Predicting Infection of Organization Endpoints by Cybersecurity Threats using Ensemble Machine Learning Techniques

Vishakha Bhattacharjee[*]

[*] Master of Science in Business Analytics, Columbia University, New York

Abstract- The proliferation of the malware industry is the result of the large volume of personal and confidential information shared on individual and public network. This has widened the scope of the organizations being vulnerable to malware - driven cybercrime. Such organized and distributed cyber-attacks can compromise the confidentiality, integrity and availability of any organization's valuable data and resources. The endpoints (Desktops, Laptops, Mobiles, Servers, etc.) are more vulnerable and hence mainly targeted by the cyber criminals. The aim of this study is to determine the probability of such endpoints being affected by cybersecurity threats, based upon certain characteristics of the particular endpoint. Using the machine learning techniques applied in this study, like missing data analysis and imputation (Multiple Imputation), ensemble learning algorithms (Bagging and Boosting), it can be predicted that which devices/systems in an organization are likely to be infected by malwares, ransomwares or other such threats. Based on such findings, proactive measures can be taken, and cyber security strategies can be devised which can help organizations prevent losses to the tune of millions of dollars and become cyber resilient.

Index Terms- Cybercrime, Cybersecurity, Ensemble Algorithms, Machine Learning, Malware Detection, Missing Data Analysis, Multiple Imputation

I. Introduction

The cybercrime industry has been gaining traction over the years, especially owing to the fact that more and more data (personal and organizational) is available on digital mediums. Today, the issues posed by cybercrime cause companies to bleed millions of dollars across the globe. In a study conducted across 507 organizations in 16 countries and regions across 17 industries by the Ponemon Institute, it was defined that the global average cost of data breach for 2019 stands at $3.92 Million, a 1.5% increase from the 2018 estimate [9]. Organizations worldwide are heavily investing into the capabilities of predictive analytics using machine learning and artificial intelligence to mitigate these challenges. As per a report by Capgemini Research Institute (2019), 48% organizations say that their budget for implementation of predictive analytics in cybersecurity will increase by 29% in the fiscal year 2020. 56% of senior executives say that cybersecurity analysts are overworked and close to a quarter of them are not able to successfully investigate all identified issues. 64% of organizations say that predictive analytics lowers the cost of threat detection and response and reduces the overall detection time by up to 12% [5].

Considering the above, the application of predictive analytics into investigating the endpoints which are likely to be infected by malware becomes imperative for organizations in the long run. The study explores this objective, using the knowledge of the specifications of certain hardware and software aspects of an organization's endpoint (Desktops, Laptops, Mobiles, Servers, etc.).

The study takes into perspective various challenges faced while approaching the objective and the ways in which the problems were mitigated. In a real-life scenario, a lot of organizations often lack important information related to the various endpoints being used by them. Some of this information could be related to hardware specifications, some related to licensing of software being used, activation and expiry dates, presence of a firewall etc. These are issues related to missing data, a problem which if not solved will affect the final classification model. The background and issues related to missing data analysis and imputation techniques have been discussed in section II of this paper. Section III discusses the background of Ensemble Learning and the techniques of Bagging and Boosting. As a part of this section, the concepts of underfitting and overfitting the data with the predictive model are also discussed. Section IV focuses on the framework of the predictive model used to classify the endpoints as likely to be affected or not. Discussed under this section are topics like data collection, feature selection, resampling procedure (K- Fold cross validation technique) and classification model building. Experimental Results are discussed under Section V, where various model diagnostics are checked and evaluated with each other so that the best model is chosen for the given dataset. The section called Conclusion follows next that discusses the success of this study and the future scopes of improvement. All the code used to develop the model and do all the other analysis related to this study are placed under the Appendix section. Lastly, the study ends with a Reference section in order to cite all the resources that were used to do the research.

II. MISSING DATA CLASSIFICATION AND MULTIPLE IMPUTATION TECHNIQUE

Missing data or missing value is nothing but the absence of data value in a variable of interest for the study being performed. Missing data is one of the primary problems faced by researchers and organizations while performing any sort of study or analytical procedure. The non-availability of data is a real-life issue and makes the difference between a great and ordinary study.

This brings about several issues while performing statistical analysis. Most statistical analysis methods reject the missing values, thus reducing the size of the dataset to work with. Often, having not enough data to work with creates models that produces results which are statistically not significant. Also, missing data might lead to cases where the results are misleading. Results are often biased towards certain segment/segments that are overrepresented in the population.

