Prediction of Malwares in Microsoft Windows Operating Systems

Olukayode Aiyeniko[1], Aishat Oladayo Jimoh-Mahmud [2], Temitope Ayanladun Oyelakun[3], Oluwaseyi Inubiwon Oluwabukola[4], Stella Kehinde Ogunkan[5]

[1]Department of Computer Science, Lagos State University, Lagos State, Nigeria

[2]Department of Computer Science, Alhikmah University, Ilorin, Kwara State State, Nigeria

[3,5] Department of Information System, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria

[4]Department of Computer Science, Kogi State Polytechnic, Kogi State, Nigeria

[5]Department of Computer Science, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria

DOI : https://doi.org/10.51584/IJRIAS.2024.907020

Received: 18 June 2024; Accepted: 29 June 2024; Published: 07 August 2024

ABSTRACT

Malware has been identified as one of the predominant cyber threats with the fast growth of the internet. Anti-malware vendors nowadays receive a huge amount of speculated malware files daily. To keep up with the flow of these malware-ridden files, machine-learning techniques are used to abstract similar malwares. This paper presents a prediction model for the prediction of malware attacks based on certain features of the machine and the Light Gradient Boost Algorithm was employed for this purpose. A dataset that came in two splits (the train and test splits) with entries of Windows machines with different specifications was acquired and preprocessed to remove irrelevant features. The train split of the dataset was then used to train the Light Gradient Boost Algorithm to derive a model which was then used for the prediction on the test split. The accuracy of the model was found to be 98% while the precision and recall of the model were also found to be 98%. This study would help Windows users know what kind of specifications of machines are more prone to malware attacks.

Keywords: Malware, Prediction, Microsoft Window Operating System

INTRODUCTION

Electronic products have become a necessity for most people nowadays [1]. Because these electronic products have taken over almost every side of human life and endeavour, the attention of hackers has also been drawn to them. This attention has yielded a considerable spike in the number of malwares that is currently in distribution [2]. The development of current technologies has produced computers one of the most commonly used electronic products [3]. However, there are always several people who want to take advantage of others by attacking others' computers. To evade property damage as much as possible, precise and efficient detection is essential. As reported by Kaspersky Labs in 2017, there was an increasing trend in the types of malwares as at least 360,000 new malicious files were detected every day for the whole of 2017.

Nowadays, the generation of technological innovation and the enlargement and extension of data in the database is very fast [4]. All things associated with technology are completely complementing data growth, as well as scientific data, social media, and financial technology [5]. Data mining tools and techniques have been employed extensively by researchers to predict future trends by making businesses more proactive and better knowledge-driven decisions [6]. With the fast advancement in data mining techniques and methods came the advent of machine learning which represents a subset of Artificial Intelligence[7]. Machine learning can also be viewed as a subdivision of the computing field where the main idea is the ability of a digital computer to learn without

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

being trained [7]. A computer program is claimed to learn from experience A of some classes of tasks B and performance measure C, if its performance at tasks within B, as estimated by C, boosts with experience A.

Fundamentally, the idea behind every machine learning task is to train a model based on some algorithm to perform certain tasks like clusterization, classification, regression and so on. An input dataset is used to train the model and the subsequent model that is built is used to make predictions. Implementation and the initial task performed are determining factors of the output of such a model. Machine learning has been applied extensively in this study to make predictions based on the dataset if a particular machine is likely to be infected with malware [8]. Under this, it is incredibly significant to find out which factors are related to the infection of computers. This study focused on the different machines involved and their probability of getting attacked, which supply necessary instructions to protect computers.

LITERATURE REVIEW

2.1 Malware

Malware is referred to as programs that have malicious code which serves as a major threat to any internet user [9]. It has a severe effect on the technological world. It has been classified to be one of the most dangerous threats on the internet [10]. Malware analysis is the process of finding the functionality, origin and potential impact of a given malware sample such as a virus, worm, Trojan horse, rootkit, or backdoor [11]. Malware or malicious software is any software that is intended to harm the user's operating system or steal sensitive data from users, organizations, or companies. Malware is immensely powerful to the extent that it can take full control of infected host and network connection deactivating all the firewalls and installed antiviruses. The problem is increasing with internet use as most web pages have been infected with diverse types of malware downloads which can be gotten by the opening of web pages [12]. According to statistics made by Google, 70% of malware is gotten from popular sites.

