Microsoft Malware Prediction

Arunachalam DhakshinaMurthy, *Department of Computer Science, Carleton University, Ottawa, Canada*

***Abstract*— Nowadays malwares have become a major threat to the information technology field. Internet has become the major reason for the rise in the malware threat. Scientists and researchers are spending a lot of time to predict the malwares prone to computer networks. Huge bytes of data is modelled to predict the probability of Windows to be hit by family of malwares based on different properties of machine. Various regression and classification algorithms like Random Forest, K Nearest Neighbor, Logistic Regression, Naive Bayes, Decision Trees, Dummy classifiers, Bagging and Boosting classifiers are tested. Performance metrics like accuracy, precision, AUC values, recall is compared between the models is compared and the best model for this dataset is identified.**

Keywords—Machine learning algorithms, Accuracy, Precision, Recall, F-1 Score, AUC score and Confusion Matrix.

I. INTRODUCTION

Malware

Malware is the overarching name for applications and other code, i.e. software that Microsoft classifies more granularly as malicious software or unwanted software. Malicious software is an application or code that compromises user security. It might steal anybody's personal information, lock the PC, spam the computer, sometimes even send or download other malicious software. In other words, it cheats, tricks or defrauds users, place users in vulnerable threats or perform other malicious activities.

Microsoft classifies most malicious software into one of the following categories:

Backdoor: A type of malware that gives malicious hackers remote access to and control of your PC.

Downloader: A type of malware that downloads other malware onto your PC. It needs to connect to the internet to download files.

Dropper: A type of malware that installs other malware files onto your PC. Unlike a downloader, a dropper doesn't need to connect to the internet to drop malicious files. The dropped files are typically embedded in the dropper itself.

Exploit: A piece of code that uses software vulnerabilities to gain access to your PC and perform other tasks, such as installing malware.

Hack tool: A type of tool that can be used to gain unauthorized access to your PC.

Macro virus: A type of malware that spreads through infected documents, such as Microsoft Word or Excel documents. The virus is run when you open an infected document.

Obfuscator: A type of malware that hides its code and purpose, making it more difficult for security software to detect or remove.

Password stealer: A type of malware that gathers your personal information, such as user names and passwords. It often works along with a key logger, which collects and sends information about the keys you press and websites you visit.

Rogue security software: Malware that pretends to be security software but doesn't provide any protection. This type of malware usually displays alerts about non-existent threats on your PC. It also tries to convince you to pay for its services.

Trojan: A type of malware that attempts to appear harmless. Unlike a virus or a worm, a Trojan doesn't spread by itself. Instead it tries to look legitimate, tricking users into downloading and installing it. Once installed, Trojans perform a variety of malicious activities, such as stealing personal information, downloading other malware, or giving attacker's access to your PC.

Trojan clicker: A type of Trojan that automatically clicks buttons or similar controls on websites or applications. Attackers can use this Trojan to click on online advertisements. These clicks can skew online polls or other tracking systems and can even install applications on your PC.

Worm: A type of malware that spreads to other PCs. Worms can spread through email, instant messaging, file sharing platforms, social networks, network shares, and removable drives. Sophisticated worms take advantage of software vulnerabilities to propagate.

II. CASE STUDY

To perform any machine learning modelling, certain steps have to be followed in order to classify or predict any data set with better accuracy.

A.) Data Collection:

This involves collection of data from various sources and combining them into a dataset. The dataset can be structured or unstructured in many cases. Microsoft Malware Prediction dataset is a structured dataset which is a combination of categorical, binary and numerical. The dataset can be imported using python library **'pd.read_csv'.** This reads the data into the python notebook and prepares the data for preprocessing

B.) Data Preparation:

For any dataset to perform well it has to be cleaned. Many features of the dataset may be improper. It can be checked for null values, statistical distribution of data, different data types present in the dataset. The dataset used in the project requires removing null values, mode imputation and categorizing columns into numerical, binary and categorical variables to perform feature extraction and feature selection

C.) Feature Engineering:

Feature engineering is a combination of feature selection and feature extraction techniques. Data normalization is a primary concern for any dataset. All the variables cannot be similar to perform training. Normalization of data makes sure that it performs equally when the dataset is trained for modelling. Encoding techniques are applied to the dataset which normalizes the values in the data. Frequency encoding is often performed for categorical dataset. One Hot encoding techniques are followed for numerical dataset. Since, malware prediction dataset is highly categorical, it is better to perform frequency encoding.

Feature selection techniques are performed to select the best features amongst the dataset to obtain better accuracy and results. Feature scores are obtained based on which feature selection is carried out.

D.) Training models:

Basically supervised modelling or unsupervised modelling is performed on the feature selected model. Supervised algorithms are used for Malware prediction dataset to obtain better accuracy compared to unsupervised techniques since the dataset is structured. The data is trained with various supervised algorithms and the best performing algorithm is chosen.

E.) Testing models:

Testing dataset is used to check whether the trained dataset performs well, even when a new similar dataset is provided. It checks whether the modelling performs similar on real time scenarios.

III. DATASET

The malware dataset is almost half a terra byte and consist of malware files of 9 different families. Each malware type has an identifier, 20 character hash value- uniquely identifying files and a class label- integer representing one of the 9 family names of malwares. For each data, the hexadecimal representation of the file's binary content without the header. The dataset also includes metadata manifest which is a log containing various metadata information extracted from the binary such as function calls, strings, etc.

The dataset consists of more than eighty features (83) that are gathered by combining heartbeats and threat reports by Windows Defender. Each row in this dataset corresponds to a machine, uniquely identified by a **Machine Identifier. Has Detections** is the target label and it indicates that Malware was detected on the machine. Each one of the 83 features plays a major role in prediction of malware. The public repository URL for this dataset is https://www.kaggle.com/c/microsoft-malware-prediction/data.

FAMILY NAME	**TYPE**	**NO OF SAMPLES**
Ramnit	Worm	1541
Lollipop	Adware	2478
Kelihos_Ver3	Backdoor	2942
Vundo	Trojan	475
Simda	Backdoor	42
Tracur	Trojan Downloader	751
Kelihos_Ver1	Backdoor	398
Obfuscator.ACY	Any kind of obfuscated malware	1228
Gatak	Backdoor	1013

Fig. 1. Table representing types of malwares

Using the information and labels in **train.csv**, one must predict the value for **HasDetections** for each machine in **test.csv**. Malware detection is typically a time-series problem, but it is made complicated by the introduction of new machines, machines that come online and offline, machines that receive patches, machines that receive new operating systems, etc. Additionally, this dataset is not representative of Microsoft customers' machines in the wild; it has been sampled to include a much larger proportion of malware machines.

