iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VIII **Month of publication:** Aug 2023

DOI: https://doi.org/10.22214/ijraset.2023.55255

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VIII Aug 2023- Available at www.ijraset.com

Malware Detection and Classification Using Machine Learning Algorithms

Ronald Chiwariro[1], Lokaiah Pullagura[2]
[1, 2]*Department of Computer Science and Engineering, Faculty of Engineering and Technology JAIN (Deemed-to-be University), Bangalore*

Abstract: ***Malicious software, or malware, can take the form of executable code or system library files in the form of viruses, worms, or Trojan horses, all of which aim to compromise system security and user privacy. Detecting malware is done by analysing malware signatures and behaviour patterns statically and dynamically. It has been proven that these methods are ineffective and time-consuming when it comes to detecting unknown malware. The latest malware can be identified using machine-learning algorithms. Hence, this study aims to determine the most effective machine-learning algorithm for malware detection and classification. XGBoost, LightGBM, and Logistic Regression were used as algorithms for developing machine learning models. Pre-processing of the data is followed by training and testing of the models. The dataset containing information about various malware and their characteristics was obtained from Kaggle. A variety of metrics are used to evaluate the performance of the three models, including accuracy, precision, true negative rate, and false negative rate. Based on the analysis, it has been found that the LightGBM algorithm has the highest accuracy and precision. As a result, it follows that the LightGBM algorithm is the most effective algorithm for detecting and classifying malware.***

Keywords: Malware Detection and Classification, XGBoost, LightGBM, Logistic Regression, Machine Learning, Confusion Matrix, Correlation, Cardinality, Feature Reduction.

I. INTRODUCTION

Malware is a record or code that is passed over a network to infect, explore, steal, or play sincerely any action preferred by the attacker. Because malware is available in such a lot of varieties, there are various methods to contaminate systems. It is either sent to the target's system directly or indirectly. They are also allowed to be spread across the internet so that any user can accidentally click or download the malware. When the user clicks or installs it, the malicious code performs actions that the user did not anticipate or intend. The performance of the malware includes the replication of the malware in diverse regions of the system, preventing document access, browser Ads flooding, and even rendering a device inoperable. There are various types of malware. Each type has its functionality and characteristics. Some of the most common malware are Trojan horse viruses, spyware, computer worms, logic bombs, etc. Trojan horses seem like innocent programs, however, while activated, they motivate damage to the host computer. In contrast to a virus, a Trojan horse does not mirror itself; instead, this malware usually tries to steal documents or passwords. Spyware infects and operates on a user's device to display or extract records from the user.

Since malware has many classifications and it is widespread, it becomes difficult for computer users to effectively detect the malware and identify its type. This problem can be sorted by a machine learning model which is incredibly efficient in the detection of malware and classifying its type. This study aims to develop three machine learning models for detecting and classifying malicious software and compares the efficiency of all three models. The results of the comparison provide the best algorithm that can be used.

II. LITERATURE SURVEY

Various researchers have accomplished studies and research about malware and its types. Some researchers also develop malware as their research or even some annotations regarding the development of malware. A study by a group of researchers from the Deakin University of Australia is one of the studies that developed an annotation for Android malware. According to their findings, Android malware poses critical protection and privacy dangers to cellular users. Because of the malware landscape's fast evolution, conventional malware detection and its circle of relative class technology have become much less effective. The annotation designed in this study is named Automatic Capability Annotation for Android Malware (A3CM). Instead of categorizing detected malware into families, A3CM predicts its security and privacy capabilities. As a result, A3CM can address unknown or zero-day malware. The semantic capabilities of A3CM may be used to seize styles of security and privateness-associated capabilities [1].

Deep learning technology is also used by researchers in the prediction of malware. Researchers from the Vellore Institute of Technology, India have used intelligent vision in their study to detect malware. They used the random forest algorithm in this study. Whilst the classifier is skilled with recognized variations from the identical family, classifying unknown malware variations with comparable traits into their respective groups is a widespread challenge. Another problem for generalizing the malware detection gadget is the identity and extraction of remarkable capabilities for every malware. The proposed algorithm can provide an average accuracy of 98%. Deep ensemble layering and occasional version complexity distinguish the proposed approach, which outperforms deep neural networks in malware detection. The proposed version recognizes unknown malware samples from common malware families [2].

Deep learning algorithms have been used in the classification of malware. The traditional artificial intelligence (AI) algorithms, especially gadget learning (ML), are not able to detect all new and complicated malware variants. To correctly fight new malware variants, novel strategies that might be exceptional from conventional strategies have to be used. The experimental consequences display that the proposed approach can correctly classify malware with excessive accuracy, outperforming contemporary strategies advanced by different researchers. On a massive scale domain, the proposed approach is efficient and decreases function space [3].

Machine learning is an effective technology that is used in various fields. A review done by researchers at the Sunyani University of Ghana explains the applications of machine learning in various fields along with its challenges. Accordingly, even though most Artificial Intelligence (AI) and Machine Learning (ML) algorithms are free, they require a new talent set which is unusual for practitioners in this subject and IT departments. Furthermore, the provision of several ML algorithms creates a project in deciding on the fine one, akin to "seeking out a needle in a haystack". Analysed 66 types of research that make use of machine learning for classification purposes, analysed their pros and cons, and concluded that the selection of the perfect algorithm should be based on various parameters. The parameters include the requirement of the algorithm, the storage space required for the algorithm to perform efficiently, the domain of the research, etc. However, certain machine learning algorithms can be used in many applications irrespective of the domain [4].

