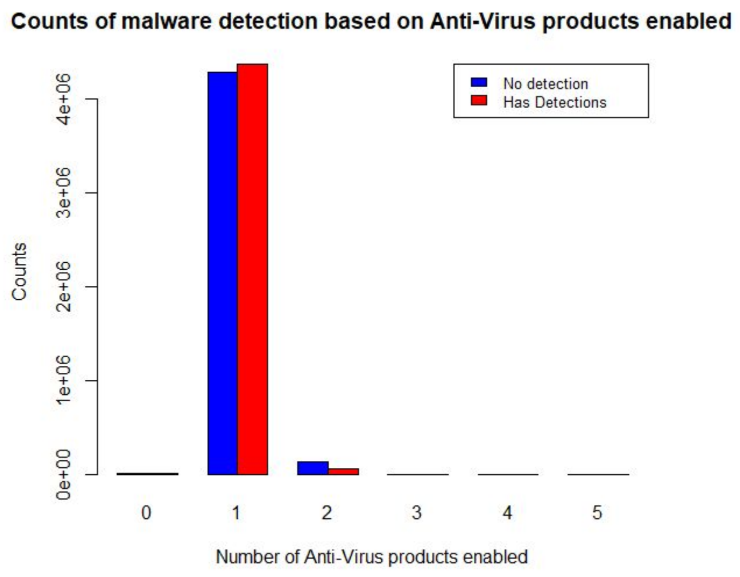
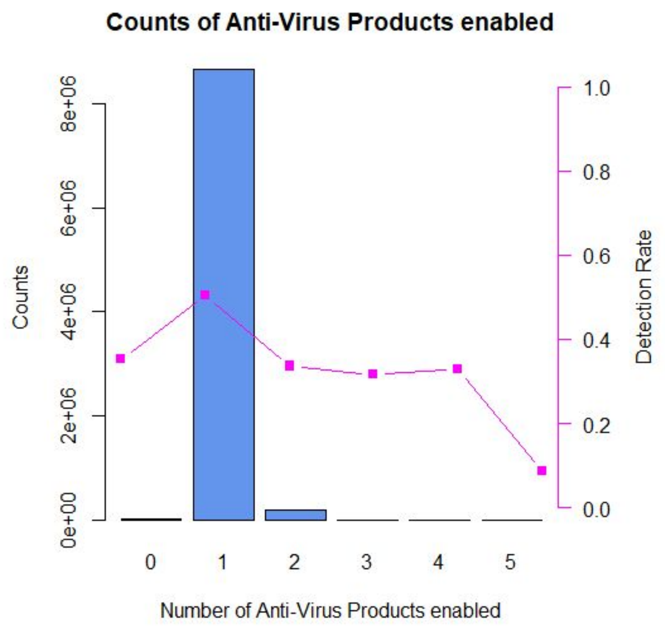


Fig. 8 Bar plot of the counts of the number of anti-virus products enabled with the malware detection rate.



We see in Table III, the rate of malware detection seems to go down as the number of antivirus products enabled increases. We want to know how effective it is to have more antivirus products enabled. Using a chi-square goodness of fit test, we find that our test statistic $\chi^2 = 25107$, giving us a p-value of 0, giving us strong evidence to reject the hypothesis that the proportion of malware based on count of antivirus products enabled do vary. Therefore, we must consider separate cases for each of the counts.

Conducting Student's t-test on each of the counts, we notice that we fail to reject the hypothesis that the true proportion of devices with malware is only less than 50 percent for one antivirus product enabled, while we fail to reject the hypothesis for the rest of the counts of antivirus products enabled. So, we have further evidence that the number of products enabled reduces malware.

Table III: Number of devices with and without malware detection according to the number of antivirus products enabled

Number of Antivirus Products Enabled						
Counts	0	1	2	3	4	5
No detection	16774	4286932	131821	4151	304	21
Has Detections	9184	4367169	66831	1924	149	2
Total Count	25958	8654101	198652	6075	453	23

Proportion with malware	0.3538	0.5046	0.3364	0.3167	0.3289	0.0870

Tying both of these together, we want to know how the number of antivirus products installed are associated with the number of antivirus products enabled. Since a majority of users have only one antivirus product enabled, we want to explore even further the effectiveness of having more than one antivirus product installed. Thus, we will observe the data by separating those who have more than one antivirus product installed and those who have at most one antivirus product installed.

Observing the results from Table IV visually, we see that the majority of people who have at most one antivirus product installed have that product enabled. This proportion of devices with malware detected is greater than our set baseline. On the other hand, we notice that those who have more than one antivirus product installed have a lower proportion of malware detected than our baseline.

