Final Project: Microsoft Malware Prediction
Can you predict if a machine will soon be hit with malware?

Yu-Chun Chen, A13356506
Yanyu Tao, A13961185
Bolin Yang, A92111272
Shuibenyang Yuan, A14031016
Haoming Zhang, A14012520
Mingxuan Zhang, A13796895

Introduction:

The war against malicious software, or malware, is one of the most significant topics in the cyber security domain since it gradually becomes a well-organized market. When malware evades the computer system, the virus can harm users. This malware problem related to everyone using computers from individuals to large companies and agencies. The malware becomes more sophisticated and can frustrate the antivirus software through encryption like inserting dead code, substituting instructions, like reordering the routine to target computers and mobile devices. So, the advanced malware is hiding inside then replicate the host protections, which is unknown and hard to discover. After installation, the infectious software of malware steals the crucial private information, such as the e-mail password and the bank account, and infects other software.

The diversity and concealment of the malicious software undermine the efficiency of traditional antivirus defenses using the signature-based. With the vast consumer and enterprises basis, Microsoft invests billion of money into this field and continues updating the operating system to improve the security. To counteract to such malware threats, Microsoft wants to develop new techniques to forecast whether the machine will be hit with malware. As with their previous, Malware Challenge (2015), Microsoft is providing Kagglers with a unique malware dataset to encourage open-source progress on practical techniques for predicting malware occurrences (Kaggle).

In this project, we are going to assess the vulnerability of the machine (computer, server, etc.) based on its configuration. The goal is to predict the probability of a Windows machine getting infected by different types of malware, based on various properties of that machine (Kaggle).

There are some related works. Since the cyber-threat associates with every consumer using the computers, many researchers tried similar attempts to our project, which quantify the risk to reduce the malware attack. In “Ensemble Models for Data-driven Prediction of Malware Infections,” the author investigated more than 1.4 million hosts and 50 malwares in the past two years worldwide. They used ESM, an ensemble-based approach combining with the non-linear model for malware spread. They found that ESM is relatively stable and accurate even though the dataset is small. G.J. Tesauro and other two researchers from IBM also investigated malware detections of boot sector viruses from the neural networks. They applied neural network into their antivirus software, which protects millions of computer users (G.J. Tesauro, J.O. Kephart, and G.B. Sorkin). Instead of the tradition signature-based method, M.G. Schultz et al. presented a data

mining framework to create a much faster detection rate. And this method can detect new malicious software, which is a vast improvement from the past (M.G. Schultz, E. Eskin, F. Zadok, and S.J. Stolfo).

Data:

Based on different properties of the machine, the probability of a Windows machine getting infected by various families of malware would vary in a certain amount. So, our dataset contains these properties, and the machine infections were generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender. Using the information and labels in train.csv, our object is to predict the value for HasDetections for each machine in test.csv (Kaggle).

The sampling methodology used to create this dataset was designed to meet certain business constraints, both regarding user privacy as well as the period during which the machine was running. Malware detection is inherently a time-series problem, but it is made complicated by the introduction of new machines, machines that come online and offline, machines that receive patches, machines that receive new operating systems, etc. Additionally, this dataset is not representative of Microsoft customers' devices in the wild; it has been sampled to include a much more significant proportion of malware machines (Kaggle).

From Kaggle, we have three datasets:

Dataset	Description
train.csv	The training dataset for prediction
test.csv	The test dataset for submission in Kaggle
sample_submission.csv	The sample submission shows how the submission may look like

Table 1: Description of Dataset

We will only use **train.csv** in our investigation since we will only do certain types of feature analysis upon certain features instead of submitting a prediction target dataset to Kaggle. From train.csv, we observed there are 82 feature columns, 1 target column (HasDetections), and 8921483 rows representing different computers. We will only select certain features to do analysis, and we will talk about the data preprocess in the beginning of investigation part of the report.

Our goal is to predict the probability of a Windows machine getting infected by different types of malware, based on various properties of that machine. From the whole dataset, each data property has different characteristics. There are three types, categorical, numerical, and binary. In these three features, most properties are numerical, which are more than 30 variables. Categorical type, the second most data feature, has 19 variables. And only 4 variables are binary.

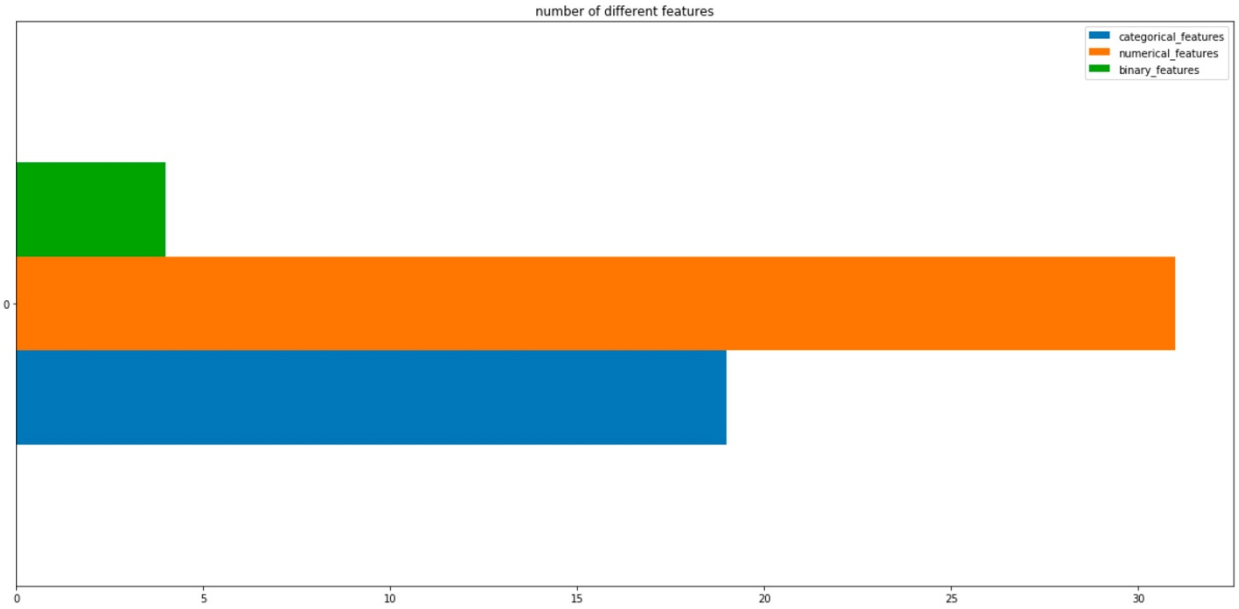

Figure 1: Distribution of Different Features

Background:

Cyber attackers create malicious software, or called malware, as threatening software to cause damage or gain the essential data to a computer. In the past years, the malware market reproduces, and it becomes one of the most pressing cybersecurity issues. Hackers improve their obfuscation technology and coding method to damage the tradition protection of antivirus, which forces antivirus companies to enhance their products defending the malware attacks. The major challenge that must overcome is to distinguish which part in the software is malware within the massive amounts of data and files.

Microsoft continues generating and running the anti-malware utilities over 150 million computers around the world, which collects millions of data to be analyzed as the potential malware. In order to analyze these data effectively, we classify them into different groups or families. This process is not an easy procedure, and the classifications are still expanding. This classification is based on their propagation process and achieved actions on the infected system (M. Asha Jerlin and C. Jayakumar). The following are the various types of malware: viruses, worms, trojans horse, spyware, adware, backdoors, rootkits, sniffers, reverse code engineering, disassemblers, debuggers, and decompiler.

The current malware analysis is the malware detection technique to catch malicious behavior in computer software and related files. There are two categories of malware analysis, one is the static method, and the other one is the dynamic method. The most significant difference between these two methods is that static method acquires the vital information by disassembling code, while dynamic method uses runtime trace report of executable files to get information (M. Asha Jerlin and C. Jayakumar). Static analysis analyzes malicious software without executing it and studies each component. Disassembler like IDA disassembles the binary files and translates

the malware's code into the assembly code. This process makes an image for the analyst, which can provide the details and particular patterns to classify the family of malware. Dynamic malware analysis is the opposite method of the static method. The dynamic method analyzes malicious files and software when they are running. During the process, infected files are simulated in the sandbox environment to avoid real infection into the system. After detecting the general behavior of malware, it may be debugged during the execution. However, the biggest problem is that dynamic analysis requires much time since researchers should prepare the exact environment for malware. And this method elevates the scalability issues (Ekta Gandotra, Divya Bansal, Sanjeev Sofat).

The malware detection technique is the essential process to analyze the role of a malware. There are two methods for malware detection, signature-based, and heuristic based. Most traditional antivirus software uses Signature-based detection. Once the malware is identified and analyzed by the system, its signature will be recorded and added to the database. Then, the computer can recognize the same malware again. However, the signature detection method has limited that there are more and more virus and virus can evolve. So, the Heuristic method analyzes the unknown malware or the new derivatives of the known malware. And it needs to upgrade the data continuously, which means that it requires a high level of CPU and other software.

Investigation

Data Preprocessing:

1. Column selection:

The malware The Malware training set obtained by Microsoft is a large 4GB dataset, the dataset provided to us is required preprocessing and data cleaning before any analysis could be done. The dataset has 83 columns and 8921483 rows. The dataset contains different aspects of different computers from software to hardware. Since we would only do the analysis upon different features rather than performing an accuracy prediction, we did not consider investigating the full scheme of the dataset.

We picked our features based on different aspects from a computer: **anti-virus software status of the computer, operating system of the computer, hardware of the computer, defender state of the system, location of the computer**, and we will investigate whether each aspect have direct relationship to the status of having virus or not, if so, which feature is the most influential under the specific feature.

The remainder of our analysis will focus on the data contained these columns:

Aspect in computer	Feature	Description	Variable Type
Anti-virus software	AVProductStatesIdentifier	NA	categorical
Anti-virus software	AVProductsInstalled	NA	categorical
Anti-virus software	AVProductsEnabled	NA	categorical
Operating System	Platform	Calculates platform name (of OS related	categorical

		properties and processor property)	
Operating System	OsBuild	Build of the current operating system	categorical
Hardware	Census_ProcessorCoreCount	Number of logical cores in the processor	numerical
Hardware	Census_PrimaryDiskTotalCapacity	Amount of disk space on primary disk of the machine in MB	numerical
Hardware	Census_TotalPhysicalRAM	Retrieves the physical RAM in MB	numerical
Hardware	Census_InternalBatteryNumberOfCharges	NA	numerical
Defender State	ProductName	Defender state information e.g. win8defender	categorical
Defender State	IsBeta	Defender state information e.g. false	binary
Location	GeoNameIdentifier	ID for the geographic region a machine is located in	categorical
Location	CountryIdentifier	ID for the country the machine is located in	categorical

Table 2: Descriptions of features we focus to analyze

2. Missing value:

Missing value is non-negligible when we deal with large data prediction. In particular, the features we chose have a certain amount of missing values:

Feature Name	Percent of missing
AVProductStatesIdentifier	0.406%
AVProductsInstalled	0.406%
AVProductsEnabled	0.406%
GeoNameIdentifier	0.0024%
Census_ProcessorCoreCount	0.0463%
Census_PrimaryDiskTotalCapacity	0.05943%

Table 3: Percentage of missing value for each feature

After investigating the Anti-virus features, we found out the **three anti-virus columns** and **two hardware columns** shown above are MAR (they have chance missing at the same time), which means we will perform a probabilistic imputation. The we found out the GeoNameIdentifier has MD since it is correlated with the column CountryIdentifier. We will impute its missing values by imputing condition on the correlated column CountryIdentifier.

3. One Hot Encoding:

Since the dataset is very large, it is more efficient to put dataset into a matrix to perform prediction. We will transform the dataset to matrix via scikit learn package's label encoder and one hot encoder in python. The features will be encoded using a one-hot encoding scheme. This creates a binary column for each category and returns a sparse matrix. The encoding process derives the categories based on the unique values in each feature.