As stated before, machine learning can be used in any field. There is various research that used machine learning in traffic detection. An essential problem in computerized training with huge datasets is modelling an efficient classifier. As a result, the computerized category is a sizable assignment that calls for the usage of schooling strategies able to assign training to facts gadgets primarily based totally on the enter sports provided to study training. Predefined training permits the popularity of the latest elements. The new version is primarily based totally on a proposed system mastering set of rules, which includes 3 layers: an enter layer, a hidden layer, and an output layer. To optimize the weights, a dependable education set of rules is proposed, and a popularity set of rules is used to validate the version. Before the evaluation step, the amassed site visitors are pre-processed. The purpose of their studies is to explain the mathematical validation of a new system mastering classifier for heterogeneous site visitors and anomaly detection [5].

The XGBoost algorithm is one of the machine learning algorithms that is used in many applications. The algorithm has usage in pedestrian detection. Pedestrian detection is extensively utilized in sensible monitoring, assisted use in automobiles, security, and smart transportation. It has been studied in lots of systems imaginative and prescient packages for plenty of years. Machine learning is trending and famous for pedestrian detection. Infrared screening is also utilized for this detection purpose. As a result, infrared and scene mild video fusion strategies may be used to enhance the detection effect, making the consequences of pedestrian actions at night-time dependable and effective [6]. The XGboost algorithm is also used in many cybersecurity-related applications such as detection of false data injection. Accordingly, the conventional strength device is step by step evolving right into a cyber-physical electricity device (CPES) with common interactions among physical and cyber components. It additionally creates new protection challenges, with disastrous outcomes for the strength device. The simulation effects display that the released stealthy False Data Injection Attacks (FDIA) can effectively skip conventional terrible records detection and for that reason compromise the CPES's records security, revealing the CPES's vulnerability and the need for a detecting method. It concluded that the FDIA defence mechanism primarily based totally on the XGBoost classifier detects cyber-assaults with excessive accuracy and robustness [7].

Light Gradient Boosting Method (LightGBM) algorithm is one of the fastest machine learning algorithms used for classification and detection. It is a gradient-boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with support for parallel and GPU learning and is capable of handling large-scale data. Due to its quick response, the LightGBM is also used in various applications. The algorithm has been utilised in the study that maps the population based on the density of the population. Accordingly, Gradient-based One-Side Sampling (GOSS) can acquire a correct statistics advantage with a tremendously small pattern dataset since information with large gradients plays a greater function within the calculation of statistics advantage. In the meantime, EFB permits the binding of together extraordinary functions to lessen the wide variety of functions. LightGBM, that's primarily based totally on GOSS and Exclusive Feature Bundling (EFB), can boost the learning process [8].

The LightGBM algorithm is also used in the medical field. It can be used in the prediction of sepsis from clinical images. Sepsis is an intense clinical situation because of the body's overreaction to an infection, which could bring about tissue damage, organ failure, or even death. The emergence of superior technology which includes synthetic intelligence and smart gadgets facilitates the speedy exploration of superior strategies for recognizing sepsis cases. The algorithm designed in this study can predict sepsis with high accuracy and precision in a faster way [9].

The Logistic Regression (LR) algorithm is one of the simplest machine learning algorithms to implement. Thus, this algorithm is also used in various applications. It is used in the estimation of lower limb muscle activations. The estimation of human muscle pastime has been studied as an essential approach in lots of fields. Electromyography (EMG), a technique that offers visible comments on the electromyogram's potential, is not an unusual application. The Logistic Regression reduces errors due to nonlinearities consisting of saturation of the enter sign and the benefit of permitting the regression coefficient to be calculated with a small quantity of data. When training the usage of outcomes of the order of numerous seconds, the proposed approach outperformed the other compared methods [10]. The LR algorithm is also compatible with other machine learning algorithms and even deep learning algorithms.

Furthermore, the combination of the Logistic Regression algorithm and Convolutional Neural Networks (CNN) is used in facial landmarks detection. Consequently, most current course-to-great facial detectors fail to discover landmarks as they should without a massive quantity of completely labelled data, that is high-priced to obtain. The LR-CNN version may be trained correctly with a small quantity of finely labelled data and a huge quantity of generated weakly labelled data using weakly supervised learning. Extensive checks on benchmark datasets display that the proposed technique can complete multi-challenge facial detection and outperform other facial element and landmark detection algorithms [11].

III. MATERIALS AND METHODS

A dataset consisting of data with numerous types of malware along with the characteristics undergoes a lot of methods to identify the best algorithm that can be used for malware detection. These methods are represented in the flowchart in figure 1.