Table IV: Comparison of proportion malware based on count of antivirus products installed and number of antivirus products enabled

Comparison of number of antivirus products installed								
	At most one antivirus product installed		More than one antivirus product installed					
Number of antivirus products enabled	0	1	0	1	2	3	4	5
No detections	8021	2794795	8753	1492137	131821	4151	304	21
Has Detections	6888	3399190	2296	967979	66831	1924	149	2
Total Count	14909	6193985	11049	2460116	198652	6075	453	23
Proportion with malware	0.4620	0.5488	0.2078	0.3935	0.3364	0.3167	0.3289	0.0870

We conduct a Kolmogorov-Smirnov test, we want to compare the two distributions of malware detection of having at most one antivirus product installed and more than one antivirus product installed. We calculate a test statistic D = 0.1603 and p-value < 2.2e-16. Since the p-value is less than a significance level of 0.05, we reject the hypothesis that these two distributions are the same. Thus, we are able to conclude that there is a difference in malware detection based on antivirus products installed, regardless of the number of products enabled.

Therefore, we are able to conclude that those who have more than one antivirus product installed are less likely to have malware than those who have only one antivirus product installed. This coincides with our findings that having more antivirus products enabled also helps reduce the chance of malware. Although it is not certain the reason behind our result, we suspect that those who only have one antivirus product installed have some sort of default antivirus. Also, we suspect that those who have more than one antivirus product installed are more likely to take care of their device and are more cautious about malware, thus resulting in lower proportion of malware detection.

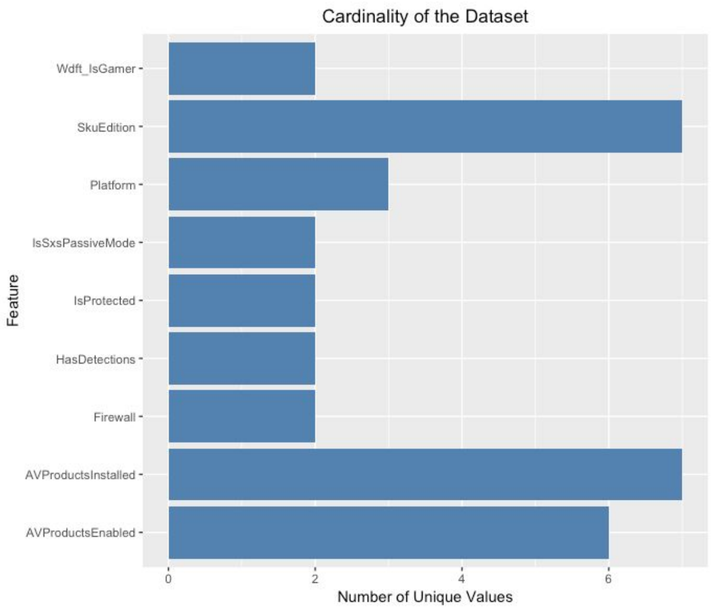
Scenario 3: Can we provide a predictive model to indicate which of the listed features is the most significant in determining a machine's malware detection status?

Our predictive modelling aims to investigate the relationship between the following variables. We are particularly curious in how these variables affect whether or not a machine *HasDectections* value is true or false (1 or 0). To better understand the data we decided to compute the cardinality of the dataset (find the number of unique values each variable contains). Fig. 9 shows that many of the variables only contain 2 unique values which are found to be 1 or 0 representing a true or false value. The variables containing more than 2 unique values are listed below with the potential values they could have in the dataset:

- *SkuEdtion*
  - Pro, Home, Enterprise LTSB, Cloud, Education, Enterprise, Invalid
- *Platform*
  - Windows 10, Windows 8, and Windows 7
- *AVProductsInstalled*
  - 1, 2, 3, 4, 5, 6, 7
- *AVProductsEnabled*
  - 0, 1, 2, 3, 4, 5

Other than these specific variables listed above, we find the counts of unique values per feature in Fig. 9 as follows:

Fig 9: The number of unique values per desired feature from our dataset.



Keeping these features in mind, we try to use boosting with decision trees using the XGBoost package in R. First, we divide the dataset of approximately 8.9 million observations into 3 parts:

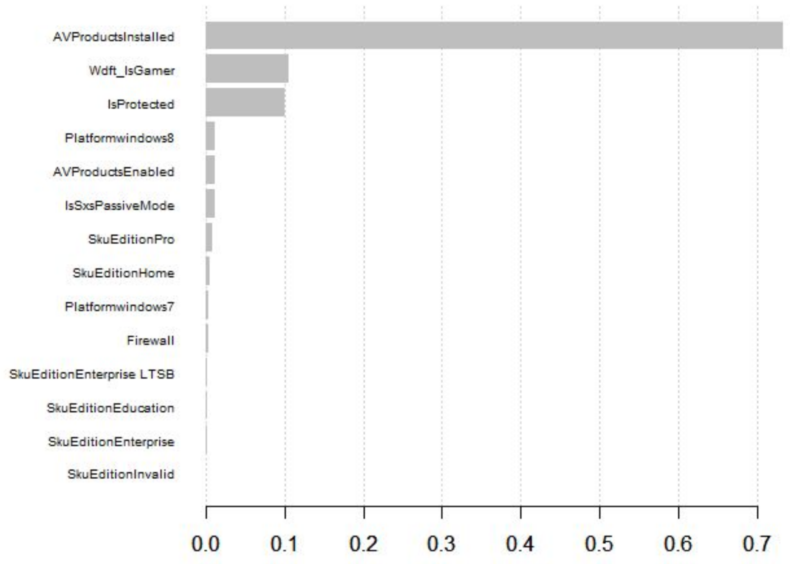
1) Training data (65% of the data)
2) Validation data (20% of the data)
3) Test data (remaining 15% of the data)

The training data, as the name suggests, is what we train the model using. Even after partitioning the data, we still have 5.5 million data points with which to train the model which is a substantial amount. The validation data is used to see how the model is performing and lets us choose whether we want to further continue tuning the model. The test data is kept entirely separate from the other data and not used until the very end when we're as done as we realistically can be with training the model.