IV. DATA PREPROCESSING AND EXPLORATORY DATA ANALYSIS

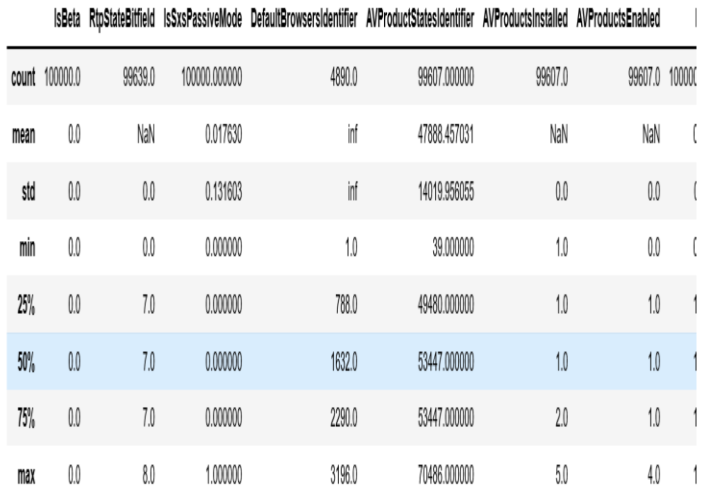
It involves finding out inaccurate, incomplete and irrelevant information in the data which could cause hindrance to accuracy of the model. This can be modified by replacing null values, finding out the most desirable column and replacing null values with mean, median or mode.

A.) Dataset information:

This dataset given is a sparse data which has all three types of columns- **categorical, numerical and binary**. The first step of pre-processing is to get the various types of data using **'.info()'** function. The data types consist of category (30), float16 (27), float32 (9), int16 (9), int32 (1), int8 (11). It is clearly seen that almost half of the data types in the data set is categorical. Handling categorical data is quite complicated as it introduces to encoding and normalization of data.

B.) Description of dataset:

Describe() method can be applied to the dataset to obtain various statistical values of data like mean, count, standard deviation and minimum and maximum values. This gives a basic idea of how a data set is distributed in a given range of values .i.e. nrows=100000. The dataset appears to be wide and has a considerable amount of **NaN** values (null values). Most of the feature contains a lot of null values and the basic idea is to ditch the features that have null values as they do not provide any information to the dataset.
The most important portion of data cleaning involves finding the unique values, percentage of null values and percentage of values in the biggest category in all the features. The feature set is sorted in the decreasing order of percentage of null values. It can be observed that the feature set **'PuaMode'** has the largest percentage of null values and apart from '**HasDetection'** which is the target set '**Census_OS_Architecture**'has least number of null values.



	IsBeta	RtpStateBitfield	IsSxsPassiveMode	DefaultBrowsersIdentifier	AVProductStatesIdentifier	AVProductsInstalled	AVProductsEnabled
count	100000.0	99639.0	100000.000000	4890.0	99607.000000	99607.0	99607.0
mean	0.0	NaN	0.017630	inf	47888.457031	NaN	NaN
std	0.0	0.0	0.131603	inf	14019.956055	0.0	0.0
min	0.0	0.0	0.000000	1.0	39.000000	1.0	0.0
25%	0.0	7.0	0.000000	788.0	49480.000000	1.0	1.0
50%	0.0	7.0	0.000000	1632.0	53447.000000	1.0	1.0
75%	0.0	7.0	0.000000	2290.0	53447.000000	2.0	1.0
max	0.0	8.0	1.000000	3196.0	70486.000000	5.0	4.0

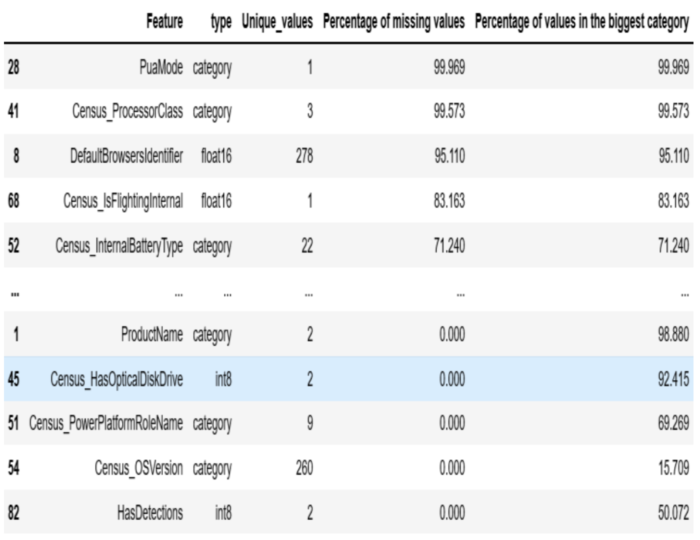
Fig. 2. Statistical Description of Dataset

C.) Data Cleaning:

Cleaning of data usually involves dealing with NaN values or null values. It is usually addressed by dropping the missing values or imputation of missing values.
Data cleaning for this type of dataset involves removing of columns having high probability of null values. A threshold value of **0.9** is considered to omit columns with high null values and unbalanced feature rate of **0.9** is considered to remove columns with unbalanced features. Missing values of columns with 70 percent threshold rate and 90 percent of features having one category is removed. Below listed are considered as **good columns**.

['MachineIdentifier', 'EngineVersion', 'AppVersion', 'AvSigVersion', 'AVProductStatesIdentifier', 'AVProductsInstalled', 'CountryIdentifier', 'CityIdentifier', 'OrganizationIdentifier', 'GeoNameIdentifier', 'LocaleEnglishNameIdentifier', 'OsBuild', 'OsSuite', 'OsPlatformSubRelease', 'OsBuildLab', 'SkuEdition', 'IeVerIdentifier', 'SmartScreen', 'Census_MDC2FormFactor', 'Census_OEMNameIdentifier', 'Census_OEMModelIdentifier', 'Census_ProcessorCoreCount', 'Census_ProcessorManufacturerIdentifier', 'Census_ProcessorModelIdentifier', 'Census_PrimaryDiskTotalCapacity', 'Census_PrimaryDiskTypeName', 'Census_SystemVolumeTotalCapacity', 'Census_TotalPhysicalRAM', 'Census_ChassisTypeName', 'Census_InternalPrimaryDiagonalDisplaySizeInInches', 'Census_InternalPrimaryDisplayResolutionHorizontal', 'Census_InternalPrimaryDisplayResolutionVertical', 'Census_PowerPlatformRoleName', 'Census_InternalBatteryType', 'Census_InternalBatteryNumberOfCharges', 'Census_OSVersion', 'Census_OSBranch', 'Census_OSBuildNumber', 'Census_OSBuildRevision', 'Census_OSEdition', 'Census_OSSkuName', 'Census_OSInstallTypeName', 'Census_OSInstallLanguageIdentifier', 'Census_OSUILocaleIdentifier', 'Census_OSWUAutoUpdateOptionsName', 'Census_GenuineStateName', 'Census_ActivationChannel', 'Census_IsFlightingInternal', 'Census_ThresholdOptIn', 'Census_FirmwareManufacturerIdentifier', 'Census_FirmwareVersionIdentifier', 'Census_IsSecureBootEnabled', 'Census_IsWIMBootEnabled', 'Census_IsTouchEnabled', 'Wdft_IsGamer', 'Wdft_RegionIdentifier', 'HasDetections']

The total number of features has been removed or reduced from 82 to 57 in total. These 57 features are considered as good columns which has the least probability of missing values.