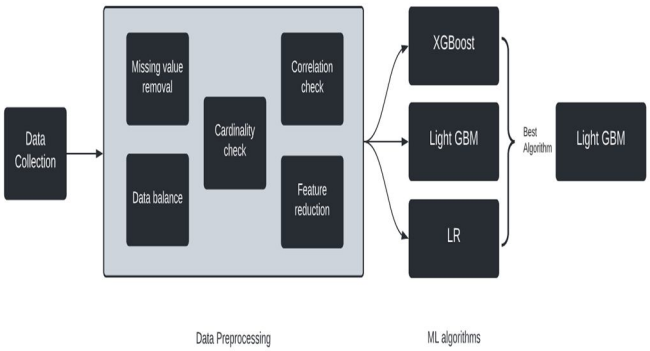


Fig. 1. Workflow of the study

From figure 1, the obtained dataset is pre-processed using various methods shown. The processed values are then used to train and test the models. The workings of the above procedures are explained clearly in the upcoming chapters.

IV. DATA COLLECTION AND PRE-PROCESSING

A dataset consisting of data with 83 columns or features about malware and its types is collected from Kaggle [12]. Kaggle is a data science and artificial intelligence platform where contests with monetary prizes are published by large companies and organizations. In addition to the competitions, users can also share their datasets and examine the datasets shared by others. The dataset consists of 8921483 samples of data with 82 features each The telemetry data containing these properties and the machine infections were generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender. . The data is of three types: categorical data, binary data, and numerical data. The distribution of the data according to datatypes is shown in Figure 2 as a pie chart.

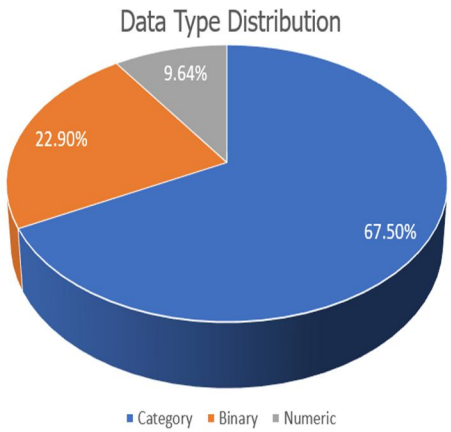


Fig. 2. Data type distribution

Due to the abundant amount of data, it becomes tough for machine learning models to process the data. Thus, the obtained dataset is pre-processed to find the necessary features that are required for malware detection. Five pre-processing methods are engaged in the reduction of this database. The methods are missing value removal, cardinality check, correlation check, feature reduction, and data balance. The methods are explained below.

A. Missing Value Removal

Out of the 83 columns, almost every column consists of some missing values. However, if only a few values are missing, they can be neglected. However, when a column consists of more than 40% of missing values, it can hinder the efficiency of the model since these columns do not contribute much to the model training. So, the columns that have more than 40% of missing values are removed from the dataset. Almost 7 columns consist of such missing values. The columns that are eliminated during this process are listed as output in Figure 3.

1) Code

```
percent = (train_data.isnull().sum()/train_data.shape[0]) * 100
new_train_data= pd.DataFrame(data=percent,columns=['Percentofnullvalues'])
new_train_data = new_train_data.sort_values(by='Percentofnullvalues',ascending=False)
print(new_train_data.head(15))
```

2) Output

Code:

```
def remove_columns(data):

    ''' Computing percent of null values in a dataset based on the feature and removing those features having 40 or more than 40
percent of null values '''
    columns_to_be_removed = []
    percent = (data.isnull().sum()/data.shape[0]) * 100

    for col in data.columns:
        if percent.loc[col] >= 40:
            columns_to_be_removed.append(col)
    new_data = data.drop(columns=columns_to_be_removed)

    return new_data
```

Fig. 3. Identifying and removing columns with more than 40% of missing values

B. Cardinality Check

Almost all datasets have categorical variables. Each categorical variable consists of unique values. A categorical feature is said to possess high cardinality when there are too many of these unique values. One-Hot Encoding becomes a big problem in such a case since we have a separate column for each unique value (indicating its presence or absence) in the categorical variable. This leads to two problems: space consumption and the curse of dimensionality. The cardinality of the data also plays an important role in the working of a machine learning model. If a column consists of only a few possible values, it will be easier for the model to be trained according to that value and vice versa. Thus, the columns with a cardinality of more than 50 are also eliminated from the dataset as a way to avoid the above-mentioned problems. The columns removed by this method are shown in Figure 4.

1) Code

```
from collections import Counter
def cumulatively_categorise(column,threshold=0.5,return_categories_list=True):

#Find the threshold value using the percentage and number of instances in the column
 threshold_value=int(threshold*len(column))

#Initialise an empty list for our new minimised categories
        categories_list=[]
#Initialise a variable to calculate the sum of frequencies
        s=0
#Create a counter dictionary of the form unique_value: frequency
        counts=Counter(column)

#Loop through the category name and its corresponding frequency after sorting the categories by descending order of frequency
        for i,j in counts.most_common():
#Add the frequency to the global sum
          s+=dict(counts)[i]

#Append the category name to the list
          categories_list.append(i)
#Check if the global sum has reached the threshold value, if so break the loop
          if s>=threshold_value:
            break
#Append the category Other to the list
        categories_list.append('Other')

#Replace all instances not in our new categories by Other
        new_column=column.apply(lambda x: x if x in categories_list else 'Other')

#Return transformed column and unique values if return_categories=True
        if(return_categories_list):
          return new_column,categories_list
#Return only the transformed column if return_categories=False
        else:
          return new_column

#Call the function with a default threshold of 50%
transformed_column,new_category_list=cumulatively_categorise(df['column'],return_categories_list=True)
```