2.2 Related Work

Maniriho et al. [13] conducted a comprehensive overview of the current state of Windows malware detection techniques, research issues, and future directions. The systematic literature review was done by examining the scientific literature on Windows malware detection based on executable files (.EXE file format) published between 2009 and 2022. The study presented new insights into the categorization of malware detection techniques based on datasets, features, machine learning and deep learning algorithms. It identified ten experimental biases that could influence the performance of malware detection techniques. It provides insights on performance evaluation metrics and discuss several research issues that impede the effectiveness of existing techniques. The study also provides recommendations for future research directions and is a valuable resource for researchers and practitioners working in the field of Windows malware detection.

Avci et al. [14] evaluated and benchmarked the performance of LSTM-based malware detection approaches on specific LSTM architectures to provide insight into malware detection. The method builds LSTM-based malware prediction models and performs experiments using different LSTM architectures including Vanilla LSTM, stacked LSTM, bi-directional LSTM, and CNN-LSTM. The study evaluated the performance of each of these architectures and different configurations. The study, as a contribution, shows that Bidirectional LSTM with hyperparameter optimization is found to be overperforming other selected LSTM architectures and alao tthat different LSTM approaches and architectures apply to the malware detection problem. Quality attributes such as efficiency and accuracy, and the software system architecture adopted for the implementation impact the selection of the LSTM approach.

A malware identification and classification scheme based on bicubic interpolation to improve the security of a plant protection information terminal system Chen et al. [16]. The study used a bicubic interpolation algorithm to restructure the generated malware images to reduce the problem of image size imbalance. The cycle-GAN model was used for data augmentation to balance the number of samples among malware families and build an efficient malware classification model based on CNNs to improve the malware identification and classification performance of the system. The accuracy of RGB and grey images generated by the Microsoft Malware

Classification Challenge Dataset (BIG2015) can reach 99.76% and 99.62%, respectively.

Performance evaluation of machine learning algorithms for malware detection [9]. Classifiers such as kNN, DT, RF, AdaBoost, SGD, extra trees and the Gaussian NB classifier were used to synthesize the study. After reviewing the test and experimental data for all five classifiers, it was discovered that the RF, SGD, extra trees and Gaussian NB Classifier all achieved 100% accuracy in the test, as well as a perfect precision (1.00), a good recall (1.00), and a good f1-score (1.00). Therefore, it is reasonable to assume that the proof-of-concept employing autonomous behaviour-based malware analysis and machine learning methodologies might identify malware effectively and rapidly.

Developed ransomware classification model in Microsoft Windows using hardware profile [16]. shows that obtaining a hardware execution profile is beneficial to identify the obfuscated ransomware too. The features obtained from hardware performance counters were evaluated to classify malicious applications into ransomware and non-ransomware categories using several machine learning algorithms such as Random Forest, Decision Tree, Gradient Boosting, and Extreme Gradient Boosting. The employed data set comprises 80 ransomware and 80 non-ransomware applications, which are collected using the VirusShare platform. The results revealed that extracted hardware features play a substantial part in the identification and detection of ransomware with an F-measure score of 0.97 achieved by Random Forest and Extreme Gradient Boosting.

METHODOLOGY

3.1 Approach to Prediction of Malware Attacks in Windows Operating Systems

Various steps were involved in the development of a predictive model for malware in the Windows Operating System. The dataset that was used in conducting this experiment was acquired from Kaggle and it came in two splits, the training and testing set. The dataset was first visualized to get the percentage of columns with high cardinality, missing values or null values, and irrelevant features. Then cleansing was performed on the dataset to remove columns with high cardinality, missing values or null values and irrelevant features. After data cleansing, the training set was split into two. The first part was used to train the light gradient model explained in the section. The second part was used to obtain the performance metrics. Then the model was used to predict a machine's likelihood of getting attacked by malware as shown in Figure 1.

