MALWARE DETECTION USING MACHINE LEARNING

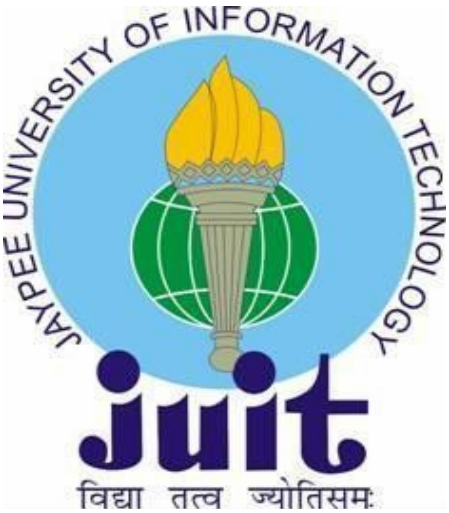
Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By
Naimish Awasthi (161455)
Nishil Pagare (161307)

Under the supersvison of
Dr. Pradeep Kumar Gupta
to



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology
Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Malware Detection using Machine Learning "** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2018 to December 2018 under the supervision of **Dr. Pradeep Kumar Gupta (Associate Professor).**
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Signature)
Naimish Awasthi (161455)
Nishil Pagare (161307)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Signature)
Dr. Pradeep Kumar Gupta
Associate Professor
Computer Science and Engineering and Information Technology
Dated:

Acknowledgement

We express our profound gratitude and deep regards to our project supervisor and mentor Dr. Pradeep Kumar Gupta for his guidance and support. He took keen interest and guided and encouraged us both in our project titled “Malware Detection using Machine Learning”. The guidance and mentorship shall carry us long way in our journey of life.
We would also like to thank all the staff members of Jaypee University of Information Technology, Waknaghat for their timely support and all the information they provided.
Lastly, we would like to thank our batch mates for their motivation without which it would be impossible to complete this project.
Till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research and we got to know a lot of new things in machine learning and deep learning.

Abstract

In recent years, the malware industry has become a well organized market involving large amounts of money. Well funded, multi-player syndicates invest heavily in technologies and capabilities built to evade traditional protection, requiring anti-malware vendors to develop counter mechanisms for finding and deactivating them. In the meantime, they inflict real financial and emotional pain to users of computer systems. One of the major challenges that anti-malware faces today is the vast amounts of data and files which need to be evaluated for potential malicious intent. For example, Microsoft's real-time detection anti-malware products are present on over160M computers worldwide and inspect over 700M computers monthly. This generates tens of millions of daily data points to be analyzed as potential malware. One of the main reasons for these high volumes of different files is the fact that, in order to evade detection, malware authors introduce polymorphism to the malicious components. This means that malicious files belonging to the same malware "family", with the same forms of malicious behavior, are constantly modified and/or obfuscated using various tactics, such that they look like many different files.

In order to be effective in analyzing and classifying such large amounts of files, we need to be able to group them into groups and identify their respective families. In addition, such grouping criteria may be applied to new files encountered on computers in order to detect them as malicious and of a certain family.

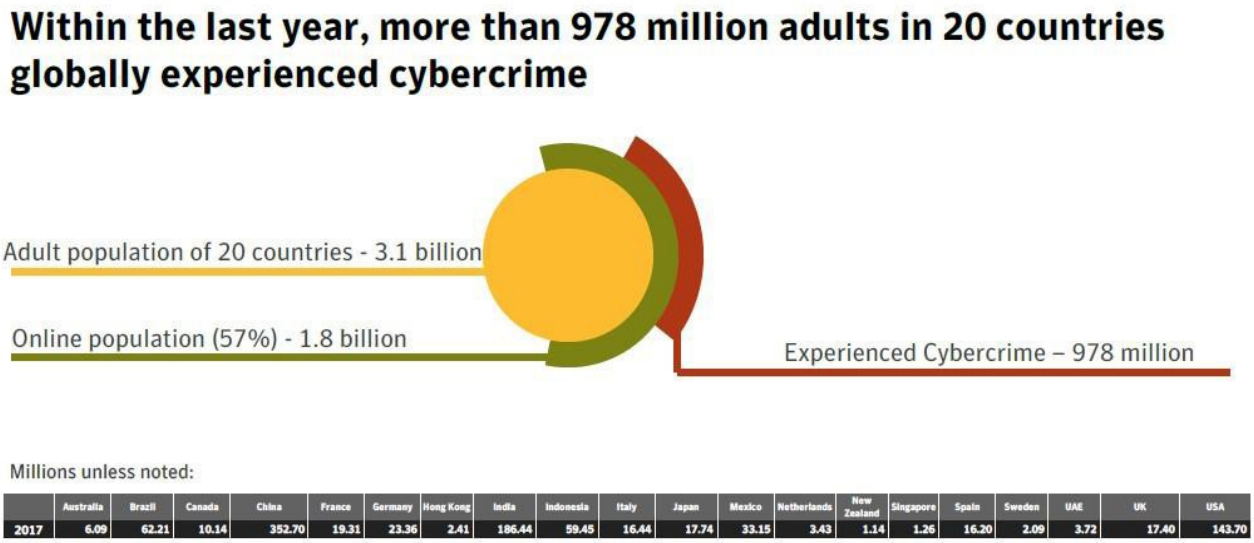
Table of Content :

INTRODUCTION

1.1 Introduction

Malware refers to malicious software perpetrators dispatch to infect individual computers or an entire organization’s network. It exploits target system vulnerabilities, such as a bug in legitimate software (e.g., a browser or web application plugin) that can be hijacked. A malware infiltration can be disastrous — consequences include data theft, extortion or the crippling of network systems.Malware is one of the most serious security threats on the Internet today. In fact, most Internet problems such as spam e-mails and denial of service attacks have malware as their underlying cause. That is, computers that are compromised with malware are often networked together to form botnets, and many attacks are launched using these malicious, attacker-controlled networks.In order to deal with the new malware generated, new techniques to detect them and prevent any damage caused by them.One of the major challenges that anti-malware faces today is the vast amounts of data and files which need to be evaluated for potential malicious intent. For example, Microsoft's real-time detection anti-malware products are present on over160M computers worldwide and inspect over 700M computers monthly. This generates tens of millions of daily data points to be analyzed as potential malware. One of the main reasons for these high volumes of different files is the fact that, in order to evade detection, malware authors introduce polymorphism to the malicious components. This means that malicious files belonging to the same malware "family", with the same forms of malicious behavior, are constantly modified and/or obfuscated using various tactics, such that they look like many different files.

In order to be effective in analyzing and classifying such large amounts of files, we need to be able to group them into groups and identify their respective families. In addition, such grouping criteria may be applied to new files encountered on computers in order to detect them as malicious and of a certain family.



	Australia	Brazil	Canada	China	France	Germany	Hong Kong	India	Indonesia	Italy	Japan	Mexico	Netherlands	New Zealand	Singapore	Spain	Sweden	UAE	UK	USA
2017	6.09	62.21	10.14	352.70	19.31	23.36	2.41	186.44	59.45	16.44	17.74	33.15	3.43	1.14	1.26	16.20	2.09	3.72	17.40	143.70

Fig 1.1: Analysis by Norton about Cybercrimes.

1.2 Problem Statement

One of the major challenges that anti-malware faces today is the vast amounts of data and files which need to be evaluated for potential malicious intent.Malicious files belonging to the same malware "family", with the same forms of malicious behaviour, are constantly modified and/or obfuscated using various tactics, such that they look like many different files.

In order to be effective in analysing and classifying such large amounts of files, we need to be able to group them into groups and identify their respective families.

1.3 Objectives

Trim our dataset
Classify malware into different families

1.4 Methodology

For this problem, we are using the unprecedented malware dataset by Microsoft for grouping variants of malware files into their respective families.

We are provided with a set of known malware files representing a mix of 9 different families. Each malware file has an Id, a 20-character hash value uniquely identifying the file, and a Class, an integer representing one of 9 family names to which the malware may belong:

1. Ramnit
2. Lollipop
3. Kelihos_ver3
4. Vundo
5. Simda
6. Tracur
7. Kelihos_ver1
8. Obfuscator.ACY
9. Gatak

the raw data contains the hexadecimal representation of the file's binary content, without the PE header (to ensure sterility). We were also provided a metadata manifest, which is a log containing various metadata information extracted from the binary, such as function calls, strings, etc. This was generated using the IDA disassembler tool.

LITERATURE REVIEW

(2.1) Terminologies

(a). Machine Learning- Machine Learning calculations are a kind of calculations that are a part of man-made brainpower and that makes the framework or the product application to be sufficiently keen to have the option to progressively precise without being expressly customized and can anticipate results. The principle thought behind these kind of calculations is that it gets input information as content or pictures and the framework or the model is prepared with the factual contributions to distinguish or foresee the yield and even refreshed the yields as new information gets accessible. It requires the calculation to look through the informational collection and search for examples or likenesses and controlling orchanging the framework as needs be.