2) Output

```
['AvSigVersion',
 'DefaultBrowsersIdentifier',
 'AVProductStatesIdentifier',
 'CityIdentifier',
 'OsBuildLab',
 'Census_OEMNameIdentifier',
 'Census_OEMModelIdentifier',
 'Census_ProcessorModelIdentifier',
 'Census_FirmwareManufacturerIdentifier',
 'Census_FirmwareVersionIdentifier']
```

Fig. 4. Columns with a cardinality of more than 50

C. Correlation Check

The term correlation is used to represent the relationship between two or more variables i.e., how the values of a specific column are affected by the change of values in other columns. The correlation between the columns is also checked to ensure the better performance of the machine learning models. Due to memory limitations, the columns are divided into batches of 15. As part of the exploratory data analysis to improve model performance, the correlation of the features with the target variable in this case "*HasDetections*" is very important. The sample of the correlation check done for the dataset is shown in figure 5.

1) Code

```
train_data[train_data.columns[1:]].corr()['HasDetections'][:].sort_values(ascending=False)
```

2) Output

```
HasDetections                                    1.000000
AVProductStatesIdentifier                        0.117404
Census_TotalPhysicalRAM                          0.057069
IsProtected                                      0.057045
Census_ProcessorCoreCount                        0.054299
Wdft_IsGamer                                     0.053891
RtpStateBitfield                                 0.041486
Census_InternalPrimaryDiagonalDisplaySizeInInches 0.034240
Census_InternalPrimaryDisplayResolutionHorizontal 0.031920
Census_OSBuildNumber                             0.029486
OsBuild                                          0.024754
Census_ProcessorModelIdentifier                  0.022711
```

3) Code

```
corr_rel = data[features]
correlation_matrix = corr_rel.phik_matrix()
plot_correlation_matrix(correlation_matrix.values, x_labels=correlation_matrix.columns, y_labels=correlation_matrix.index,
                        vmin=0, vmax=1, color_map='blue', title=r'correlation $\1-15$', fontsize_factor=1.5,
                        figsize=(25,25))
plt.tight_layout()
```

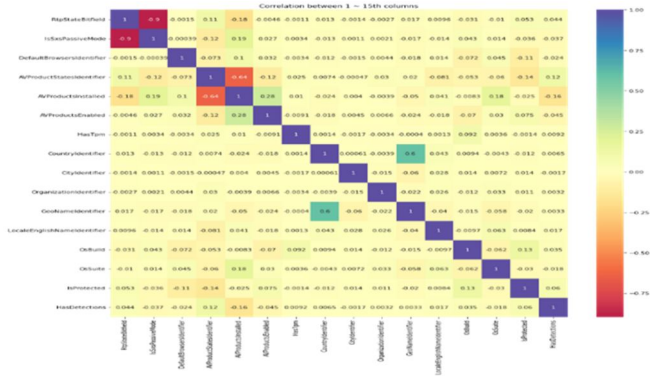


Fig. 5. Correlation check

D. Feature Reduction

Even after the removal of several columns using the above-mentioned pre-processing techniques, some features of the dataset are still unnecessary in the process of malware detection. Thus, the columns that are irrelevant to this classification are also removed. The number of columns after every step of pre-processing is shown in figure 6.

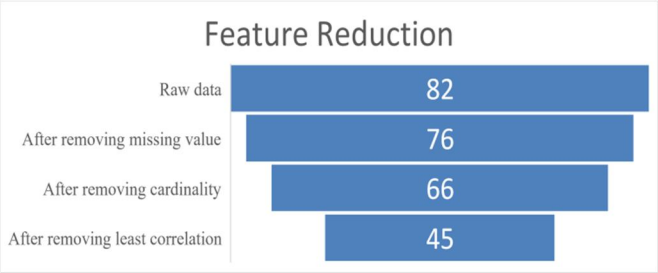


Fig. 6. Feature reduction

The train.csv file contains entries, where each entry corresponds to a machine that is uniquely identified by a *MachineIdentifier* and *Has Detections*, which is the ground truth and indicates that Malware was detected on the machine. After data pre-processing, the training dataset contains 1500000 rows and 45 columns which are split into a training set and a validation set. The validation set is for parameter hyper-tuning and optimising the best model. They are equally split into two, the training set and the validation set with 750000 rows and 45 columns for the training set and validation set each. The data is then fitted into the model training. The training sample is shown in Figure 7.

```
Train on 750000 samples, validate on 750000 samples
Epoch 1/3
 - 69s - loss: 0.6441 - accuracy: 0.6216 - val_loss: 0.6347 - val_accuracy: 0.6295
Train AUC: 0.693089888008839
Validation AUC: 0.6906604230388648
Epoch 2/3
 - 66s - loss: 0.6387 - accuracy: 0.6281 - val_loss: 0.6281 - val_accuracy: 0.6365
Train AUC: 0.6977973033795017
Validation AUC: 0.694683409002442
Epoch 3/3
 - 69s - loss: 0.6363 - accuracy: 0.6303 - val_loss: 0.6286 - val_accuracy: 0.6379
Train AUC: 0.7008944098845151
Validation AUC: 0.6968994090877807
```

Fig. 7. Train and validation samples

The test.csv file contains 1500000 rows with the selected features the same as those the models are trained on less the label. Since this is a real-time competition dataset, using the information and labels in train.csv, the models must predict the value for '*HasDetections*' for each machine in test.csv.