After trying out a variety of different parameters for the model, changing the learning rate, max depth of the tree, and number of iterations of training, the lowest training error rate and we can attain after training the model for 25 rounds with the latest choice of parameters is about 0.4241. The model also ends up having an error rate of 0.42 on the test set which conversely means that the accuracy rate is about 60%, which is non-trivially better than the accuracy rate for guessing at random for binary classification, which would be 50%.

Figure 10 displays the importance of every feature in predicting if a system is prone to be affected by malware or not:

Fig. 10: The calculated importance of each feature in predicting the infection status of a machine



We find that the most important factor in determining if a system is prone to being infected is the number of antivirus products installed on a machine, followed by whether the system belongs to a gamer or not, followed by which operating system the user has where Windows 8 was the most significant one.

DISCUSSION AND CONCLUSION

Our investigation was divided into two main parts: analysis and prediction. Both means of investigation revolved around the same dataset, but accomplished drastically different goals. Our analysis of the dataset provided insight regarding specific questions asked about our data, whereas our prediction used all the information regarding specific features of our data to answer a general question.

In our analysis, we posed two questions. The first of the two was whether or not machines used by gamers were more likely to contract malware infection. Using a bootstrap method to generate a model distribution, we find that with this model is fairly normal and verify the shape of this distribution using a Kolmogorov-Smirnov test. Proceeding with a Student's t-test, we confirm that the proportion of gamer devices with malware is not the same as the discussed 50% baseline, in fact, it was greater than the 50% baseline, so we come to the conclusion that gamer machines are more likely to contract some sort of malware infection.

The second question was on the basis of how protected a consumer's machine is depending on the number of antivirus products installed. We were able to determine three main points. We first found that there is enough statistical evidence to conclude that the proportion of devices with malware is lower than our 50-percent baseline for machines with more antivirus products installed. We then found that there is enough statistical evidence to conclude that the number of antivirus products enabled reduces the risk of malware detection. With these two conclusions, we finally found that there is enough statistical evidence to conclude that machines with more than one antivirus product installed is better protected than those with at most one product installed, which answers our question for this scenario.

We conclude our investigation by providing a predictive model, indicating what type of machine is most likely to contact malware, given the features described in our DATA section. These features are the *HasDetections*, *IsSxsPassiveMode*, *AVProductsInstalled*, *AVProductsEnabled*, *Platform*, *SkuEdition*, *IsProtected*, *FirewallI*, and *Wdft_IsGamer* columns from our dataset. From these columns, we have found a predictive model with a ~60% success rate in determining the malware detection status per machine, where the most significant factor in determining this status was *AVProductsInstalled*.

We understand that our predictive model may have been restricted by the number of features used in training the model. Though the previous literature in which the author used a minimal set of features to classify malware, this may not have been the best approach for our study. If we used more features, we could have been able to train a more accurate model, as we see that our model is just a bit better than random classification.

METHODS AND THEORY

Skewness

In probability theory and statistics, asymmetry or distortion is an indicator of the probability distribution asymmetry of a real-valued random variable. The value of a degree may be positive or negative and may not be defined. If the degree is negative, the probability density function has a long tail on the left side of the function and a median containing more data on the right side. When the degree is positive, it has a long tail on the right side of the probability density function and indicates that the data are distributed more on the left side. If the mean and median are the same, the degree is zero.

Skewness coefficient

Skewness coefficient is the average of the third power of the standardized data.

$Skewness\ coefficient\ = \frac{1}{n}(\sum_{i=1}^{\infty} \frac{Xi - \overline{X}}{S})^{3}$ where S is the standard deviation, with

$S^{2} = \frac{1}{n-1} \sum_{i=1}^{n} (X_{i} - \overline{X})^{2}$ [19]

Kurtosis

Kurtosis is a measure of the degree of sharpness of the probability distribution. It is used to measure how intensively the observations are centered. If the kurtosis value (K) is close to 3, the scatter is close to the normal distribution. (K < 3), the distribution can be judged to be a flat distribution more gently than the normal distribution, and if the kurtosis is a positive number larger than 3 (K> 3), the distribution can be considered to be a more pointed distribution than the normal distribution.

Kurtosis coefficient

Kurtosis coefficient is the average of the fourth power of the standardized data.

$$Kurtosis\ coefficient\ = \frac{1}{n}\left(\sum_{i=1}^{\infty} \frac{Xi - \overline{X}}{S}\right)^4$$

where S is the standard deviation.