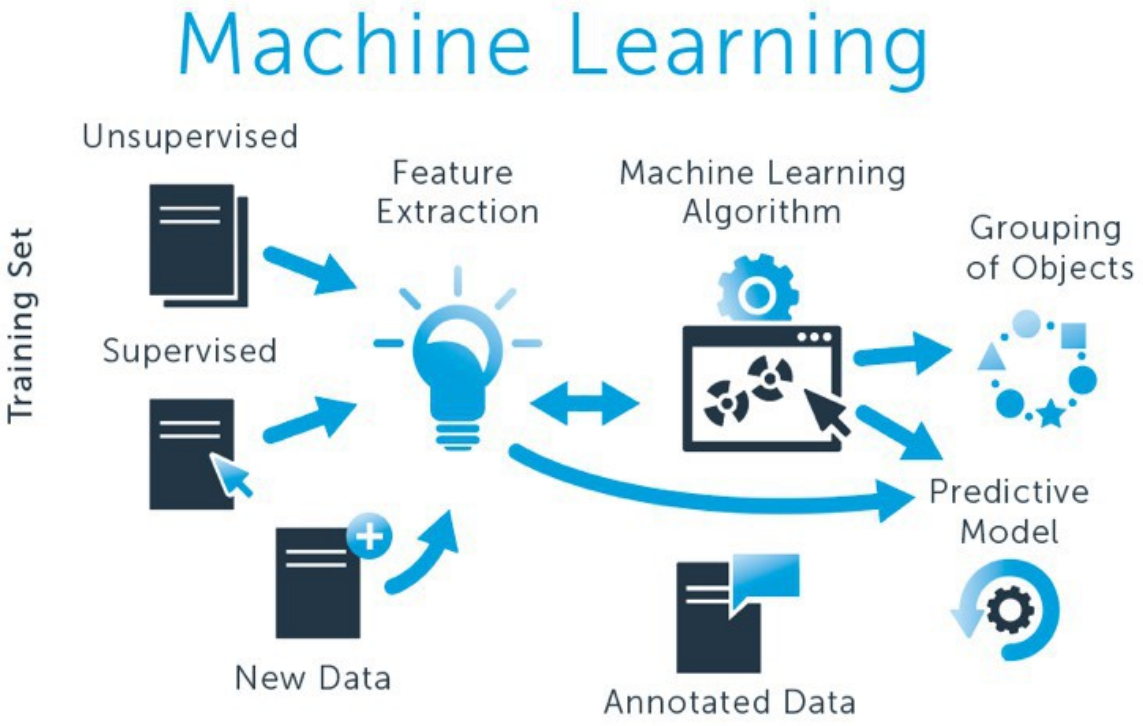


Fig 2.1: Introduction to ML

(b). How machine learning works

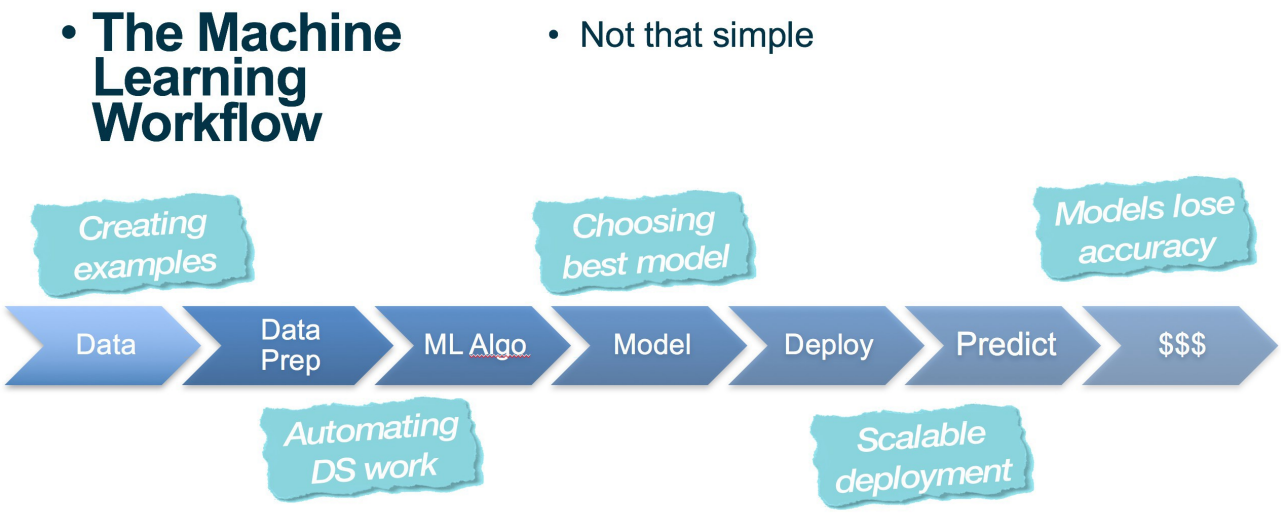


Fig 2.2: ML algorithm workflow

The methodology of AI starts with the variety of data or discernments as the information dataset which can be as pictures, substance, tables, etc. Further,various predefined AI counts are applied to the information data which either orchestrate the data into social occasions or perceives plans among the dataset to anticipate the yield and give appropriate results. Man-made intelligence figurings are roughly requested into controlled and solo learning computations.

(c). Types of Machine Learning

Supervised Machine learning- This sort of estimations work for a dataset which is presently being set up by past yields and consequences of the past using named data to anticipate the aftereffect of the new data. For this circumstance the known dataset is destitute down, the count by then conveys an initiated limit which help in gauge of the yield estimations of new data. It can similarly inspect the data and the outcome and appear differently in relation to the as of late set away data with see botches and as prepared tochange and set up the model suitably..

Unsupervised Machine Learning- This sort of not exactly equivalent to coordinated AI estimations as this counts are used when the model isn't set up before neither it is described nor it is named. Independent learning estimations make the system to initiate a covered structure or model in the

unlabelled dataset and anticipate potential results with the usage of such models while emptying the abnormalities.

Semi-Supervised Machine Learning-The semi-coordinated AI figurings are counts which uses advantages of both controlled and solo AI estimations for setting up the dataset and likewise makes a great deal of productive and momentous classifiers. In these sort of estimations, the model uses both named and unlabelled data for the planning and it by and large requires a constrained amount of named data and a reasonably colossal number of unlabelled data which are used simultaneously to set up the model. This is used for redesigning the precision and the gauge limits of the model and as needs be consistently used by virtue of data which requires both talented and significant hotspots for getting ready and picking up from it.

Reinforcement Machine Learning-The rule thought of help learning is that it is compensate based getting ready in which the model collaborates with the earth by doing exercises and discovering botches or rewards. The most relevant characteristics of help learning are the experimentation and conceded compensate. For this circumstance the model increases from its mistakes or bungles and the model is made to team up with the machines to normally choose the outcome and the ideal direct to overhaul the working and for execution help.

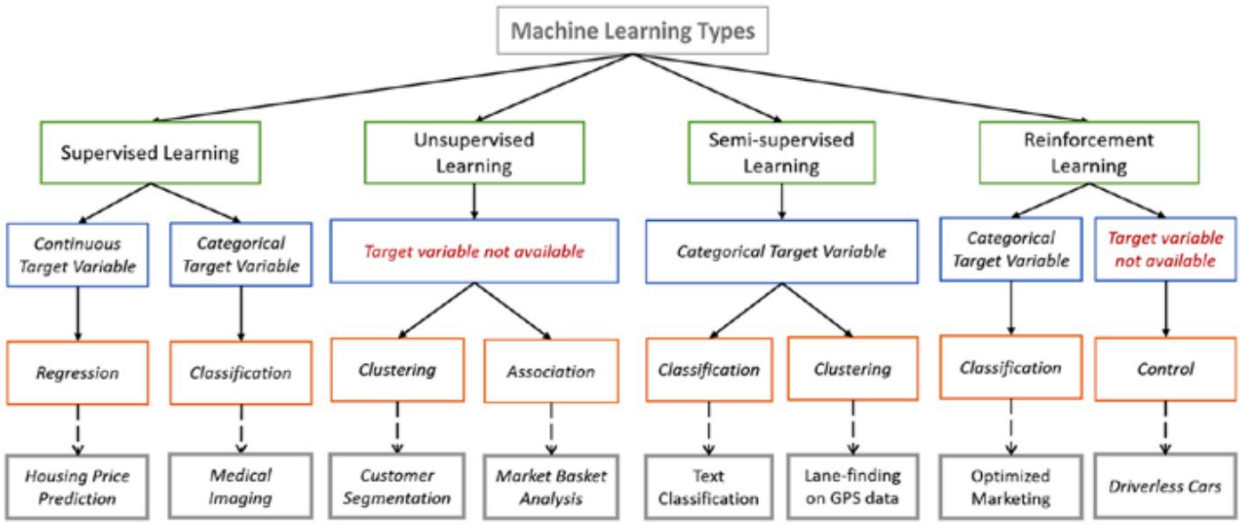


Fig 2.3: Types of machine learning

(d). Machine Learning Algorithms-

This segment contains of different ML calculations which are to be utilized in the task and talked about-

- **Nearest Neighbors-** KNN or K-nearest neighbors is a sort of calculation which can be utilized both for relapse and grouping issues however is for the most part utilized in arrangement issues. This calculation is simple in elucidation and requires exceptionally low computation time and along these lines is a generally utilized ML calculation. The K in this calculation is the quantity of neighbors which are characterized by the client. In this calculation we utilize the Euclidean separation to quantify the K nearset neighbors of the information point and foresee the yield as per its neighbors.