4) Data Balance

The data must be balanced so that the training of the models will not be biased. If one class of data exceeds the number in another class of data, it results in the model being trained more for the former class. So, the values from the dataset are checked for class balance. If a value is more than another, one of the values is altered to maintain balance in data. The bar graph representing the classes of data after data balance is shown in figure 8.

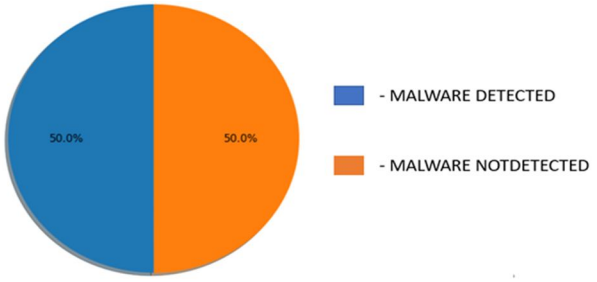


Fig. 8. Data after data balance

The pre-processed data can be used for both training and testing the machine learning models. The data is then again split into two parts in the ratio of 4:1. The larger part will be used to train the models and the smaller part will be used for testing.

V. PAGE FORMATTING

A Machine Learning model can be thought of as a program that has been trained to discover patterns in data and make predictions in new data. These models are represented by a mathematical function that accepts requests in the form of input data, makes predictions based on that data, and then responds with an output. These models are first trained on a collection of data before being given an algorithm to reason over that data, extract patterns from fed data, and learn from it. These models can be used to predict the unknown dataset once they have been trained. The Models under evaluation are based on XGBoost, LightGBM, and Logistic Regression algorithms.

A. XGBoost

Extreme Gradient Boosting (XGBoost) is a scalable, allotted Gradient-Boosted Decision Tree (GBDT) system for acquiring knowledge of the library. It is the main system for gaining knowledge of the library for regression, classification, and rating problems, and it helps parallel tree boosting. XGBoost is optimized to be especially efficient, flexible, and portable. The XGBoost algorithm is also capable of producing the smallest Root Mean Square Error (RSME) value in various scenarios. Gradient Boosting Tree carries out unsupervised studying through studying from information without a predefined model. It is called an "All in One" algorithm. It is likewise called an end-to-end algorithm. It is a perfect mixture of software programs and hardware optimization strategies that produce perfect consequences whilst the usage of the fewest computing assets within the shortest quantity of time. It is extra fast than Gradient Boosting. With its integrated features, it is intended to address lacking information [13].

B. LightGBM

Light Gradient Boosting Machine (LightGBM) makes use of selection tree algorithms to carry out the ranking, and classification The advantages are that LightGBM is a histogram-primarily based set of rules that play fee bucketing, which ends up in quicker training speed and accuracy. It additionally requires less memory. This proves to be a large gain whilst operating on massive datasets and constrained hardware environments. The LightGBM algorithm provides the smallest RMSE value than many other algorithms making it more effective [14].

C. Logistic Regression

Logistic Regression is a Machine Learning algorithm used mainly for binary category problems. It is used to predict specific structured and unbiased variables. A logistic regression instance could be the usage of a system to decide whether someone is likely to be inflamed with a particular disease [15]. Linear regression is straightforward to apprehend and explain, and it could be regularised to avoid overfitting. Furthermore, linear fashions may be effortlessly up to date with new statistics and the use of stochastic gradient descent. It additionally has an extreme prediction rate.

VI. PERFORMANCE METRICS

The performance of Machine Learning algorithms; classification algorithms, and regression algorithms can be evaluated using a variety of metrics. Because how the performance of ML algorithms is assessed and compared is reliant on the metric used, it must be carefully selected to measure ML performance. The Confusion Matrix is the simplest technique to assess the performance of a classification task with two or more types of output. A confusion matrix is a table containing two dimensions, “Actual” and “Predicted”, which in turn intersect as “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)” and “False Negatives (FN)” on both dimensions, as illustrated in Figure 9 below.

Fig. 9. Confusion Matrix

- True Positive (TP): the ground truth is that it is positive, and the test predicts that it will be positive. It is Malware, and the exam correctly identifies it.
- True Negative (TN): the ground truth is that it is negative, and the test predicts that it will be negative. It is not Malware, and the test correctly reflects this.
- False Negative (FN): the ground truth is positive, but the test predicts a negative. The file is malicious, yet the test incorrectly indicates that it is legitimate. In statistics, this is known as a Type II error.
- False Positive (FP): Although the ground truth is negative, the test indicates a positive outcome. It is not Malware, but the test incorrectly indicates that it is. In statistics, this is known as a Type I error.

  We can quantify the accuracy of our tests quantitatively by calculating ratios between these numbers. Therefore, the calculations used for the comparison of the models' performances are based on the equations mentioned below.

