MICROSOFT MALWARE PREDICTION USING MACHINE LEARNING MODELS

Yashasri Edukulla[1] Sai Kumar Manchikanti[2] Arun Reddy [3]

I. PROPOSAL

Malware is a software that is intentionally created to escape the security measures enforced by the concerned company on any of its machines. This industry is determined to hurt these measures and invade enterprise information, which could result in a loss to the company. Additionally, the malware industry is financially sound to improve their ways to affect the machine. To counter this attack, companies come up with antivirus, firewalls, security gates and other access permissions for any user to enter. With more than two billion customers all around the world, Microsoft is trying to create a machine learning algorithm that could predict future occurring Malware attack. Seeing the forthcoming challenges in the security industry all along with the world, we have chosen it as our project.

II. DATA ACQUISITION

This problem is one of the competitions posted on Kaggle competitions website, nine months ago. This competition is hosted by Microsoft, Windows Defender ATP Research, Northeastern University College of Computer and Information Science, and Georgia Tech Institute for Information Security and Privacy.

III. INTRODUCTION

In today's fast-moving technological world, one of the major and serious threats is on the Internet, which is known as Malware. Malware is "any code included, changed, or removed from code to intentionally cause hurt or demolish the system's expecting function". Such software has been utilized to compromise computer frameworks, to crush their data, and to render them useless. It has moreover been utilized to accumulate data, such as passwords and credit card numbers, and to disperse data, such as obscenity, all without the information of a system's clients. As increasingly novice clients get advanced computers with high-speed associations to the Web, the potential for advance mishandle is incredible. These are designed by human predators which make them be polymorphic and tangible to change as they propagate into the machine that they will infect. An analysis report according to a study, conducted by Fire-Eye in June 2013, 47 percent of the organizations experienced malware security incidents/network breaches within the past year. The malware is ceaselessly developing in volume (developing danger scenes), assortments (inventive noxious strategies) and speed (ease of dangers) This malware have certain similar behavioural patterns that will give us an idea of their purpose. The destination of this machine learning project's journey is to predict the probability of a Windows Machine that is soon going to get affected by various families of malware using its basic properties. The properties are nothing but the data-set on which our training and testing models are implemented. The data was generated by combining the heartbeat and threat reports which are collected by Microsoft endpoint protection solution, Windows Defender. The size of the data-set is very complex holding 3,00,074 rows and a voluminous amount of 82 columns. All the columns are considered to be features which are classified into 1 dependant-features, 52 categorical-features, 23 of which are binary encoded to preserve the privacy of the information and the rest are names of different properties like Engine-Name, Machine-Identifier etc respectively. The solution to this problem is very complex as it is a time – series problem. Additionally, another challenge is to implement these models on such a large data-set, that takes hours of computations in the system in order to process precisely.

IV. DATA DESCRIPTION

The data-set contains 82 features /columns. The description of each column is given below:

1) MachineIdentifier: Indicates Machine ID. The machine ID is generally created from an irregular source either when the system is being installed or during the very first boot and then it remains consistent for every other boot.

2) ProductName: This indicates the name of the application being installed, significantly for display purpose.

3) EngineVersion: This indicates the Antimalware Engine version.

4) AppVersion: The appVersion property returns a value based on the application name, application version, and platform.

5) AvSigVersion: This is Antivirus signature version, which indicates the version of the antivirus currently running.

6) IsBeta: Boolean Value that indicates whether the update is a beta release or not.

7) RtpStateBitfield: RTP stands for Realtime protection state, this property indicates either it is enabled or disabled.

8) IsSxsPassiveMode: This indicates the mode of the operation for windows defender which is either active mode or passive mode. If there exists any other third-party antivirus on the system, the Defender enters Passive mode. Passive mode offers reduced functionality.

9) DefaultBrowsersIdentifier: This indicates the ID for the machine's default browser.

10) AVProductStatesIdentifier: This indicates the ID for the specific configuration of a user's antivirus software.

11) AVProductsInstalled: This indicates the number of anti-virus products installed on the system.

12) AVProductsEnabled: No Info

13) HasTpm: Tpm Stands for a Trusted Platform Module. This property is True if machine has tpm. Tpm is a specialized chip on an endpoint device which stores RSA encrypted keys specific to the host system for hardware authentication.