MISSING DATA CLASSIFICATION: The classification of missing data was first discussed buy Rubin in his paper titled ‘Inference and Missing Data’ [10]. According to his theory, every data point in a dataset has some probability of being missing. Based on this probability, Rubin classifies missing data into the following types:

1. Missing Completely at Random (MCAR): The missingness of data in this case is not related to the other responses or information in the data in anyway. The probability of any data point being missing in the dataset remains equal for all the other data points. In simpler words, there is no identifiable pattern in the missingness of the data. An important thing to be noticed when dealing with MCAR is that analysis done on MCAR remains produces unbiased results.

2. Missing at Random (MAR): MAR is a wider classification of missing data when compared to MCAR, and in some terms more realistic too. In the case of MAR, the probability of the missingness of data is similar for certain subsets of the data defined for the statistical study [3]. The missingness of the data can be attributed to the other data that is present and hence can be predicted. Again, in simpler words, in case of MAR, there is a pattern to the missingness of the data.

3. Not Missing at Random (NMAR): Data which is not classified as MCAR and MAR is classified as NMAR.

MULTIPLE IMPUTATION: When it comes to handling missing data for any type of statistical analysis, the standard approach has been deleting the records with the missing data, also known as the Listwise deletion method. While Listwise deletion might pose as a convenient method and works well with data that is of nature MCAR (provides unbiased estimates), often a lot of inconsistencies arise from using it [4], especially in cases where a lot of data is missing. Other techniques that work on the method of deleting missing data report similar inconsistencies in analysis and loss of valuable information. With single imputation techniques like Mean Imputation, problems arise in terms of highly biased estimates, even in case of MCAR type of missing data [7]. Regression Imputation techniques produces outputs where the standard error obtained by comparing the observed and imputed data is too low. To mitigate this issue, Stochastic Regression Imputation [1] is used and some amount of noise is added to the imputed estimates. Single imputation methods work on the assumption that the estimated imputed value is the true value, neglecting the uncertainties related to the prediction of the imputed value.

Multiple Imputation method for estimating missing data was proposed by Rubin (1987). The method starts with a dataset that contains missing values and then goes on to create several sets of imputed values for the missing data using a statistical model like linear regression. This is followed by calculating the parameters of interest for each of the imputed data set, which are finally pooled together into one estimate. Figure 1 shows a pictorial representation of a Multiple Imputation method of the 4th order.

The standard error which was too small in case off single imputation techniques in mitigated well by using multiple imputation. Multiple imputation performs well not only with MCAR data but also with MAR data. The variation in data that is received across the multiple imputed dataset helps in offsetting any sort of bias. This is achieved by adding the uncertainties that were missing as part of the single imputation techniques. This in turn increases the precision and results in robust statistics which leads to better analysis that might be performed on the data. One of the most popular methods of performing multiple imputation in R is using the MICE (Multivariate Imputation via Chained Equations) package [2].

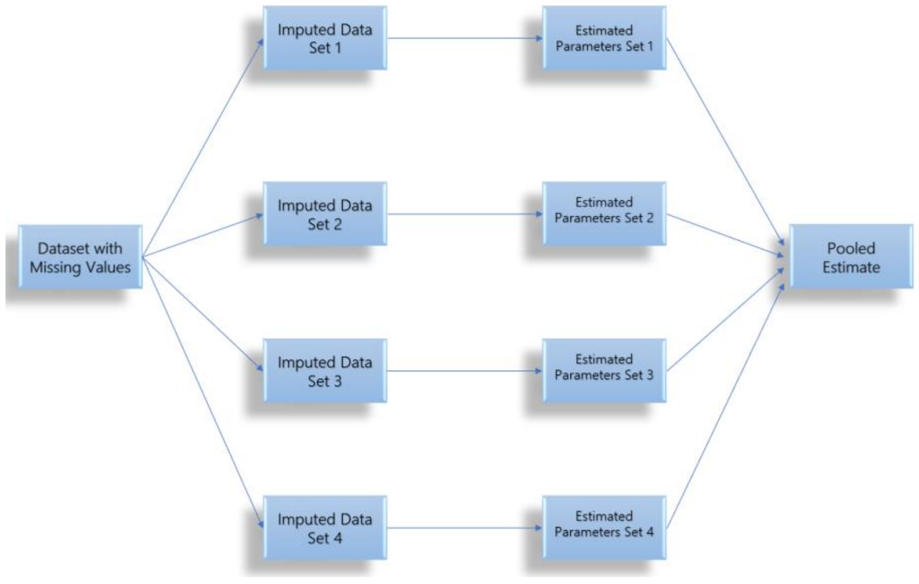
Dataset with Missing Values
Imputed Data Set 1
Imputed Data Set 2
Imputed Data Set 3
Imputed Data Set 4
Estimated Parameters Set 1
Estimated Parameters Set 2
Estimated Parameters Set 3
Estimated Parameters Set 4
Pooled Estimate