A. Accuracy

It is the most often used indicator for assessing the performance of classification algorithms. It's the number of correct predictions divided by the total number of predictions. To calculate the accuracy of our classification model, we may utilise the accuracy score function in sklearn.metrics.

$$Accuracy = \frac{Correct\ Predictions\ (TP+TN)}{Total\ Predictions\ (TP+FN+TN+FP)} \quad (1)$$

B. True Negative Rate (TNR)

The true negative rate, also known as specificity, is the likelihood that a true negative will test negative.

$$TNR = \frac{TN}{TN+FP} \quad (2)$$

C. True Positive Rate (TPR)

The probability that an actual positive will test positive is also known as sensitivity or recall.

$$TPR = \frac{TP}{TP+FN} \quad (3)$$

D. False Negative Rate (FNR)

The probability that a true positive will be missed by the test, is also known as the miss rate.

$$FNR = \frac{FN}{FN+TP} \quad (4)$$

E. False Positive Rate (FPR)

It is the chance that a false alarm will be triggered, resulting in a positive result while the genuine value is negative.

$$FPR = \frac{FP}{FP+TN} \quad (5)$$

F. Precision

The number of correct malware classifications returned by the model.

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

G. F1 Score

From the above performance measures, the F1 Score can also be calculated. The F1 score is made up of two components: precision and recall. The F1 score's purpose is to combine the precision and recall metrics into a single metric. At the same time, the F1 score was created to operate well with data that is imbalanced. F1 Score is the harmonic mean of precision and recall. Because they are both rates, the harmonic mean is a logical choice.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \qquad (7)$$

Because the F1 score is the average of Precision and Recall, it gives Precision and Recall equal weight:

- A model will have a high F1 score if both Precision and Recall are high.
- If both Precision and Recall are low, a model will have a low F1 score.
- If one of Precision or Recall is low and the other is high, a model will get a medium F1 score.

VII. EXPERIMENTAL SETUP

This section describes the parameter selection of the machine learning algorithms used for experimentation. Some classifiers we intend to use are sensitive to parameters. We perform hyperparameter tuning to find the best parameters.