T-test

The t test is a name of a statistical test method that utilizes that statistics follow the t distribution when assuming that the null hypothesis is correct. It is a parametric test method that assumes that the population follows a normal distribution and uses that the t distribution does not depend directly on the original average or standard deviation (but depends on the degree of freedom). It is used for testing whether there is a significant difference in mean between two sets of specimens. One of statistical hypothesis tests. In the Japanese Industrial Standard, it is defined as "a statistical test that assumes that the test statistic follows the t distribution under the null hypothesis." The t-test is also called the Student's t-test.

One sample T-test

One sample T-test is for testing the null hypothesis that the population mean value μ is equal to a specific value μ_0 .

$$t = \frac{\overline{X} - \mu}{s/\sqrt{n}}$$

Notation

$\overline{X}$: *Sample mean*

s : *Standard deviation*

μ : *Population mean value*

The t test can be broadly divided as follows.

1. A test of whether the means are equal, assuming that both populations follow a normal distribution.
2. If specimens are paired, that is, if there is a special relationship between each member of a set of specimens and another set of specific members (for example, when investigating the same person twice before and after, When comparing between husband and wife, etc).

3. If the samples are independent and it can be assumed that the variances of the two groups being compared are equal (assumption of equidistribution).
4. When the specimens are independent and equal variance can not be assumed (heterogeneous dispersion). This is exactly called Welch's t test.

A test of whether the average of the population according to the normal distribution is equal to a specific value.
Test on whether the slope of the regression line is significantly different from 0.

Independent two-samples T test

It is used to test the null hypothesis of whether the mean value μ_1 of the first population is equal to the mean value μ_2 of the second population. In other words, test the null hypothesis of whether. $\mu_1 - \mu_2 = 0$.

Equal sample size and equal variance

For explanation, two specimens of sample sizes n and m, $x_1, ..., x_m$ and $y_1, ..., y_n$ are normal distributions N (μ_1 , σ_1^2) and N (μ_2 , σ_2^2) (σ_1 , σ_2 are unknown). At this time, we perform a test on the difference of the population mean of two samples, δ, and set the hypothesis as follows.

Null hypothesis $H_0 : \delta = \mu_0$

Alternative hypothesis $H_1 : \delta \neq \mu_0$

Here, the difference d = $\bar{x}$ - $\bar{y}$ between the average values x - ~ N (μ_1 , $\frac{\sigma_1^2}{m}$) and y - ~ N (μ_2 , $\frac{\sigma_2^2}{n}$) of the two specimens is also normal distribution N ((μ_1 - μ_2), σ^2 ($\frac{1}{m} + \frac{1}{n}$)) is used.

Kolmogorov-Smirnov

The Kolmogorov-Smirnov test is a type of hypothesis tests in statistics, and based on a finite number of samples, whether the probability distributions of the two populations are different or whether the probability distribution of the population is presented as a null hypothesis It is used to investigate whether it is different from the distributed distribution. Often abbreviated as KS test.

One sample Kolmogorov-Smirnov tests compares the empirical distribution with the cumulative distribution function shown in the null hypothesis. The main application is fitness test for normal distribution and uniform distribution. As for the test on the normal distribution, a slight improvement by the lift force is known. In the case of normal distribution, the Shapiro - Wilk test and Anderson - Darling test are generally more powerful methods than the Lily force test.

Two sample Kolmogorov-Smirnov tests is one of the most effective and general nonparametric methods to compare two specimens. This is because this approach depends on both the position and the shape of the empirical distribution for the two specimens.

Empirical distributions Fn for samples $y_1, y_2, ..., y_n$ are given as follows.

$$F_n(x) = \frac{\#\{1 \le i \le n \mid y_i \le x\}}{n}$$

Let F(x) be the distribution presented in the null hypothesis or the other empirical distribution, then the two one-sided KS test statistics are given by

$$D_n^+ = \sup_x (F_n(x) - F(x))$$
$$D_n^- = \sup_x (F(x) - F_n(x))$$

Assuming that the null hypothesis that the two distributions are equal is not rejected, the probability distribution that the above two statistics should obey is not dependent on the shape of the distribution as long as the distribution presented by the hypothesis is a continuous distribution.
In the one sample Kolmogorov-Smirnov test, when the number of samples n is sufficiently large, the distribution of the examination amount under the assumption that the empirical distribution Fn (x) follows the null hypothesis is

$$\text{Prob}(\sqrt{n}D_n \le x) = 1 - 2\sum_{i=1}^{\infty}(-1)^{i-1}e^{-2i^2x^2} = \frac{\sqrt{2\pi}}{x}\sum_{i=1}^{\infty}e^{-(2i-1)^2\pi^2/(8x^2)}$$

Therefore, when the significance level is α, the verification amount Dn is $n\sqrt{D_n} > K_\alpha$ (where K_α is the number satisfied with $Prob(\ n\sqrt{D_n} > K_\alpha) = 1 - \alpha$) is satisfied, the null hypothesis is rejected and it is suggested that the empirical distribution Fn (x) is different from the distribution F (x) presented in the null hypothesis.

Goodness of Fit test
Goodness of Fit test is the test that if the sample data fits a certain distribution.
We first introduce discrete models: n observations are grouped into t classes.
Then, we use hypothesis test:

$$H_0\colon p_1 = p_{1_o},\ p_2 = p_{2_o}, \ldots, p_t = p_{t_o}$$
$$H_1\colon p_i \ne p_{i_o} \text{ for at least one } i$$

which is equivalent to test if $(X_1, X_2, ..., X_t)$ comes from a multinomial population with probabilities $p_1 = p_{1_o}, p_2 = p_{2_o}, ..., p_t = p_{t_o}$
Example: we test $H_0 : p_1 = 0.3, p_2 = 0.5, p_3 = 0.2$.
So, in this case, $p_{1_o} = 0.3$, $p_{2_o} = 0.5$, $p_{3_o} = 0.2$.

Chi-Squared Test Statistics (known parameter)

Theorem: Let $r_1, r_2, ..., r_t$ be the set of possible outcomes (or ranges of outcomes) associated with each of n independent trials, where $P(r_i) = p_i, i = 1, 2, ..., t.$ Then, let
$X_i = number\ of\ times\ r_i\ occurs,\ i = 1, 2, ..., t.$
Then, The random variable

$$D = \sum_{i=1}^{t} \frac{(X_i - np_i)^2}{np_i}$$

which is $X^2 test\ statistic$ has approximately a $X^2 distribution$ with t-1 degrees of freedom. For the approximation to be adequate, the t classes should be defined so that $np_i \geq 5$, for all i.
For t=2, $X^2 test$ using D is equivalent to perform a test using Z test statistic.
We can derive

$$\begin{aligned} D &= \frac{(X_1 - np_1)^2}{np_1} + \frac{(X_2 - np_2)^2}{np_2} \\ &= \frac{(X_1 - np_1)^2}{np_1} + \frac{[n - X_1 - n(1-p_1)]^2}{n(1-p_1)} \\ &= \frac{(X_1 - np_1)^2(1-p_1) + (-X_1 + np_1)^2 p_1}{np_1(1-p_1)} \\ &= \frac{(X_1 - np_1)^2}{np_1(1-p_1)} \quad = \left[\frac{X_1 - E(X_1)}{\sqrt{\text{Var}(X_1)}}\right]^2 \end{aligned}$$

Notation : If X^2 is large, then it is a poor fit.
If X^2 is small, then it is a good fit.

Decision rule: Let $k_1, k_2, ..., k_t$ be the observed frequencies for the outcomes $r_1, r_2, ..., r_t$, respectively, and let $np_{1_o}, np_{2_o}, ..., np_{t_o}$ be the corresponding expected frequencies based on the null hypothesis. At the α level of significance, H_0 is rejected if

$$d = \sum_{i=1}^{t} \frac{(k_i - np_{i_o})^2}{np_{i_o}} \geq \chi^2_{1-\alpha, t-1}$$

where $np_{i_o} \geq 5$ for all i.

Here, we summarize four steps of $X^2 test$ with known parameters.
Step1: State a null hypothesis H_0 and an alternative hypothesis H_1 .

$$H_0: p_1 = p_{1_o}, p_2 = p_{2_o}, \dots, p_t = p_{t_o}$$

$$H_1: p_i \neq p_{i_o} \text{ for at least one } i$$

Step 2: The data are divided into several classes. The test statistic

$$D = \sum_{i=1}^{t} \frac{(X_i - np_i)^2}{np_i}$$

Has approximately a X^2_{t-1} distribution if H_0 is true.
Step 3: Compare the observed test statistic with $X^2_{1-\alpha,t-1}$ or calculate the p-value.
Step 4: Conclusion.

Chi-Squared Test Statistics (unknown parameter)
Suppose that a random sample of n observations is taken from $f_Y(y)$ *or* $p_x(k)$, a pdf having s with unknown parameters. Let $r_1, r_2, ..., r_t$ be a set of mutually exclusive ranges or outcomes associated with each of the n observations. Let $\widehat{p_i}$=estimated probability of r_i, i=1,2,...,t as calculated from $f_Y(y)$ *or* $p_x(k)$ after the pdfs, s with unknown parameters have been replaced by their maximum likelihood estimates. Let X_i denote the number of times that r_i occurs, i=1,2,...,r. Then, the random variable

$$D_1 = \sum_{i=1}^{t} \frac{(X_i - n\widehat{p_i})^2}{n\widehat{p_i}}$$

Has approximately a X^2 distribution with t-1-s degrees of freedom. For the approximation to be Fully adequate, the r_i 's should be defined so that $n\widehat{p_i} \geq 5$ for all i.
Decision rule: $k_1, k_2, ..., k_t$ are the observed frequencies of $r_1, r_2, ..., r_t$, respectively, and $n\widehat{p}_{1_o}, n\widehat{p}_{2_o}, ..., n\widehat{p}_{t_o}$ are the corresponding estimated expected frequencies based on the null hypothesis. If,

$$d_1 = \sum_{i=1}^{t} \frac{(k_i - n\widehat{p}_{io})^2}{n\widehat{p}_{io}} \geq X^2_{1-\alpha,t-1-s}$$