FIGURE 1: STEPS OF MULTIPLE IMPUTATION OF ORDER 4

III. ENSEMBLE LEARNING METHODS

Ensemble learning methods use the combined computational power of multiple models to classify and solve the problem at hand. When compared to ordinary learning algorithms that create only one learning model, ensemble learning methods create multiple such models and combine them to make the final model that makes more efficient classifications. Ensemble learning is also called as committee-based learning or learning multiple classifier systems [11]. Figure 2 shows a common ensemble learning architecture.

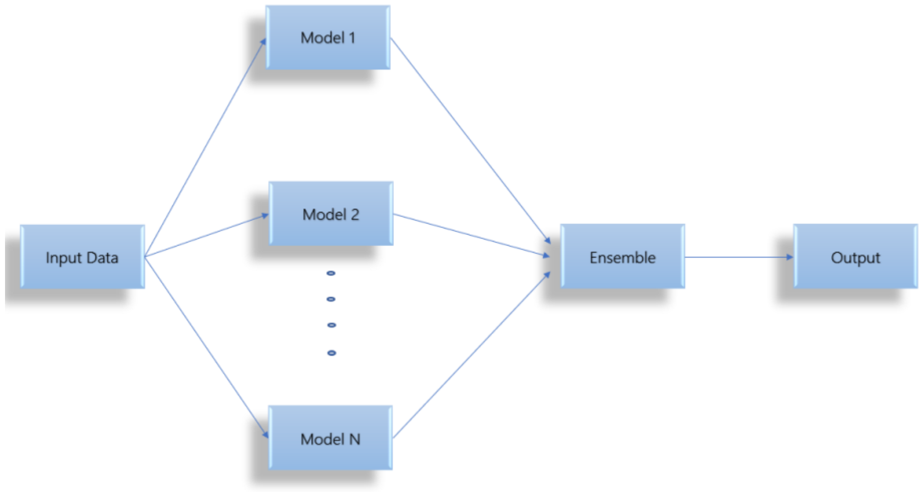

FIGURE 2: COMMON ENSEMBLE LEARNING METHOD ARCHITECTURE

Ensemble learning methods are used and appreciated because of their ability to boost the performance of weak learners, often known as base learners. This in turn produces predictions with higher accuracy and stronger generalization performance. The models created also are more robust in nature and respond well to noise in data. For this study, two advance forms of ensemble learning methods have been used, Bagging and Boosting. They are discussed below:

BAGGING ENSEMBLE LEARNING: Bagging is an acronym for Bootstrap Aggregating and is used to solve both classification and regression problems. The method of Bagging involves creating multiple samples which are random in nature with replacement. These samples are used to create models, the results from which are amalgamated together. The advantage of using Bagging algorithms lies in the fact that they reduce the chances of a predictive model overfitting the data. Since every model is built on a different set of data, the variance error component of the reducible error in the model is low, which means that the model handles the variance in test data well. The Bagging learning architecture is pictorially depicted in Figure 3.

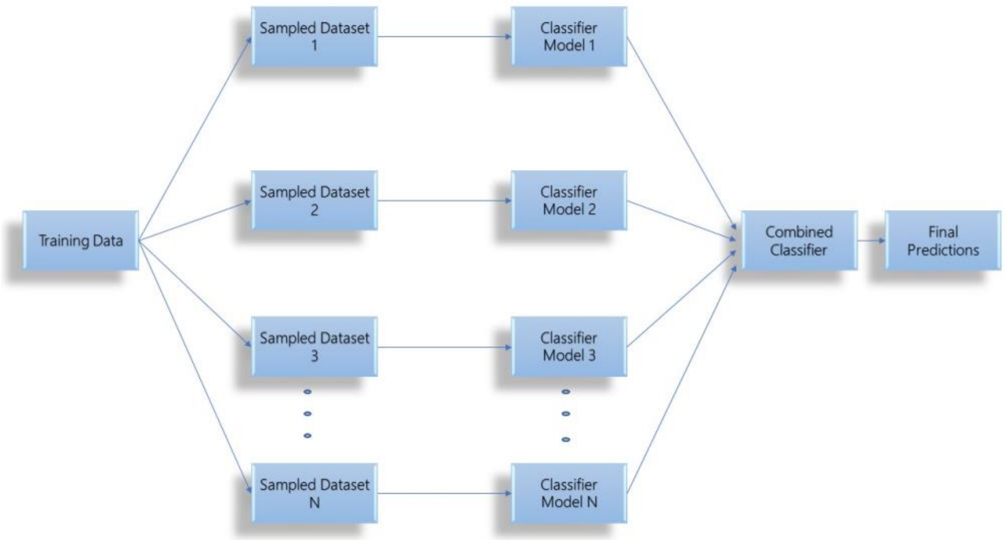

FIGURE 3: BAGGING ENSEMBLE LEARNING ARCHITECTURE

BOOSTING ENSEMBLE LEARNING: Boosting ensemble learning works on an iterative approach of adjusting weights of an observation present in the training dataset based upon the performance of the previous classification model. The weight for an observation is increased if it is classified incorrectly and decreased if classified correctly. Boosting ensemble learning has it's in advantage in cases where the bias error component of reducible error is high. Boosting decreases this bias error and helps in building stronger predictive models.

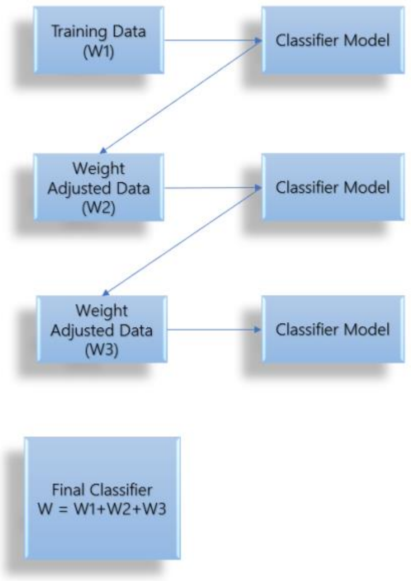
Training Data (W1)
Classifier Model
Weight Adjusted Data (W2)
Classifier Model
Weight Adjusted Data (W3)
Classifier Model
Final Classifier W = W1+W2+W3

FIGURE 4: BOOSTING ENSEMBLE LEARNING ARCHITECTURE

IV. SOLUTION FRAMEWORK AND RESULTS

In order to identify the endpoints which are highly likely to be infected by malware, a solution was designed with the framework as described below:

DATA CLEANING AND IMPUTATION: The data for the study was sourced from the Microsoft Malware Classification Challenge (BIG 2015) [8]. For the purpose of this study, 50,000 random rows of data were taken, as would be the case in any mid-size organization in terms of number of endpoints. The variables present in the data and their description is described in Table 1. The dataset has a mixture of both categorical and numerical data. In this study, one of the primary challenges faced was the presence of a large percentage of missing data in the dataset. The data upon further analysis was categorizes as Missing at Random (MAR). The missing data is imputed using the Multiple Imputation technique by using the MICE package available in R. Further, exploratory data analysis helps in understanding class imbalance and correlation.

Variable	Description
MachineId	Individual machine ID
ProductName	Type of Endpoint Protection enabled e.g. win8defender
HasTpm	True if the machine has tpm (Trusted Platform Module) enabled
Platform	Version of Windows installed
Processor	Process architecture of the installed operating system
SkuEdition	SKU Edition of the Windows Version
IsProtected	If the Machine is Protected by an active and up-to-date Antivirus Product
Firewall	If the Windows Firewall is enabled
AdminApprovalMode	Whether the "administrator in Admin Approval Mode" user type is disabled or enabled
DeviceType	Type of the Device eg - Notebook, Laptop, Desktop
PrimaryDiskTotalCapacity	Amount of disk space on primary disk of the machine in MB
PrimaryDiskTypeName	Friendly name of Primary Disk Type - HDD or SSD
SystemVolumeTotalCapacity	The size of the partition that the System volume is installed on in MB
HasOpticalDiskDrive	True indicates that the machine has an optical disk drive (CD/DVD)
TotalPhysicalRAM	Retrieves the physical RAM in MB
AutoUpdate	Friendly name of the WindowsUpdate auto-update settings on the machine.
GenuineStateOS	Indicates the authenticity of the OS version
IsSecureBootEnabled	Indicates if Secure Boot mode is enabled
IsPenCapable	Is the device capable of pen input ?
IsAlwaysOnAlwaysConnectedCapable	Retreives information about whether the battery enables the device to be AlwaysOnAlwaysConnected
IsGamer	Indicates whether the device is a gamer device or not based on its hardware combination.
IsInfected	Indicates if the machine has been diagnosed as Malware affected

TABLE 1: LIST OF ATTRIBUTES USED

CROSS VALIDATION AND MODEL BUILDING: Instead of splitting the data into training and testing data subsets, a K- Fold cross validation approach is adopted. The K-Fold cross validation technique randomly creates K subsets from the data of approximately equal size. From these subsets, the first subset is treated as the validation or the test set and the remaining K-1 subsets are used to train the predictive model [6]. For the purpose of this study, the value of K is taken as 10. Four ensemble learning models were created on this 10-Fold cross validated data. They are described below:

1. C5.0 Boosting Algorithm
2. Stochastic Grading Boosting Algorithm
3. Bagged CART Algorithm
4. Random Forest Algorithm

Model Evaluation: The 4 ensemble models created on the given data are evaluated on the terms of parameters like model accuracy and the Area Under Curve (AUC), the curve being the Receiver Operating Characteristic (ROC) curve. These parameters help in determining how accurate and robust the model is.

1. Model Accuracy: The model accuracy is a simple evaluation criteria and is calculated by taking the total number of correct classifications that the model makes (True Positives and False Negatives) and dividing it by the total number of samples on which the model is tested. The model accuracies for the various ensemble learning algorithms is described in Table 2.

2. Area Under Receiver Operating Characteristic Curve (AUROC): The AUROC is a performance measure that helps in understanding how well the model does in distinguishing two classes. In the case of the study, the AUC score will help in distinguishing between infected and not infected endpoints in an organization. The best possible value for AUC is 1, in which case the model perfectly distinguishes between the infected and not infected endpoints. AUC is one of the best measures of selecting models for practical usage. The AUC scores for the ensemble models are described in Table 2. Figure 5 compares the distribution of the AUC values for all the models using a Box-Whisker plot.

Model	Accuracy	AUC	Sensitivity	Specificity
C5.0 Boosting Algorithm	0.546	0.5651978	0.5167168	0.575051
Stochastic Grading Boosting Algorithm	0.5496	0.5716335	0.4785676	0.620366
Bagged CART Algorithm	0.529	0.5406808	0.528008	0.52996
Random Forest Algorithm	0.5442	0.5668337	0.3377704	0.749531

Table 2: Performance Evaluation of Ensemble Models

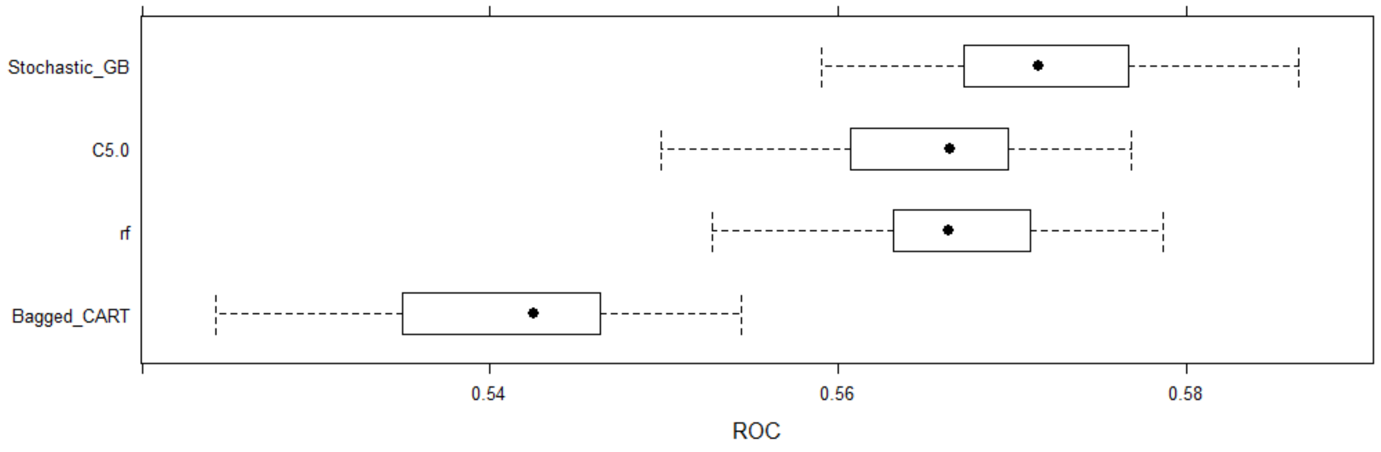

Figure 5: Comparing the Distribution of AUC Values for Various Models

MODEL SELECTION: Based on the evaluation parameters described above, i.e. the model accuracy and the AUCROC score, the model created using the Stochastic Gradient Boosting algorithm is selected. The reason for the selection of this particular model is the highest model accuracy and the highest AUC score. This indicates that out of all the models created on the dataset, the Stochastic Gradient Boosting algorithm-based model is the most accurate in terms of correct class identification and robustness. The model that gave the best result for this study had a Tree Depth of 3 and number of trees as 100.

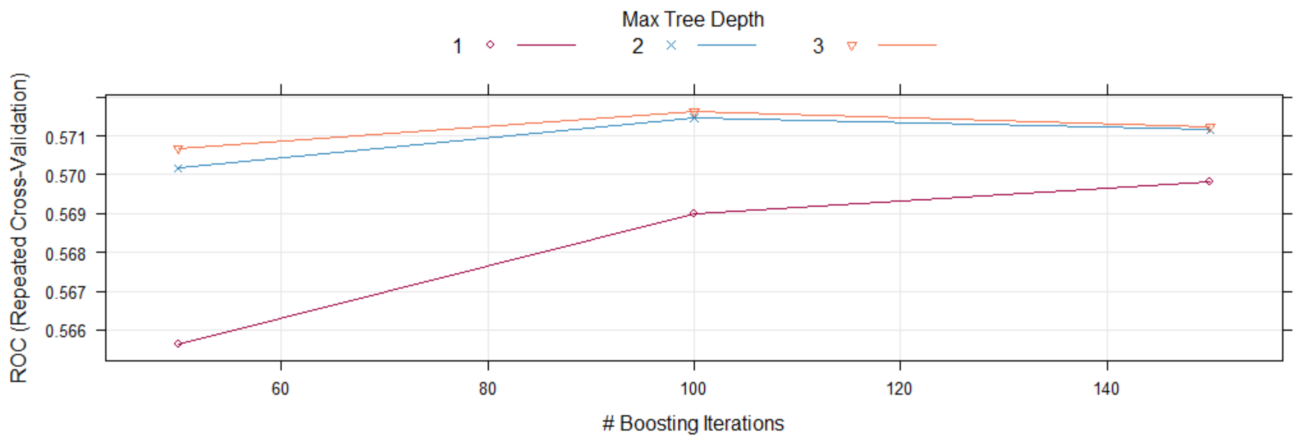

FIGURE 6: AUCROC PROFILE OF STOCHASTIC GRADIENT BOOSTING MODEL COMPARED AGAINST TUNING PARAMETER MAX TREE DEPTH

V. CONCLUSION

The study works towards establishing the hypothesis that with the correct knowledge of the specifications of organizational endpoints (both software and hardware), it is possible to predict the likeliness of an endpoint to get infected by malware attacks and other cybersecurity threats. In order to achieve the aim of this study, several real-life challenges related to data were faced, like understanding missing data, the imputation of missing data using multiple imputation technique, challenges with cross validation of data and performance evaluation of the models. The list of variables used in the study for building the model is not exhaustive in nature, several new metrics and variables can be added based upon the availability and applicability with respect to various organizations to build more accurate and robust models.

APPENDIX

R Code for the study is shared below:

```
#########################################################
##PROJECT: Predicting Infection of Organization Endpoints by Cybersecurity Threats
#using Ensemble Machine Learning Techniques

##AUTHOR: VISHAKHA BHATTACHARJEE
##Version 1.1
##Date: 30/03/2020
#########################################################

#############INSTALLING PACKAGES
install.packages('mice')
install.packages('caret')

#############LOADING PACKAGES
library(mice)
library(caret)
```

```
############READING DATA
df = read.csv('Train.csv')
View(df)
sapply(df, function(x) sum(is.na(x)))
#Firewall - Factor - logreg
#IsProtected - Factor - logreg
#SystemVolumeTotalCapacity - numeric - pmm
#PrimaryDiskTotalCapacity - numeric - pmm
#IsAlwaysOnAlwaysConnectedCapable - Factor - logreg
#AdminApprovalMode - Factor - logreg
#TotalPhysicalRAM - Numeric - pmm
#IsSecureBootEnabled - Factor - logreg
#IsGamer - Factor - logreg


##############converting into factors(categorical variables)
df$HasTpm = as.factor(df$HasTpm)
df$IsProtected = as.factor(df$IsProtected)
df$Firewall = as.factor(df$Firewall)
df$AdminApprovalMode = as.factor(df$AdminApprovalMode)
df$HasOpticalDiskDrive = as.factor(df$HasOpticalDiskDrive)
df$IsSecureBootEnabled = as.factor(df$IsSecureBootEnabled)
df$IsPenCapable = as.factor(df$IsPenCapable)
df$IsAlwaysOnAlwaysConnectedCapable = as.factor(df$IsAlwaysOnAlwaysConnectedCapable)
df$IsGamer = as.factor(df$IsGamer)
df$IsInfected = as.factor(df$IsInfected)

str(df)
ncol(df)

###############REMOVING MachineId FROM DATA FRAME
df = df[,-c(1)]


##############IMPUTATION OF MISSING DATA USING MICE
init = mice(df, maxit=0)
meth = init$method
predM = init$predictorMatrix

#Excluding the output column IsInfected as a predictor for Imputation
predM[, c("IsInfected")]=0

#Excluding these variables from imputation
meth[c("ProductName","HasTpm","Platform","Processor","SkuEdition","DeviceType","HasOpticalDiskDrive","IsPenCapable","IsI
nfected")]=""


#Specifying the imputation methods for the varaibles with missing data
meth[c("SystemVolumeTotalCapacity","PrimaryDiskTotalCapacity","TotalPhysicalRAM")]="cart"
meth[c("Firewall","IsProtected","IsAlwaysOnAlwaysConnectedCapable","AdminApprovalMode","IsSecureBootEnabled","IsGame
r")]="logreg"
meth[c("PrimaryDiskTypeName","AutoUpdate","GenuineStateOS")]="polyreg"

#Setting Seed for reproducibility
set.seed(103)
```

```
#Imputing the data
imputed = mice(df, method=meth, predictorMatrix=predM, m=5)
imputed <- complete(imputed)

sapply(imputed, function(x) sum(is.na(x)))

sum(is.na(imputed))

#####IMPUTING MANUALLY A FEW BLANK ROWS OF DATA
imputed$PrimaryDiskTypeName[imputed$PrimaryDiskTypeName == ""] = "UNKNOWN"

imputed$AutoUpdate[imputed$AutoUpdate == ""] = "UNKNOWN"

imputed$GenuineStateOS[imputed$GenuineStateOS == ""] = "UNKNOWN"

#writing imputed data into a new file for future use and backup
write.csv(imputed,'MICE_Imputed_Data.csv')


########################MODEL CREATION
imputed = read.csv('MICE_Imputed_Data.csv')
str(imputed)

##############converting into factors(categorical variables)
imputed$HasTpm = as.factor(imputed$HasTpm)
imputed$IsProtected = as.factor(imputed$IsProtected)
imputed$Firewall = as.factor(imputed$Firewall)
imputedAdminApprovalMode = as.factor(imputed$AdminApprovalMode)
imputed$HasOpticalDiskDrive = as.factor(imputed$HasOpticalDiskDrive)
imputed$IsSecureBootEnabled = as.factor(imputed$IsSecureBootEnabled)
imputed$IsPenCapable = as.factor(imputed$IsPenCapable)
imputed$IsAlwaysOnAlwaysConnectedCapable = as.factor(imputed$IsAlwaysOnAlwaysConnectedCapable)
imputed$IsGamer = as.factor(imputed$IsGamer)

#Converting the output variables classes into single letters F and T from 0 and 1 respectively
#This is required by the CARET package for ROC metric functionality
imputed$IsInfected[imputed$IsInfected == 0] = "F"
imputed$IsInfected[imputed$IsInfected == 1] = "T"

#Converting the output variable to a factor variable
imputed$IsInfected = as.factor(imputed$IsInfected)

##Omitting two columns in the begining of the data that do not contribute
imputed = imputed[,-c(1,2)]
nrow(imputed)
sum(is.na(imputed))
sapply(imputed, function(x) sum(is.na(x)))
View(imputed)


##Function twoClassSummary is used to call upon metrics like ROC in CARET package function train()
head(twoClassSummary)

###################BOOSTING ALGORTIHMS
```

```
control <- trainControl(method="repeatedcv", number=10, repeats=3, classProbs = TRUE, summaryFunction = twoClassSummary)
seed <- 7

##Only one metric can be set at a time. Other metric like Accuracy can also be checked
metric <- "ROC"

################### C5.0
set.seed(seed)
fit.c50 <- train(IsInfected~., data=imputed, method="C5.0", metric=metric, trControl=control)
print(fit.c50)
confusionMatrix(fit.c50)


################### Stochastic Gradient Boosting
set.seed(seed)
fit.gbm <- train(IsInfected~., data=imputed, method="gbm", metric=metric, trControl=control, verbose=FALSE)
print(fit.gbm)
confusionMatrix(fit.gbm)
############ summarize results
boosting_results <- resamples(list(c5.0=fit.c50, gbm=fit.gbm))
summary(boosting_results)
dotplot(boosting_results)


#Plotting the Boosting Algorithms
trellis.par.set(caretTheme())
plot(fit.c50)
plot(fit.gbm)




##########################BAGGING ALGORTIHMS
control <- trainControl(method="repeatedcv", number=10, repeats=3, classProbs = TRUE, summaryFunction = twoClassSummary)
seed <- 7
metric <- "ROC"

################## Bagged CART
set.seed(seed)
fit.treebag <- train(IsInfected~., data=imputed, method="treebag", metric=metric, trControl=control)
print(fit.treebag)
confusionMatrix(fit.treebag)


################## Random Forest
set.seed(seed)
fit.rf <- train(IsInfected~., data=imputed, method="rf", metric=metric, trControl=control)
print(fit.rf)
confusionMatrix(fit.rf)


#################### summarize results
bagging_results <- resamples(list(treebag=fit.treebag, rf=fit.rf))
summary(bagging_results)
dotplot(bagging_results)
```

```
###Plotting Bagging Algorithms
trellis.par.set(caretTheme())
plot(fit.treebag)
plot(fit.rf)



########Generating aggregated statistics and plots of all models
model_list = list(C5.0 = fit.c50,
                  rf = fit.rf,
                  Bagged_CART = fit.treebag,
                  Stochastic_GB = fit.gbm)

resamp <- resamples(model_list)
resamp

summary(resamp)
lattice::bwplot(resamp, metric = "ROC")
```