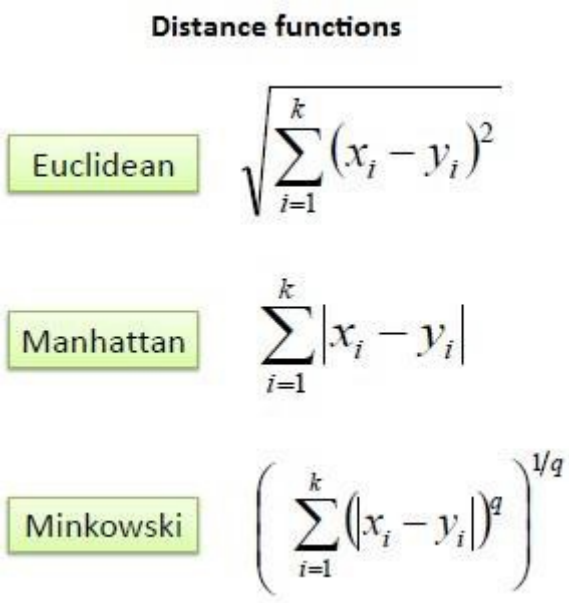


Fig 2.4: Formulae used for calculating nearset distance.

-Naïve Bayes- Naive Bayes hypothesis is a kind of order calculation which can be utilized for both double and multi class arrangement issues. This hypothesis is called so on the grounds that it has its underlying foundations of Bayes hypothesis.Innocent Bayes is regularly spoken to by probabilities. In this model theinformation is put away as probabilities for an educated model.

Equation- **P(h|d) = (P(d|h)*P(h))/ P(d)**

-Decision Trees- Decision tree calculation is a sort of directed learning calculation wherein an information structure is utilized to tackle an issue. For this situation the leaf hub is alluded to as the class mark and the interior hubs of the tree speak to the qualities. They can take care of the issues of both arrangement and relapse. At first, we consider the entire dataset as the root and clear cut element esteems are liked and the persistent qualities are first made discrete qualities before utilizing them to fabricate the model. At that point measurable techniques are utilized for requesting the properties as inner hub or root.

Equation-

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

Where:

- S - represents the training set
- A - represents the attribute that we are comparing
- $|S|$ - represents the number of elements in the training set
- $|S_v|$ - represents the number of elements where the attribute has a particular value.

$$Entropy(S) = \sum_i -p_i log_2 p_i$$

Linear Regression- Linear regression is a calculation which utilizes the measurable ideas and models a connection between the information and yield numerical qualities. The model is spoken to by a direct condition which joins theinfo estimations of a particular set and predicts the yield for a lot of that information esteems.

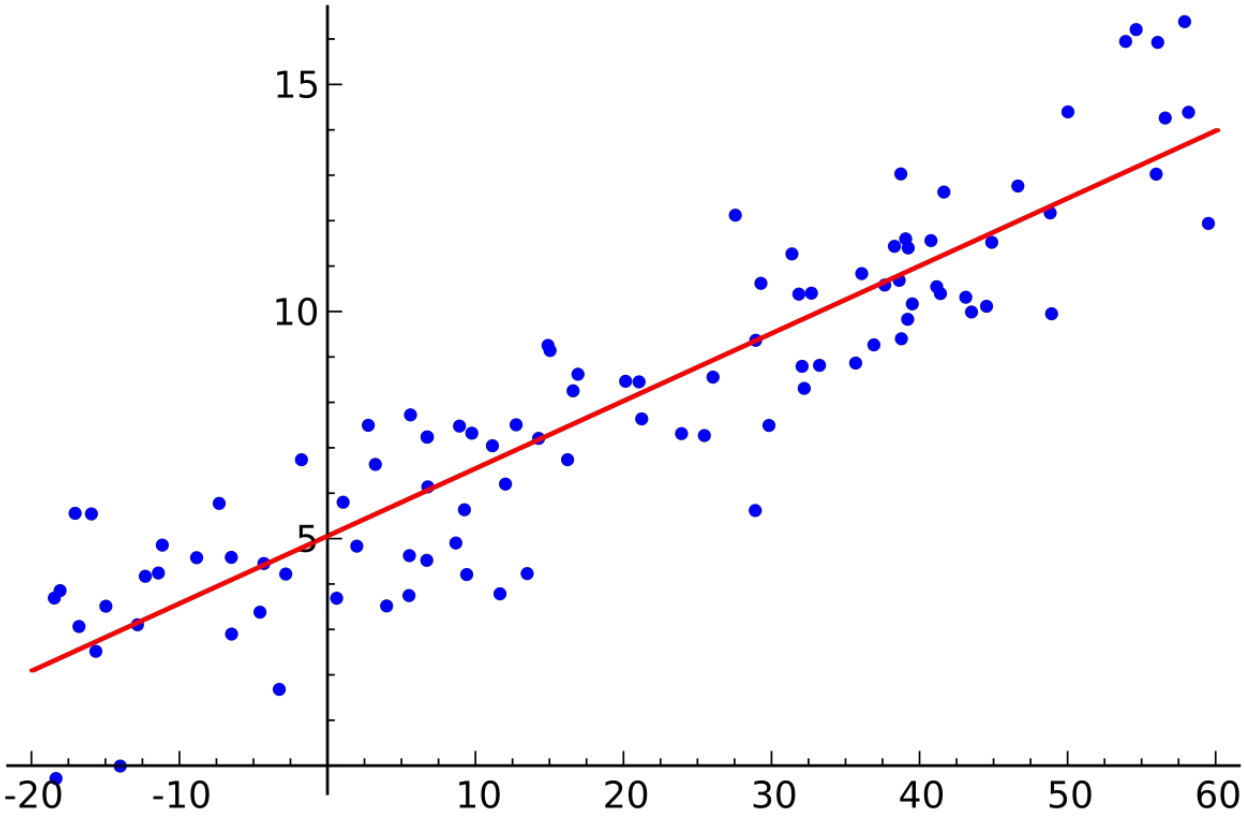


Fig 2.5: Pictorial representation of example of LR.

Support Vector Machines-SVM is a regulated AI calculation which is ordinarily utilized for both relapse and characterization issues. It is broadly utilized in arrangement issues where every datum thing is plotted in n-dimensional space and n characterizes the highlights present and the estimation of each element is the estimation of each organize. Further, a different hyper plane is made to separate the two classes.

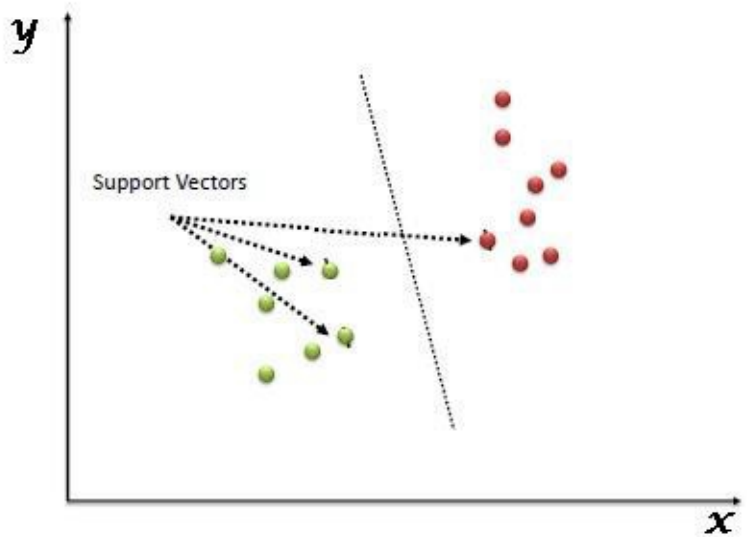


Fig 2.6: Portrayal of SVM.

Comparing Supervised Learning Algorithms : Table

Algorithm	Problem Type	Results interpretable by you?	Easy to explain algorithm to others?	Average predictive accuracy	Training speed	Prediction speed	Amount of parameter tuning needed (excluding feature selection)	Performs well with small number of observations?
KNN	Either	Yes	Yes	Lower	Fast	Depends on n	Minimal	No
Linear regression	Regression	Yes	Yes	Lower	Fast	Fast	None (excluding regularization)	Yes
Logistic regression	Classification	Somewhat	Somewhat	Lower	Fast	Fast	None (excluding regularization)	Yes
Naive Bayes	Classification	Somewhat	Somewhat	Lower	Fast (excluding feature extraction)	Fast	Some for feature extraction	Yes
Decision trees	Either	Somewhat	Somewhat	Lower	Fast	Fast	Some	No
Random Forests	Either	A little	No	Higher	Slow	Moderate	Some	No

Table 2.1:Distinction between different ML Algorithms

(e). Interpretation of Performance Measures

There are different strategies to assess the exhibition of the calculations. One of these techniques is to decide the region under the bend or the ROC bend and different parameters which are otherwise called Confusion Metrics. To assess the exhibition proportion of the order model for a dataset that gives the genuine qualities are known, the perplexity framework table is utilized.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Table 2.2: Confusion Matrix

True Positives (TP) - These are the qualities which are effectively anticipated and are sure qualities which can be depicted as the positive estimation of real class and positive estimation of anticipated class. It is signified by TP.

.

True Negatives (TN) - These are the qualities which are wrongly anticipated however is valid in genuine for example - when we have positive estimations of genuine class however refutation in anticipated class.

.

False Positives (FP) –These are the qualities which are wrongly anticipated however is valid in genuine for example - when we have positive estimations of real class however invalidation in anticipated class.

.

False Negative (FN) – These are the qualities which are wrongly anticipated and negative in real class.

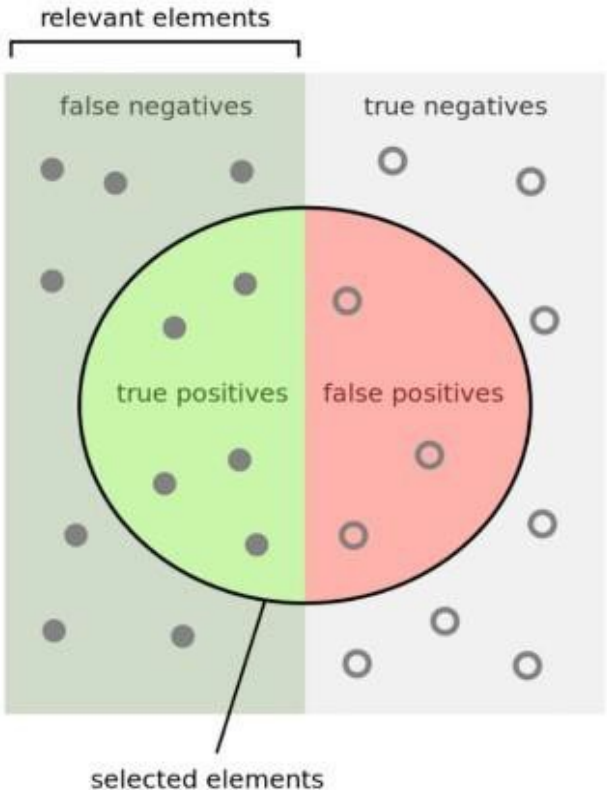


Fig 2.7: Pictorial representation of confusion matrix

Further, we investigate more parameters of execution which are exactness, accuracy, Recall and F1 score.

· **Accuracy** – Accuracy is the most common execution measure and it is basically an extent of adequately foreseen recognition to the total discernments. One may envision that, in case we have high precision, our model is perfect. Really, precision is an uncommon measure yet exactly when you have symmetric datasets where estimations of bogus positive and bogus negatives are generally same. Thusly, you have to look at changed parameters to survey the execution of your model.

Equation = TP+TN/TP+FP+FN+TN

· **Precision -** Precision is the extent of precisely foreseen positive discernments to the total foreseen positive recognitions. The request that this estimation answer is of all voyagers that named as suffer, what number of truly persevere? Highaccuracy relates to the low bogus positive rate.

Equaation = TP/TP+FP

· **Recall (Sensitivity)** - Review is the extent of viably foreseen positive recognitions to the all discernments in real class - yes. The request survey answers is: Of the significant number of explorers that truly suffer, what number of did we mark?

Equation = TP/TP+FN

· **F1 Score** -F1 Score is the weighted typical of Precision and Recall. Therefore,this score thinks about both bogus positives and bogus negatives. Naturally it isn't as clear as precision, yet F1 is regularly more important than accuracy, especially in case you have an uneven class movement. Accuracy works best if bogus positives and bogus negatives have practically identical cost. In case the cost of bogus positives and bogus negatives are out and out various, it's more brilliant to look at both Precision and Recall.

Equation = 2*(Recall * Precision) / (Recall + Precision)

(f) **Deep Learning-**

Deep learning is a progressively perplexing and insightful sub classification of AI which has its calculations motivated by the working and structure of the human cerebrum comprehensively known as Artificial neural system. What's more, it additionally alludes to the assortment of methods which are utilized for learning in neural system with various layers. Counterfeit Neural Network or ANN are the kind of neural system model which takes its motivation and chips away at the essential thought of the sensory systems and the handling of data in human cerebrum to gain from information. Here, the learning components can be either directed, semi-managed or solo. Profound learning has been demonstrated effective in different fields and brought about progressively practical learning of machines.

Its uses ranges from the field of medication structure to the traffic expectation and furthermore for the item acknowledgment. A profound neural system is not quite the same as a neural system due to the quantity of layers. While the execution of profound learning, we experienced two primary issues, for example, 1) the computational power required for the way toward preparing the model was higher than that of the framework accessible and in this way requires more opportunity forcalculation. 2) Another issue that experienced during the usage is the inclination disappearing issue, that is, in a neural system that has actuation capacities, for example, the hyperbolic digression or the sigmoid and the angle extend is (- 1,1) or [0,1), the backpropagation is normally processed by chain rule, increasing k to this little numbers from the yield layer through a k-layer organize, which implies that the slope diminished exponentially with k. Coming about of this is the front layers of the model trains slowly than the other layers.

(g) **Perceptrons**

The most essential and the early regulated learning calculation is the perceptron and it is the littlest structure square of the system that is Artificial Neural Network.

The working of the perceptrons is by taking numerous information sources (x1,x2… … xj) and creation of a solitary yield (y). Likewise, weighted data sources were additionally considered to help decide the significance of particular contributions to the yield. The subsequent yield is either 0 or 1 and it is possibly dictated by checking if the weighted whole is more prominent than 0 or under 0.

Weighted Sum- $\Sigma j\ wj * xj + b$
{ 1: if $w * x + b > 0$
Output =
0: if $w * x + b <= 0$ }

Be that as it may, as the perceptron just gives the yield as 0 or 1, it makes it extremely troublesome or about difficult to broaden the working and functionalities of the model to have the option to chip away at order errands having numerous classes. Besides, this issue can be settled by having various perceptrons in a layers ensuring that each perceptron in the layer acquires a similar info and all the perceptrons are liable for the yield work. The Artificial Neural Network (ANN) is just perceptrons with more than one layers though the perceptron is simply an ANN with single layer, which is frequently the yield layer having just 1 neuron.

(h) **Loss Function**

The exhibition of the neural system is estimated by a capacity which is called as the cost capacity or the misfortune work which helps in estimating the inconsistency between the expectation by the calculations and the right name if the forecast or the aggregate arrangement of forecast is given alongside the name or alot of marks. Among the different cost work accessible the most basic and the usually utilized in neural systems is the mean squared blunder (MSE).

The mean squared error can be defined as:

$$L(W,b) = 1/m(\sum_{i=1}^{m} ||h(x^i) - y^i||^2)$$

where:

- m is the number of training examples
- x^i is the i^{th} training sample
- y^i is the class label for the i^{th} training sample
- $h(x^i)$ is the algorithm's prediction for the i^{th} training sample

A definitive objective of preparing the neural systems is to limit the expense/misfortune capacity and locate the individual loads and inclinations that do as such. For this method, we utilized a calculation which called as angle drop calculation.

SYSTEM DEVELOPMENT

(3.1) System Requirements

The calculations that are being actualized in this undertaking requires some
nonexclusive framework as it requires preparing of calculations.