H_0 should be rejected. (The r_i 's should be defined so that $n\widehat{p}_{i_o} \geq 5$ for all i.)
Here we summarize five steps of $X^2 test$ with unknown parameters.
Step 1: Find the maximum likelihood estimator for s unknown parameters.
Step 2: : Plug the estimated parameters into the distribution to be tested, calculate the estimated probabilities $\widehat{p}_{1o}, \widehat{p}_{2_o}, ..., \widehat{p}_{t_o}$,then state the null hypothesis H_0 as

$$H_0 : p_1 = \widehat{p}_{1o}, p_2 = \widehat{p}_{2_o}, ..., p_t = \widehat{p}_{t_o}$$

Step 3:The test statistic

$$d_1 = \sum_{i=1}^{t} \frac{(k_i - n\widehat{p}_{io})^2}{n\widehat{p}_{io}} \geq X^2_{1-\alpha,t-1-s}$$

Has approximately a X^2_{t-1-s} distribution if H_0 is true.
Step 4: Compare the observed test statistic with $X^2_{1-\alpha,t-1-s}$.
Step 5: Write a conclusion.

P-value

If we let the significance level as 5%, then the result of the hypothesis test is rejecting the null hypothesis at a significance level of 5%. However, it would arise questions that "what happens when the significance level is 1% or 0.1%?" and "what is the limit?" So, P-value solves that question.
If P-value is smaller than a significance level, then we reject the null hypothesis H_0. In other words, the limit value of the significance level such that the null hypothesis is rejected is P-value.
It P-value is small, which means
① the probability of taking an extreme value is smaller than the statistic calculated from the data.
② we can reject null hypothesis.
Therefore, the smaller the P-value is, the more reliable to reject the null hypothesis.

Decision tree
Decision trees are graphs for making decisions in the field of decision theory, and are used to make plans and reach goals. Decision trees are created to help make decisions. Decision trees are a special form of tree structure.
In the field of machine learning, decision trees are predictive models, and observations on an item lead to conclusions about the target value of the item. Internal nodes correspond to variables, and branches to child nodes indicate possible values of the variable. The leaves represent predicted values of the objective variable with respect to the variable value represented by the route from the root.

The machine learning method of making decision trees from data is called decision tree learning, or simply called decision tree.

Decision trees are a commonly used method in data mining because classification models using decision trees can easily interpret the process leading to their classification. In that case, the decision tree has a tree structure in which the leaves represent classifications and the branches represent a collection of features leading up to the classifications.

The decision tree can be learned by dividing the original set into subsets based on the attribute value test. This process is repeated recursively for all subsets. Repetition ends when division becomes impossible or when individual elements of the subset fall into one classification.
There are two types of decision tree.
a.Regression tree
→ It is not used for classification but for approximation of functions that take real values.
b.Classification tree
→ when y is a classification variable.

XG Boost Classification

What is XG Boost?

・Ensemble learning combining gradient boosting and random forest (classifier composed of a plurality of classifiers).

・It is said that gradient boosting is implemented in C ++ and accelerated, and it is said to process about 10 times faster than Gradient Boosted Tree (GBT).

・ Tuning with multiple parameter adjustments is required, and optimization involves multiple grid searches and cross validations.

・ In order to increase the generalization ability, lower the learning rate (parameter eta in the XGBoost package) and perform optimization each time.

Basically what is done internally is to make multiple decision trees. However, there is a feature in how to make it.

Let's define the symbol here.

Let m be the explanatory variable xi = (xi1, ..., xim) T, and let yi, i = 1, ..., n be the target variables. Here n is the number of data. Also, let $\widehat{y}$ i be the predicted value for data xi.

First, build one decision tree. Then you can use that decision tree to make predictions. Let yi be a predicted value for data xi obtained from the first decision tree. At this time, the error between the measured value yi and the predicted value ŷ (1) i is

ε (1) i = yi-ŷ (1) i

It will be. What the above error represents is the part that could not be learned in the first decision tree, that is, the part that could not be explained. yi-ŷ (1) If i is close to 0, measurement and prediction agree with each other, so learning is possible. In other words, the i-th data could be explained by one decision tree. It means that if it is a value away from 0, the prediction is wrong and learning is not good.

Well, what if you could predict the error? Actual value-predicted value = error Therefore, if an error is known, it means actual value = predicted value + error. If an error is added to the predicted value, the actual value is obtained. So, with XGBoost, the second decision tree is constructed using the part that could not be predicted in the first decision tree, that is, the error as the objective variable. We make a model to predict the error.