14) CountryIdentifier: This indicates the ID for the country the machine is located in. Wikipedia cites 255+ countries and independent territories.

15) CityIdentifier: This indicates the ID for the city the machine is located in.

16) OrganizationIdentifier: This indicates the ID for the organization the machine belongs in.

17) GeoNameIdentifier: This indicates the ID column for the 292 geographic regions, a machine is located in.

18) LocaleEnglishNameIdentifier: This indicates the English name of Locale ID of the current user. As the same language may be spoken in multiple countries (often with subtle differences) and a single country may speak multiple languages thus A locale is the area where a language is spoken which may or may not align with political or geographical boundaries.

19) Platform: No Info

20) Processor: This indicates the process architecture of the installed operating system.

21) OsVer: This indicates the Version of the current operating system.

22) OsBuild: This indicates the Build of the current operating system.

23) OsSuite: This indicates the Product suite mask for the current operating system.

24) OsPlatformSubRelease: This returns the OS Platform sub-release such as Windows Vista, 7, 8.

25) OsBuildLab: This indicates the Build lab information that generated the current OS.

26) SkuEdition: No Info.

27) IsProtected: This indicates Boolean values, returns TRUE if there is at least one active and up-to-date antivirus product running on this machine if there is no active AV product on this machine then it returns FALSE and NULL is returned if there are no Antivirus Products.

28) AutoSampleOptIn: No Info

29) PuaMode: Pua stands for “The Potentially Unwanted Applications” this feature in Windows Defender Antivirus can identify and block from downloading and installing on endpoints in your network. These applications are not considered infections, malware, or different sorts of dangers, yet may perform activities on endpoints that antagonistically influence their exhibition or use. PUA can likewise allude to applications that are considered to have a poor notoriety.

30) SMode: This is a set of mode in which only Microsoft store aps can be installed This field is set to true when the device is known to be in ’S Mode’.

31) IeVerIdentifier: This retrieves which version of Internet Explorer is running on this device.

32) SmartScreen: SmartScreen is a registry entry enabled string value. Windows Defender SmartScreen helps to protect employees of a certain organization if any employee tries to visit the websites previously reported as phishing or malware websites, or if an employee tries to download potentially malicious files. This only applies to Win 10 and Win 10 mobile

33) Firewall: Boolean Value, this is true for Windows 8.1 and above if windows firewall is enabled

34) UacLuaenable: This property reports whether the "administrator is in Admin Approval Mode" user type is disabled or enabled in User Access Control

35) Census-MDC2FormFactor: This is A grouping based on a combination of Device Census level hardware characteristics. The logic used to define Form Factor is rooted in business and industry standards and aligns with how people think about their device.

36) Census-DeviceFamily: This Indicates the type of device that an edition of the OS is intended for. Usually a very high proportion of Windows Desktop and very few of Windows Server and Windows.

37) Census-OEMNameIdentifier: No Info

38) Census-OEMModelIdentifier: No Info

39) Census-ProcessorManufacturerIdentifier: No Info

40) Census-ProcessorModelIdentifier: No Info

41) Census-ProcessorClass: No Info

42) Census-ProcessorCoreCount: This indicates the Number of logical cores in the processor. Examples are Dual, Quad and Octa

43) Census-PrimaryDiskTotalCapacity: Amount of disk space on primary disk of the machine in Mega Bytes. Examples are 250gb, 500gb and 1Tb

44) Census-PrimaryDiskTypeName: This is the Name given to Primary Disk Type - HDD or SSD.

45) Census-SystemVolumeTotalCapacity: This is the size of the partition that the System volume is installed on in MB.

46) Census-HasOpticalDiskDrive: Boolean Value and a True indicates that the machine has an optical disk drive (CD/DVD). Most systems don’t have a disk drive. This column could provide an idea of how old the chassis is as older chassis might correlate with higher malware rates.

47) Census-TotalPhysicalRAM: This Retrieves the physical RAM in MB.

48) Census-ChassisTypeName: This Retrieves a numeric representation of what type of chassis the machine has.

49) Census-InternalPrimaryDiagonalDisplaySizeInInches: This Retrieves the physical diagonal length in inches of the primary display.

50) Census-InternalPrimaryDisplayResolutionHorizontal: This Retrieves the number of pixels in the horizontal direction of the internal display.