· Windows 10 (64-bit)
· ANACONDA
· Python
· 4 GB RAM
· Intel(R) Core(TM) i3-3120M CPU @ 2.50 GHz

(3.2) Reason for using Python

Python is a programming language with an enormous group of spectators and it is extremely straightforward and can be effectively coherent. Besides, python offers the assortment of bundles which makes the most scary calculations or undertakings less difficult. Python has libraries for pretty much every usable document for example - with working with pictures, working with content or working with sound records. In any event, when working with another OS, python is entirely pliant.
Python has an enormous network which makes it simpler to look for help and tips and deceives.

(3.3) Reason for using ANACONDA

Anaconda is broadly well known as it gives every one of the libraries preintroduced and make the client free from issue of the generally introducing all libraries. It has around 100 bundles which can be utilized for information science, AI or factual investigation.

(3.4) SCIKIT LEARN

Scikit learn is a library in python typically utilized for AI and is fit for including different relapse, grouping and bunching calculations.

(3.5) PANDAS

It is an open source python library which provides better results. This library is anything but difficult to utilize and even give information structure and information examination devices. This library is broadly utilized in every scholarly field, business and mechanical fields.

(3.6) KERAS

It is an open source python library that is commonly utilized for neural system. It is regularly intended to run quick experimentation of numerous intricate profound learning calculations. It as a rule centers around being more easy to use, progressively particular and increasingly extensible.

(3.7) Pillow

It is an open source python library that is utilized for imaging and furthermore includes bolsters in the content for opening, likewise controlling, further sparing a wide range of picture record positions. It offers standard methods for doing picture control. This incorporates per-pixel controls and concealing and straightforwardness dealing with.

(3.8) TENSERFLOW

It is a free open source python library that is use for dataflow and differentiableprogramming which is utilized over a specific scope of errands. The tenserflow canbe characterized as an emblematic math library which can likewise be utilized forsome, AI applications for instance neural systems.

In order to be effective in analysing and classifying such large amounts of files, we need to be able to group them into groups and identify their respective families. In addition, such grouping criteria may be applied to new files encountered on computers in order to detect them as malicious and of a certain family.

For this problem, we are using the unprecedented malware dataset by Microsoft for grouping variants of malware files into their respective families.

We are provided with a set of known malware files representing a mix of 9 different families. Each malware file has an Id, a 20-character hash value uniquely identifying the file, and a Class, an integer representing one of 9 family names to which the malware may belong:

1. Ramnit
2. Lollipop
3. Kelihos_ver3
4. Vundo
5. Simda
6. Tracur
7. Kelihos_ver1
8. Obfuscator.ACY
9. Gatak

the raw data contains the hexadecimal representation of the file's binary content, without the PE header (to ensure sterility). We were also provided a metadata manifest, which is a log containing various metadata information extracted from the binary, such as function calls, strings, etc. This was generated using the IDA disassembler tool.

Feature Engineering :

Initial feature engineering consisted of extracting various keyword counts from the ASM files as well as the entropy and file size from the BYTE files of the 10868 malware samples in the training set.

Image files of the first 1000 bytes of the ASM and BYTE files were created and combined with keyword and entropy data. This resulted in a set of 2018 features.

Flow control graphs and call graphs were generated for each ASM sample. A feature set was then generated from the graphs, including graph maximum delta, density, diameter and function
counts etc.

Feature Selection :

Statistical analysis of the feature set using chi-squared tests to remove features that are independent of the class labels or have low variance. The BYTE file images were found to be weak
learners and were removed from the feature set. A comparison of the best features from the chi-squared
tests with reduced feature sets of between 10% - 50% of the original features.

In this feature selection we will closely examine each and every feature, how a particular feature going to affect the malware category, in this we will only keep necessary features like features not having variance very close to 0 or very close to 1 and we will apply some pre-defined tests to get correct features for training and testing in order to achieve high accuracy with minimum error.

There might be high possibility of data missing from the dataset, in order to overcome this problem, we have may possibilities like deleting that particular entry or filling the null entries on basis of their frequency or median/mean for categorical or Numerical data respectively.

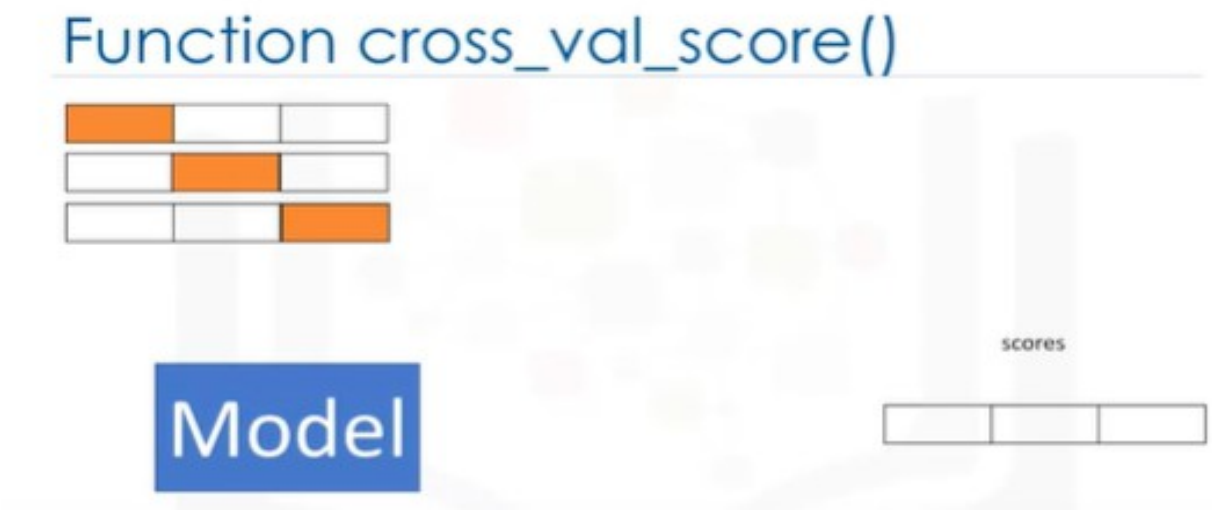
Now Firstly load our data set

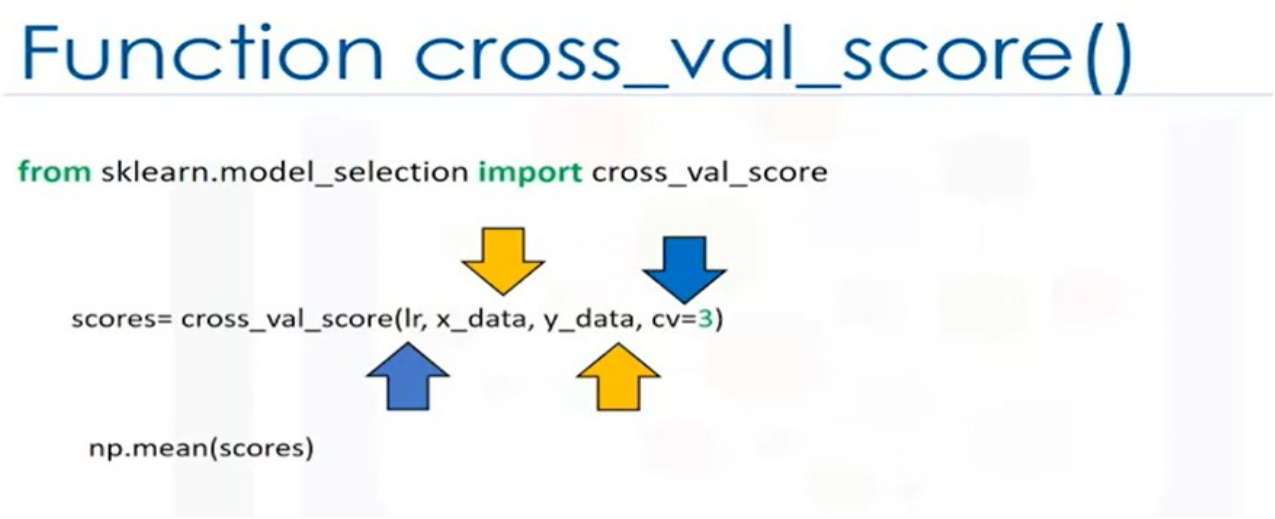
```
In [2]: # First load the ASM training data and training labels
        sorted_train_labels = pd.read_csv('data/sorted-train-labels.csv')
        combined_train_data = pd.read_csv('data/final-combined-train-data-30percent.csv')
        call_graph_features_train = pd.read_csv('data/final-call-graph-features-10percent.csv')
```

```
In [3]: sorted_train_labels.head()
```