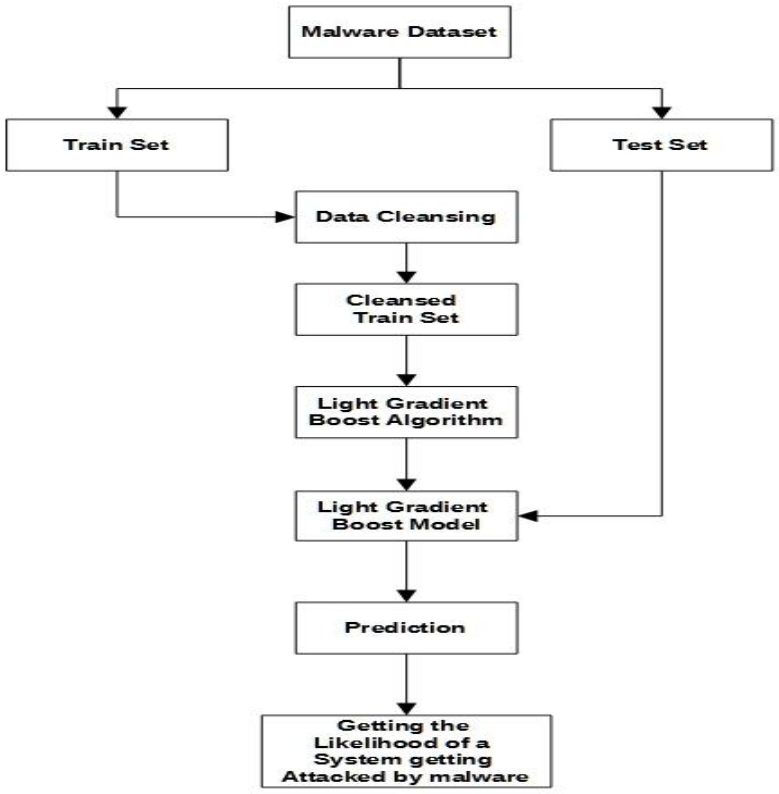

Figure 1: Framework of Prediction Model

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

3.2 Dataset Acquisition and Preprocessing

The dataset is divided into two sets, the training set and the testing set, and the dataset holds many properties of dissimilar computer hardware and software-related information. It accommodates a total of over eight million rows and eighty-three columns. Each row is set aside for a unique machine identifier. The other columns supply different software and hardware specifications of the machine.

Table 1:Features of the Dataset

The dataset holds many features and some of features are shown in Table 1.

S/N	Feature	Description	Used in Training/Testing
1	HasDetections	It tells whether the machine is affected by malware or not	Yes
2	MachineIdentifer	Individual machine ID	Yes
3	TPM (Trusted Protection Module)	Shows whether the machine has TPM capabilities	Yes
4	Platform Identifier	OS properties and microprocessor info	No
5	City	City identifier where the machine is accessed	No
6	Country	Country identifier from where the machine is accessed	No
7	OS Version	Version of the Operating System	Yes
8	Is protected	Shows if the system is protected or not. The field holds if at least one of the installed antiviruses is up to date.	Yes
9	Census TotalPhysicalRAM	Details of physical RAM	No
10	Firewall Details	It shows the firewall information and holds if the firewall is active	Yes
11	isVirtualDevice	Virtual devices hosting on the specific machine	Yes
12	Secure Boot	Shows if secure boot is enabled or not	Yes
13	Primary Disk Type	Shows Information about the primary disks, e.g., HDD, SSD.	Yes

The features contained in the dataset are not limited to Table 1.

3.2.1 Analysis of Data

The dataset has three types of columns contained in it. The Numerical, Binary and Categorical Types as described in Figure 3. The numerical column deals with all entries that have a numerical value. The Binary column deals with all entries in the column that either have a ‘0’ or a ‘1’ as an entry. An entry with ‘0’ means ‘no’ and an

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

entry with '1' means 'yes'. The categorical column type holds every other kind of entry. The column type distribution in the dataset was gotten using the following Python script and is shown in Figure 2

```
In [6]:
#Column Type Distribution
total = train.shape[0]
missing_df = []
cardinality_df = []
for col in train.columns:
    missing_df.append([col, train[col].count(), total])
    cardinality = train[col].nunique()
    if cardinality > 2 and col != 'MachineIdentifier':
        cardinality_df.append([col, cardinality])

missing_df = pd.DataFrame(missing_df, columns = ['Column', 'Number of records', 'Total']).sort_values("Number of
cardinality_df = pd.DataFrame(cardinality_df, columns = ['Column', 'Cardinality']).sort_values("Cardinality", asc
type_df = [['Binary columns', len(binary_columns)],['Numerical Columns', len(true_numerical_columns)], ['Categori
type_df = pd.DataFrame(type_df, columns = ['Type', 'Column count']).sort_values('Column count', ascending = True)

f, ax = plt.subplots(figsize = (7, 7))
sns.barplot(x = "Column count", y = "Type", data=type_df, label="Missing", palette = 'Spectral')
plt.show()
```