References

[1] Buck, S. F. 1960. "A Method Of Estimation Of Missing Values In Multivariate Data Suitable For Use With An Electronic Computer". Journal Of The Royal Statistical Society: Series B (Methodological) 22 (2): 302-306. doi:10.1111/j.2517-6161.1960.tb00375.x.

[2] Buuren, Stef van, and Karin Groothuis-Oudshoorn. 2011. "Mice: Multivariate Imputation By Chained Equations Inr". Journal Of Statistical Software 45 (3). doi:10.18637/jss.v045.i03.

[3] Buuren, Stef van. n.d. Flexible Imputation Of Missing Data. Taylor & Francis Group, LLC., 6.

[4] lbid., 8-16.

[5] Capgemini Research Institute. 2020. "Reinventing Cybersecurity With Artificial Intelligence The New Frontier In Digital Security". Accessed March 17.

[6] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. An Introduction To Statistical Learning With Applications In R. 6th ed. New York: Springer., 181.

[7] Jamshidian, Mortaza, and Peter M. Bentler. 1999. "ML Estimation Of Mean And Covariance Structures With Missing Data Using Complete Data Routines". Journal Of Educational And Behavioral Statistics 24 (1): 21. doi:10.2307/1165260.

[8] "Microsoft Malware Classification Challenge (BIG 2015) | Kaggle". 2020. Kaggle.Com. https://www.kaggle.com/c/malware-classification.

[9] Ponemon, Larry. 2020. "What's New In The 2019 Cost Of A Data Breach Report". Security Intelligence. https://securityintelligence.com/posts/whats-new-in-the-2019-cost-of-a-data-breach-report/.

[10] Rubin, Donald B. 1976. "Inference And Missing Data". Biometrika 63 (3): 581-592. doi:10.1093/biomet/63.3.581.

[11] Zhou, Zhi-Hua. 2012. Ensemble Methods: Foundations And Algorithms. Boca Raton: CRC Press.,15-18.

Authors

First Author – Vishakha Bhattacharjee, Master of Science in Business Analytics, Columbia University, New York.