Out[3]:

	Id	Class
0	01IsoiSMh5gxyDYTI4CB	2
1	01SuzwMJEIXsK7A8dQbl	8
2	01azqd4InC7m9JpocGv5	9
3	01jsnpXSAlgw6aPeDxrU	9
4	01kcPWA9K2BOxQeS5Rju	1





Here lr represent the model type, and you Can see our Data Splitted into 3 Folds and from each fold one part(white) used for training and other part(orange) for testing and collectively returning mean of scores

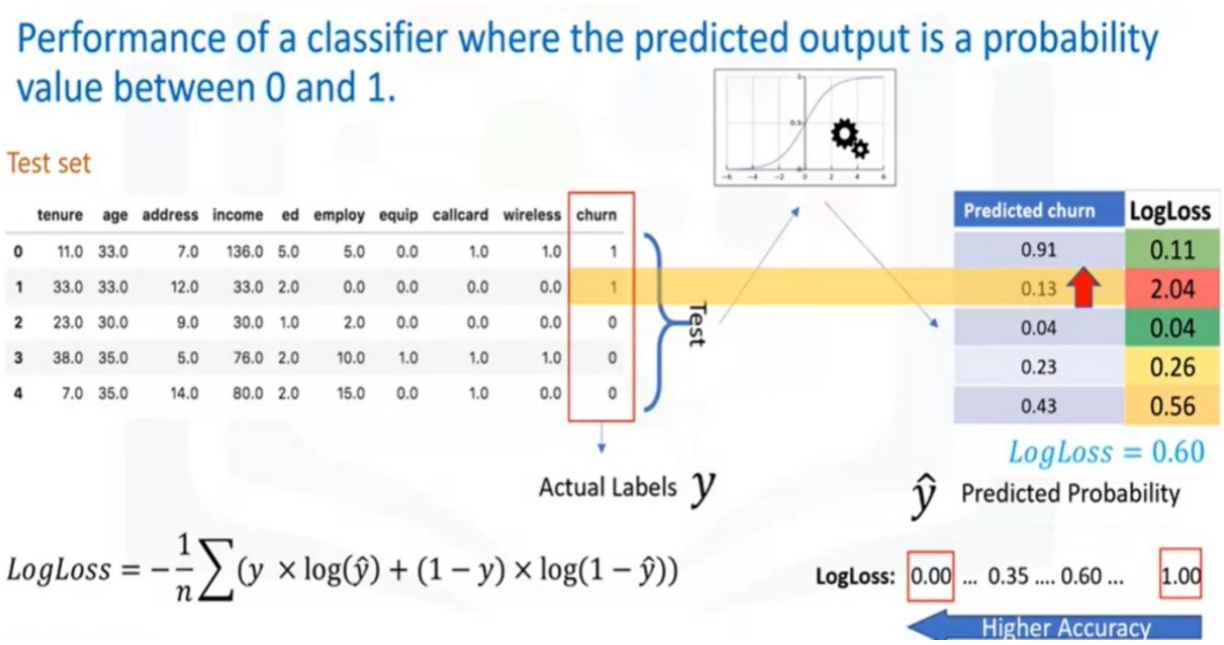
CV= how many folds you want in data

So similarly we are splitting our data here into 10 folds to get more accuracy in our scores

An extra-trees classifier.

This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

So Log Loss is basically probability of Comparison between the lograthmic value of Predicted data and Original Data. And as Log loss approaches to 0 it shows model with higher accuracy.



Selection Comparison

Testing with an ExtraTreesClassifier and 10-fold cross validation produced the following results:

- Original ASM Keyword Counts (1006 features):
  logloss = 0.034

Since when we performed our Testing on Original Data Set Without excluding any features and considering all features equally important we get to know that the accuracy of our result was less as compared to other in this comparison we got log loss of 0.034.

- 10% Best ASM Features with Entropy and Image Features (202 features):
  logloss = 0.0174

Now when we started testing the Dataset by considering only 20% of features we saw an increment in accuracy and decrease in log loss.

- 20% Best ASM with Entropy and Image Features (402 features):
- logloss = 0.0164

- 30% Best ASM with Entropy and Image Features plus Feature Statistics (623 features):
  multiclass logloss = 0.0133
  accuracy score = 0.9978

Confusion Matrix:

```
[[1540    0    0    0    0    1    0    0    0]
 [   1 2475    2    0    0    0    0    0    0]
 [   0    0 2942    0    0    0    0    0    0]
 [   1    0    0  474    0    0    0    0    0]
 [   2    0    0    0   38    2    0    0    0]
 [   3    0    0    0    0  748    0    0    0]
 [   1    0    0    0    0    0  397    0    0]
 [   0    0    0    0    0    0    0 1225    3]
 [   0    0    0    0    0    0    0    8 1005]]
```

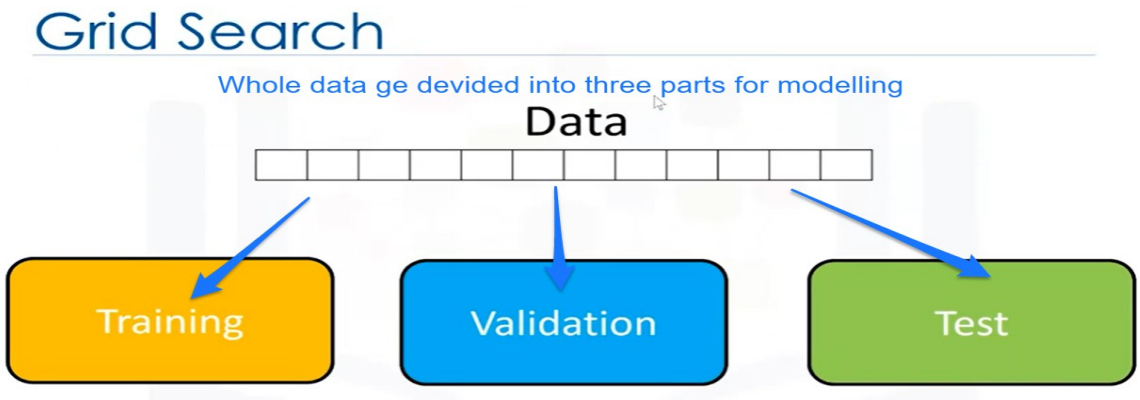
- 40% Best ASM and image features with feature statistics:
  ExtraTreesClassifier with 1000 estimators on 10868 training samples and 823 features
  using 10-fold cross validation:
    multiclass logloss = 0.0135
    accuracy score = 0.9976
    Confustion Matrix:

```
[[1541    0    0    0    0    0    0    0    0]
 [   1 2475    2    0    0    0    0    0    0]
 [   0    0 2942    0    0    0    0    0    0]
 [   1    0    0  474    0    0    0    0    0]
 [   5    0    0    0   37    0    0    0    0]
 [   5    0    0    0    0  746    0    0    0]
 [   1    0    0    0    0    0  397    0    0]
 [   0    0    0    0    0    0    0 1227    1]
 [   0    0    0    0    0    0    0    9 1004]]
```

GRID SEARCH


hyperparamters
Grid Search
Model 1
Model 2


hyperparamters
Grid Search
Model 1


Grid Search
Whole data ge devided into three parts for modelling
Data
Training
Validation
Test

Model Selection

Selection of candidate models using GridSearchCV to find optimal classifier hyper-parameters.

- SVM:
- ExtraTrees:
- XGBoost: 30% Best Features

logloss: 0.0080
accuracy: 0.9981
Confusion Matrix:

```
[[1540    0    0    0    0    1    0    0    0]
 [   2 2475    0    1    0    0    0    0    0]
 [   0    0 2941    0    0    0    1    0    0]
 [   0    0    0  474    0    1    0    0    0]
 [   1    0    0    0   41    0    0    0    0]
 [   4    0    0    0    1  746    0    0    0]
 [   0    0    0    0    0    0  398    0    0]
 [   0    0    0    0    0    0    0 1227    1]
 [   0    0    0    0    0    0    0    8 1005]]
```

- NaiveBayes:
- KNN:

Initial feature engineering consisted of extracting various keyword counts from the ASM files as well as the entropy and file size from the BYTE files of the 10868 malware samples in the training set. Image files of the first 1000 bytes of the ASM and BYTE files were created and combined with keyword and entropy data. This resulted in a set of 2018 features. Flow control graphs and call graphs were generated for each ASM sample. A feature set was then generated from the graphs, including graph maximum delta, density, diameter and function counts etc.

Performance Analysis

Feature Engineering :

Initial feature engineering consisted of extracting various keyword counts from the ASM files as well as the entropy and file size from the BYTE files of the 10868 malware samples in the training set.