4. Smoothing:

Since the dataset is large and messy, then the dataset has a lot of noise preventing us to do analysis. We need to smooth the dataset to improve the quality of prediction and visualization. We tried different ways to smooth, the first thing we did is to drop outliers of the data set. For instance, The Census Primary Disk Total Capacity has some fairly large value but with low counts: we assume that was a large server, it will largely skew our analysis if we consider the outliers like such. We did our first step to drop the outliers. Then we used conditional addictive smoothing technique for each feature to reduce the noise of rarely seen categories. For instance, in the AV Product States Identifier column, there are many small rarely seen AV Product States Identifier making the visualization messy. We consider smoothing those rarely unseen categories to improve out visualization, making it clearer to be analyzed. Finally, we tried to use rolling window technique to Hardware attributes, making the hardware quantitative data smoother to be analyzed.

Scenario 1: Antivirus

Q: Whether the features of antivirus software have direct relationship to the status of being infected by virus or not? If it is, which one is the most influential?

To begin, since this is a big project with tons of data, we need to find some relevant features to our target (whether the computer is infected with virus) in order for us to do analysis. Just the train data is about 4.08 GB, there are more than 80 different features for each sample in the training set. In our dataset, each row in this dataset corresponds to a machine, uniquely identified by a MachineIdentifier. And for each row, there is a column called "HasDetections", which provide the information that whether the machine is infected with virus or not, if it is infected, the value is 1, and 0 for no infection. After some cleaning and thoughtful thinking about where we should begin, we choose the following three features, which are 'AVProductStatesIdentifier', 'AVProductsInstalled', 'AVProductsEnabled'. "AVProductStatesIdentifier" stands for ID for the specific configuration of a user's antivirus software. 'AVProductsInstalled' stands for whether the antivirus software is installed on the machine. 'AVProductsEnabled' stands for whether the installed antivirus software is enabled or not. All of the three factors are directly linked to the antivirus software, which we believe is very crucial in determining whether the machine is infected or not, since the function of antivirus is to protect virus from getting into the machine. In the following exploration, we want to how much influence each factor will affect the infection condition of machine.

For the "AVProductStatesIdentifier", we found that in our training dataset, there are 28970 different antivirus product configurations, for convenience, we choose to analyze the top 100 antivirus product configuration. We plot the first five configuration to see the distribution.

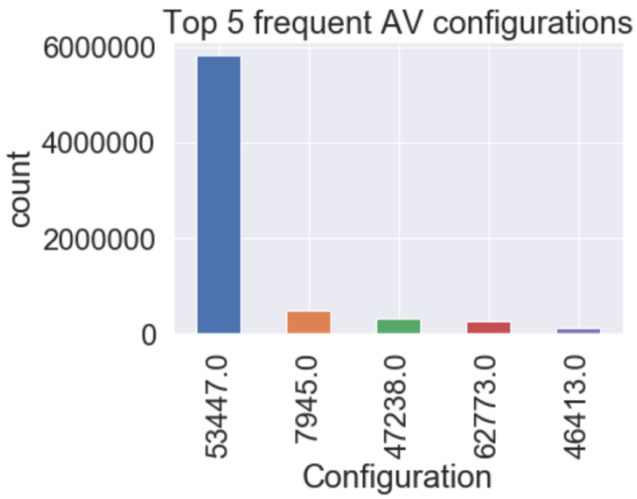

Figure 2: Bar plot for the distribution of top 5 "AVProductStatesIdentifier"

In the bar plot, we observe the even in the top 5 most frequent use of configurations, we get a huge percentage of machine with '53447' configuration, which means that in our overall population, there are a lot of users who will have the same configuration as '53447'. Since this group of machine has such a high occurrence, we want to see whether his configuration has the ability to defeat virus. We then plot a bar plot on the proportion of '53447' being infected or not.

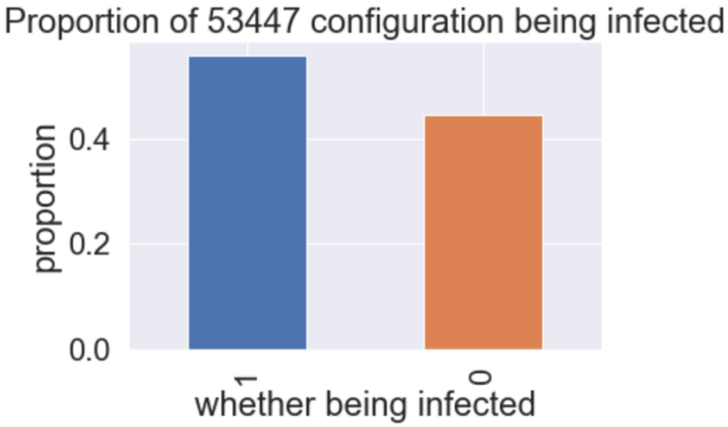

Figure 3: Bar plot for the distribution of 53447 configuration being infected or not

From figure 3, we can see there with the most popular antivirus, there is still quite a big proportion of machine being infected by the virus. Thus, we cannot fully depend on this type of antivirus based on its "popularity".

Then, we want to find out if the configurations of different anti-virus software truly affect the state of being infected by the virus or not, or it just occur at random. For each configuration, we create a control group with randomly generated value for whether the machine has detections of virus with the rate of being affected from the total group, and then use it to compare to the observed proportion value of the antivirus being affected or not. We plot the two distribution for both data and the sample for the top 100 configurations.

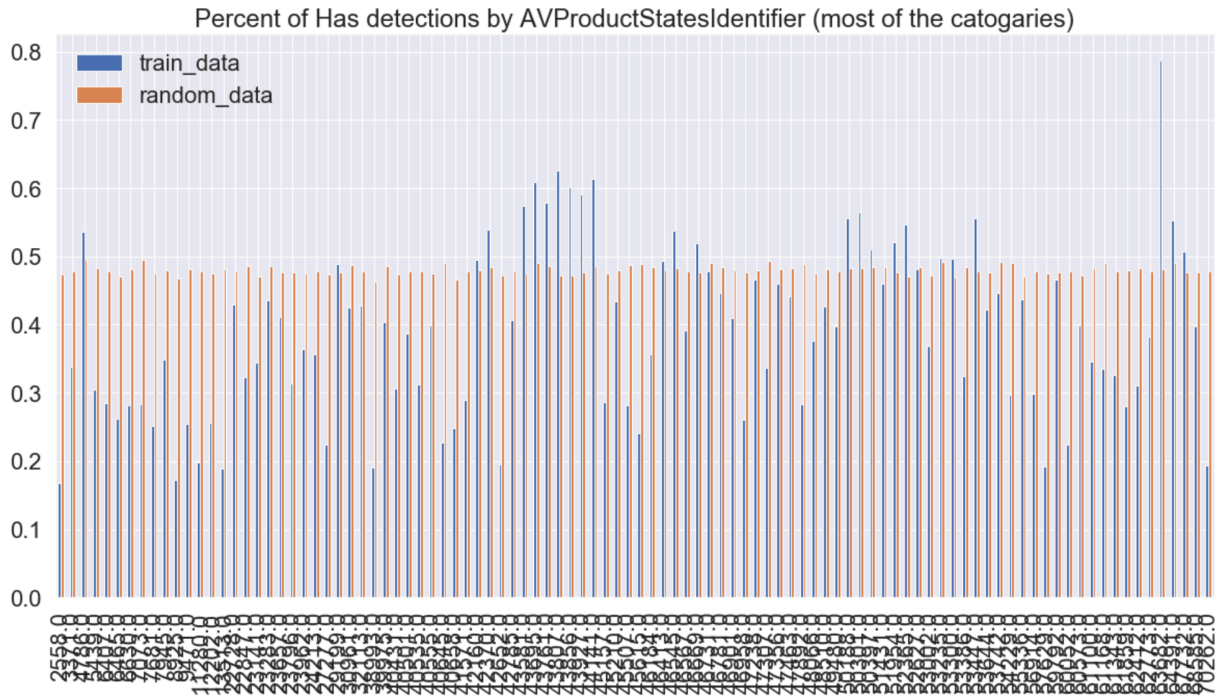

Figure 4: Bar plot for the distribution of the status of being affected by virus of top 100 "AVProductStatesIdentifier"

We observe that the distribution of the control group is almost uniformly distributed indicting the no effect on the target. By comparing the distribution of the control group and the original data, we do find a difference in the two distribution. In order to verify our observation, we then do a Kolmogorov-Smirnov test on the two distribution of machine being infected (data and the sample). The test-statistic is 0.99, which has p-value of 1.2251433537012255e-44 significantly lower than then 0.05 alpha level, thus we reject the null hypothesis that the antivirus product does have an influence in whether the machine is being affected or not. Therefore, if later we want to do a classification to predict status, we can use this feature, as it is influential in the prediction.

Next feature we want to look at is 'AVProductsInstalled'. There are 7 state in this column, except first 3, there is little data in the rest of states, thus we choose to only analyze the first three stats. If the value is 1, it means that the machine does not install their antivirus product. If the value is 2, it means that the machines does install their antivirus product. If the value is 3, it means that there are some mistakes in data that including all other state of this feature. As we did for "AVProductStatesIdentifier", we want to find out if machine is installed with the antivirus product would truly affect the state of being infected by the virus or not, or it just occur at random. This,

for each configuration, we create a control group with randomly generated value for whether the machine has detections of virus with the rate of being affected from the total group, and then use it to compare to the observed proportion value of the antivirus being affected or not. We plot the two distribution for both data and the sample for all three states.

	train_data	random_data
1.0	0.548581	0.498079
2.0	0.396906	0.497881
3.0	0.291596	0.497114

Table 4: Table for the proportion of machine being affected by virus in different state of "AVProductsInstalled"

Figure 5: Bar plot for the distribution of the status of being affected by virus in different state of "AVProductsInstalled"

In table 4 and figure 5, we can again observe that the control group has uniformly distributed data, thus having 0.5 for each state. Comparing with the distribution of our original data, we do observe a difference, but we are not sure. Thus, we do the K-S test on the distribution. The test-statistic is 0.666667, which has p-value of 0.3197243332709645 higher than then 0.05 alpha level, thus we cannot reject the null hypothesis that whether antivirus product is installed does not have an influence in whether the machine is being affected or not. Therefore, if later we want to do a classification to predict status of being infected by virus, we are not sure about using this feature.

Similarly, we also want to analyze the effect of "AVProductsEnabled" column on the status of the machine being infected of virus or not. At first, we believe that this is the most important feature, since this is the most important feature, we believe, that have the ability to prevent virus from entering the machine. There are 5 state for "AVProductsEnabled" column, only the first two have significant amount of data, thus we only focus on those two. If the value is 1, it means that the machine does not enable their antivirus product. If the value is 2, it means that the machines does enable their antivirus product. As we did for "AVProductStatesIdentifier", we want to find out if machine is enabled with the antivirus product would truly affect the state of being infected by the virus or not, or it just occur at random. This, for each configuration, we create a control group with randomly generated value for whether the machine has detections of virus with the rate of being affected from the total group, and then use it to compare to the observed proportion value of the antivirus being affected or not. We plot the two distribution for both data and the sample for both states.

	train_data	random_data
1.0	0.504636	0.496984
2.0	0.336422	0.497433

Table 5: Table for the proportion of machine being affected by virus in different state of "AVProductsEnabled"