Figure 2: Code Snippet for Showing Column Distribution

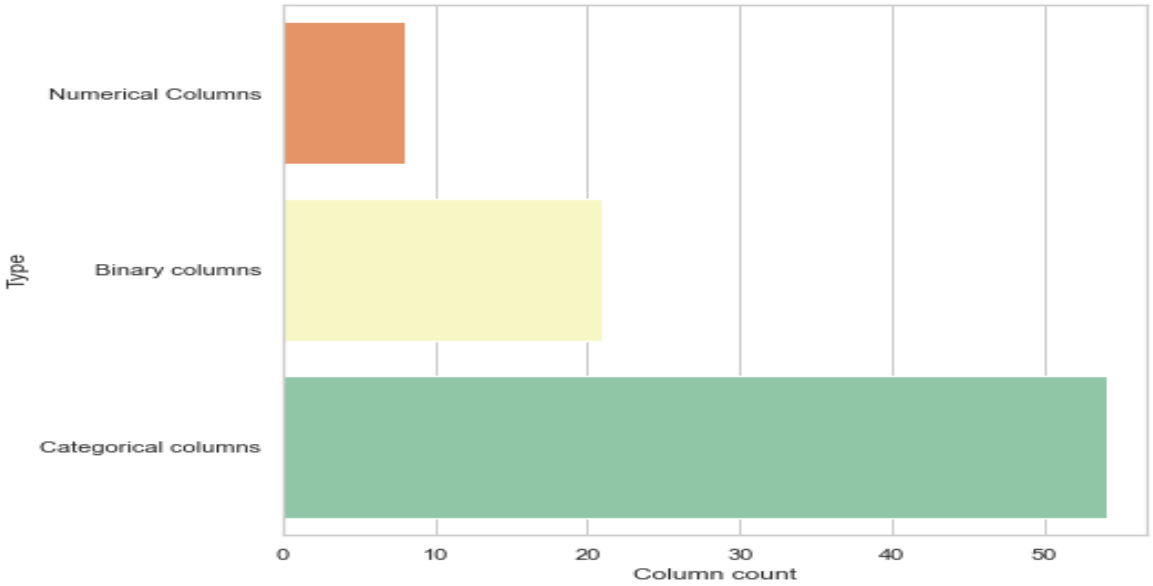

Figure 3: Column Type Distribution

As seen in Figure 3, the graph gives more categorical columns than there are binary and numerical.

3.2.2 Null Values

In the dataset, there are some columns with null values. These were visualized using the code snippet as discussed in Figure 4. The graphical representation is shown in Figure 5.

```
In [7]:
#Null values count
f, ax = plt.subplots(figsize=(10, 15))
sns.set_color_codes("pastel")
sns.barplot(x = "Total", y = "Column", data = missing_df, label = "Missing", color = "b")
sns.barplot(x = "Number of records", y = "Column", data = missing_df, label = "Existing", color = "orange")
ax.legend(ncol = 2, loc = "upper right", frameon = True)
plt.show()
```

Figure 4: Null Values Count Snippet

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

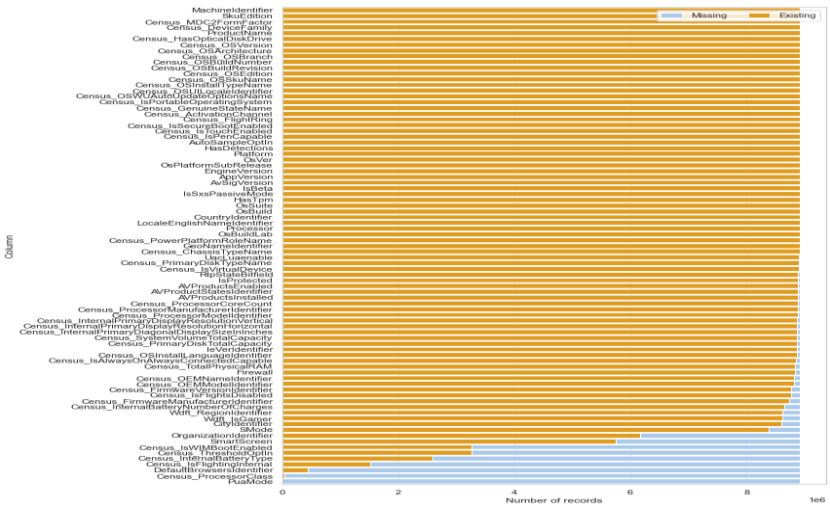

Figure 5: Null Values Count Graph

From Figure 4, it shows that there are seven features with more than 50% missing values.