Image files of the first 1000 bytes of the ASM and BYTE files were created and combined with keyword and entropy data. This resulted in a set of 2018 features.

Flow control graphs and call graphs were generated for each ASM sample. A feature set was then generated from the graphs, including graph maximum delta, density, diameter and function
counts etc.

Feature Selection :

Statistical analysis of the feature set using chi-squared tests to remove features that are independent of the class labels or have low variance. The BYTE file images were found to be weak
learners and were removed from the feature set. A comparison of the best features from the chi-squared
tests with reduced feature sets of between 10% - 50% of the original features.

In this feature selection we will closely examine each and every feature, how a particular feature going to affect the malware category, in this we will only keep necessary features like features not having variance very close to 0 or very close to 1 and we will apply some pre-defined tests to get correct features for training and testing in order to achieve high accuracy with minimum error.

There might be high possibility of data missing from the dataset, in order to overcome this problem, we have may possibilities like deleting that particular entry or filling the null entries on basis of their frequency or median/mean for categorical or Numerical data respectively.

Selection Comparison

Testing with an ExtraTreesClassifier and 10-fold cross validation produced the following results:

- Original ASM Keyword Counts (1006 features):
  logloss = 0.034

Since when we performed our Testing on Original Data Set Without excluding any features and considering all features equally important we get to know that the accuracy of our result was less as compared to other in this comparison we got log loss of 0.034.

- 10% Best ASM Features with Entropy and Image Features (202 features):
  logloss = 0.0174

Now when we started testing the Dataset by considering only 20% of features we saw an increment in accuracy and decrease in log loss.

- 20% Best ASM with Entropy and Image Features (402 features):
- logloss = 0.0164

- 30% Best ASM with Entropy and Image Features plus Feature Statistics (623 features):
  multiclass logloss = 0.0133
  accuracy score = 0.9978

Confusion Matrix:

```
[[1540    0    0    0    0    1    0    0    0]
 [   1 2475    2    0    0    0    0    0    0]
 [   0    0 2942    0    0    0    0    0    0]
 [   1    0    0  474    0    0    0    0    0]
 [   2    0    0    0   38    2    0    0    0]
 [   3    0    0    0    0  748    0    0    0]
 [   1    0    0    0    0    0  397    0    0]
 [   0    0    0    0    0    0    0 1225    3]
 [   0    0    0    0    0    0    0    8 1005]]
```

- 40% Best ASM and image features with feature statistics:
  ExtraTreesClassifier with 1000 estimators on 10868 training samples and 823 features
  using 10-fold cross validation:
    multiclass logloss = 0.0135
    accuracy score = 0.9976
    Confustion Matrix:
    [[1541 0 0 0 0 0 0 0 0]
    [1 2475 2 0 0 0 0 0 0]
    [0 0 2942 0 0 0 0 0 0]
    [1 0 0 474 0 0 0 0 0]
    [5 0 0 0 37 0 0 0 0]
    [5 0 0 0 0 746 0 0 0]
    [1 0 0 0 0 0 397 0 0]
    [0 0 0 0 0 0 0 1227 1]
                        [0 0 0 0 0 0 0 9 1004]]

```
names = ["Nearest Neighbors",
         #"Linear SVM",
         #"RBF SVM",
         "Decision Tree",
         "Random Forest",
         "AdaBoost",
         "Naive Bayes",
         #"Linear Discriminant Analysis",
         #"Quadratic Discriminant Analysis",
         "Extra Trees"]
classifiers = [
    KNeighborsClassifier(),
    #SVC(kernel="linear", C=0.025, probability=True),
    #SVC(gamma=2, C=1, probability=True),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=1000),
    AdaBoostClassifier(),
    GaussianNB(),
    #LinearDiscriminantAnalysis(),
    #QuadraticDiscriminantAnalysis(),
    ExtraTreesClassifier(n_estimators=1000)]
```

```
def run_cv(X,y, clf):

    # Construct a kfolds object
    kf = KFold(len(y),n_folds=10,shuffle=True)
    y_prob = np.zeros((len(y),9))
    y_pred = np.zeros(len(y))

    # Iterate through folds
    for train_index, test_index in kf:
        #print(train_index.shape)
        X_train = X.loc[train_index,:]
        X_test = X.loc[test_index,:]
        y_train = y[train_index]

        clf.fit(X_train,y_train)
        y_prob[test_index] = clf.predict_proba(X_test)
        y_pred[test_index] = clf.predict(X_test)

    return y_prob, y_pred

# iterate over classifiers
ytrain = np.array(y)
for name, clf in zip(names, classifiers):
  print(name)
  prob, pred = run_cv(X,ytrain,clf)
  print "logloss: %.3f" % log_loss(y, prob)
  cm = confusion_matrix(y, pred)
  print(cm)
#score = clf.score(X_test, y_test)
```

Nearest Neighbors
1.708
[[61 1 3 0 0 1 0 0 0]
 [7 111 0 0 1 2 0 0 0]
 [0 1 147 0 0 0 0 0 0]
 [0 0 0 2 1 0 0 2 1]
 [0 0 0 0 3 0 0 0 0]
 [1 1 3 0 0 18 0 1 4]
 [0 1 2 0 1 1 18 0 0]
 [0 0 1 3 0 1 0 55 4]
 [0 0 2 2 0 0 0 1 36]]

Decision Tree
1.546
[[55 2 0 0 0 5 2 1 1]
 [3 114 0 0 0 0 2 1 1]
 [0 0 141 0 0 1 5 0 1]
 [2 0 0 0 1 1 1 0 1]
 [0 0 0 2 0 1 0 0 0]
 [4 1 0 0 0 19 3 1 0]
 [2 0 0 0 0 0 21 0 0]
 [2 0 1 0 0 4 0 57 0]
 [1 0 0 0 2 3 0 0 35]]

Random Forest
0.297
[[63 0 2 0 0 0 0 0 1]
 [4 115 0 0 0 0 0 2 0]
 [0 0 147 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 0 3 0]
 [3 0 0 0 0 22 1 1 1]
 [0 0 2 0 0 0 20 1 0]
 [2 0 0 0 0 0 0 62 0]
 [0 0 0 0 0 0 0 1 40]]

AdaBoost
2.015
[[0 53 1 0 1 0 1 10 0]
[0 118 3 0 0 0 0 0 0]
[0 11 136 0 0 0 1 0 0]
[0 5 1 0 0 0 0 0 0]
[0 2 1 0 0 0 0 0 0]
[0 25 1 0 0 0 1 1 0]
[0 2 3 0 0 0 17 1 0]
[0 61 0 0 0 0 1 2 0]
[0 36 0 0 0 0 1 2 2]]

Naive Bayes
6.851
[[35 5 1 0 0 14 1 2 8]
[2 66 2 1 0 7 0 0 43]
[0 0 147 0 0 0 0 0 1]
[0 0 0 4 0 1 1 0 0]
[0 0 0 3 0 0 0 0 0]
[1 1 3 0 0 18 0 2 3]
[0 0 0 0 0 0 21 0 2]
[0 0 1 0 0 2 3 57 1]
[0 1 0 0 0 0 0 4 36]]

Extra Trees
0.185
[[64 0 0 0 0 0 0 0 2]
[2 116 0 0 0 0 0 3 0]
[0 0 146 0 0 0 0 2 0]
[0 0 0 4 0 0 0 2 0]
[0 0 0 0 2 0 0 1 0]
[2 0 0 0 0 24 0 1 1]
[0 0 1 0 0 0 20 2 0]
[0 0 1 0 0 0 0 63 0]
[0 0 0 0 0 0 0 0 41]]

The performance of the ExtraTreesClassifier is optimal at around 30% of ASM and image features with highest variance plus sample statistics, entropy and file size. Adding call graph features produced a marginal improvement. It is possible that better classification accuracy would be achieved by using an ensemble of different classifiers with the ASM, image and call graph feature sets as separate inputs to the various classifiers.