Suppose you build a second decision tree. Then you can use the decision tree to predict errors. Let ε ˆ (2) i be the predicted value for data xi obtained from the second decision tree. That is, ε ˆ (2) i is the predicted value of the part ε (1) i that could not be explained by the first decision tree. At this time, the predicted value for the actual value yi is as follows by adding the predicted value of the second decision tree, ie, the predicted error, to the predicted value of the first decision tree:

ŷ (2) i = ŷ (1) i + ε ˆ (2) i
Therefore, the error between the actual measured value and the predicted value when the second decision tree is constructed is

ε (2) i == yi−ŷ (2) iyi− (ŷ (1) i + ε) (2) i)
It will be. Here ε (2) i is the part that could not be learned even if the second decision tree was constructed. So we build a third decision tree with ε (2) i as the objective variable. Again, the predicted value for data xi after the third decision tree has been constructed is the value predicted by the first decision tree + the error that could not be learned by the first decision tree The predicted value of (the result of the second decision tree) + the predicted value of the error that could not be learned even after constructing the two decision trees (the result of the third decision tree), You:

ŷ (3) i == ŷ (2) i + ˆ (3) i ŷ (1) i + ˆ (2) i + ˆ (3) i
Repeat the same operation K times below. The number of repetitions K is a parameter determined by the analyst, and needs to be determined well by cross validation.

In this way, boosting is a method of promoting data learning by constructing a new model using information on models learned so far.

At the beginning of this article, what is being done inside XGBoost is to make multiple decision trees. More precisely, when boosting, that is, when creating a new decision tree using the results of the previous decision tree, the error between the measured value and the predicted value is minimized in a sense It is XGBoost that built the decision tree algorithm. Here, "in a certain sense" means that the world's machine learning algorithm constructs an algorithm so as to minimize (or maximize) the objective function, but has not yet described the objective function in XGBoost is.

The point to be emphasized here is that instead of simply constructing a new decision tree with error as an objective variable, building a new decision tree using the information of the previous decision trees . So, when building a new decision tree, it leads to the story of how to use the information up to the previous time.

Decision trees are algorithms that divide explanatory variables and, in an easy-to-understand manner, construct a tree diagram. From this fact, using the information up to the previous time when constructing a decision tree means using the information up to the previous time to optimize the division of the explanatory variable. And, I will explain later, but the information up to the previous time can be taken out as the gradient of the objective function. G of XGBoost is G of Gradient.

<u>Formulation of XGBoost</u>
Now, what we need to explain is "how to build a decision tree using the information up to the previous time". So, for now, let the f-th decision tree itself be abstracted into the function fk. Passing

the data xi to the function f returns the predicted value for that data. In the example so far, f1 (xi) = ŷ (1) i, f2 (xi) = ε̂ (2) i, f3 (xi) = ε̂ (3) i. And the predicted value when boosting K times, that is, when the Kth decision tree is constructed, is as follows:

ŷ i = ∑ K k = 1 fk (xi)
We still do not know what f is like, that is, "what kind of algorithm is the decision tree using the previous information?" So, by doing various calculations, we will look at what kind of properties of this f should be satisfied.

Let's express the previous story as a formula.

Introduce the loss function l (a, b). As a loss function, for example, in the case of regression in many cases the squared error

Squared error = (actual value-predicted value) 2
Is used. The loss function is differentiable, and is set to take smaller values as a (for example, actual values) and b (for example, predicted values) are closer, and to take larger values as they are farther. Therefore, the question "how should we construct the t-th decision tree using the information of the previous t-1 decision trees" is formulated as follows:

$$\min_{f_t} \sum_{i=1} l(y_i, \hat{y}_i^{(k)})$$

$$\Longleftrightarrow$$

$$\min_{f_t} \sum_{i=1}^{n} l\left(y_i, \sum_{k=1}^{t} f_k(x_i)\right)$$

$$\Longleftrightarrow$$

$$\min_{f_t} \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right)$$

This means that we need to construct a decision tree ft that minimizes the above equation.

However, overfitting to the data used for learning would lead to a phenomenon called overfitting, which would reduce the prediction performance. This is particularly noticeable in the above formulation. The reason is that we are only thinking that the prediction error will be small with yi, i = 1, ..., n actually available at hand, and we do not consider the prediction for future data at all is.

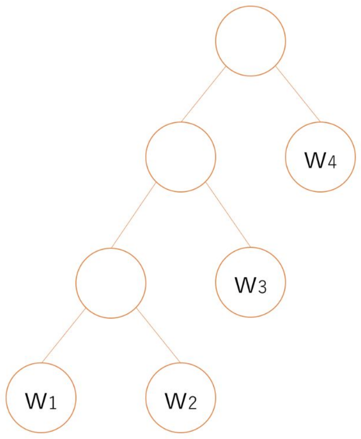
So we define the following penalty terms:

Ω (ft) = γT + 12λ || w || 2
Then let (t) (ft) be the penalty term added to the loss function above, and set the problem you want to solve finally:

$$\min_{f_t} \mathcal{L}^{(t)}(f_t) = \min_{f_t} \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$
$$= \min_{f_t} \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \gamma T + \frac{1}{2}\lambda||w||^2$$

Here, T is the number of final nodes (that is, the size of the tree) when constructing the decision tree. The larger the tree size T, the easier it is to over-learn, and the smaller the part of l, the smaller the objective function (ft). ft) works to be larger. As a result, too large T trees are not favored. γ is a penalty for the size of T, and when γ = 0 it is fine to over-learn, so I just want to make the part of l smaller! ! This means that by setting γ> 0, the larger the value of γ, the smaller the value of T, that is, the smaller tree is preferred. Therefore, γ is a parameter that analysts need to decide in advance.