3.2.3 Column Cardinality

The cardinality of a column refers to the number of times an entry appears in that column. The cardinality of each column is shown using the code snippet as shown in Figure 6. The result is shown in Figure 7

```
In [9]: #Column Cardiality
        f, ax = plt.subplots(figsize = (10, 15))
        sns.set_color_codes("pastel")
        sns.barplot(x = "Cardinality", y = "Column", data = cardinality_df, label = "Existing", color = "black")
        plt.show()
```

Figure 6: Column Cardinality Snippet

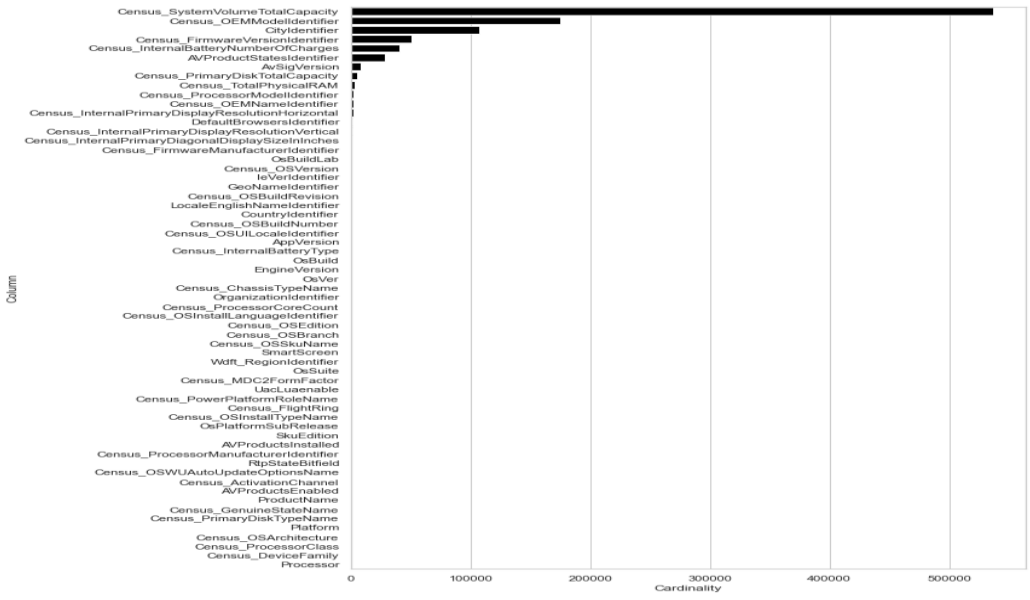

Figure 7: Column Cardinality Graph

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

Some of the features have too many categorical values (high cardinality), which can be a problem for some models. As seen from the graph above, it can be deduced that the column with the highest cardinality is Census_System Volume Capacity, followed by Census_OEM Model Identifier and so on.

RESULTS AND DISCUSSION

4.1 Result for Implementation of the Model on the Test Split

The test set was loaded into Jupiter after which the model was applied to it for the prediction of the likelihood of a computer being infected by malware. The process is illustrated using Figure 8 and the outputs are shown in Figure 9

```
In [68]:
#Pre-proccess Test
submission = pd.DataFrame({"MachineIdentifier":test['MachineIdentifier']})
test.drop('MachineIdentifier', axis=1, inplace=True)

for col in categorical_columns:
    test[col] = indexer[col].get_indexer(test[col])
```

```
In [70]:
#Make predictions
predictions = model.predict(test, num_iteration=model.best_iteration, predict_disable_shape_check = True)
```

```
In [71]:
#Output results
submission["HasDetections"] = predictions
submission.to_csv("submission.csv", index=False)
submission.head(10)
```

Figure 8: Prediction on the Test set

Out[71]:

	MachineIdentifier	HasDetections
0	0000010489e3af074adeac69c53e555e	0.41
1	00000176ac758d54827acd545b6315a5	0.41
2	0000019dcefc128c2d4387c1273dae1d	0.40
3	0000055553dc51b1295785415f1a224d	0.42
4	00000574cefffeca83ec8adf9285b2bf	0.39
5	000007ffedd31948f08e6c16da31f6d1	0.42
6	000008f31610018d898e5f315cdf1bd1	0.40
7	00000a3c447250626dbcc628c9cbc460	0.36
8	00000b6bf217ec9aef0f68d5c6705897	0.40
9	00000b8d3776b13e93ad83676a28e4aa	0.40