	Feature	type	Unique_values	Percentage of missing values	Percentage of values in the biggest category
28	PuaMode	category	1	99.969	99.969
41	Census_ProcessorClass	category	3	99.573	99.573
8	DefaultBrowsersIdentifier	float16	278	95.110	95.110
68	Census_IsFlightingInternal	float16	1	83.163	83.163
52	Census_InternalBatteryType	category	22	71.240	71.240
...	...	...	...	...	...
1	ProductName	category	2	0.000	98.880
45	Census_HasOpticalDiskDrive	int8	2	0.000	92.415
51	Census_PowerPlatformRoleName	category	9	0.000	69.269
54	Census_OSVersion	category	260	0.000	15.709
82	HasDetections	int8	2	0.000	50.072

Fig. 3. Percentage of NULL values

D.) Class imbalance:

To check for class imbalance, the total number of target variable **HasDetections** is obtained. Since it is binary type of variable, it can have only two values. The target variable seem to be balanced because it has almost equal number of majority and minority class variables.

Majority class variable count: 50072
Minority class variable count: 49928

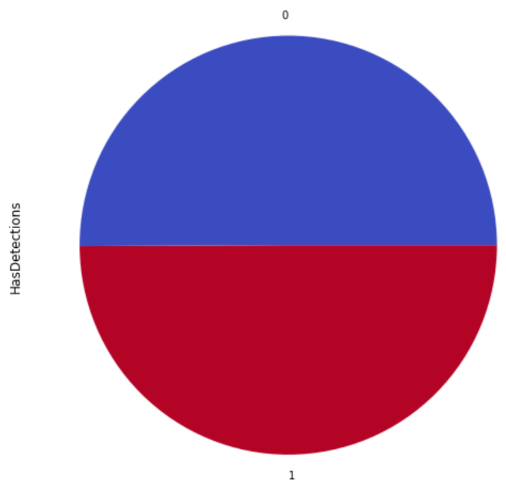


Fig. 4. Class Imbalance-Pie chart

E.) Separation of numerical and categorical data:

The dataset consist of numerical, binary and categorical columns and the main objective is to separate them because categorical variables has to be handled differently compared to binary and numerical columns. Initially separation of numeric data and categorical data is performed based on the data types. Numerical data also contains binary values as a part of them. Then, binary values are separated from numerical columns by finding columns with only 2 unique elements.
MachineIdentifier is a unique identifier and hence can be removed from the categorical columns.
It is observed that numerical features appears to be more than categorical features as binary features have the least count.

Length of Categorical Columns: 20
Length of Numerical Columns: 32
Length of Binary Columns: 4

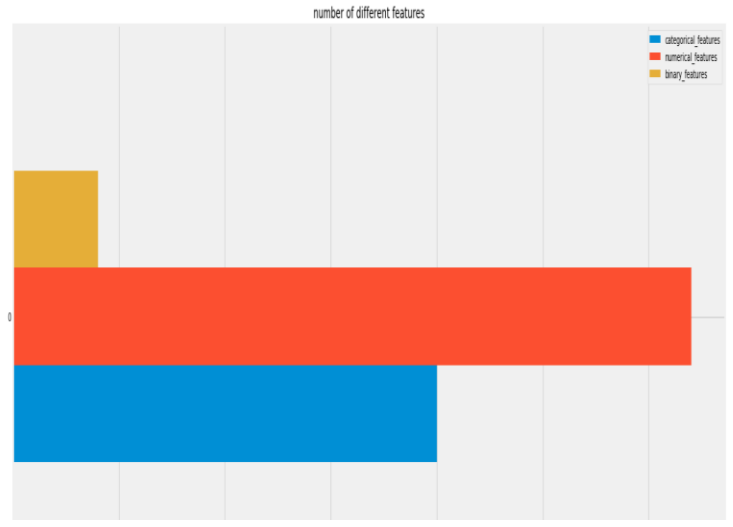


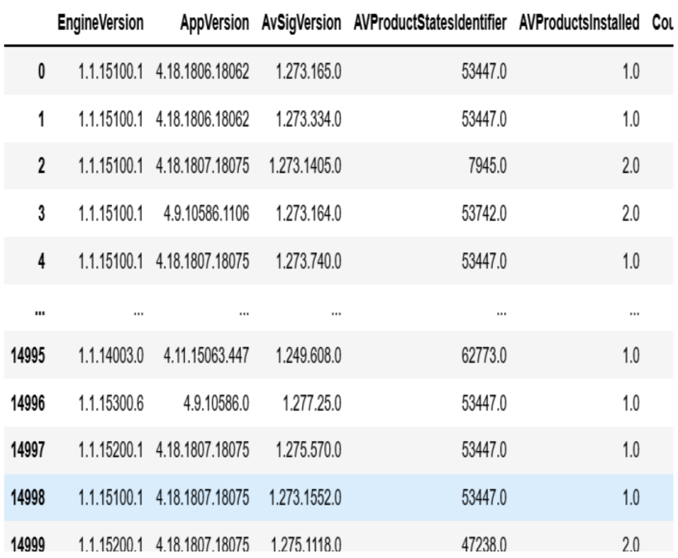
Fig. 5. Plot of different types of variables

F.) Filling Null values with Statistical Mode:

Dropping missing values: It involves dropping columns having highest percentage of missing values.
Imputing missing values: This involves replacing missing values with statistical values like mean, median or mode.

- Mean Imputation: This involves replacing the null values with the mean of the non-missing values in case of numerical feature.
- Median Imputation: This involves replacing the missing values with the median of non-missing values in case of numerical features
- Mode Imputation: This involves replacing the null values with the mode of the non-missing values for both categorical and numerical features.

For Malware prediction dataset, mode imputation is applied because there are plenty of categorical features in the dataset and hence performing mean imputation and median imputation would be ideally wrong.
The mode imputation shall be performed for both the training and testing dataset.