Also, w is a vector of values that the decision tree ft can return. For example, in the case of the tree below, T = 4, and the values that can be returned are four: w = (w1, w2, w3, w4). In other words, w is a set of values that ŷ (t) i can take.



λ is a parameter for adjusting the size of the value w that can be returned by this decision tree ft. Setting a large λ makes small w to be preferred. The predicted value after the t-th decision tree is constructed is the sum of the results up to the t-1st result plus the t-th result ∑t-1k = 1fk (xi) + ft (xi) Therefore, the fact that the value of ft (xi) is small is like updating the difference between the measured value and the predicted value. On the other hand, setting a small λ will boldly update the error between the measured value and the predicted value so that it quickly becomes smaller. In other words, λ is a parameter to adjust the width of update when constructing a new decision tree. A small update will increase the number of K's, which means building a lot of trees, so it will take longer to learn, but while it can express various patterns, it will also be a factor in over-learning. If bold updating is performed, the difference between the actual measurement value and the predicted value decreases quickly, but if the difference decreases quickly, the number K of decision trees finally built will decrease, so the input data Can not capture the various patterns of As balance is important, λ must be determined by the analyst in advance by cross validation.

EDA (Exploratory Data Analysis)

Exploratory data analysis was advocated by the famous statistician JWTukey from around 1960. In interpreting the data, it is not a "model first" but a realistic situation before assuming a model. It is an approach that emphasizes the initial phase of analysis, in which information suggested by is collected in multiple ways.

Before that, we prepared models in advance and applied data to calculate probability. However, in reality, it is not easy to prepare an optimal model in advance from complex real data structures. As a result, you need to look at the data and then modify or select a model.

In addition, as it emphasizes methods that can be used easily and easily by anyone, rather than the theoretical aspects of mathematical statistics, it is also effectively used for "data mining" at business sites. It can be said that it is a possible approach.

<u>Characteristics</u>

1. Robust and robust method of resistance
   "Resistant robustness" here means that it is not susceptible to outliers. Although real data often contains outliers, the mean and variance are susceptible to outliers and are not very resistant. However, medians and quartiles are often used in exploratory data analysis because they are not susceptible to outliers. For example, in the case of 99 pieces of data and 10 pieces of 10000, the average value is 109.9, but the median is 10. Also, in the general least squares regression analysis, if there are outliers (to square the distance), the outliers will be pulled large by that, but in exploratory data analysis, methods resistant to outliers have been developed, S-PLUS also includes such a method.
2. Analysis of residuals from models
   Fitting data into a model and assessing this degree of fit is not a complete analysis. Whether you are interpreting experimental data, or when you compare forecast and actual values in your business, it is very important to feed the assessment back to the model. By further analyzing the residuals after fitting, you can fit the original data to a better model. S-PLUS is very easy to re-analyze the output residuals, and there are several methods to check the normality of residuals.
3. Utilization of data re-expression (variable conversion)
   Depending on the data, log conversion or reciprocal calculation makes it easier to observe the data. Also, in some cases, it may be easier to compare the stratified data equally distributed. For time-series data, using moving averages instead of the original data makes it possible to make observation easier. Not only is S-PLUS easy to convert these variables, it is also possible to use the converted data as input immediately.
4. Invention and development of graphic display
   When displaying data in two dimensions, various methods have been developed that can represent the features of data at a glance. In general, stem leaves and box and beard

figures are often used. In addition, with regard to the method of expressing multivariate data of three or more dimensions in two dimensions, it is possible to express clearly by brushing and face graph etc. S-PLUS offers many of these unique graphics.

ACKNOWLEDGEMENTS

CISION PR Newswire, Cybersecurity Ventures. (2017, October 19). *Cybercrime damages are predicted to cost the world $6 trillion annually by 2021*[Press release]. Retrieved from https://www.prnewswire.com/news-releases/cybercrime-damages-are-predicted-to-cost-the-world-6-trillion-annually-by-2021-300540158.html

Lemonnier, J. (2015, June 6). *What is Malware? How Malware Works & How to Remove it* [Web log post]. Retrieved March 20, 2019, from https://www.avg.com/en/signal/what-is-malware

Microsoft Malware Prediction. (n.d.). Retrieved from https://www.kaggle.com/c/microsoft-malware-prediction/data

Microsoft. (n.d.). *Windows Defender Antivirus*[Press release]. Retrieved from http://info.microsoft.com/rs/157-GQE-382/images/Windows 10 Security Whitepaper.pdf

Raman, K. (n.d.). *Selecting Features to Classify Malware*[Scholarly project]. Retrieved from http://www.covert.io/research-papers/security/Selecting Features to Classify Malware.pdf

Rankin, B. (2018, April 5). *A Brief History of Malware — Its Evolution and Impact* [Web log post]. Retrieved March 20, 2019, from https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/