Figure 9: Likelihood of each machine identifier being attacked by Malware

4.2 Result of Data Cleansing

The first step conducted in the pre-processing of the dataset involved the removal of the columns with high cardinality. This was achieved using the code snippet as shown in Figure 10. The columns that were dropped are represented in Figure 11.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

```
#Remove Columns with High Cardinality
high_cardinality_columns = [c for c in categorical_columns if train[c].nunique() > 500]
high_cardinality_columns.remove('MachineIdentifier')
train.drop (high_cardinality_columns, axis = 1, inplace = True)
print('Columns with high cardinality: ', high_cardinality_columns)
```

Figure 10: Drop Columns with High Cardinality Snippet

```
Columns with high cardinality:  ['AvSigVersion', 'DefaultBrowsersIdentifier', 'AVProductStatesIdentifier', 'CityI
dentifier', 'OsBuildLab', 'Census_OEMNameIdentifier', 'Census_OEMModelIdentifier', 'Census_ProcessorModelIdentifi
er', 'Census_FirmwareManufacturerIdentifier', 'Census_FirmwareVersionIdentifier']
```

Figure 11: Columns with High Cardinality

Unwanted columns and columns with more than 40% missing values were also removed. This was achieved using the following code snippet in Figure 12.

In [12]:
```
#Remove Colums with more than 40% missing data
high_null_columns = [c for c in train.columns if train[c].count() < len(train) * 0.6]
train.drop(high_null_columns, axis = 1, inplace = True)
print('Columns with more than 40% null values: ', high_null_columns)
```

```
Columns with more than 40% null values:  ['PuaMode', 'Census_ProcessorClass', 'Census_InternalBatteryType', 'Cens
us_IsFlightingInternal', 'Census_ThresholdOptIn', 'Census_IsWIMBootEnabled']
```

In [13]:
```
#Remove Unwanted Columns
unwanted_columns =['MachineIdentifier']
train.drop(unwanted_columns, axis = 1, inplace = True)
```

Figure 12: Processing the Dataset for Light Gradient Boost Model

4.3 Train/validation of Random Split

The training dataset was divided into train and validation for the Light Gradient Boost Model using the code snippet as discussed in Figure 13. The model was ranked and visualized using the code snippet as shown in Figure 14.

In [22]:
```
# Get labels
labels = train['HasDetections']
train.drop('HasDetections', axis=1, inplace=True)
```

In [23]:
```
X_train, X_val, Y_train, Y_val = train_test_split(train, labels, test_size=0.15,random_state=1)

binary_columns, true_numerical_columns, categorical_columns = update_feature_lists()
```

In [49]:
```
# Label encoder
indexer = {}
for col in categorical_columns:
    _, indexer[col] = pd.factorize(X_train[col])

for col in categorical_columns:
    X_train[col] = indexer[col].get_indexer(X_train[col])
    X_val[col] = indexer[col].get_indexer(X_val[col])
```

Figure 13: Code Snippet for Light Gradient Boost Model

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

In [25]:
```
params = {'num_leaves': 60,
          'min_data_in_leaf': 100,
          'objective':'binary',
          'max_depth': -1,
          'learning_rate': 0.1,
          "boosting": "gbdt",
          "feature_fraction": 0.8,
          "bagging_freq": 1,
          "bagging_fraction": 0.8 ,
          "bagging_seed": 1,
          "metric": 'auc',
          "lambda_l1": 0.1,
          "random_state": 133,
          "verbosity": -1}
```

Figure 14: Train/Validation Split Snippet

4.4 Model Training

The code snippet for the training of the Light Gradient Boost model is shown in Figure 15. The importance of each feature to the building is discussed in Figure 16

In [54]:
```
model = lgb.train(params, lgb_train, 10000, valid_sets=[lgb_train, lgb_val], early_stopping_rounds=200, verbose_e
```

```
Training until validation scores don't improve for 200 rounds
[100]   training's auc: 0.740788        valid_1's auc: 0.559059
[200]   training's auc: 0.745803        valid_1's auc: 0.556957
Early stopping, best iteration is:
[11]    training's auc: 0.72079 valid_1's auc: 0.57865
```

Figure 15: Training the LGB Model Snippet

In [55]:
```
#Model feature importance
lgb.plot_importance(model, figsize=(15, 10))
plt.show()
```