Cross-Validation of Model using cross validation Score

```
# Set the parameters by cross-validation
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]


print("# Tuning hyper-parameters for SVC")
print()

clfgrid = GridSearchCV(SVC(C=1), tuned_parameters, cv=10, n_jobs=4)

start = time()
clfgrid.fit(X_train, y_train)

print("Classification report:")
print("GridSearchCV took {:.2f} seconds.".format(time() - start))
print(" ")
y_pred = grid_search.predict(X_test)
print(classification_report(y_test, y_pred))
print(" ")
y_prob = grid_search.predict_proba(X_test)
print("logloss = {:.3f}".format(log_loss(y_test, y_prob)))
print("score = {:.3f}".format(accuracy_score(y_test, y_pred)))
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

Classification report:
GridSearchCV took 2183.53 seconds.

	precision	recall	f1-score	support
1	0.99	1.00	1.00	135
2	1.00	1.00	1.00	251
3	1.00	1.00	1.00	292
4	1.00	1.00	1.00	45
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	84
7	1.00	0.98	0.99	47
8	0.99	1.00	1.00	127
9	1.00	0.99	1.00	104
avg / total	1.00	1.00	1.00	1087

logloss = 0.014
score = 0.998

```
[[135   0   0   0   0   0   0   0   0]
 [  0 251   0   0   0   0   0   0   0]
 [  0   0 292   0   0   0   0   0   0]
 [  0   0   0  45   0   0   0   0   0]
 [  0   0   0   0   2   0   0   0   0]
 [  0   0   0   0   0  84   0   0   0]
 [  1   0   0   0   0   0  46   0   0]
 [  0   0   0   0   0   0   0 127   0]
 [  0   0   0   0   0   0   0   1 103]]
```

Shannon's Entropy by malware class. A score of 0.0 means the bytes are all the same value, a score of 1.0 means every byte in the file has a different value.Shannon's Entropy by file size. A score of 0.0 means the bytes are all the same value, a score of 1.0 means every byte in the file has a different value.Assembler register EDX by ESI counts.

Selecting Model as ExtraTreeClassifier

```
n_jobs=4, criterion='gini')
p1, pred1 = run_cv(X,y,clf1)
print("logloss = {:.3f}".format(log_loss(y, p1)))
print("score = {:.3f}".format(accuracy_score(y, pred1)))
cm = confusion_matrix(y, pred1)
print(cm)
(array([  38,   41,   63, ..., 10845, 10854, 10866]), array([    0,     1,
2, ..., 10864, 10865, 10867]))
(array([   1,    7,   11, ..., 10855, 10857, 10862]), array([    0,     2,     3, ...,
10865, 10866, 10867]))
(array([   8,   13,   18, ..., 10842, 10848, 10860]), array([    0,     1,
2, ..., 10865, 10866, 10867]))
(array([   3,   17,   19, ..., 10851, 10858, 10863]), array([    0,     1,
2, ..., 10865, 10866, 10867]))
(array([   4,   12,   23, ..., 10823, 10827, 10831]), array([    0,     1,
2, ..., 10865, 10866, 10867]))
(array([   2,    6,   32, ..., 10856, 10859, 10864]), array([    0,     1,     3, ...,
10865, 10866, 10867]))
(array([  10,   29,   37, ..., 10840, 10843, 10861]), array([    0,     1,
2, ..., 10865, 10866, 10867]))
(array([  16,   20,   21, ..., 10807, 10819, 10833]), array([    0,     1,
2, ..., 10865, 10866, 10867]))
(array([   0,   28,   39, ..., 10814, 10865, 10867]), array([    1,     2,
3, ..., 10863, 10864, 10866]))
(array([   5,    9,   14, ..., 10820, 10834, 10839]), array([    0,     1,     2, ...,
10865, 10866, 10867]))
logloss = 0.038
score = 0.993
```

```
[[1537    0    0    0    0    0    0    4    0]
 [   5 2469    2    0    0    1    0    1    0]
 [   0    0 2941    0    0    1    0    0    0]
 [   1    0    0  468    0    1    0    5    0]
 [   1    1    0    0   37    0    0    2    1]
 [   3    1    0    0    1  742    0    2    2]
 [   0    0    0    0    0    0  398    0    0]
 [  25    0    3    2    0    3    2 1193    0]
 [   3    0    0    0    0    2    0    2 1006]]
```

```
X = combined_train_data.iloc[:,1:]
ylabels = sorted_train_labels.iloc[:,1:]
y = np.array(ylabels - 1)
y = y.flatten()
y
```

```
array([1, 7, 8, ..., 3, 3, 3], dtype=int64)
```

```
xgclf = xgb.XGBClassifier(objective="multi:softprob", nthread=4)
```

```
params     = {"n_estimators": [1000, 2000],
              "max_depth": [5, 10],
              "learning_rate": [0.1, 0.05]}
```

```
# run grid search
grid_search = GridSearchCV(xgclf, param_grid=params)
start = time()
grid_search.fit(X, y)
```

```
print("GridSearchCV took {:.2f} seconds.".format((time() - start)))
print(" ")
y_pred = grid_search.predict(X)
print(classification_report(y, y_pred))
print(" ")
y_prob = grid_search.predict_proba(X)
```

```
print("logloss = {:.3f}".format(log_loss(y, y_prob)))
print("score = {:.3f}".format(accuracy_score(y, y_pred)))
cm = confusion_matrix(y, y_pred)
print(cm)
```

GridSearchCV took 6196.85 seconds.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1541
1	1.00	1.00	1.00	2478
2	1.00	1.00	1.00	2942
3	1.00	1.00	1.00	475
4	1.00	1.00	1.00	42
5	1.00	1.00	1.00	751
6	1.00	1.00	1.00	398
7	1.00	1.00	1.00	1228
8	1.00	1.00	1.00	1013
avg / total	1.00	1.00	1.00	10868

```
logloss = 0.000
score = 1.000
[[1541    0    0    0    0    0    0    0    0]
 [   0 2478    0    0    0    0    0    0    0]
 [   0    0 2942    0    0    0    0    0    0]
 [   0    0    0  475    0    0    0    0    0]
 [   0    0    0    0   42    0    0    0    0]
 [   0    0    0    0    0  751    0    0    0]
 [   0    0    0    0    0    0  398    0    0]
 [   0    0    0    0    0    0    0 1228    0]
 [   0    0    0    0    0    0    0    0 1013]]
```

Selecting Model as XGBoostClassifier

```
xgclf = xgb.XGBClassifier(n_estimators=1000,
objective="multi:softmax", nthread=4)
prob, pred = run_cv(Xpoly,y,xgclf)
print("logloss: {:.4f}".format(log_loss(y, prob)))
print("accuracy: {:.4f}".format(accuracy_score(y, pred)))
cm = confusion_matrix(y, pred)
print(cm)
```

```
[    0     1     2 ..., 10865 10866 10867] [    8    16    21 ..., 10844 10849
10864]
[    0     1     2 ..., 10864 10865 10866] [   23    25    38 ..., 10847 10857
10867]
[    0     1     2 ..., 10865 10866 10867] [    7     9    10 ..., 10819 10861
10863]
[    0     1     2 ..., 10865 10866 10867] [   29    36    40 ..., 10823 10824
10832]
[    0     1     2 ..., 10865 10866 10867] [    6    14    18 ..., 10808 10836
10860]
[    0     1     2 ..., 10865 10866 10867] [   11    26    32 ..., 10840 10841
10848]
[    0     1     2 ..., 10864 10866 10867] [    4    12    17 ..., 10854 10858
10865]
[    0     2     4 ..., 10865 10866 10867] [    1     3     5 ..., 10853 10855
10859]
[    1     2     3 ..., 10865 10866 10867] [    0    31    54 ..., 10850 10851
10856]
[    0     1     3 ..., 10864 10865 10867] [    2    15    30 ..., 10843 10862
10866]
```

```
logloss: 0.0081
accuracy: 0.9982
```

```
[[1541    0    0    0    0    0    0    0    0]
 [   2 2475    0    0    0    0    0    0    1]
 [   0    0 2941    0    0    0    1    0    0]
 [   0    0    0  474    0    1    0    0    0]
 [   1    0    0    0   41    0    0    0    0]
 [   4    0    0    0    1  746    0    0    0]
 [   0    0    0    0    0    0  398    0    0]
 [   0    0    0    0    0    0    0 1226    2]
 [   0    0    0    0    0    0    0    7 1006]]
```

The Confusion Matrix for ExtraTreeClassifier and XGBoostClassifier

Each Row and Column of the matrix is representing the Category of malware and X-axis represent Original Value and y-axis Represent Predicted Value.

For Comparison let's see the 8th Category of Malware in ExtraTreeClassifier, and XGBoostClassifier. Since for XGBoostClassifier the Model Correctly Predicted 1226 Category as Category-8 and remaining 2 as category-9, while talking about ExtraTreeClassifier it predicted only 1193 as category-8 while wrongly predicted 25 category-1, 3 category-3, 2 category-4, 3 category-6 and 2 category-7 as category-8, which shows XGBoost model is a good classifier for predicting category,

Since we can see accuracy of XGboostClassifier is 99.82% while that of ExtraTreeClassifier is 99.7%.

CONCLUSION

The best accuracy scores were achieved with XGBoost (99.82%) and ExtraTreesClassifier (99.76%) using a feature set of 623 ASM, image and entropy features. Marginal improvements could be achieved using additional features and ensemble methods, however due to the limited sample size further efforts are unlikely to produce significant improvements in prediction accuracy.