	EngineVersion	AppVersion	AvSigVersion	AVProductStatesIdentifier	AVProductsInstalled	Cou
0	1.1.15100.1	4.18.1806.18062	1.273.165.0	53447.0	1.0	
1	1.1.15100.1	4.18.1806.18062	1.273.334.0	53447.0	1.0	
2	1.1.15100.1	4.18.1807.18075	1.273.1405.0	7945.0	2.0	
3	1.1.15100.1	4.9.10586.1106	1.273.164.0	53742.0	2.0	
4	1.1.15100.1	4.18.1807.18075	1.273.740.0	53447.0	1.0	
...	...	...	...	...	...	
14995	1.1.14003.0	4.11.15063.447	1.249.608.0	62773.0	1.0	
14996	1.1.15300.6	4.9.10586.0	1.277.25.0	53447.0	1.0	
14997	1.1.15200.1	4.18.1807.18075	1.275.570.0	53447.0	1.0	
14998	1.1.15100.1	4.18.1807.18075	1.273.1552.0	53447.0	1.0	
14999	1.1.15200.1	4.18.1807.18075	1.275.1118.0	47238.0	2.0	

Fig. 6. Mode Imputed Dataset

Now, the dataset is perfectly cleaned to start with feature engineering which involves feature extraction, feature selection , finding the top most features, etc.,

V. FEATURE ENGINEERING

Feature engineering is the use of knowledge from the dataset to create features and apply machine learning models. It typically involves extracting of features from a cleaned dataset or raw dataset. Applying feature engineering before modelling can improve the overall performance of the dataset.
The dataset has a large number of categorical and numerical columns. To perform modelling without normalizing all the categorical features may lead to bad results. Categorical data usually has different dataset which are mostly unique and hence normalization is necessary to obtain desired results.

A.) Normalization of Categorical features:

It is observed that most of the categorical features have high cardinality i.e. high number of unique values. Since the dataset is huge, one hot encoding is not desirable as it may create large number of dummy variables and thereby degrade the time to compute the result.

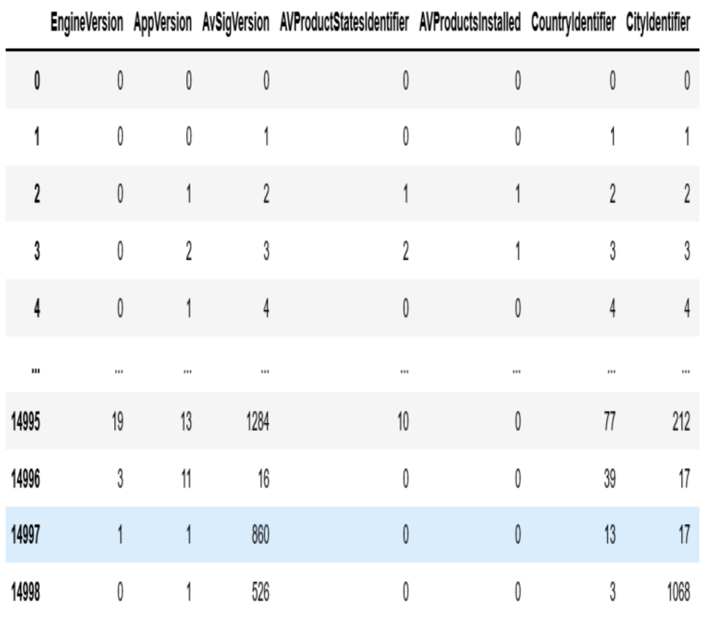
Frequency encoding is desirable for categorical variables as it replaces the variables with the frequency of occurrence throughout the training data.

During frequency encoding of categorical columns, the function checks for the occurrence of each value and replace them with the count number.

Frequency encoded variables having high cardinality:

Census_OEMModelIdentifier
CityIdentifier
Census_FirmwareVersionIdentifier
Census_ProcessorModelIdentifier
Census_OEMNameIdentifier
AvSigVersion

With the help of TQDM, frequency encoded variables are pipelined which can make its execution time faster.



	EngineVersion	AppVersion	AvSigVersion	AVProductStatesIdentifier	AVProductsInstalled	CountryIdentifier	CityIdentifier
0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
2	0	1	2	1	1	2	2
3	0	2	3	2	1	3	3
4	0	1	4	0	0	4	4
...	...	...	...	...	...	...	...
14995	19	13	1284	10	0	77	212
14996	3	11	16	0	0	39	17
14997	1	1	860	0	0	13	17
14998	0	1	526	0	0	3	1068

Fig. 7. Frequency Encoded Dataset

B.) Feature Selection:

Feature selection is the most sought method in machine learning modelling to select the K best features which can perform well in the model. It sorts all the feature in the order of ‘most important’ based on their scores.
Two methods of feature selection is applied on the dataset to check which one performs better and the best one is taken for the training.

C.) Extra Trees Classifier:

Extra Trees Classifier is a kind of randomized tree model which can be used to reduce over-fitting of the model.
This type of feature selection method is a part of ensemble library which uses a number of decision trees to improve the predictive accuracy on the dataset in particular. These are the feature scores of all the training dataset.

```
[0.01085481 0.01615204 0.01322754 0.00953613
0.00935683 0.0142741 0.01504324 0.0095153 0.01218211
0.01426732 0.01954508 0.01548321 0.03747182 0.21632776
0.01033438 0.03390342 0.00643279 0.01733835 0.02375248
0.01651002 0.0245505 0.00590096 0.03714041 0.02955643
0.03322362 0.03091839 0.05097411 0.01182731 0.02925145
```

0.03463765 0.01639889 0.04173036 0.00538653 0.01380157 0.01714857 0.0508926 0.01838416 0.01521488 0.01155288].

The top thirty features are shown below,

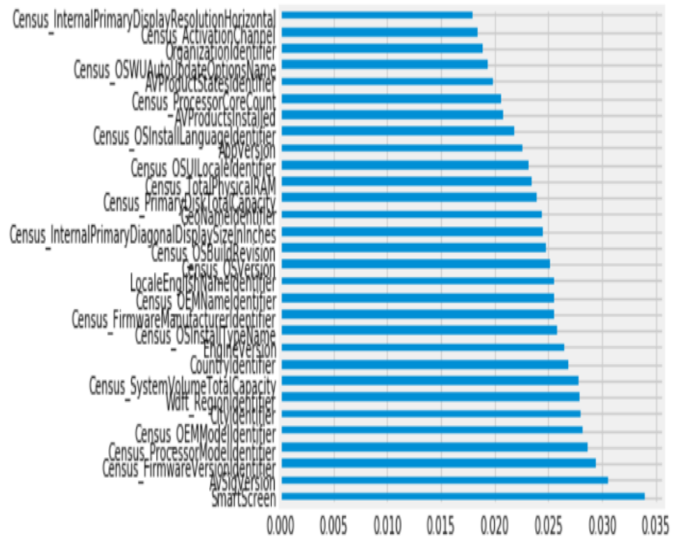


Fig. 8. Top 30 features of dataset

Features	Feature Scores
SmartScreen	0.036957
AvSigVersion	0.032328
Census_FirmwareVersionIdentifier	0.029729
Census_ProcessorModelIdentifier	0.028523
Census_SystemVolumeTotalCapacity	0.028248
Census_OEMModelIdentifier	0.028131
EngineVersion	0.026989
CityIdentifier	0.026439
GeoNameIdentifier	0.026281
CountryIdentifier	0.026221
Wdft_RegionIdentifier	0.025679
Census_OEMNameIdentifier	0.025618
Census_OSVersion	0.025323
Census_OSInstallTypeName	0.025201
Census_TotalPhysicalRAM	0.024808
Census_FirmwareManufacturerIdentifier	0.024561
Census_OSBuildRevision	0.023927
Census_PrimaryDiskTotalCapacity	0.023877
Census_InternalPrimaryDiagonalDisplaySizeInInches	0.023578
LocaleEnglishNameIdentifier	0.023529

Fig. 9. Table representing top 20 features of dataset

D.) Principal Component Analysis:

PCA is a dimensionality reduction technique which uses mathematical concepts like linear algebra with matrix operations to project the original data into scores having matrix with reduced dimensions.

It is necessary to encode the dataset after frequency encoding as it doesn't require all the columns to undergo modelling. It is necessary to select the topmost best columns which is supposedly expected to perform well based on the PCA scores provided by principal component analysis technique. It deals with the variance-covariance of the training dataset.

The explained variance ratio of the PCA model is provided below,

[9.99999915e-01 7.72483193e-08 8.09935061e-09] [[-9.22810717e-11 -1.63962293e-10 -1.24694545e-08 -9.51875399e-10 -1.94184636e-11 -8.69276440e-10 1.95896390e-09 2.28515069e-12 -3.35954370e-10 -8.42434512e-11 -1.90199943e-10 7.62507990e-11 -1.93259890e-10 -1.31236550e-09 7.50999221e-11 -5.26425403e-10 4.88151222e-12 4.41642109e-10 1.46493595e-09 -2.84473922e-08 1.32223418e-10 1.33148983e-11 2.31913413e-08 8.38324956e-06 8.69769645e-11 -1.28914824e-07 4.89385392e-07 4.98955860e-10 1.64476419e-09 3.40207923e-08 2.75115080e-08 4.27057608e-10 3.09206752e-11 1.00000000e+00 -1.74335671e-09 -2.40334724e-10 -1.53125406e-10 -1.63144409e-09 1.94115535e-10 1.91587679e-10 -4.34824613e-11 2.00047034e-10 1.82103070e-10 2.74093898e-11 1.88744903e-11 9.43871460e-11 -0.00000000e+00 -6.22930748e-14 -1.94298257e-10 1.23552894e-08 -8.19870965e-11 -0.00000000e+00 -2.62106329e-11 2.49352071e-11 4.15445459e-11]
The fitted components of the model,
[4.68775512e-08 -1.93162296e-07 -2.60453516e-06 7.63011144e-06 1.19100066e-07 1.51332478e-06 -1.85862380e-06 -2.50542830e-08 1.61875037e-06 5.06385188e-07 -1.79658736e-07 -1.32444254e-07 -1.63533450e-07 -3.97835618e-06 -1.58680168e-07 -7.27250354e-07 6.67484476e-10 -1.65154916e-07 -5.77796726e-06 -8.58180611e-06 3.42540837e-07 5.08485414e-08 -1.50010788e-05 7.32906730e-01 -4.90798294e-07 6.80327350e-01 1.55624880e-03 -2.77606863e-07 9.89454160e-07 1.50147879e-05 -1.62558047e-05 -7.25826182e-08 -2.21984788e-08 -6.05719729e-06 -1.52566742e-06 -2.41734998e-07 -1.73359711e-07 -1.22111668e-06 -3.45493034e-07 -3.32553488e-07 -1.12143935e-07 -7.25651665e-07 -7.56089216e-07 -3.69090183e-08 -4.77572483e-08 -1.24205407e-07 -0.00000000e+00 4.91411500e-11 -3.54597798e-07 -2.47269557e-06 1.81520975e-07 -0.00000000e+00 -5.63381311e-08 8.24398773e-09 3.94033376e-08].

After feature engineering is performed, it is based on the use to select the K best features required for the modelling of dataset. For this dataset, topmost 20 features have been selected to perform training.

VI. MACHINE LEARNING PROCESS

The training dataset which was a raw dataset with several categories columns, null values, etc, is fit for applying machine learning models after excessive amount of information was extracted from the dataset during feature engineering. In general six types of machine learning models have been applied to the training dataset and the performance of the models are compared based on their accuracy. The motive of comparing all the algorithms is that how different types of models- Naïve Bayes, K-Nearest Neighbor, Decision Tree, Rule based Learning model(Dummy classifier),Ensemble model(Random Forests) perform for the dataset. Different types of algorithms have different approaches towards the algorithm and hence it will provide insight on how each model performs for the particular dataset.
All the necessary libraries are imported from sckit-learn and the train-test split is performed on the dataset to split them into 0.7 percent of training size and 0.3 percent of testing size.

A.)Random Forest Algorithm:

The goal is to create a model that predicts the value of a target variable by learning multiple decision rules for the malware prediction dataset. Random Forest itself is an improved version of decision tree model as it consist many decision trees to build a model with 'feature randomness' and 'bagging'.
Parameters like 'n_estimators', 'min_sample_leaf', 'max_features', 'oob_score' and 'n_jobs' are assigned for the algorithm to perform better.
The performance of this model is measured using metrics like accuracy score, precision score, recall score and F-1 score.
The classification report stating all the performance metrics is obtained for this model.
The accuracy of this model is found out to be 61.25%.