Figure 16: Model of Feature

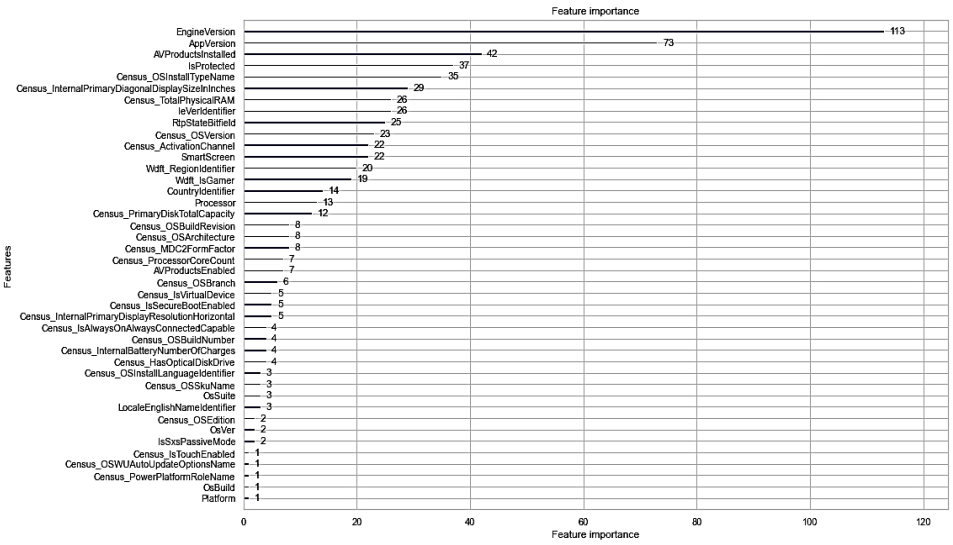

Figure 15: Importance of Feature in the Model

As seen in the graph above, the most important feature that was used by the model is the Engine Version column.

4.6 Performance Evaluation Metrics

The metrics that were used in evaluating the model are shown in Figure 16 and also Figure 17 indicates the confusion matrix metrics.

```
-----Train-----
                    precision    recall  f1-score   support

HasDetections = 0       0.48      0.98      0.65   1409602
HasDetections = 1       0.52      0.02      0.04   1510157

          accuracy                           0.48   2919759
         macro avg       0.50      0.50      0.34   2919759
      weighted avg       0.50      0.48      0.33   2919759

-----Validation-----
                    precision    recall  f1-score   support

HasDetections = 0       0.48      0.98      0.65    249299
HasDetections = 1       0.53      0.02      0.04    265953

          accuracy                           0.48    515252
         macro avg       0.50      0.50      0.34    515252
      weighted avg       0.51      0.48      0.33    515252
```

Figure 16: Evaluation Metrics for the Model

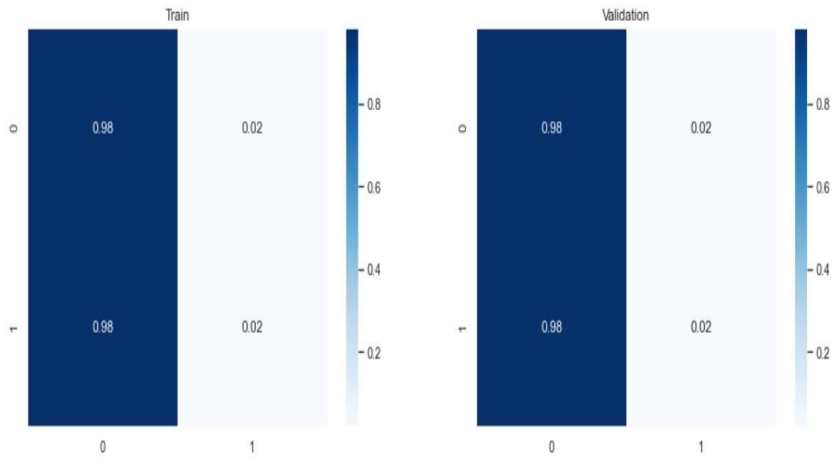

Figure 17: Confusion Matrix of LGBM

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume IX Issue VII July 2024

The True Positive (TP) of the train split = 0.98

The False Positive (FP) of the train split = 0.02

The False Negative (FN) of the train split = 0.02

The False Negative (TN) of the train split = 0.98

The accuracy of the model on the train split = 0.98.

The precision is calculated thus:

$$Precision = \frac{TP}{(TP+FP)} \quad (1)$$

Therefore, the precision of the model on the train split = 0.98.

The recall is calculated thus:

$$Recall = \frac{TP}{(TP+FN)} \quad (2)$$

Therefore, the recall of the model on the train split = 0.98.

To have a joint effect of the Precision and Recall, the F1 Score is calculated. The F1 score is the harmonic mean of the Precision and Recall.

The F1 Score is calculated thus:

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (3)$$

Therefore, the F1 score = 0.98%.