51) Census-InternalPrimaryDisplayResolutionVertical: This Retrieves the number of pixels in the vertical direction of the internal display.

52) Census-PowerPlatformRoleName: This Indicates the Original equipment manufacturer (OEM) preferred power management profile.

53) Census-InternalBatteryType: This indicates and labels the Battery type in the system.

54) Census-InternalBatteryNumberOfCharges: No Info.

55) Census-OSVersion: This indicates the Numeric OS version.

56) Census-OSArchitecture: This indicates the Architecture on which the OS is based.

57) Census-IsPortableOperatingSystem: No Info

58) Census-OSBranch: No Info

59) Census-OSBuildNumber: No Info

60) Census-OSBuildRevision: No Info

61) Census-OSEdition: No Info

62) Census-OSSkuName: No Info

63) Census-OSInstallTypeName: No Info

64) Census-OSInstallLanguageIdentifier: No Info

65) Census-OSUILocaleIdentifier: No Info

66) Census-OSWUAutoUpdateOptionsName: No Info

67) Census-GenuineStateName: No Info

68) Census-ActivationChannel: No Info

69) Census-ThresholdOptIn: No Info

70) Census-FirmwareManufacturerIdentifier: No Info

71) Census-FirmwareVersionIdentifier: No Info

72) Census-IsFlightingInternal: No Info

73) Census-IsFlightsDisabled: No Info

74) Census-IsFlightingInternal: No Info

75) Census-IsSecureBootEnabled: This Indicates if Secure Boot mode is enabled.

75) Census-IsWIMBootEnabled: This Indicates if WimBoot mode is enabled it is an alternative way for OEMs to deploy Windows.

76) Census-IsVirtualDevice: This Identifies if it is a Virtual Machine.

77)Census-IsTouchEnabled: This Identifies if it is a touch device or not.

78)Census-IsPenCapable: This Identifies if it is a touch device or not.

79)Census-IsAlwaysOnAlwaysConnectedCapable: No Info

80)Wdft-RegionIdentifier: No Info

81)Wdft-IsGamer: This Indicates whether the device is a gamer device or not based on its hardware combination and specification.

The columns that have No info against the subcategory are not defined to be anything by Microsoft because of security issues in explicitly telling them out.

V. DATA PREPROCESSING

A. Graphical Analysis

Graphs are an extremely adaptable and effective way to represent information. Conventional relational databases, with their fixed patterns, make it difficult to store connections between distinctive substances, yet such associations are an abundant and vital portion of the real world. In a graph database, these associations are simple to store and query. Besides, regularly the relationships between numerous things (e.g. the associations between family individuals) collectively give crucial data, and chart databases make this easy to analyze. We have drawn several graphs depicting every relationship the input feature(s) share with the output feature(s).

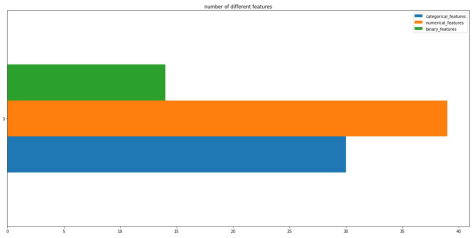

Fig. 1. Snapshot of various variable buckets

B. Cor-relational Data

The initial step to pre-process data is by understanding the importance it stands in affecting the output variable, which is namely provided by correlation. In python, the correlation could be predicted in the range of numbers, giving us a brief description of how correlated they are to each other. The highest number indicated against the column tells us that they are highly correlated and thereby the presence of the column acts as a duplicate to the column that it is being compared to and henceforth it could be dropped. Similarly, the smallest number indicates that they have the lowest correlation between each other. In our data-set, analyzing the columns we have found these columns to have the least correlation.

1) Platform
2) Census-OSSkuName
3) Census-OSInstallLanguageIdentifier
4) Processor

As a first step to building a good model, we have dropped these columns. The correlation between these columns and the dependent variable column could be drawn by using heat maps in python.