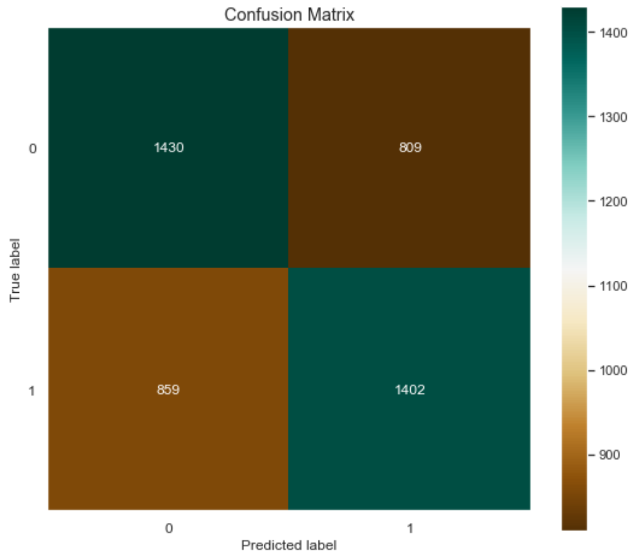


Fig. 10. Confusion Matrix-Random Forest

B.) Decision Trees Algorithm:

The goal is to create a model that predicts the value of a target variable by learning simple decision rules for the malware prediction dataset.
Parameters like 'n_samples and 'n_features' can be assigned to improve the accuracy of the model
The performance of this model is measured using metrics like accuracy score, precision score, recall score and F-1 score.
The classification report stating all the performance metrics is obtained for this model. The confusion matrix is plotted using matplot.
The accuracy of this model is found out to be 54.68%.

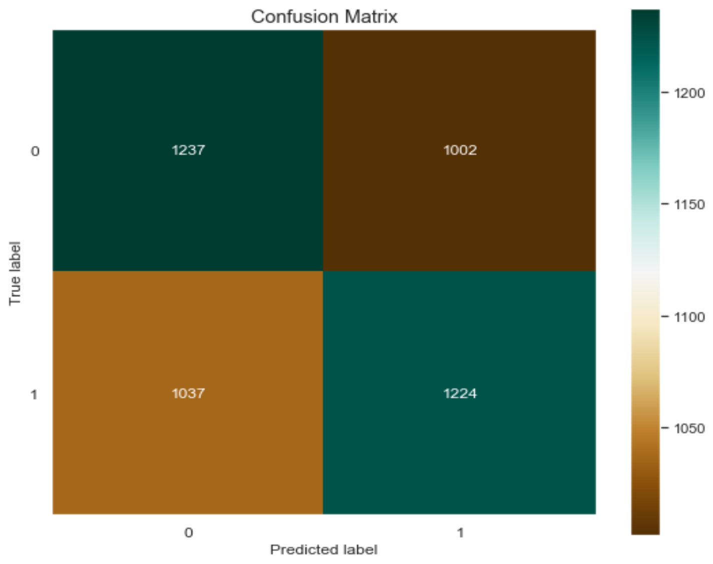


Fig.11.Confusion Matrix – Decision Trees

C.) Naive Bayes Algorithm:

The goal is to create a model that predicts the value of a target variable by learning simple Naïve Bayes algorithm for the malware prediction dataset.

Gaussian Naïve Bayes algorithm is a simple supervised algorithm and a probabilistic classifier that uses Bayesian network for modelling. Similar to all models, the classification report which has precision, recall, f1 score is calculated and the confusion matrix is plotted using matplot function. Further, the AUC score for the algorithm is also obtained.

The accuracy of the model was found to be 54.60%.

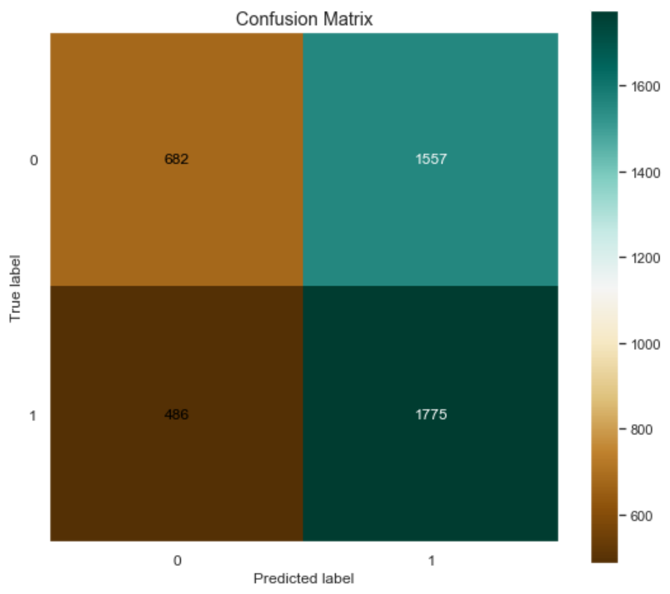


Fig. 12. Confusion Matrix-Naïve Bayes

D.) K-Nearest Neighbors:

It is a simple supervised algorithm which can be used to classify or predict algorithms based on K-neighbors. It can be observed from the dataset that value of K is proportional to the accuracy of the data. As the K value was increased from 3 to 9, the accuracy also increased considerably. Similar to all models, the classification report which has precision, recall, f1 score is calculated and the confusion matrix is plotted using matplot function. AUC score is calculated using sklearn.metrics function.

The accuracy for KNN=3 is 53.46%
The accuracy for KNN=5 is 52.89%
The accuracy for KNN=7 is 53.90%
The accuracy for KNN=9 is 53.92%.

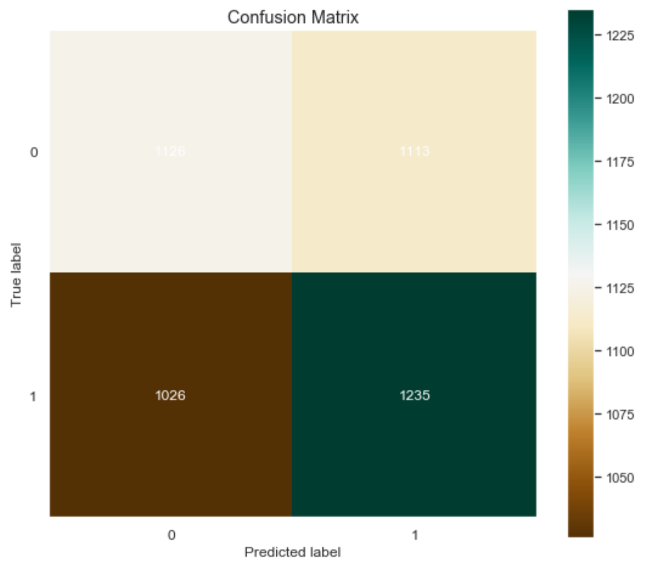


Fig. 13. Confusion Matrix-KNN

E.)Logistic Regression:

It is linear based model which 'OVR' one versus rest type of algorithm to model the malware prediction dataset. Since logistic regression is a linear model, it uses linear function to make predictions on the input features.

Parameters like solver ('lbfgs') and multi class('ovr') are used. Similar to all models, the classification report which has precision, recall, f1 score is calculated and the confusion matrix is plotted using matplot function. The AUC score which gives out whether the predictions are right or wrong is obtained for logistic regression model.