CONCLUSION

The Light Gradient Boost Algorithm was utilized in experimenting and the outcome was that each machine was given a value standing for its likelihood of being attacked by malicious software. From the result, it can be deduced that the likelihood of a machine getting infected by malware increased as the value approached 1. Based on the result obtained, the light Gradient Boost Algorithm had an accuracy of 98%**.**

REFERENCES

1. W. Niu and M. Fan, “Control and Research of Computer Virus by Multimedia Technology,” Int. J. Inf. Syst. Supply Chain Manag., vol. 17, no. 1, pp. 1–17, 2023, doi: 10.4018/IJISSCM.333896.
2. A. K. S. Vijayanand. C. D, “Impact of Malware in Modern Society Impact of Malware in Modern Society Abstract :,” J. Sci. Res. Dev., vol. 2, no. 3, pp. 593–600, 2019.
3. Q. Pan, W. Tang, and S. Yao, “The application of LightGBM in microsoft malware detection,” J. Phys. Conf. Ser., vol. 1684, no. 1, pp. 1–7, 2020, doi: 10.1088/1742-6596/1684/1/012041.
4. S. Chandrakant Narwade and N. S. Ratnaparkhi, “An Overview Paper on Data Mining Techniques and Applications,” Int. J. Classif. Res. Tech. Adv. Multidiscip. Peer-Reviewed Online Open Access J. Impact Factor 5, vol. 455, no. 3, pp. 48–53, 2023, [Online]. Available: www.ijcrta.org
5. C. Okoro, J. W. Etukudo, and C. J. Obizuo, “Impact of Financial Technology o n Financial Institutions ’ Performance . Evidence From Nigerian Commercial Banks,” J. Account. Financ. Manag., vol. 10, no. 3, pp. 1–25, 2024, doi: 10.56201/jafm.v10.no3.2024.pg111.134.
6. S. Gheware, A. S. Kejkar, and S. M. Tondare, “Data Mining : Task , Tools , Techniques and Applications,” Int. J. Adv. Res. Comput. Commun. Eng., vol. 3, no. 10, pp. 1–5, 2014, doi:

10.17148/IJARCCE.2014.31003.

7. I. Sabah, L. L. Nahmatwlla, and W. A. Ahmed, "Role of AI and Machine Learning Based on Principle of Web Technology and Cloud Computinmg," J. Biomech. Sci. Eng., pp. 117–141, 2023, doi: 10.17605/OSF.IO/4A9MH.
8. N. Z. Gorment, A. Selamat, L. K. Cheng, and O. Krejcar, "Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions," IEEE Access, vol. 11, pp. 141045–141089, 2023, doi: 10.1109/ACCESS.2023.3256979.
9. Muhammad Shoaib Akhtar and Tao Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," Symmetry (Basel)., vol. 14, pp. 1–11, 2022.
10. S. Franjić, "Cybercrime is very dangerous form of criminal behavior and cybersecurity," Emerg. Sci. J., vol. 4, no. Special Issue, pp. 18–26, 2020, doi: 10.28991/esj-2020-SP1-02.
11. S. Talukder and Z. Talukder, "A Survey on Malware Detection and Analysis Tools," Int. J. Netw. Secur. Its Appl., vol. 12, no. 2, pp. 37–57, 2020, doi: 10.5121/ijnsa.2020.12203.
12. C. Eze, A. O & Chukwunonso, "Malware Analysis and Mitigation in Information Preservation," J. Comput. Eng., vol. 20, no. 4, pp. 53–62, 2018, doi: 10.9790/0661-2004015362.
13. P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on Windows malware detection: Techniques, research issues, and future directions," J. Syst. Softw., vol. 209, p. 111921, 2024, doi: 10.1016/j.jss.2023.111921.
14. C. Avci, B. Tekinerdogan, and C. Catal, "Analyzing the Performance of Long Short-Term Memory Architectures for Malware Detection Models," Concurr. Comput. Pract. Exp., vol. 35, no. 6, p. 1, 2023, doi: 10.1002/cpe.7581.
15. Z. Chen, S. Xing, and X. Ren, "Efficient Windows malware identification and classification scheme for plant protection information systems," Front. Plant Sci., vol. 14, no. February, pp. 1–12, 2023, doi: 10.3389/fpls.2023.1123696.
16. S. Aurangzeb, R. N. Bin Rais, M. Aleem, M. A. Islam, and M. A. Iqbal, "On the classification of Microsoft-Windows ransomware using hardware profile," PeerJ Comput. Sci., vol. 7, pp. 1–24, 2021, doi: 10.7717/peerj-cs.361.