C. Missing Data Values

As the dataset is raw and doesn't contain any order of organization, certain columns while the collection is not filled or are placed as null columns or either is left empty. We cannot work with these columns as they do not provide any substantial change that indicates upcoming malware. The threshold value is also set (0.75) that indicates: any column having null value percentage beyond 75 must be removed. Thereby, we have deleted these columns too as filling them with any other content would be inappropriate. The columns are:

1) PuaMode
2) Census-ProcessorClass

D. Skewed Features

Data Skewness is asymmetry in a measurable distribution, in which the bend shows up misshaped or skewed either to the left or to the right. Skewness can be measured to characterize the degree to which a distribution contrasts from a normal distribution. In a skewed distribution the

mean, median and mode are all different, whereas, in the asymmetric bell curve, the mean, median and mode are all same. A negatively skewed distribution means the opposite: that the extreme data results are smaller. A positively skewed distribution means that the extreme data results are larger. A data transformation may be used to decrease skewness. A distribution that's symmetric is simpler to handle and decipher than skewed dissemination. More particularly, a normal or Gaussian conveyance is regularly regarded as perfect because it is expected by many factual methods. Even after applying various mathematical transformations to the skewed data, it did not seem to come into a proper scale and thereby we have dropped these columns too. They are:

1) Census-IsWIMBootEnabled
2) IsBeta
3) Census-IsFlightsDisabled
4) Census-IsFlightingInternal
5) AutoSampleOptIn
6) Census-ThresholdOptIn
7) SMode
8) Census-IsPortableOperatingSystem
9) Census-DeviceFamily
10) UacLuaenable
11) Census-IsVirtualDevice

E. Unrelated Data

Data that has a name as the machine's unique identifier or any other naming related pattern could be removed as they don't affect the output of the implemented model. The columns under this category are:

1) MachineIdentifier
2) AppVersion
3) AvSigVersion

F. Dealing with Categorical Data

Data that have repeated string values could be encoded with numerical values. One hot encoding could also be done but a lot of unique string values would encourage too many column's that would increase overhead complexity even more. Assigning each unique string with a number would ease the complexity and computation because the processor that is processing the data would only have to deal with a single digit or maximum double-digit numbers rather than an entire combination of 26 letters. This way we have encoded the entire categorical variable columns which are:

1) OsPlatformSubRelease
2) SkuEdition
3) SmartScreen
4) Census-MDC2FormFactor
5) Census-PrimaryDiskTypeName
6) Census-ChassisTypeName
7) Census-PowerPlatformRoleName
8) Census-InternalBatteryType
9) Census-OSArchitecture
10) Census-OSBranch
11) Census-OSEdition
12) Census-OSInstallTypeName
13) Census-OSWUAutoUpdateOptionsName
14) Census-GenuineStateName
15) Census-ActivationChannel
16) Census-FlightRing

G. NULL VALUES

Some features do contain null values, but not as many as the case that has happened in missing values. These null values cannot be processed by just keeping them empty. In python, SimpleImputer function could be used to fill these null values according to three strategies – mean, median, most frequent and constant. If this method needs to be implemented on a large number of columns it is going to take a couple of hours to execute it finely. Thereby, to eliminate long processing hours we have divided our columns into two lists – most-frequent-list1 and most-frequent-list2. The null values were filled with mean and most frequent strategies, in our implementation model.

H. Feature Scaling

Once the data is cleaned with all the dirty values, they are all scattered and lying numerically in different directions. This means that our data is now raw needs normalization. It will make the data fall into a single range of values. It helps in fastening the calculations that are involved in the algorithm. In our model, we have used the StandardScaler() method to implement normalization. Now, the data is fine and is ready to be implemented on the desired model of choice.

VI. MODELS IMPLEMENTED

A. LOGISTIC REGRESSION

Regression analysis may be a frame of predictive demonstrating strategy which explores the relationship between a subordinate (target) and free variable (s) (indicator). This strategy is utilized for forecasting, time series modelling and finding the causal impact relationship between the factors. For illustration, the relationship between why Does body weight, calorie intake, fat intake, and age have an influence on the probability of having a liver failure? Regression analysis is a critical tool for modelling and analyzing information. Here, we fit a curve/line to the data points, in such a way that the differences between the separations of data points from the curve or line are minimized. Benefits of utilizing regression analysis:

1. It shows the significant relationships between the dependent variable and independent variable.

2. It also tells us the quality of effect of different independent variables on a dependent variable. Logistic Regression is a classification algorithm that is used to predict the probability of a categorical variable (dependant). In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X. In our model, we have taken Logistic Regression as the first model, this is because of our dependable variable which is also a

binary column which depicts 0 as no and 1 as yes for the probability of getting attacked with malware in near future. The parameters set in our model is default parameters, which are listed below: [C=1.0, class-weight=None, dual=False, fit-intercept=True, intercept-scaling=1, l1-ratio=None, max-iter=100, multi-class='warn', n-jobs=None, penalty='l2',random-state=None, solver='lbfgs', tol=0.0001, verbose=0, warm-start=False] The accuracy observed with these parameters is on a broad scope is 62 per cent.

```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print("Report:",report)

Report:               precision    recall  f1-score   support

           0       0.61      0.63      0.62     29801
           1       0.62      0.60      0.61     30214

    accuracy                           0.62     60015
   macro avg       0.62      0.62      0.62     60015
weighted avg       0.62      0.62      0.62     60015
```

Fig. 2. Snapshot of various variable buckets

B. XGBOOST

As XGBoost provides a wrapper class to allow models to be treated like classifiers or regressors in the sci-kit-learn framework, we have implemented logistic regression by considering 52 dependent parameters, we have used XGBClassifier which is an XGBoost model for classification, XGBoost dominates structured or tabular data-sets on classification and regression predictive modelling problems thus increasing the accuracy of the model. The XGBoost library implements the gradient boosting decision tree algorithm. This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines. This approach supports both regression and classification predictive modelling problems. The score reports show its reports:

```
[0]   validation_0-auc:0.689974       validation_1-auc:0.671925
Multiple eval metrics have been passed: 'validation_1-auc' will be used for early stopping.

Will train until validation_1-auc hasn't improved in 100 rounds.
[100]   validation_0-auc:0.758549       validation_1-auc:0.706223
[200]   validation_0-auc:0.788679       validation_1-auc:0.710267
[300]   validation_0-auc:0.805004       validation_1-auc:0.711731
[400]   validation_0-auc:0.816869       validation_1-auc:0.712189
[500]   validation_0-auc:0.828105       validation_1-auc:0.712715
[600]   validation_0-auc:0.838662       validation_1-auc:0.712713
[700]   validation_0-auc:0.84765        validation_1-auc:0.712927
[800]   validation_0-auc:0.856299       validation_1-auc:0.712741
Stopping. Best iteration:
[703]   validation_0-auc:0.848075       validation_1-auc:0.712959
```

Fig. 3. Report 1

```
              precision    recall  f1-score   support

           0       0.65      0.65      0.65     30027
           1       0.65      0.65      0.65     29988

    accuracy                           0.65     60015
   macro avg       0.65      0.65      0.65     60015
weighted avg       0.65      0.65      0.65     60015


accuracy_score 0.6482046155127885

roc-auc score for the class 1, from target 'HasDetections'  0.7129587434045564

elapsed time in seconds:  494.4030694961548
```

Fig. 4. Report 2

1) Matrix: This Values are drawn for our machine learning model after performing XGboost Algorithm with various dependent variables and HasDetections Column True Positive Rate (TPR): 19393 False Positive Rate (FPR): 10634 False Negative Rate (FNR): 10479 True Negative Rate (TNR):19509

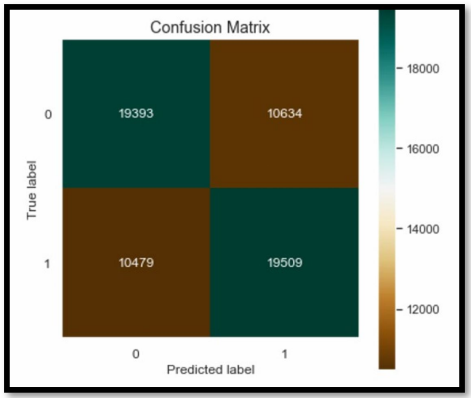

Fig. 5. Confusion Matrix

2) ROC Curve: AUC : ROC curve is a performance measurement for our problem at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much model is capable of distinguishing between various parameters. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, Higher the AUC, better the model is at distinguishing between the parameters and HasDetectionsColums.

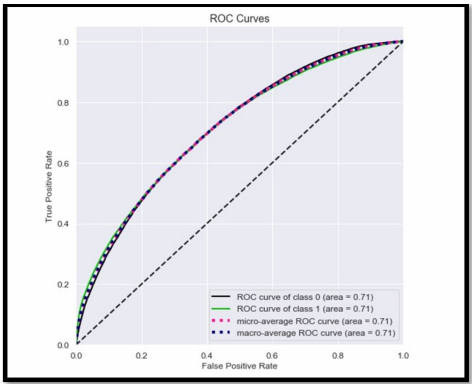

Fig. 6. ROC Curves

3) Lift Curve: As we used the classification model within XGboost we implemented a lift curve which is a way of visualizing the performance of our classification model.

4) KS Statistic Plot:

5) Precision-Recall Curve: Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. The precision-recall curve shows the trade-off between precision and recall

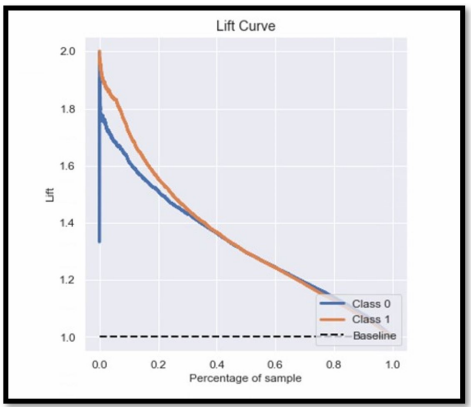

Fig. 7. Lift Curve

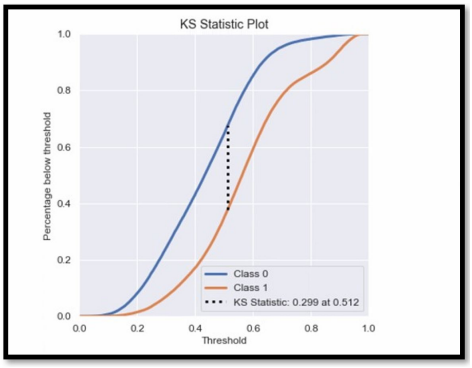

Fig. 8. Statistic Plot

for different threshold in our model. A high area under the curve represents both high recall and high precision, where high precision relates to a low false-positive rate, and high recall relates to a low false-negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning most of all positive results (high recall).

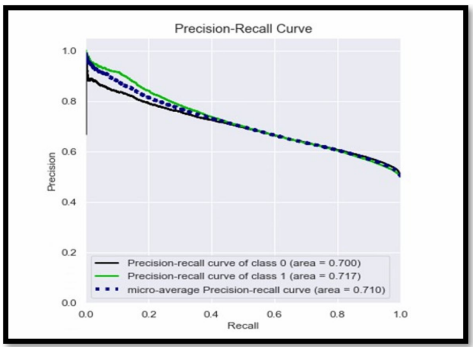

Fig. 9. Precision-Recall Curve

6) Cumulative Gains Curve:

7) Hyperparameter Tuning: In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value

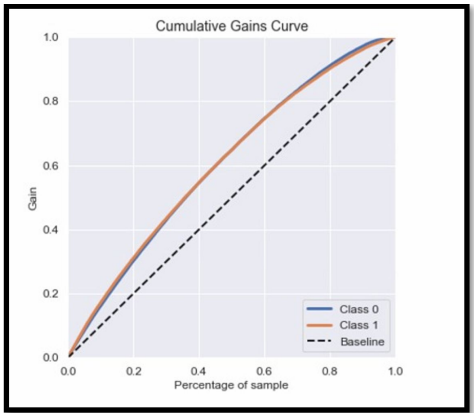

Fig. 10. Cumulative-Gain Curve

is used to control the learning process. Hyperparameter tuning works by running multiple trials in a single training job. Each trial is a complete execution of your training application with values for your chosen hyperparameters, set within limits you specify. The AI Platform training service keeps track of the results of each trial and adjusts for subsequent trials. When the job is finished, you can get a summary of all the trials along with the most effective configuration of values according to the criteria you specify. Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid GridSearchCV(logreg,param,scoring='roc-auc',refit=True,cv=10) The parameters implemented in this model are: param = 'C':[0.001,0.003,0.005,0.01,0.03,0.05,0.1,0.3,0.5,1,2,3,3,4,5,10,20] params = 'num-leaves': 256, 'min-data-in-leaf': 42, 'objective': 'binary', 'max-depth': 5, 'learning-rate': 0.05, "boosting": "gbdt", "feature-fraction": 0.8, "bagging-freq": 5, "bagging-fraction": 0.8, "bagging-seed": 11, "lambda-l1": 0.15, "lambda-l2": 0.15, "random-state": 42, "verbosity": -1 The best obtained value for the curve is - Best roc-auc: 0.5469, with best C: 'C': 0.5

VII. Artificial Neural Network

Neural networks are a set of algorithms, modelled freely after the human brain, that are planned to recognize patterns. They translate sensory information through a kind of machine recognition, labelling or clustering raw input. The designs they recognize are numerical, contained in vectors, into which all real-world data, be it pictures, sound, content or time arrangement, must be translated. The Design of an Artificial Neural Network: ANN is a set of associated neurons organized in layers:

(A) Input layer: brings the initial information into the framework for advance processing by consequent layers of artificial neurons.

(B) Hidden layer: a layer in between input layers and yield layers, where manufactured neurons take in a set of weighted inputs and create an output through an activation function.

(C) Output layer: the final layer of neurons that produces given outputs for the program.

Once the basic model is constructed, we have to decide upon which activation function that is supposed to be used. The activation functions are among the following categories:

1)Sigmoid: A sigmoid function is a function having a characteristic "S"- curve or sigmoid curve. Often, sigmoid function refers to the unique case of the logistic function which generates a set of probability outputs between 0 and 1 when fed with a desired set of inputs.

2)Softmax: Unlike the Sigmoid activation function, the Softmax activation function is used for multi-class classification. It calculates the probability distribution of the event over 'n' different events. In usual way of telling, this function will calculate the probabilities of each target class over all possible target classes.

3)Relu: Instead of the sigmoid activation function, most recent artificial neural networks use rectified linear units (ReLUs) for the hidden layers. A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input.

Our model has taken Relu activation function to be the parameter for all the hidden and input layers. Analysing the performance, we have got better results when the number of layers is 58, excluding the output layer. The accuracy performance observed is fifty per cent approximately. The output dimension mentioned is decided by approximately (no of features/2). Further analysis is also considered by looking at the confusion matrix, where the True Positives are 29801 and False negatives are 30214.

VIII. Concluding Analysis

After implementing all the three models over this data-set, we have found that XGBoost Algorithm has produced higher accuracy as such. It has produced 62 percent of correctiveness. The prediction analysis also gives us a taste of challenge involved in working with a huge data-set. The accuracy seems very high as it is equal to impossible: in predicting whether a system in near future gets attacked by malware just by basic properties of a machine. Algorithms run very slowly on such huge data-set which is the main drawback. Deleting more columns would wipe out some significant features and reduce the complexity. In order to face the challenge, we removed only those columns that had negligible relationship with the output feature.

IX. Learning Experience

We have learnt a lot about how changing parameters affect our accuracy score. We also learnt how to deal with big data set by using memory usage reduction algorithm techniques. The challenges that we faced were quite many, firstly deciding upon which data should be removed and which data should be held on for processing, secondly doing all the loop analysis and coding from scratch on our own, thirdly waiting long hours for code to get executed and finally dealing with how neural networks and dense neurons analyse the input data to produce output data.

X. Work Division

DATA Preprocessing is done by all of us together in DataScience Lab at Sacramento State University. ARUN REDDY - Implementation of XGBOOST, Report Writing and Presentation SAI KUMAR MANCHIKANTI - Implementation of Artificial Neural Network, Report Writing and Presentation YASHASRI.E - Implementation of Logistic Regression, Report Writing and Presentation

XI. References

References

[1] https://medium.com/datadriveninvestor/what-ive-learned-microsoft-malware-prediction-competition-on-kaggle-3c8189dcc850
[2] https://github.com/miteshjoshi/malware-prediction/blob/master/papers/kolter06a.pdf
[3] https://www.kaggle.com/c/microsoft-malware-prediction/notebooks
[4] https://www.kaggle.com/artgor/is-this-malware-eda-fe-and-lgb-updated
[5] https://www.kaggle.com/jiegeng94/everyone-do-this-at-the-beginning
[6] https://ieeexplore.ieee.org/document/8681127
[7] https://ieeexplore.ieee.org/document/6287147?reload=truearnumber=6287147
[8] https://ieeexplore.ieee.org/document/8258483
[9] https://ieeexplore.ieee.org/document/8463441