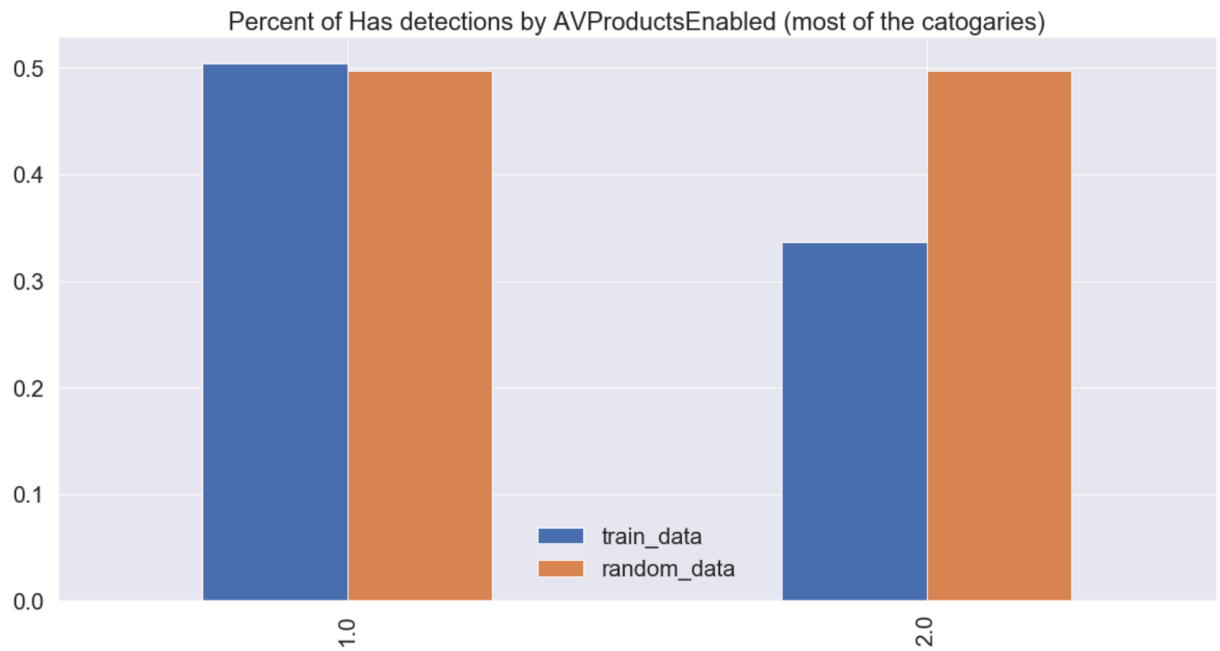

Figure 6: Bar plot for the distribution of the status of being affected by virus in different state of "AVProductsEnabled"

In table 5 and figure 6, we can again observe that the control group has uniformly distributed data, thus having 0.5 for each state. Comparing with the distribution of our original data, we do observe a difference, but we are not sure. Thus, we do the K-S test on the distribution. The test-statistic is 0.5, which has p-value of 0.8438198245415606 higher than then 0.05 alpha level, thus we cannot reject the null hypothesis that whether antivirus product is enabled does not have an influence in whether the machine is being affected or not. Therefore, if later we want to do a classification to predict status of being infected by virus, we are not sure about using this feature.

In order to validify our analysis, for each feature, we try to only use that feature to predict whether the machine is infected or not. We use random forest model to compute the accuracy and compare it to the control group. Noted, this is only a validation step to verify our analysis, the formal prediction on the target will be included in the later part.

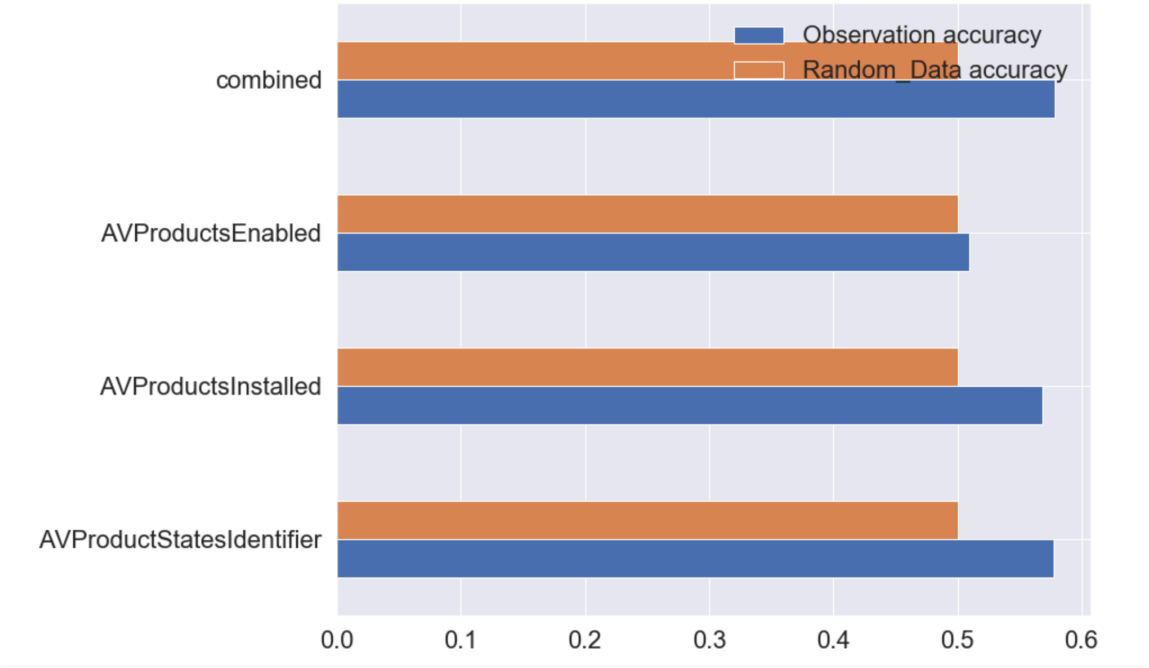

Figure 7: Barh plot for the accuracy for predicting being affected by virus through only using one feature

We can see that both the "AVProductsIdentifier" and "AVProductsInstalled" perform well in predicting, gaining an accuracy close to 0.6 compare to 0.5 for the control, while "AVProductsEnabled" has the accuracy rate around 0.5. In conclusion, we believe that if we want to pick features from antivirus product side, "AVProductsIdentifier" and "AVProductsInstalled are two good features to use.

Scenario 2: Operation System

Q: Has any particular operation platform been detected uncommon amount of virus? Or even more specifically, has any build of operation system been detected uncommon amount of virus? Does the uncommon number of detected virus suggest that this particular operation system is correlated to virus detection?

Operation Platform

From *Figure 8* below, we see that our data includes sample models that are mostly using operation system windows10. We can also observe that the number of computers that have virus detection is about the same as that of computers that do not for each kind of operation system.

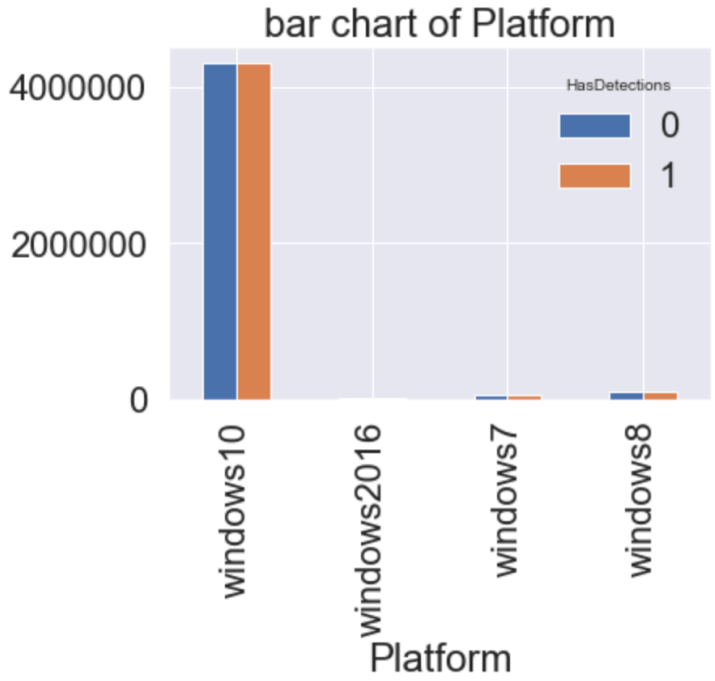

Figure 8. Bar Plot of Computers with Different Operation Platforms

From Figure 9 below, we can observe that percentage of virus detection in each operation platforms. In the figure, we also simulate a random distribution by shuffling the label of HasDetection. By the bar chart, we can see that the percentage of virus detection only deviates a little from randomly simulated percentage of detection. The biggest difference we see comes from the operation platform windows2016.

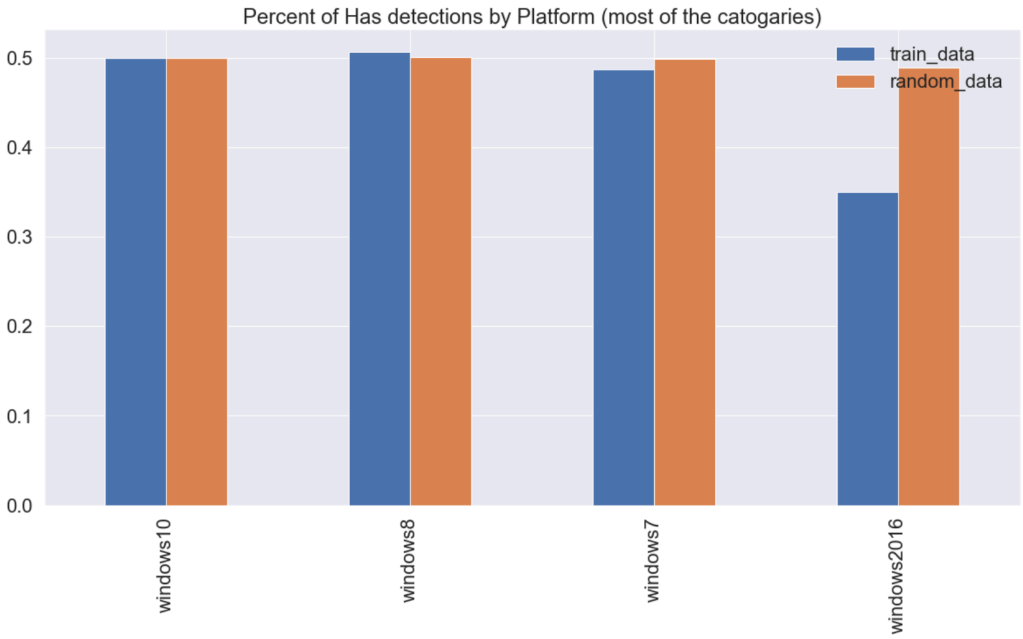

Figure 9. Percentage of Virus Detection in Each Operation Platform

To further investigate, we use Two-sample Kolmogorov-Smirnov test to test if the distribution of computers that have been detected virus for each operation system and the distribution of computers that have not been detected virus for each operation system follow the same distribution. Our null hypothesis is that both distributions follow the same distribution. The p-value we got for KS test is 0.107, which is greater than our significance level 0.05. This p-value indicates that we are unable to reject the null hypothesis that distributions of computers with each operation platforms, whether having been detected virus or not, follow the same distribution.

Operation Build

We also do the same investigation for builds of operation system. From *Figure 10* below, we can see that kernel density estimations of computers for each build are still similar whether they have been detected virus or not.

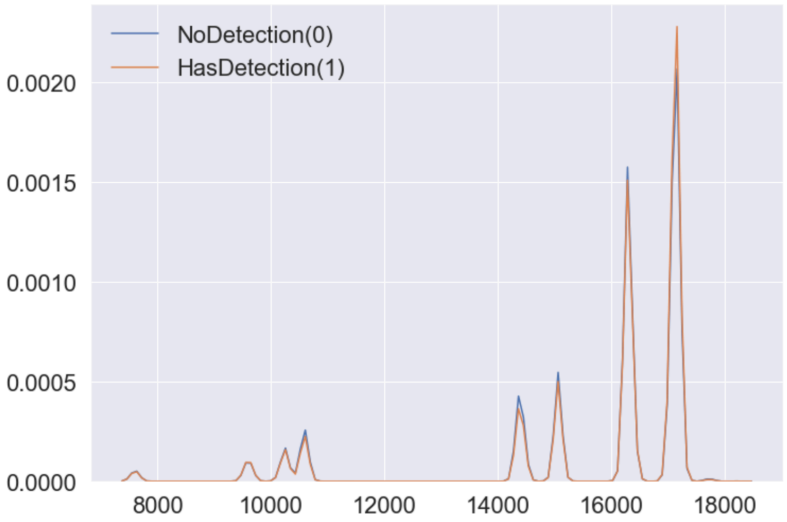

Figure 10. Kernal Density Estimation of Computers in Each Build

From *Figure 11* below, we can observe that percentage of virus detection in each operation build. Again, we also simulate a random distribution of rate of virus detection by shuffling the label of HasDetection. By the bar chart, we can see that, just like operation platform, the percentage of virus detection only deviates a little from randomly simulated percentage of detection.