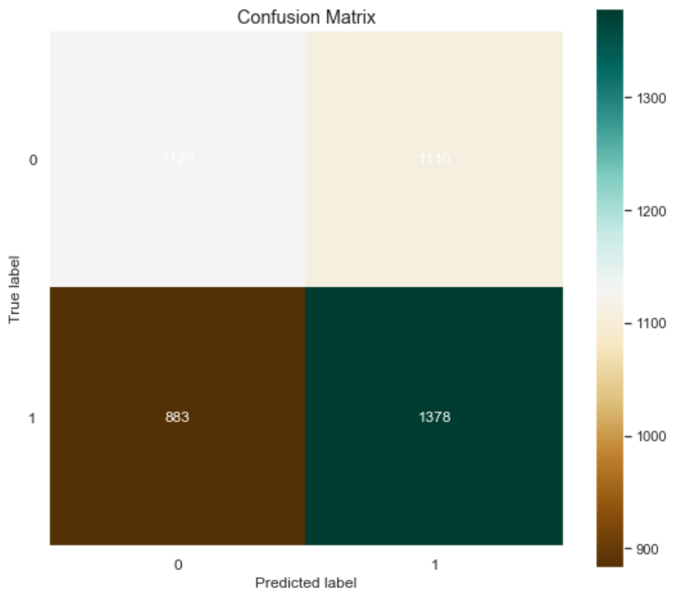


Fig. 14. Confusion Matrix-Logistic Regression

The accuracy of Logistic Regression is 55.72%.

F.)Dummy Classifiers:

This is a simple rule based algorithm used for predicting the dataset based on simple rules. The parametric strategy of 'most frequent' is used in the dataset to get all the most frequently classified labels. The random state and constant have the default values as none.

The accuracy of Dummy Classifier is found to be 55.72%
All the performance metrics like precision, recall, f-1 score and AUC score are calculated.

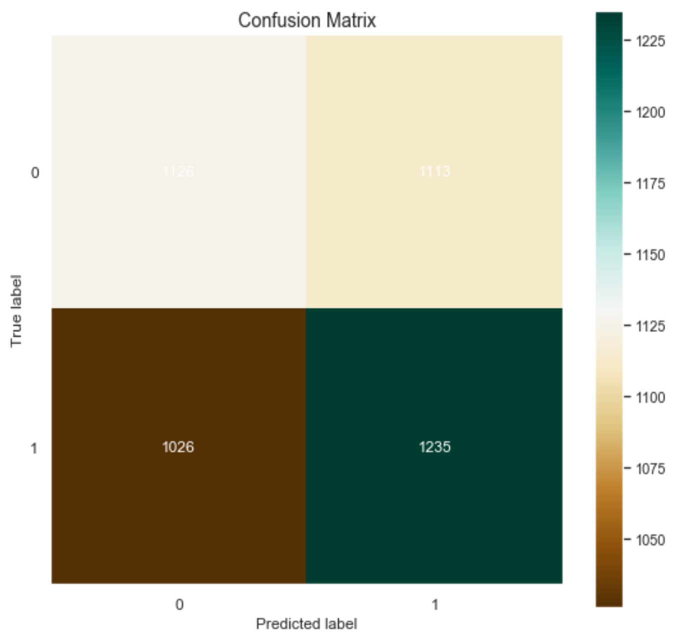


Fig. 15. Confusion Matrix- Dummy Classifier

G.)Algorithms for Boosting:

In general, these algorithms are typically used in supervised techniques to reduce bias reduction to improve the accuracy of the model.

I) XG-Boost:

This type of supervised gradient boosting algorithm is used to speed up the performance of the model. In case of malware predictions, these types of algorithms are necessary because they tend to improve the speed with which malwares are being predicted. This will surely improve the user experience and feedback.

Parameters chosen for this dataset is provided below,
Learning rate = 0.03
N_Estimators = 3000
Max_Depth = 11
Minimum Child Weight = 9
Gamma = 0.2
Sub_sample = 1
Co_sample_bytree = 0.4
Objective = 'binary logistic'
N thread = -1
Scale_pos_weight = 1
Reg_alpha = 0.6
Reg_lamda = 3
Seed = 42
The model accuracy is calculated as 62.52%.

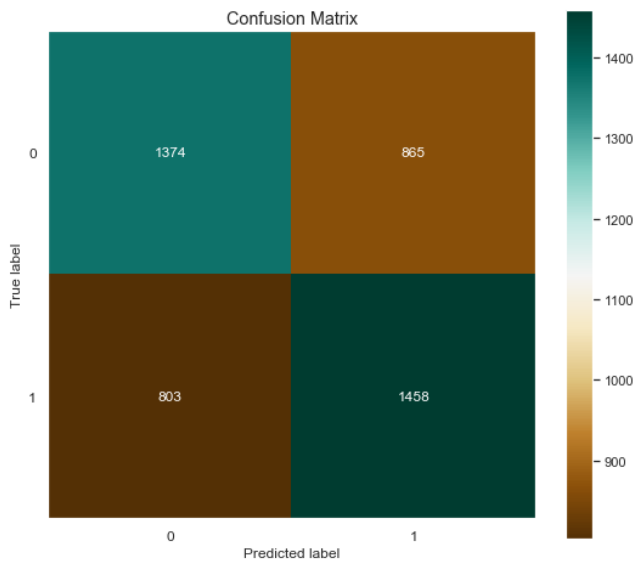


Fig. 16. Confusion Matrix- XG Boost

All the performance metrics is calculated using classification report.

II) AdaBoosting:

This type of algorithm is used to model accuracy based adaptive boosting. It uses decision trees as weak learners to improve over fitting of the model.
Similar to all other algorithms, the performance metrics are calculated.

The accuracy is found out to be 62.51%.

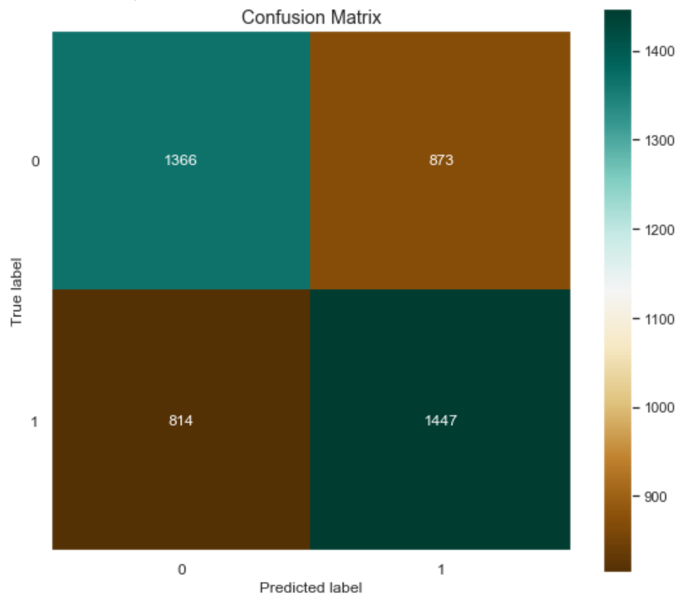


Fig. 17. Confusion Matrix-AdaBoost

H.) Bagging Classifier with Random Forest:

The bagging classifier is a variance reduction technique used to improve the accuracy of the model. The stability of the model is also altered.