A. Xtreme Gradient Boosting

The Extreme Gradient Boosting algorithm is a decision tree-based machine learning algorithm that uses a process called boosting to help improve performance. The most powerful ML algorithms like XGBoost are famous for picking up patterns and regularities in the data by automatically tuning thousands of learnable parameters. The hyperparameters include the maximum depth of the tree, the number of trees to grow, the number of variables to consider when building each tree, the minimum number of samples on a leaf, and the fraction of observations used to build a tree. Since there are many parameters for the optimisation of XGBoost, Grid search CV was used to automatically find the optimum combination of hyperparameters for the best result. The selected parameters used for the final solution are shown in Figure 10.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eval_metric='mlogloss',
              gamma=0, gpu_id=-1, importance_type='gain',
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=16,
              num_parallel_tree=1, objective='multi:softprob', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', use_label_encoder=False,
              validate_parameters=1, verbosity=None)
```

Fig. 10. XGBoost Grid search Hyperparameters

B. Light Gradient Boosting Method

Gradient boosting utilizes tree-based learning, and its base classifier is based on decision trees. Unlike most traditional boosting methods whereby decision trees are grown breadthwise, LGBM grows leaf-wise. Growing leaf-wise usually results in a lower loss, but it may cause overfitting when the data size is small. That is not the case for our project since our dataset has 1.5 million observations. To ensure that the classifier can obtain the complete pattern of the training dataset, the K-Fold cross-validation method is used to train the classifier. It is also to ensure that there is ample data for training and validation. In K-Fold cross-validation, the dataset is divided into n subsets. This method will run for n iterations. For every iteration, one of the n subsets will be used as a validation set, while the other n-1 subsets will form a training set. Every n subset is a validation set exactly once and a training set n-1 times. This reduces the bias and variance since we are using the entire dataset for training. Train the classifier with a learning rate of 0.1, and it stops when the validation score stops improving after 100 rounds. To prevent overfitting, introduce bagging and feature subsampling, with a fraction of 0.8, and a frequency of 1. This means that for every iteration, 0.8 of the training data will be used for bagging and feature subsampling.

C. Logistic Regression

Logistic regression uses a sigmoid function to help map the probabilities into any values between 0 and 1. The advantage of Logistic regression is that it is good for binary classification and suitable for this problem as the probabilities of whether malware is detected are any values between 0 and 1. Five different solvers were tried: liblinear, Stochastic Gradient Descent (SGD), Stochastic Average Gradient Descent (SAG), SAGA (a variant of SAG) and limited-memory Broyden-Fletcher-Goldfarb-Shanno Algorithm(LBFGS). For each solver, an L2 regularization (penalty=L2) is used to reduce overfitting. The rest of the parameters are kept as default (C=1.0, class weight=None). In terms of training time, SAGA and SAG are the most appropriate for this problem as they are optimized for very large datasets and take around 20 minutes to train the classifier compared to other classifiers such as liblinear which takes 3 hours to train. LBFGS is not optimized for large datasets, but in this case, the data used has been standardized, which results in a faster training time of around 20 minutes. C represents the inverse of regularization strength. The smaller the value of C, the stronger the regularization. The best classifier for logistic regression was the LBFGS solver with C=0.2.

VIII. RESULTS AND DISCUSSION

Three machine learning models were developed using machine learning algorithms which are the XGBoost algorithm, the LightGBM algorithm, and the Logistic Regression algorithm. The models were then trained and validated using the processed dataset before testing for efficiency. The performance of the three models is plotted on a graph for comparison. The graph is shown in Figure 11.

Fig. 11. Performance of all algorithms

Accuracy is the proportion of correct predictions out of all predictions. From figure 11, the results show that LightGBM algorithm achieves 93.3%, followed by the XGBoost algorithm at 89.3% and finally, Logistic Regression at 84%. It can therefore be concluded that LightGBM has managed to detect and classify 93.3% of the test data correctly and has outperformed XGBoost and Logistic Regression by 4% and 9.3% respectively in terms of accuracy.

Furthermore, precision is the proportion of correct positive predictions of all cases classified as positive. This is particularly an important metric in malware classification since misclassification leads to legit files being deleted or regarded as malicious. LightGBM achieved 93.2% followed by XGBoost with 87.7% and Logistic Regression with 80.9%. It can further be concluded that LightGBM classification is more desirable due to the high precision compared to XGBoost and Logistic Regression by 5.5% and 12.3% respectively. In addition, the True Negative Rate (TNR) also termed specificity or selectivity is a performance metric that measures the probability that your model will predict negative when the true value is negative. It is closely related to the True Positive Rate (TPR), which is completely analogous. This is also crucial in this use case since misclassification leads to malware being treated as legitimate, which results in system compromise. The LightGBM algorithm also has a True Negative Rate of 90.4% followed by XGBoost with 85% and Logistic Regression at 76% thus 5.4% and 14.4% progressed than the other two algorithms under comparison respectively. LightGBM True Positive Rate is 95.4%, XGBoost 94% and Logistic Regression 89% which is 1.4% and 6.4% progressed than XGBoost and Logistic Regression respectively.