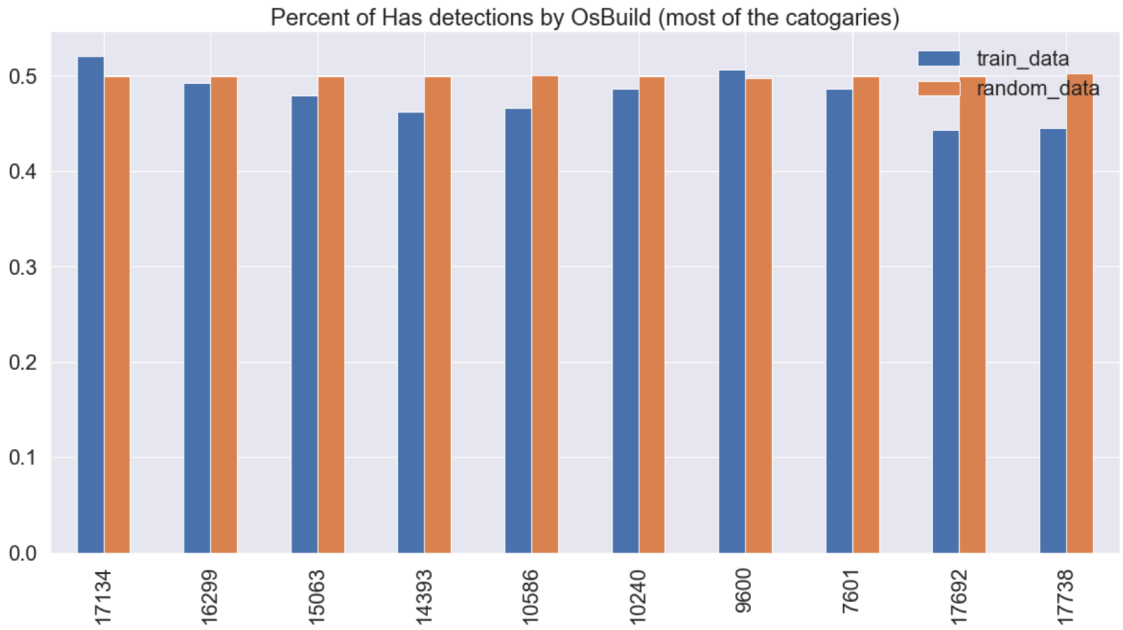

Figure 11. Percentage of Virus Detection in Each Operation Build

Then, we perform KS test for operation build. This time, we have p-value of 1.89e-05, which is smaller than our significance level 0.05, and we can reject the null hypothesis that the

distribution of computers in each build, whether having virus detection or not, follow the same distribution.

Random Forest Classifier

To further confirm the difference between distributions of computers with virus detection in each operation build and computers without, and also to test if operation build, or operation platform is a useful feature for malware detection, we use random forest classifier to test the accuracy of malware prediction using operation build. The table below shows the result of prediction accuracy of using operation build as a feature and using random classifier that label a computer as being detected virus randomly. From the table we see that our prediction accuracy of virus using operation build are almost the same as the random classifier, suggesting that although the distributions of computers in each build, whether having virus detection or not, follow the same distribution, using operation build as a feature does not better predict malware in computers compared to a random classifier.

	Observation accuracy	Random_Data accuracy
Platform	0.500503	0.500177
OsBuild	0.518036	0.500423
combined	0.518036	0.500423

Table 6. Prediction Accuracy of Virus with Operation System

Scenario 3: Hardware

Q: We are then interested in how hardware can affect our prediction of malware. Does larger disk capacity tell us anything about its likelihood of being detected virus? Does more core signify any message? How about processor? If they do, how do they together help our prediction?

Processor

From *Figure 12* below, we see that most of sample computer in our data are using x64 as processor, while a small proportion of them use x86. From the bar chart, we see that computers with x64 has slightly higher number of computers with virus detection compared, while computers with x86 has slightly lower number of computers with virus detection.

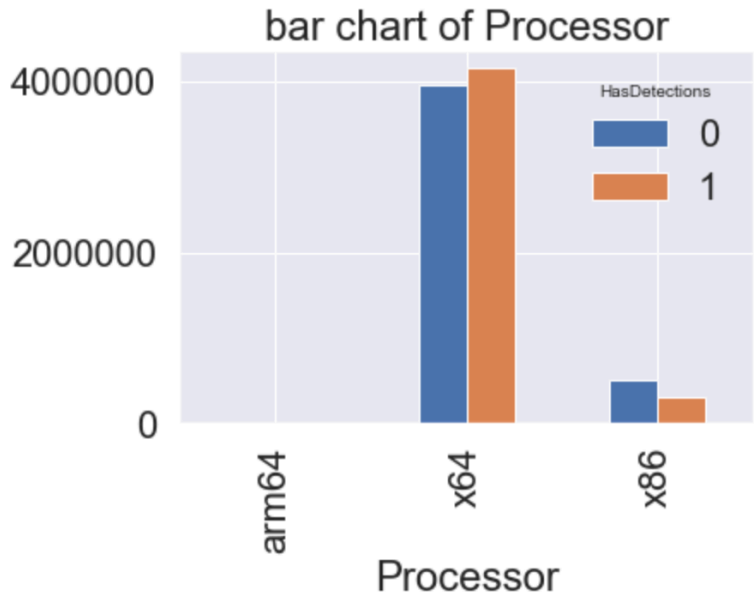

Figure 12. Bar Plot of Computers with Different Processors

From Figure 13 below, we can observe the percentage of virus detection in each used processor. In the figure, we also simulate a random distribution by shuffling the label of HasDetection. By the bar chart, we see that the percentage of virus detection for arm64 processor is very different from the simulated percentage. However, this might be due to the fact that our data includes few computers using arm64 as processor. Still, we see that the percentage of virus detection for computers using x86 are lower than the simulated percentage.

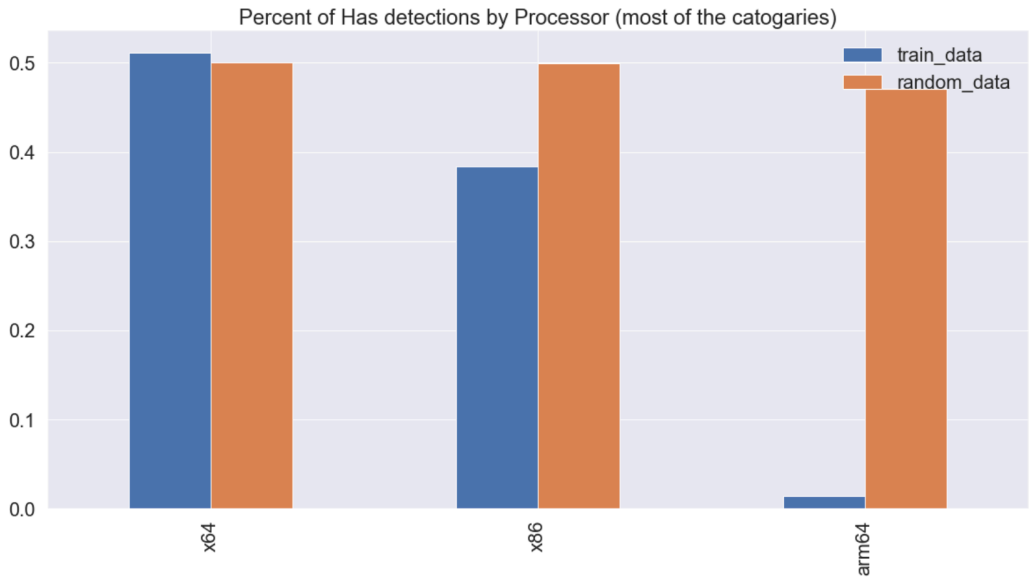

Figure 13. Percentage of Virus Detection in Each used Processor

To further investigate, we again use Two-sample Kolmogorov-Smirnov test to test if the distribution of computers that have been detected virus for each used processor and the distribution of computers that have not been detected virus for each processer follow the same distribution. Our null hypothesis is that both distributions follow the same distribution. The p-value we got for KS test is 0.32, which is greater than our significance level 0.05. This p-value indicates that we are unable to reject the null hypothesis that distributions of computers with each used processor, whether having been detected virus or not, follow the same distribution.

Processor Core Count

We also want to see if numbers of logical cores in processor affect out prediction result. From *Figure 14* below, we can see that kernel density estimations of computers for processor core count are somehow different between those with virus detections and those without.

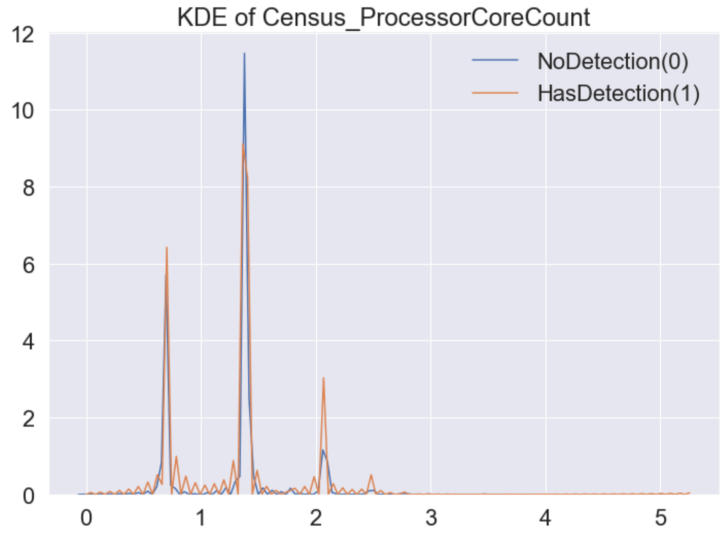

Figure 14. Kernel Density Estimation of Computers for Processor Core Count

From *Figure 15* below, we can observe the percentage of virus detection in different number of cores in processor. By the bar chart, we can see that computers with some particular number of cores in processor show different rate of virus detection compared to randomly simulated rate.

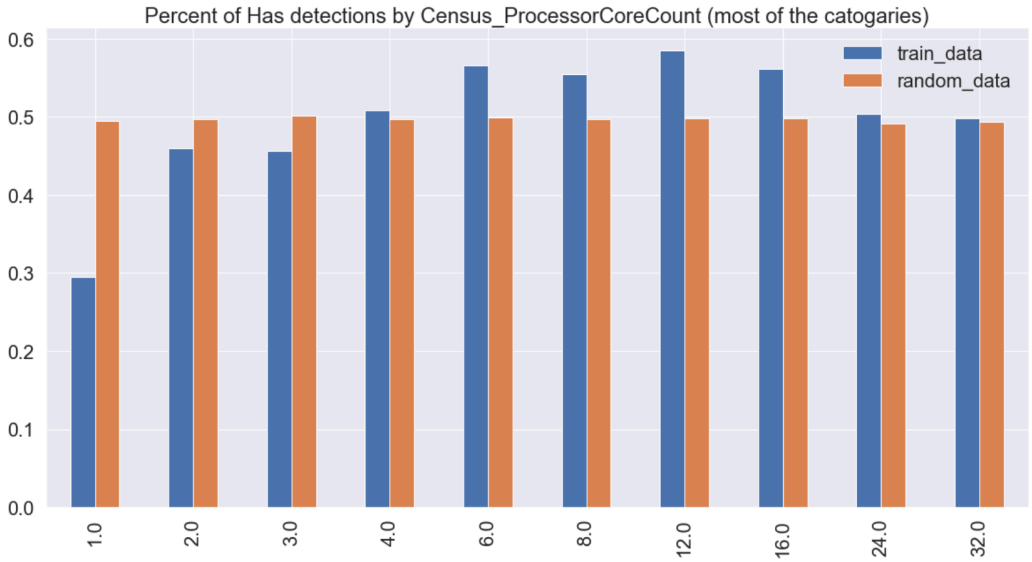

Figure 15. Percentage of Virus Detection in Different Amount of Processor Cores

Thus, we perform KS test to confirm our graphical analysis. The test result shows a p-value of 0.00017, which is smaller than our significance level 0.05, and we can reject the null hypothesis that the distribution of computers for different number of processor cores, whether having virus detection or not, do not follow the same distribution.

Disk Capacity

Then, we would like to see if disk capacity affects out prediction result. From *box plot* below, we can see that the distributions of disk capacity between computers with and without virus detection are very similar, with computers without virus detection having lower range of disk capacity. This is probably due to the fact that disk capacity essentially comes from a fixed number of same choices.

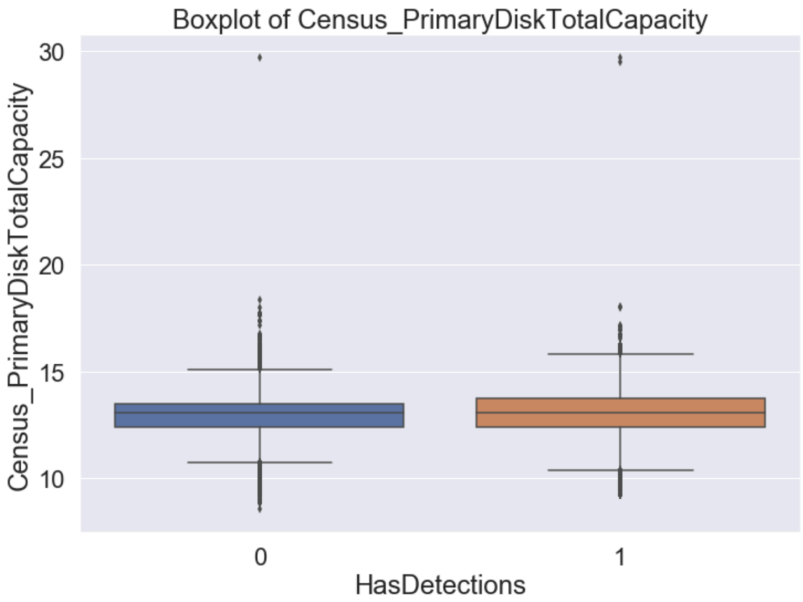

Figure 16. Box plots of Disk Capacity for Computers with or without Virus Detection

From *Figure 17* below, we can observe the percentage of virus detection in different disk capacity. By the bar chart, we can see that computers with some particular disk capacity show different rate of virus detection compared to randomly simulated rate.

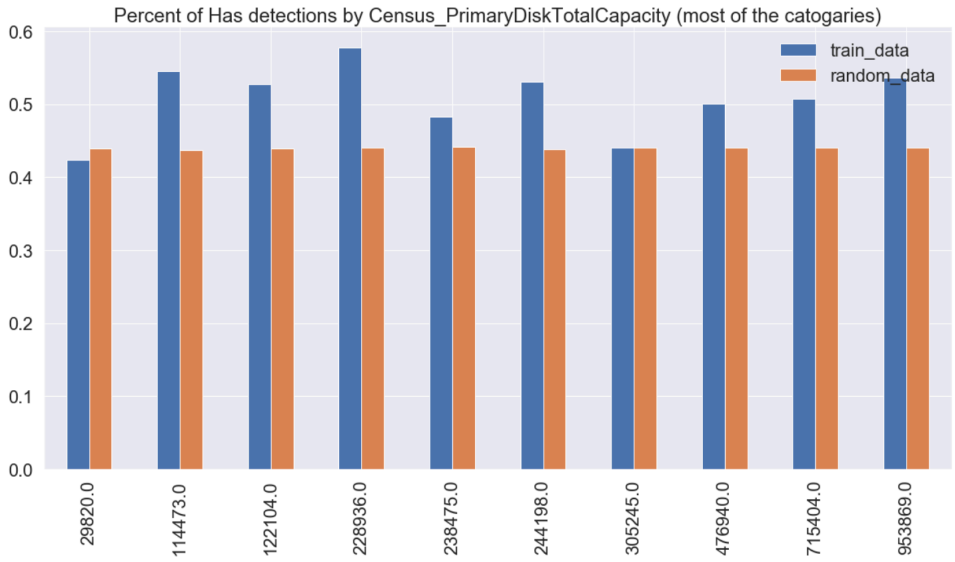

Figure 17. Percentage of Virus Detection in Different Disk Capacity

We will perform KS test again to confirm our graphical analysis. The test result shows a p-value of 1.88e-05, which is smaller than our significance level 0.05, and we can reject the null hypothesis that the distribution of computers for different disk capacity, whether having virus detection or not, do not follow the same distribution.

Random Forest Classifier

We will then use random forest classifier, with all three columns of hardware as features, to cross-validate our prediction accuracy for malware in computers. The table below shows the result of prediction accuracy of using each column of hardware as a feature, along with using all three, and using random classifier that label a computer as being detected virus randomly. From the table we see that our prediction accuracy of virus using hardware perform slightly better predictions compared to the random classifier, suggesting that although distributions of computers with or without virus detection for different processor types follow the same distribution (from KS test), the feature of processor is still useful with other features of hardware in malware prediction.

	Observation accuracy	Random_Data accuracy
Census_ProcessorCoreCount	0.524082	0.499893
Census_PrimaryDiskTotalCapacity	0.533193	0.499547
Processor	0.521055	0.49966
combined	0.543938	0.499912

Table 7. Prediction Accuracy of Virus with Hardware

Scenario 4: Defender State

Q: Do Beta versions more likely to be infected by virus? Do different types of defender perform differently at blocking viruses?

In this scenario, we are going to investigate the features of defender state and whether they are influential to computers being infected by virus. After performing some basic cleaning to our dataset, we choose the features of "IsBeta" and "ProductName" as the two most relevant features among all to have affected the performance of the systems at blocking virus. "IsBeta" identifies whether the system is a beta version, with 0 as official model and 1 as beta model. A Beta model may be more easily to be infected by virus than an official model that has been fully tested and optimized. "ProductName" provides the information of the types of defender being installed in the computers. Different types of defender are probably preforming differently at blocking virus.

First, we do our analysis based on "IsBeta." From the table and bar chart below, we see that our training set has a uniform distribution while our random set has a nearly uniform distribution. Comparing the two distributions, the difference is small. We can also confirm by doing a K-S test. The test statistic is 0.5 with a p-value of 0.8438198245415606. Since the p-value

is much higher than the type I error rate of 0.05, we choose to believe the null hypothesis that the two distributions are very similar, in which having a beta model will not give a higher chance of being infected by virus.

	Train_data	Random_data
0	0.499793	0.499793
1	0.492537	0.499793

Table 8: Table of Proportion of Machine Being Detected According to "IsBeta"

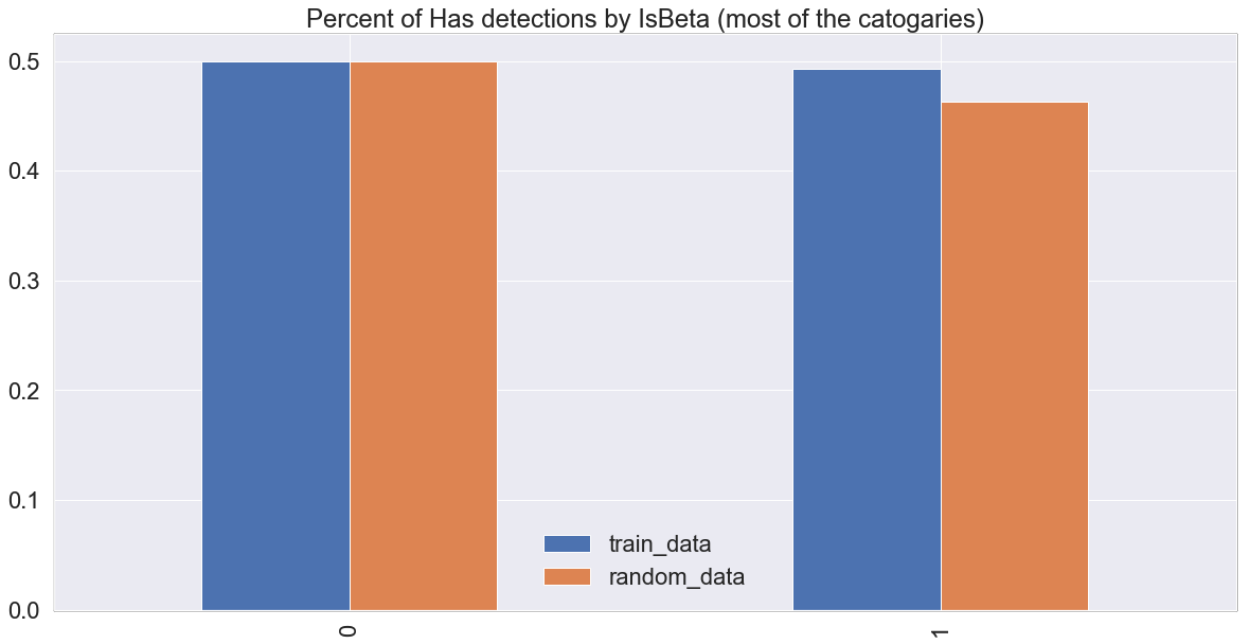

Figure 18: Bar Chart of Distribution of Machine Being Detected According to "IsBeta"

As having a beta version or not does not make a significant difference in machines being infected by virus, we then want to make investigation on the performances of difference types of defender at blocking virus. There are total of six different types of defender in our training set. By looking at the table and bar chart below, neither our training set nor random set are uniform. In our training set, there is an exceptional low detection rate with the defender named windowsintune, while the random set has an extra high detection rate with the same defender. Thus, the two distributions are different. To confirm, we again perform the K-S test with a test statistic of 0.8333333333333334 and get a p-value of 0.012238153125878112. The p-value is smaller than $\alpha = 0.05$, we reject the null hypothesis and think that different types of defender may perform differently at blocking virus.

	Train_data	Random_data
fep	0.428571	0.571429
mse	0.484448	0.499626
mseprerelease	0.490566	0.603774
scep	0.454545	0.409091
Win8defender	0.499958	0.499794
Windowsintune	0.125000	0.750000

Table 9: Table of Proportion of Machine Being Detected According to "ProductName"

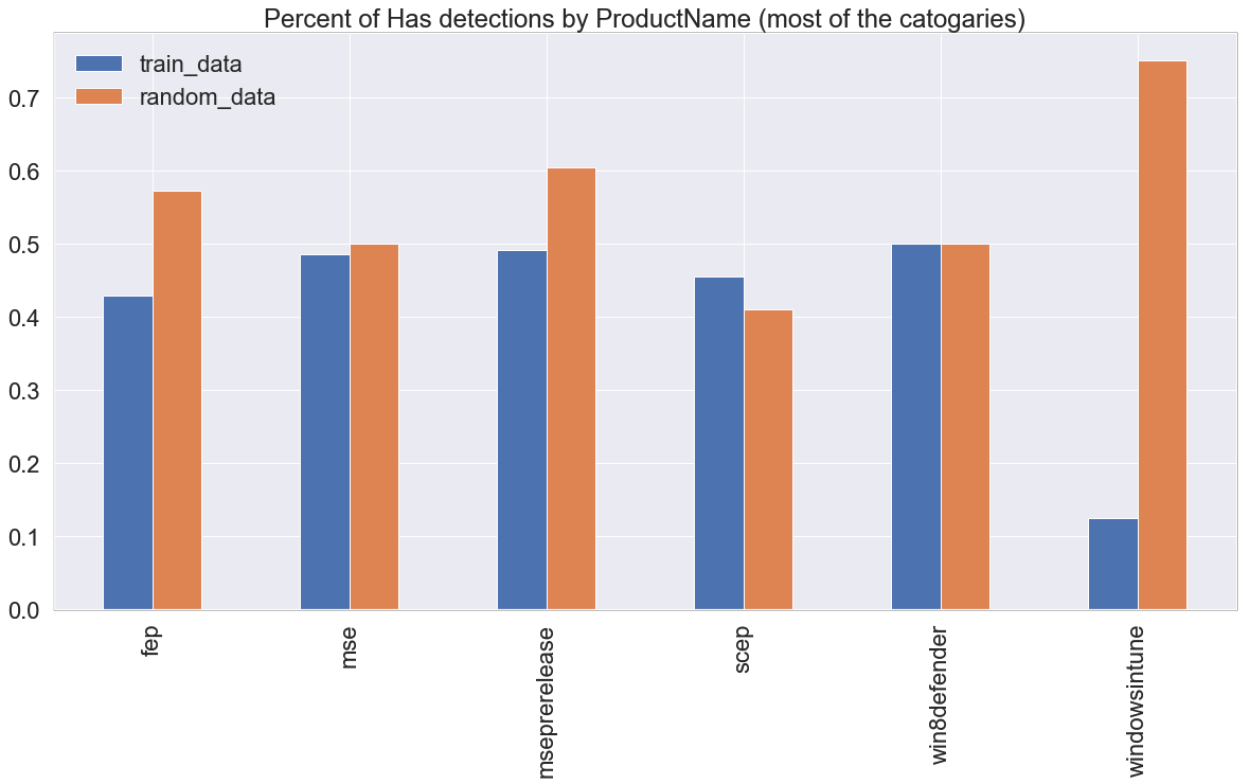

Figure 19: Bar Chart of Distribution of Machine Being Detected According to "ProductName"

Yet, we still want to validate our conclusions from the above two K-S tests. We thus use random forest model from sklearn library along with each feature individually to make prediction of machine infection. From the bar chart below we can see that both observation accuracy and random data accuracy are uniformly distributed at 0.5, which means that both features are not

useful feature in predicting machine infection. Thus, we cannot include either of "IsBeta" or "ProductName" in our later classification. Even though we reject the null hypothesis in the K-S test of "ProductName," it is still not a very effective feature in predicting whether machine is infected by virus or not.

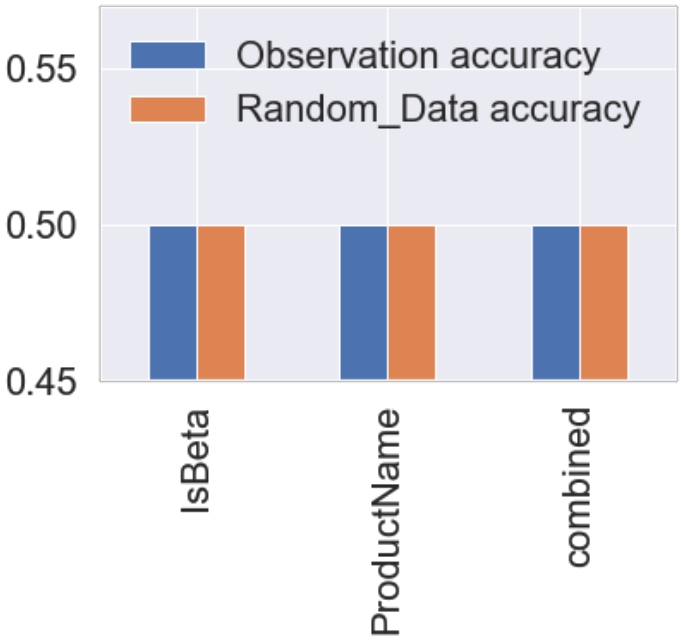

Figure 20: Bar Chart of Random Forest Predictions of Machine Infection Accuracy with Respect to Features of Defender State

Scenario 5: Geography

Q: Is geography an effective factor of machine virus infection? Do computers at various locations or countries have different infection rate?

In the last scenario, we would like to explore the relationship of machine infection with the features of geography. We wonder if computers at certain areas will have different infection rate than those from other areas. We choose two geo-related features from the training data named "GeoNameIdentifier" and "CountryIdentifier." "GeoNameIdentifier" gives area code of the location of every machine. "CountryIdentifier" gives country code of every machine. These two features provide a fairly detailed information about the geography of each computer. Because of the difficulty of allocating such gigantic dataset, we choose 40 most common codes from both "GeoNameIdentifier" and "CountryIdentifier" as our test samples.

First, we do our analysis based on "GeoNameIdentifier." From bar chart below, we see that our random dataset is uniformly distributed while out training set is not. Comparing the two distributions, the difference is obvious. To confirm our observation, we perform a K-S test again.

The test statistic is 1.0 with a p-value of 6.133847783205273e-19. Since the p-value is extremely small and smaller than the type I error rate of 0.05, we confirm that the two distributions are significantly different, and we reject the null hypothesis and tend to believe locations probably is an effective factor of machine infection.

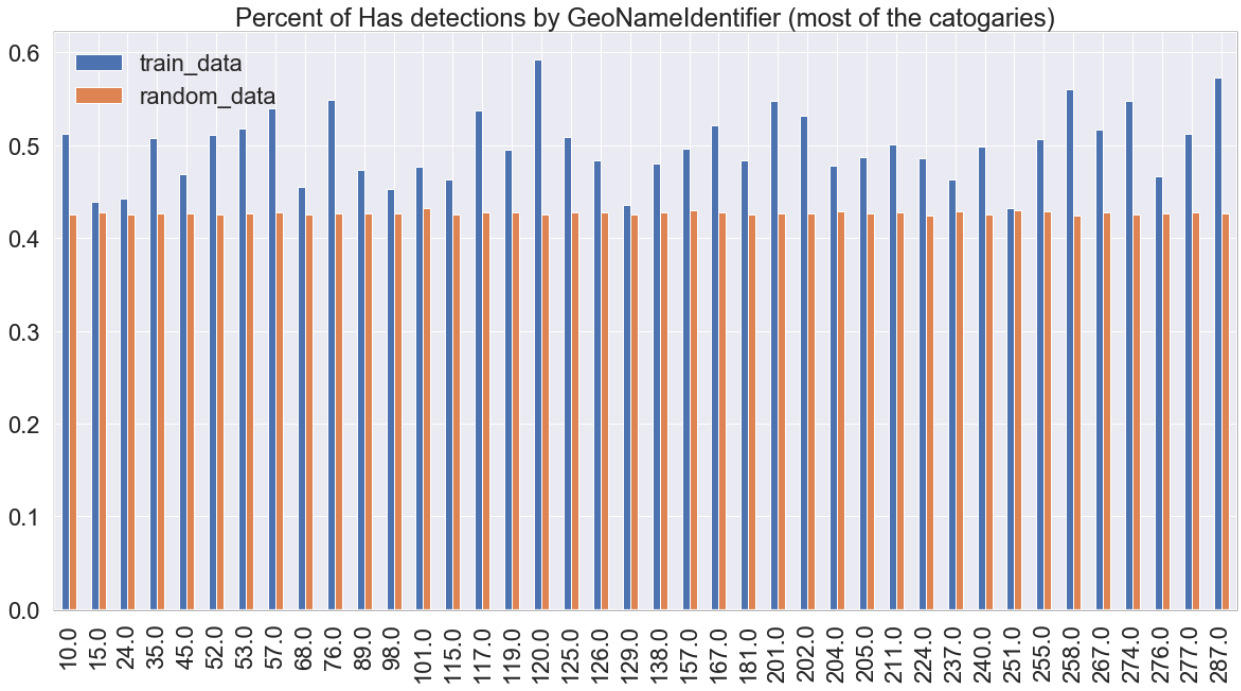

Figure 21: Bar Chart of Distribution of Machine Being Detected According to "GeoNameIdentifier"

With the above test, we have somewhat expectation of a similar conclusion with the test of "CountryIdentifier." From the bar chart below, we again see a uniformly distributed random set and a non-uniformly distributed training set. The two distributions are apparently different from one another. A K-S test with test statistic of 1.0 gives a p-value of 6.133847783205273e-19. Since the p-value is significantly lower than 0.05. We reject the null hypothesis, in which machines from different countries do have different virus infection rate.

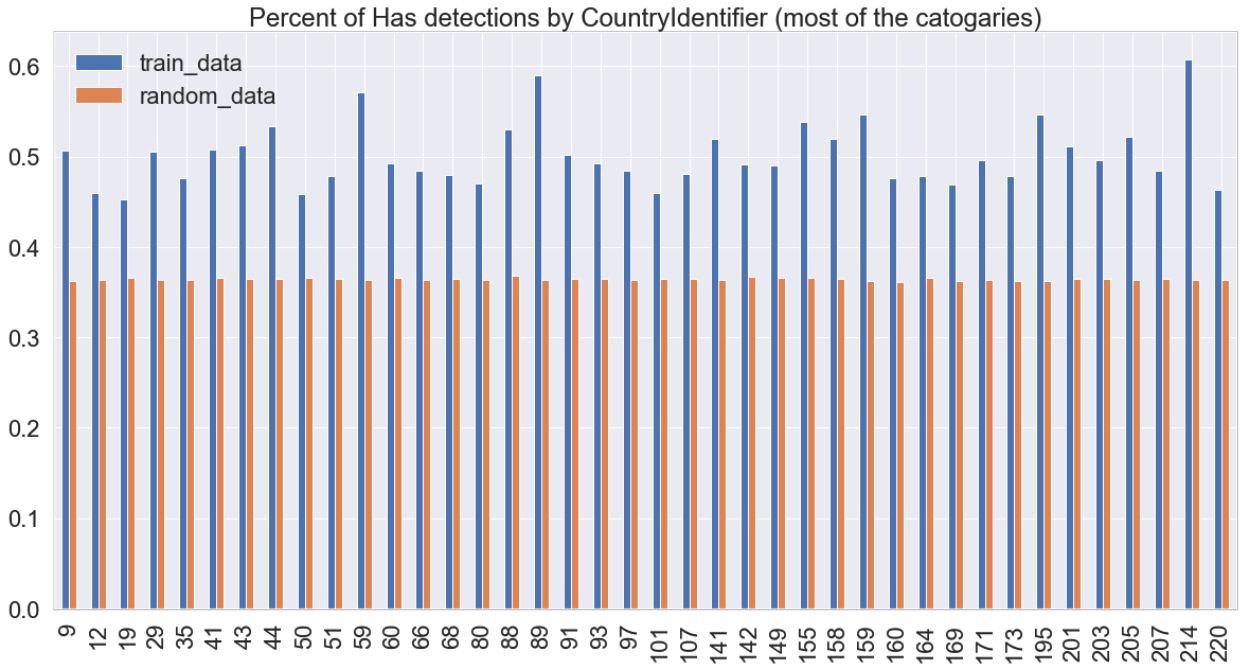

Figure 22: Bar Chart of Distribution of Machine Being Detected According to "CountryIdentifier"

To verify our conclusions from the above two K-S tests, we use random forest model from sklearn library along with each feature individually to make prediction of machine infection. From the below bar chart, we can see that as the random data accuracy is uniformly 0.5, both "GeoNameIndentifier" and "CountryIdentifier" produce predictions with higher accuracy. At this point, we are confident to include both features in the classification portion since both features are influential to computer infection. In conclusion, geography an effective factor of machine virus infection.

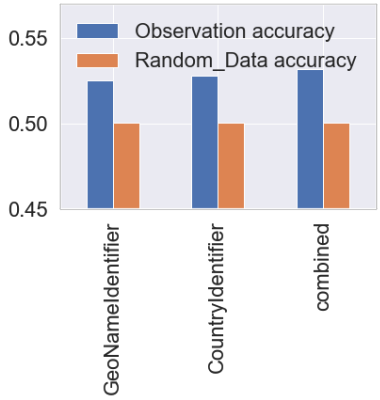

Figure 23: Bar Chart of Random Forest Predictions of Machine Infection Accuracy with Respect to Features of Geography

To more intuitively visualize the difference of machine affection rate according to various area codes and countries, we plot two kernel density estimation graphs respectively. From the graphs below, we can clearly visualize that the distribution of no detection and that of with detection do not align with each other in both plots.

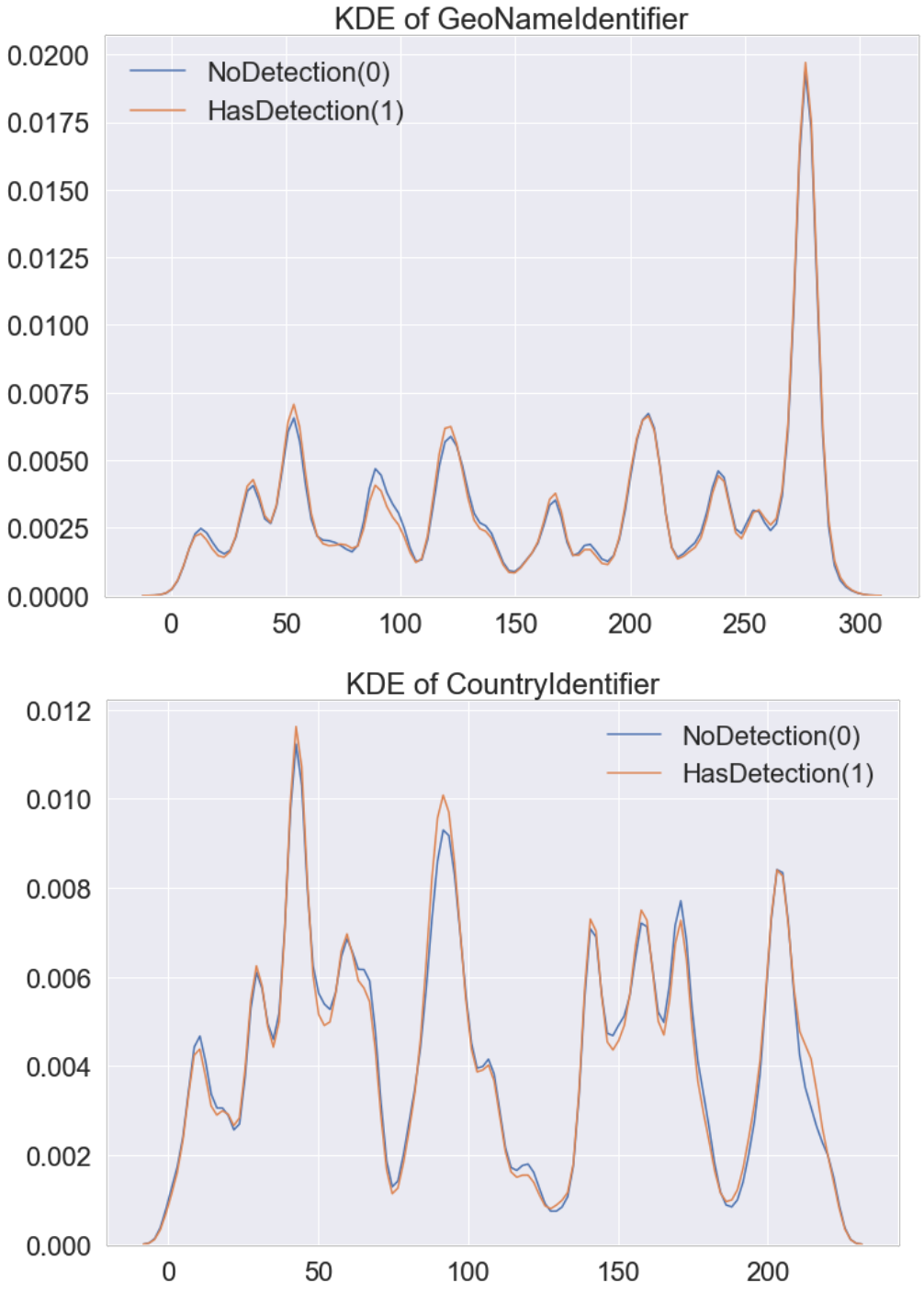

Figure 24: KDE Plot of "GeoNameIdentifier" and "CountryIdentifier"

Classification:

Objective: we want to reach highest accuracy that we can get using the important and related features that we analyzed.

From scenario 1, 2, 3, 4, and 5, we search for the related features to the status of machine whether being infected by virus from antivirus, operating system, hardware, defender state and geography aspects. Combining all the features that are useful, we get the lists of columns. They are 'AVProductStatesIdentifier', 'AVProductsInstalled', 'GeoNameIdentifier', 'CountryIdentifier', 'OsBuild', 'Census_ProcessorCoreCount', 'Census_PrimaryDiskTotalCapacity', and 'Processor'. Altogether, there are eight features, which either is identified by the K-S test or marked by its accuracy in classifying the virus status. Therefore, we want to use these features to train our model and to see if we can get good accuracy with them.

Before we use any classifier, we preprocess the data by using one hot encoding method to transform the related data to either 0 or 1 of the data in the column. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Since there is no meaning of averaging the categorical values, it is better to transform the data using the one hot encoding method. After that, we split our data into 75:25, training set and validation set. And then, we use three different models to train and fit our data. The three classifiers are Linear SVC, Linear SVC with stochastic gradient descent (SGD) learning, and Random Forest. Since our data is so huge, we also prepare a control group containing the data coming from random generated sample of whether machine being infected by the virus or not.

Linear SVC: Observation training accuracy: 0.500329, random data accuracy: 0.500157

SGDC: Observation training accuracy: 0.595651, random data accuracy: 0.500559

Random Forest: Observation training accuracy: 0.593908, random data accuracy: 0.499594

We observe that the control group, the random generated data has average accuracy around 0.5, which is exactly what we expected, as the data is so huge. By comparing the accuracy for all of the three classifiers, we found that except LinearSVC, the SGDC and Random Forest with those eight features significantly increase the chance of predicting whether the machines are infected by the malware. We do not know exactly the reason why the Linear svc does not work well in this dataset, but we guess it is due to the fact that the malware affects multiple dimensions of the machines, which could not be measured by the linear relationship. Thus, it is not appropriate to address this question/problem with single linear model.

Since we only use the features that we believe significantly important to predict whether the machine is infected by the malware, we want to discover if more accuracy could be done by including other relatively less important features. We import the package of recursive feature elimination from sklearn. The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. After fit and transform our data, we found 50 significant features, which includes all the 8 features we found in the previous scenario study. The top 20 features are “CountryIdentifier”, “Census_OSVersion”, “GeoNameIdentifier”, “Census_OSBuildRevision”, “OsBuildLab”, “LocalEnglishNameIdentifier”,

“Census_FirmwareManufactuerIdentifier”, “AppVersion”, “AVProductStatesIdentifier”, “SmartScreen_AVProductsInstalled”, “AvSigVersion”, “AvSigVersion_2”, “Census_OEMModelIdentifier”, “Census_FirmwareVersionIdentifier”, “non_primaru_drive_MB”, “Census_SystemVolumnTotalCapacity”, “CityIdentifier”, “new_num_1”, “Census_ProcessorModelIdentifier”, and “Census_OSVersion_3”. We can see from the label, our scenarios are in good direction, since most of the labels are associated with geography, machine versions and antivirus status. After getting all the features, we proceed to use random forest classifier, which is proven to be useful in the previous classification with our 8 features. The training accuracy rise to 0.67, significantly improved from our own classification, which means that if Microsoft needs to improve their defenders, the 20 features that they need to pay special attention to are the features we mentioned above.

Discussion & Analysis:

Our goal in this project is to find related features and then use them to make classification on whether the machine is infected by the virus or not. We divide our task into 5 parts, scenario 1 talks about antivirus, scenario 2 is about operating system, scenario 3 is about hardware, scenario 4 is about defender state, and scenario 5 is about geography. At the end, we use all the features that we think important to train and fit the data and then use to predict. We also include an automatic feature selection method to reconfirm our analysis and result.

From scenario 1, we first want to analyze the effect of the features of antivirus product on the status of being affected by virus. After comparing three features by using tabular value, graphical analysis and statistical analysis (K-S test) for “AVProductsIdentifier”, “AVProductsInstalled”, and “AVProductsEnabled”, we find out that there is no significant influence that “AVProductsEnabled” would bring to the status of being affected by virus. It contradicts to our normal belief that if we enable our antivirus product, the machines we use will be free of the virus. For “AVProductsInstalled”, even though the distribution is not different from the control group indicated by the K-S test, it still has high accuracy in predicting the status of being affected by virus. For “AVProductsInstalled”, K-S test clearly reject the null hypothesis that it does have effect on the status of being affected by virus. Therefore, for the later classification part, two features about antivirus products part that needed to be included are “AVProductsIdentifier” and “AVProductsInstalled.

From scenario 2, we see that distributions of computers with each operation platform, whether with virus detection or now, essentially follow the same distribution according to the KS test. On the other hand, distributions of computers with each operation build, whether with virus detection or now, do not follow the same distribution according to the KS test. However, by comparing malware prediction accuracy with random forest classifier using operation platforms and builds as features with the accuracy with random classifier, we see that operation system is not the best choice to infer about malware.

From scenario 3, we see that distributions of computers with each processor, whether with virus detection or now, essentially follow the same distribution according to the KS test. On the other hand, distributions of computers with different number of logical cores in processor and disk capacity, whether with virus detection or now, do not follow the same distribution according to

the KS test. However, by combining all three features of hardware, we see that we have better malware prediction accuracy using random forest classifier compared to random classifier that assigns label of virus randomly.

From scenario 4, we first want to investigate whether beta model is more likely to be infected by virus compare to official model. Though we think that beta model may be weaker at blocking virus, after some graphical analysis and K-S test analysis, we reach to the conclusion that beta model and official model are infected by virus with same chance. Then, we would like to know if different types of defender perform differently at blocking virus. Doing similar graphical analysis and statistical analysis, we find out that different types of defenders perform differently at blocking virus. However, after using random forest model from sklearn library along with each feature individually to make prediction of machine infection. We see that both features are not effective in predicting whether machine is infected by virus or not, and we cannot include either of them in classification despite the results from K-S tests.

From scenario 5, we would like to make investigation on the influence of geography to machine virus infection. We perform our graphical analysis and statistical analysis to the feature of "GeoNameIdentifier" and "CountryIndentifier." The conclusions we draw from both K-S tests are to reject null hypothesis, in which computers at various locations or countries have different infection rate. The random forest model also confirms the conclusions. Thus, we will include both features in further classification.

From classification, we combine all the important features from the previous scenerios. They are 'AVProductStatesIdentifier', 'AVProductsInstalled', 'GeoNameIdentifier', 'CountryIdentifier', 'OsBuild', 'Census_ProcessorCoreCount', 'Census_PrimaryDiskTotalCapacity', and 'Processor'. Using these 8 features, we tried three different classifiers, Linear SVC, Linear SVC with stochastic gradient descent (SGD) learning, and Random Forest. We found out that single linear model does not work well in detecting the malware, the best classifier we get is the random forest classifier. After using the recursive feature elimination (RFE) method to automatically select features, we found that all eight features are included in the features generated by the RFE, which means that our analysis are correct. There are 20 different features that have high weight in classifying malware, (labels are in Classification part) if Microsoft needs to improve their defenders, they need to pay special attention to those features.

Data limitation: Since our dataset is so huge, we could only use part of the datasets or features to predict malware as we do not obtain knowledge of handling huge data. If we could have more time or more resource to GPU, we can finetune our model in order to reach even higher accuracy. Also, the features are not enough, as we can see from the other Kaggle team, no one can reach 0.8 or above accuracy rate. If Microsoft could release more information under privacy condition, we can obtain a better classifier.

Theory:

Goal:

a. To determine the rank of weight of different features that affect the detection of malwares in selecting features.

- **Mean Square Error**
  - The mean square error of an estimator $\hat{\theta}$ for a parameter θ is
    $$MSE(\hat{\theta}) = E[(\widehat{\theta} - \theta)^2]$$
  - It measures the average of the squares of the errors – that is, the average squared difference between the estimated values and what is estimated.
  - The MSE is a measure of the quality of an estimator – it is always non-negative, and values closer to zero are better.
  - Many people memorize MSE as Var + Bias squared.
  - Many of the estimators we use are UNBISED, but sometimes an estimator with a small bias will have a small MSE.
  - Theorem: Under certain regularity conditions, as the sample size increases, the Maximum-likelihood estimator, $\hat{\lambda}$ satisfies
    $$\hat{\lambda} \sim N(\lambda, \frac{1}{nI(\lambda)})$$
    Where $I(\lambda)$ is called the Fisher's Information Matrix.

- **Boxplot**: This plot provides an effective means of displaying the range of the data (minimum to maximum), likely range of the data (IQR, from lower quantile to upper quantile) and the typical value (the median). Also, box plot can clearly show the outliers (more than 3IQR away from the box) and suspected outliers (more than 1.5IQR away from the box)

- **Central Limit Theorem:**
  - If $X_1, \ldots, Xn$ are independent, identically distributed with mean μ and variance σ^2 then, for large n, the probability distribution of $Z = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$ is approximately standard normal, no matter if X_i is normal distributed or not.

- **The Probability Model**:
  - By the rule of simple random sample, each unit in population has the same chance (n/N) of being in the sample. However, there is dependence between selections.
    $P(\text{unit \#1 in the sample}) = \frac{n}{N}$
    $P(\text{ unit \#1 and unit \#2 are in the sample}) = \frac{n(n-1)}{N(N-1)}$
  - In general, let I(1), I(2), … represent the first, second, … number drawn from the list 1, 2, …, N. Then, $P(I(1) = j_1, I(2) = j_2, \ldots, I(n) = j_n) = \frac{1}{N(N-1)\ldots(N-n+1)}$

- **Smoothing**
  - Smoothing extracts trends from data by reducing the variance of nearby observations.
    - Origins in engineering (audio/images)
    - Good for visualization
    - Careful building inferential models on smoothed data!
    - inferential models assess what is/isn't noise

- Additive Smoothing
  - Given a dataset of observations x of size N, with d categories,
  - Smooth the empirical probability a value occurs:
    - $\mathrm{p_i} = \frac{\mathrm{x_i}}{N}$
    - $p_i = \frac{(x_i + a)}{N + ad}$
  - Where α reflects a guess that each category has an additional count α
  - Where $1/d$ is the uniform probability, if each category is equally likely.
- Conditional Additive Smoothing
  - examine a categorical attribute x using a second, Boolean attribution b.
    - $p_i = \frac{b_i}{x_i}$
    - $p_i = \frac{b_i + a * \left(\frac{b}{N}\right)}{x_i + a}$
  - Additive smoothing interpolates between:
    - incidence rate of b per group, and
    - overall incidence rate b/N.
  - The smoothing factor has the same interpretation as the unconditional case.
    - b/N scales α by the incidence rate.

- **Missing value**
  - Missing by Design (MD):
    - The field being absent is deterministic.
    - A function of the rows of the dataset that can:
      - exactly predict when a column will be null,
      - with only knowledge of the other columns.
  - Missing Completely at Random (MCAR):
    - The missing value isn't associated to the (actual, unreported) value itself, nor the values in any other fields.
      - The missingness is unconditionally uniform across rows.
      - Example 1: additional questions in a survey are posed on a random sample of respondents.
      - Example 2: Water damage to paper forms prior to entry (assuming shuffled forms).
  - Not Missing at Random (NMAR):
    - A missing value depends on the value of the (actual, unreported) variable that's missing.
    - Example 1: people with high income are less likely to report income.
- **Data imputation**
  - Listwise deletion
    - Procedure: drop rows with null value
    - If MCAR, doesn't change statistics of the data
    - If MCAR and small, may have high variance
  - Mean imputation
    - Procedure: fill null value with mean value.

        - If MCAR, gives unbiased estimate of mean; variance is too low.
        - Analogue for categorical data: imputation with the mode.
    - conditional mean / regression imputation
        - Procedure: fill null with mean conditional on a column
        - If MAR, gives unbiased estimate of mean; variance is too low.
        - Increases correlations between the columns.
        - If dependent on more than one column: use linear regression to predict missing value.
    - Probabilistic imputation
        - Procedure: draw from empirical distribution of observed data to fill missing values.
        - If MCAR, gives unbiased estimate of mean and variance.
        - Extending to MAR case: draw from conditional empirical distributions
        - if conditional on a single categorical column c2:
    - Multiple imputation
        - Procedure:
            - Apply probabilistic imputation multiple times, resulting in N imputed datasets.
            - Do analyses separately on the N imputed datasets (e.g. compute correlation coefficient).
            - Plot the distribution of the results of these analyses

- **Linear models with stochastic gradient descent**
    - stochastic gradient descent
        - a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression
        - The advantages of Stochastic Gradient Descent are:
            - Efficiency.
            - Ease of implementation (lots of opportunities for code tuning).
        - The disadvantages of Stochastic Gradient Descent include:
            - SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
            - SGD is sensitive to feature scaling.
    - This estimator implements regularized linear models with stochastic gradient descent (SGD) learning.
        - $E(w, b)$ by considering a single training example at a time.

$$E(w,b) = \frac{1}{N}\sum_{i=1}^{n} L\big(y_i, f(x_i)\big) + \alpha R(w)$$

            - where L is a loss function that measures model (mis)fit and R is a regularization term (aka penalty) that penalizes model complexity; $\alpha > 0$ is a non-negative hyperparameter.

        - the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning.

- **Feature selection with Recursive feature elimination**
    - Feature selection module from sklearn
        - module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.
    - Recursive feature elimination
        - selecting features by recursively considering smaller and smaller sets of features.
        - Procedure:
            - Initialize an estimator that is trained on the initial set of features and the importance of each feature.
            - the least important features are pruned from current set of features.
            - Repeat on the pruned set until the desired number of features to select is eventually reached

- **Kolmogorov–Smirnov Test:**
    - The Kolmogorov–Smirnov test is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K–S test), or to compare two samples (two-sample K–S test).
    - According to Method of Moments, the empirical distribution function F_n for n i.i.d observations X_i is defined as $F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I\{X_i > x\}$, where $F_n(x) \rightarrow F(x)\ as\ n \rightarrow \infty$ by the Law of Large Number.
    - For the one-sample K–S test, we have $H_0: p = p_0\ and\ H_a: p \neq p_0$. The test statistic $D_n = \sqrt{n}Sup|F_n(x) - F(x)| = \sqrt{n}Max\{Max\left\{\left|F(X_i) - \frac{i}{n}\right|, \left|F(X_i) - \frac{i-1}{n}\right|\right\}\ for\ 1 \leq i \leq n\}$. Reject H_0 when $D_n \geq s$.
    - For the two-sample K–S test, we have $H_0: p_x = p_Y\ and\ H_a: p_X \neq p_Y$. The test Statistic $D_{n,m} = \sqrt{\frac{nm}{n+m}}Sup|F_n(x) - G_m(x)|$. Reject H_0 when $D_{n,m} \geq s$.
- **Histogram**:
    - A histogram consists of rectangles whose areas represent the percentages of the data lying in the specific interval. It provides an accurate representation of distributions of numerical data and an estimate for the probability distribution of population.
- **Monte Carlo Simulation**
    - Monte Carlo simulations are used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random

variables. The essential idea is using randomness to solve problems that might be deterministic in principle.
    - In our study, we applied Monte Carlo simulation and randomly generated a new data set to apply Kolmogorov–Smirnov Test on our training set.
- **Probability Method**:
    - Probability method provides us a scientific way to do select sample with knowing the relation between the sample and the population. Meanwhile, we can get the chance of each possible example. In our study, **we applied simple random sample.**
- **Addictive Smoothing:**
    - Smoothing is a technique to extract trends from data by reducing the variance of nearby observations.
    - For a given dataset off observations x of size N with d categories, the empirical probability a value occurs is $p_i = \frac{x_i}{N}$. We smooth the probability by defining $p_i = \frac{x_i+\alpha}{N+\alpha d}$, where α is a constant reflecting a guess that each category has an additional count α.
- **Decision Trees:**
    - Decision Trees is a popular method to solve classification or regression predictive modeling problems. The tree is represented by a series of binary splits where each internal node represents a value query on one of the variables — e.g. "Is X_3> 0.4". If the answer is "Yes", go right, else go left. The terminal nodes are the decision nodes. For classification tasks, each terminal node is typically dominated by one of the classes. The tree is grown using training data, by recursive splitting. New observations are predicted by passing their X down to a terminal node of the tree, and then using majority vote for classification or taking average for regression.
    - The advantage of using Decision Trees includes:
        - It can handle huge datasets
        - It can handle mixed predictors, both quantitative and qualitative
        - It can easily ignore redundant variables
        - It can handle missing data elegantly
- **Random Forests:**
    - Random Forests is another popular algorithm for classification or regression predictive modeling problems. The algorithm predicts the value by building multiple trees and combining the results into a final one.
    - The training process of random forests applies some general techniques including bootstrap, bagged trees/ decision trees:
        - Here N is the number of data points in the training set; p is the number of features; m is number of features to be considered at each tree split, typically m = $\sqrt{p}$.
        - To grow a tree, we first need to generate a training set by sampling N data points from our original dataset (bootstrap sampling). For each tree grown

on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored. This is called the "out-of-bag" error rate.
        - At each tree split, a random sample of m features is drawn, and only those m features are considered for splitting.
    - Random forests try to improve on bagging by "de-correlating" the trees. Each tree has the same expectation.
- **One Hot Encoding:**
    - One Hot Encoding is feature engineering process to convert categorical features into a meaningful form by using a one-of-K scheme. In the other word, we perform 'binarization' on each category and include it as a feature to train the model.
- **Support-vector Machine:**
    - Support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. The algorithm constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.
    - Any hyperplane can be written as the set of points satisfying $\vec{w} \cdot \vec{x} - b = 0$. For the linearly separable cases, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The two hyperplanes can be described by $\vec{w} \cdot \vec{x} - b = 1$ (anything on or above is of one class) and $\vec{w} \cdot \vec{x} - b = 1$ (anything on or below is of the other class. The distance between the two hyperplanes are $\frac{b}{||\vec{w}||}$. Thus, to maximize the distance, we need to minimize $||\vec{w}||$.
    - In our study, we used the LinearSVC from sci-kit learn library to apply Support-vector machine on our classification task.
- **K-Fold Cross-Validation:**
    - K-Fold Cross-Validation is a technique to assess how the results of a prediction model will generalize to an independent data set.
    - We first shuffle the dataset randomly and split them into k equal-sized groups. For each unique group:
        - Take the group as a hold out or test data set
        - Take the remaining groups as a training data set
        - Fit a model on the training set and evaluate it on the test set
        - Retain the evaluation score and discard the model
    - Finally, we summarize the skill of model using the sample of model evaluation scores.

- **Hypothesis test**:

- Hypothesis test is a method of inference which refers to the formal procedures used by statisticians to accept or reject statistical hypotheses.
- During the test, we first propose a null hypothesis, denoted by H_0, which is usually the hypothesis that sample observations result purely from chance and an alternative hypothesis, denoted by H_a, which is the hypothesis that sample observations are influenced by some non-random cause. Secondly, we select an appropriate test and state the relevant test statistic T. Thirdly, we derive the distribution of the test statistic under the null hypothesis from the assumptions and compute from the observations the observed value t_obs of the test statistic T. With the observed value t_obs and the distribution, we can get the p-value for our observations. Finally, we compare the p-value to the significance level (α) and decide to either reject the null hypothesis in favor of the alternative or not reject it.
- When we reject the null hypothesis, we don't know if we have been unlucky with our sampling and observed a rare event or if we are making the correct decision. Thus, we have this table to define the 2 types of error we could possibly make in the hypothesis tests.

	Decision		
Truth		Fail to reject H_0	Reject H_0
	H_0 True	No error	Type 1 Error
	H_a True	Type 2 Error	No error

Works Cited:

AshaJerlin, M., & Jayakumar, C. (2015). "A Dynamic Malware Analysis for Windows Platform-A Survey." *Indian Journal of Science and Technology*, *8*(27), 1.

Chanhyun Kang, Noseong Park, B. Aditya Prakash, Edoardo Serra, and V. S. Subrahmanian. (2016). "Ensemble Models for Data-Driven Prediction of Malware Infections." In *Proceedings of the 2016 ACM International Conference on Web Search and Data Mining.* ACM.

Gandotra, E., Bansal, D. and Sofat, S. (2014). "Malware Analysis and Classification: A Survey." *Journal of Information Security*, **5**, 56-64. doi: 10.4236/jis.2014.52006.

Kaggle. "Microsoft Malware Prediction: Can You Predict if a Machine Will Soon Be Hit with Malware." https://www.kaggle.com/c/microsoft-malware-prediction.

Schultz, M., Eskin, M., Zadok, E., and Stolfo, F. (2001). "Data Mining Methods for Detection of New Malicious Executables." In *Proceedings of 2001 IEEE Symposium on Security and Privacy*, IEEE, Oakland, CA, May 2001, 38-49. DOI = 10.1109/SECPRI.2001.924286

Tesauro, G.J., Kephart, J.O., Sorkin, G.B. (1996). "Neural Networks for Computer Virus

Recognition.” IEEE Expert 11(4), 5–6 (1996)