The parameters considered for bagging classifiers are base estimators which is the simple learning model (Random Forest), n_estimators (10) and random state (0).
For bagging classifier with random forest, the accuracy is found out to be 98.15%.

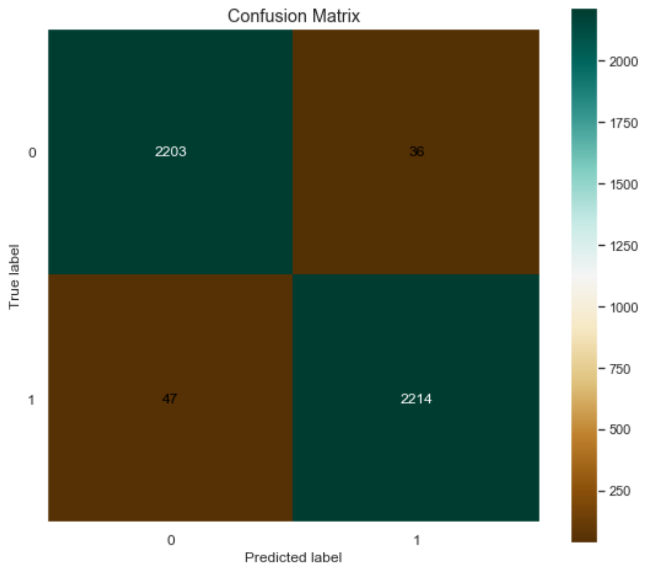


Fig. 18 Confusion Matrix-Bagging

I.) Statistical Significance Test:

Paired T test is chosen as the statistical comparison tool to compare two algorithmic models. It can be observed there is statistical significance between Random Forest and KNN. They have the same distribution. They satisfy the null hypothesis condition. It is also the same with Logistic Regression and Decision Trees.

The statistical value and p value between Random Forest and KNN is found to be 0.559 and 0.590 respectively and for Logistic Regression and Decision Trees it is found to be 0.546 and 0.598.

VII. COMPARISON OF MODELS

Various performance measures have been measured. Since the dataset is based on malware prediction it is better to compare the accuracy of the models rather than comparing precision, recall and the scores of the model. Malwares have to be predicted accurately so that next time when there is any attack, it should not misclassify and affect the functioning of the systems. For this kind of dataset, the exactness and completeness of the model is least considered compared to the true positives and true negatives.

MODELS	ACCURACY
RANDOM FOREST	61.25
K-NEAREST NEIGHBOUR	52.89
LOGISTIC REGRESSION	54.14
DUMMY CLASSIFIER	54.13
DECISION TREE	54.37
NAÏVE BAYES	54.44
XG-BOOST	61.43
BAGGING ALGORITHM	98.48
ADA BOOSTING	59.99

Fig. 19 Table representing accuracy of models

The accuracy of the models is compared and it can be found out among simple learners i.e. Naïve Bayes, KNN, Decision Trees, Logistic Regression and Dummy Classifier, Bayesian algorithm performs better. Ensemble techniques like Random Forest improves the accuracy with randomization of trees. The main motive of using ensemble techniques is to improve the accuracy of the dataset and it is achieved by introducing Random Forest algorithms.

Additional boosting and bagging techniques were used to improve the accuracy of the model. XG Boost and AdaBoost were compared with each other and it was found out that, gradient boosting technique performed better than the adaptive boosting algorithm. These algorithms can also be considered as ensembles as it used multiple algorithms to model the dataset.

It can be observed that bagging algorithm used along with random forest model has the highest accuracy among the models performed. It has significantly improved the accuracy to 98 percent.

MODELS	PRECISION	RECALL	F1 SCORE	AUC SCORE
RANDOM FOREST	63	63	63	62.8
K-NEAREST NEIGHBOUR	52	50	51	52.4
LOGISTIC REGRESSION	56	50	53	52.4
DUMMY CLASSIFIER	56	50	53	55.6
DECISION TREE	54	54	55	54.6
NAÏVE BAYES	58	30	40	52.4
XG-BOOST	63	61	62	62.9
BAGGING ALGORITHM	98	98	98	98.15
ADA BOOSTING	63	61	62	62.50

Fig. 19 Table representing performance metrics of all the models

For simple learners,

Naïve Bayes > Decision Tree > Logistic Regression > Dummy Classifier > KNN

For ensemble learners,

Bagging > Boosting.

VIII. CONCLUSION AND FUTURE WORK

The study involves comparison of various supervised models – Naïve Bayes, Decision Trees, Logistic Regression, Dummy classifier, KNN, AdaBoost, XGBoost and AdaBoost classifiers. The performance metrics is calculated for all the algorithms and compared with each other to conclude the better performing algorithm. Accuracy is considered as a best fit to compare all the models with each other.

It can be can be concluded that among simple learners Naïve Bayes performs well. The accuracy of Naïve Bayes is by far greatest and it can be able to predict the malwares with good precision. The decision trees has almost the same accuracy as Naïve Bayes and it performs better than other three simple learners. When considering the accuracy in general, bagging classifiers with Random forest is the best with almost accuracy greater than 98 percent.

Paired T test is by far chosen as the best parameter to compare the statistical significance between the algorithms and null hypothesis test is satisfied.
Thus, supervised model comparison is performed for different algorithms and the best algorithm is chosen for the dataset.

The dataset with simple learners does not provide more than 60 percent accuracy which is not encouragable for malware predictions. Even though tuning with parameters for simple learners and ensembles did not give desired result, advanced supervised or semi-supervised algorithms can be used to get the desired accuracy. Adversial Validation when performed normally is over-fitting the model. It can be tuned with necessary parameters to obtain perfect accuracy.
Since the dataset is large the computer should not take large time in running the processes. In real time scenarios, features like hardware limitations, network bandwidth, resource utilization and user experience should be taken in account.

REFERENCES

[1] https://scikit-learn.org
[2] https://machinelearningmastery.com/handle-missing-data-python/
[3] https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02/
[4] https://www.kaggle.com/c/microsoft-malware-prediction