Furthermore, LightGBM has the lowest False Negative Rate of 4.5% compared to XGBoost and Logistic Regression with 6% and 10% accordingly. The False Positive Rate of LightGBM is 9.5%, XGBoost 16% and Logistic Regression 21%. It is therefore evident that LightGBM achieves the desired performance metrics in relation to Malware detection and Classification on the given dataset. However, further hyperparameter tuning can improve the results. The main concerns are memory usage and the time it takes to run a given model.

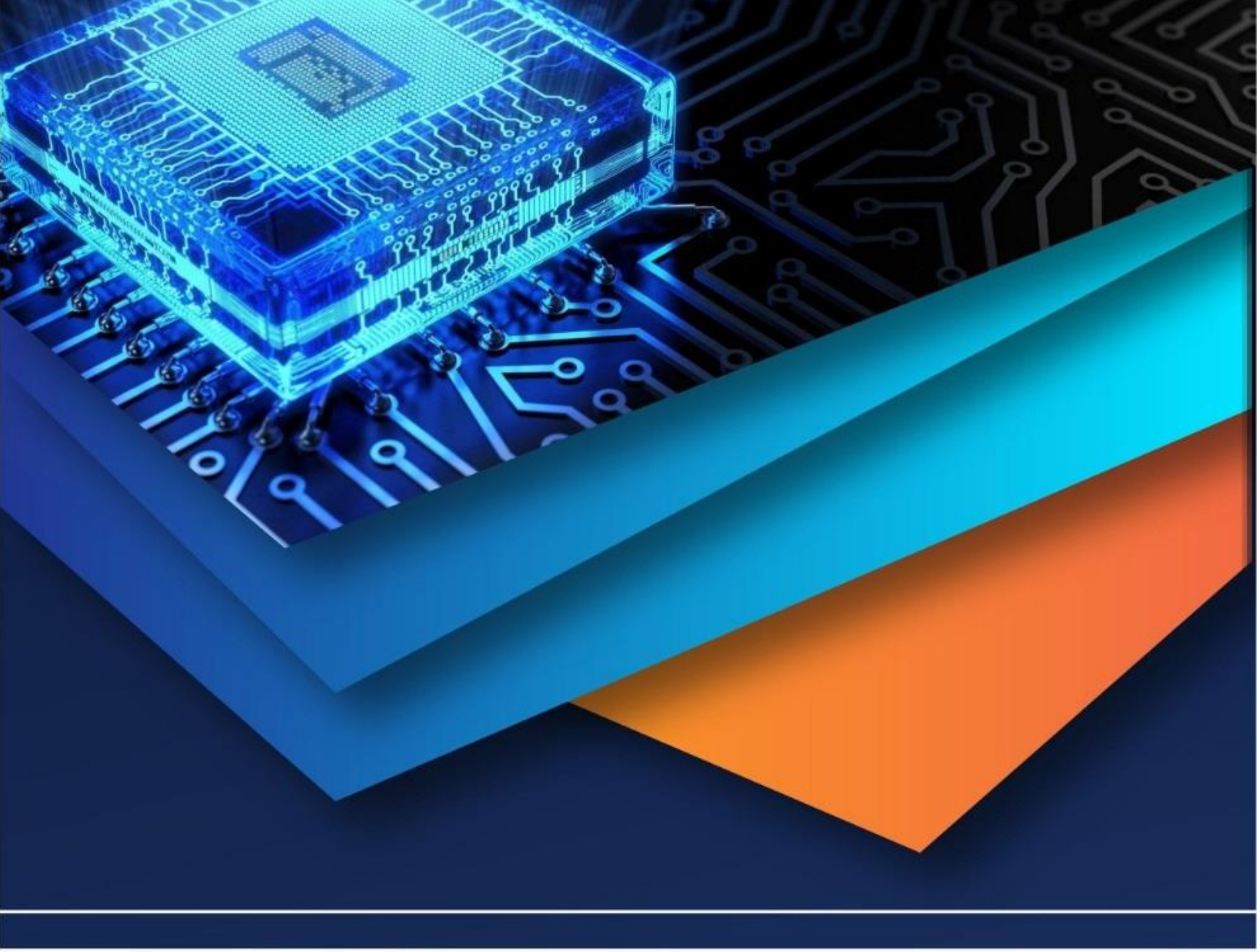
These results on the dynamic prediction of malware are particularly positive to the threat model described in the paper. The advantage is that zero-day attacks can also be detected and classified even though they are not present in the anti-virus signature database. Additionally, device security is guaranteed due to the malware prediction capability compared to relying on anti-virus software that required virus definitions for comparison before classifying files as malicious.

IX. CONCLUSION

The XGBoost, LightGBM, and Logistic Regression algorithms were used to construct three different machine learning models that were trained and validated on Kaggle's Malware dataset. A graph was used to compare the performance of all three algorithms using the test.csv dataset provided on the same platform. Based on our findings, the LightGBM algorithm is best suited for this use case of malware detection and classification. LightGBM exhibited superior performance in terms of precision, true positive rate, false positive rate, true negative rate, and false negative rate over the other two algorithms. Therefore, this model can be deployed on a website or software application to detect malware and classify it in real-time. To improve results, hybrid approaches can also be considered.

REFERENCES

[1] J. Qiu et al., "A3CM: Automatic Capability Annotation for Android Malware," in IEEE Access, vol. 7, pp. 147156-147168, 2019, DOI: 10.1109/ACCESS.2019.2946392.
[2] S. A. Roseline, S. Geetha, S. Kadry et. al., "Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm," in IEEE Access, vol. 8, pp. 206303-206324, 2020, DOI: 10.1109/ACCESS.2020.3036491.
[3] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, DOI: 10.1109/ACCESS.2021.3089586.
[4] I. K. Nti, J. A. Quarcoo, J. Aning et. al, "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects," in Big Data Mining and Analytics, vol. 5, no. 2, pp. 81-97, June 2022, DOI: 10.26599/BDMA.2021.9020028.
[5] A. Guezzaz, Y. Asimi, M. Azrour et. al, "Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection," in Big Data Mining and Analytics, vol. 4, no. 1, pp. 18-24, March 2021, DOI: 10.26599/BDMA.2020.9020019.
[6] Y. Jiang, G. Tong, H. Yin et. al, "A Pedestrian Detection Method Based on Genetic Algorithm for Optimize XGBoost Training Parameters," in IEEE Access, vol. 7, pp. 118310-118321, 2019, DOI: 10.1109/ACCESS.2019.2936454.
[7] W. Xue and T. Wu, "Active Learning-Based XGBoost for Cyber-Physical System Against Generic AC False Data Injection Attacks," in IEEE Access, vol. 8, pp. 144575-144584, 2020, DOI: 10.1109/ACCESS.2020.3014644.
[8] X. Zhao, N. Xia, Y. Xu et. al, "Mapping Population Distribution Based on XGBoost Using Multisource Data," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 11567-11580, 2021, DOI: 10.1109/JSTARS.2021.3125197.
[9] S. Chami and K. Tavakolian, "Early Prediction of Sepsis from Clinical Data Using Single Light-GBM Model," 2019 Computing in Cardiology (CinC), 2019, pp. Page 1-Page 4, DOI: 10.23919/CinC49843.2019.9005718.
[10] M. Sekiya, S. Sakaino and T. Toshiaki, "Linear Logistic Regression for Estimation of Lower Limb Muscle Activations," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 27, no. 3, pp. 523-532, March 2019, doi: 10.1109/TNSRE.2019.2898207.
[11] R. Zhang, C. Mu, M. Xu et. al, "Facial Component-Landmark Detection with Weakly-Supervised LR-CNN," in IEEE Access, vol. 7, pp. 10263-10277, 2019, DOI: 10.1109/ACCESS.2018.2890573.
[12] [Dataset]Microsoft, 2019, Microsoft Malware Prediction, Kaggle, https://www.kaggle.com/competitions /Microsoft-malware-prediction/data, accessed on 2 August 2022.
[13] S. Zhang, D. Zhang, J. Qiao, X. Wang and Z. Zhang, "Preventive control for power system transient security based on XGBoost and DCOPF with consideration of model interpretability," in CSEE Journal of Power and Energy Systems, vol. 7, no. 2, pp. 279-294, March 2021, DOI: 10.17775/CSEEJPES.2020.04780.
[14] Y. Wu and Q. Wang, "LightGBM Based Optiver Realized Volatility Prediction," 2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE), 2021, pp. 227-230, DOI: 10.1109/CSAIEE54046.2021.9543438.
[15] Y. Wang, Y. Ou, X. Deng, L. et. al, "The Ship Collision Accidents Based on Logistic Regression and Big Data," 2019 Chinese Control and Decision Conference (CCDC), 2019, pp. 4438-4440, DOI: 10.1109/CCDC.2019.8832686.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY