Web Based Malware & Virus Prediction System Using Machine Learning

BY

Tanvir Rezwan
ID: 191-15-2522

Mominul Islam
ID: 191-15-2624

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Fatema Tuj Johora
Senior Lecturer
Department of CSE
Daffodil International University

Co-Supervised By

Md Mahfujur Rahman
Senior Lecturer
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

23 January 2023

APPROVAL

This Project/internship titled **“Web Based Malware & Virus Prediction System Using Machine Learning”**, submitted by Tanvir Rezwan, Mominul Islam ID No: 191-15-2522, 191-15-2624 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on *23-01-2023*.

DECLARATION

We hereby declare that, this project has been done by us under the **supervision of Ms. Fatema Tuj Johora and co-supervision of Md Mahfuzur Rahman, Department of CSE** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

Fatema

____

Ms. Fatema Tuj Johora
Senior lecturer
Department of CSE
Daffodil International University

Co-Supervised by:

____

Md Mahfujur Rahman
Senior Lecturer
Department of CSE
Daffodil International University

Submitted by:

[illegible]

____

TANVIR REZWAN
ID: 191-15-2522
Department of CSE
Daffodil International University

Mominul

____

MOMINUL ISLAM
ID: 191-15-2624
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Supervisor Ms. Fatema Tuj Johora and Co-Supervisor Md Mahfujur Rahman, Senior Lecturer**, Department of CSE, Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of "Machine Learning, Data Mining, Deep learning, Artificial Intelligence and Computer Networking" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Prof. Dr. Touhid Bhuiyan** and Head**,** Department of CSE, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

Malware-based cyber-attacks are increasing and have advanced in sophistication across practically all networks. Attacks using sophisticated malware are also the most complex, which makes it difficult to detect them. Advanced malware can hide many of its traces using various techniques, including metamorphic engines causing significant financial and privacy losses to a variety of organizations As a result, malware analysis methods now face a considerable barrier in predicting and detecting such threats. With the tremendous technological development and cutting-edge research, researchers and anti-virus organizations have begun using machine learning and deep learning methods for malware analysis and detection. Through this proposed work we have proposed a system to detect the malware on Windows systems. According to the types of features, appropriate algorithm is chosen to check the malware and virus. Total 4 different algorithms named LightGBM, XGboost, and Logistic Regression, CatBoost Regressor is used here. To handle the missing values, skewness check, correlation test, feature engineering is applied with accuracy of 74%. For practical feeling of the proposed system an E-commerce website also designed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Since all current malware applications tend to have multiple polymorphic layers to avoid detection or to use side mechanisms to automatically update themselves to a newer version at short intervals of time in order to avoid detection by any antivirus software, malware detection through standard, signature-based methods is becoming increasingly challenging. The interested reader can view an example of dynamical file May 18, 2022 DRAFT analysis for malware identification through virtual environment emulation. computer. Le infectors and stand-alone malware make up a straightforward taxonomy of malware. One or more of malware's goals can be to gain access to private networks, steal confidential information, commandeer computer systems to use their resources, or interfere with communication or computing processes. Malware is purposefully malicious software that can spread throughout a network, remain undetected, and alter or destroy an infected machine or network, even when it is disguised as genuine software from an apparently reliable source. Other goals can be to disrupt services for certain corporations, steal identities, or even wage cyberwar against another state. Entrepreneurs, businesses, big corporations, or even governments could be the victims. Malware commonly comes in the form of worms, trojan horses, adware, ransomware, etc. There is a description of traditional techniques for finding metamorphic viruses in. A summary of the various machine learning techniques that have been suggested for malware detection is provided. Here, we provide a few examples of these techniques in use. In, it is discovered that boosted decision trees working with engrams outperform the Naive Bayes classifier and Support Vector Machines in terms of output. use automatic association rule extraction on Windows API execution sequences to differentiate between malicious and legitimate software files. Additionally employing association rules, but with honeytokens with well-known parameters. To determine whether a specific program file is (or is not) a variant of a prior program file, hidden Markov models are utilized. Uses Profile Hidden Markov Models to accomplish a similar objective, they have been successfully employed in the past for bioinformatics sequence analysis. This article explores neural networks' ability to recognize polymorphic malware. Self-Organizing Maps are utilized in Windows exactable files to find patterns of activity for infections. In this essay.

1.2 Motivation

Cybercriminals seem to always remain one step ahead of the curve, despite major advancements in malware prediction, prevention, and cybersecurity in general. With more people and businesses depending on the internet, both the volume and intensity of cyberattacks have only grown. According to projections, the cost of cybercrime would reach $6 trillion USD globally by 2021. The fight against malware may not have a promising future for humanity, but thanks to the many brilliant brains working on cybersecurity, the harm caused by malware may soon be lessened.

1.3 Rationale of the Study

Malware assaults are one of the growing problems for consumers and companies utilizing Windows systems in terms of data security and integrity. Microsoft Windows OS dominates the market with an 87% market share, according to NetMarketShare (source: Wikipedia), so the biggest challenge for Microsoft is to provide antivirus defense or other security products that accurately and efficiently determine whether a machine is infected with malware or not in order to keep their data secure and remain immune to such attacks. I therefore created a machine learning algorithm that predicted the likelihood that a machine would be infected with malware.

1.4 Research Questions

1. Does your system use sample data to forecast a real output?

2. Can it use a machine learning technique to categorize malware-benign in provided data/file?

3. Do all algorithms operate flawlessly (yes/no)?

4. Do we know the valid accuracy rate to predict the malware?

I am confident that our machine can anticipate accurate results based on prior Microsoft Windows Defender as well as customer satisfaction because we trained it with a huge number of data sets and acquired good accuracy.

We achieved the best result using LGBM algorithm also get similar accuracy using XGB, Logistic Regression, and CatBoost algorithm.

1.5 Expected Outcome

In our malware predict system, given data from our dataset is used to generate an expected result. Here, we combined test and training data to produce a more accurate result. According to our findings, the accuracy completely depends on the training data set. After all of our system's necessary steps have been completed, our machine learning system will be prepared. We have used a variety of tactics in order to obtain accurate results. We have got almost 74% accuracy from “LGBM”, 71% accuracy from “XGB”, 72% accuracy from “Logistic Regression”, 73% accuracy from “CatBoost Regressor” which we applied to this system.

1.6 Report Layout

Chapter 1 In this section, we discussed the motivation behind our work as well as its goals and expected outcomes.
Chapter 2 describes the theoretical underpinnings of our research and examines previous comparable publications, studies, and the scope of problems and challenges.

Chapter 3 we discuss our research topic, the tool we used, how we collected the data, how we analyzed the data statistically, and how we put it all into practice.

Chapter 4 In this chapter, we present the results of our research experiments, descriptive analyses, and conclusions.

Chapter 5 This chapter includes a summary of the forecast and conclusion as well as additional study methods.

CHAPTER 2
BACKGROUND

2.1 Introduction

Malware is created to be aggressive, persistent, and hostile. This attack, which tries to harm the system, compromise it, partially take over some activities, or entirely shut it, targets workstations, networks, platforms, devices, and smart applications. It interferes with routine activity, just way the flu virus does. A number of machine learning algorithms have been developed to help with the problem of predicting malware assaults. Currently, the Light Gradient Boosted Machine classification technique is the one that is most anticipated to boost in find malware (LGBM). Auto AI is used by several large organizations to protect their systems. Malware assaults are one of the growing problems for consumers and companies utilizing Windows systems in terms of data security and integrity. Microsoft Windows OS currently holds a market share of 87%, according to NetMarketShare firm (source: Wikipedia), which makes providing antivirus defense or other security products that accurately and efficiently determine whether a machine is infected with malware or not Microsoft's biggest challenge in order to keep their data secure and remain impervious to such attacks. Microsoft challenged the data science community to create machine learning systems that can assess or anticipate whether a machine would soon be infected by malware as part of their larger business plan. The aim of this project is to reach everyone easily by an E-Commerce website and achieve an increase accuracy on the Microsoft Kaggle's Malware Prediction dataset.

2.2 Related Works

[1] The authors of (LIN, 2019) described a technique that utilized Kaggle's Microsoft Malware Prediction dataset and a Naive Transfer Learning strategy. Based on the training data, they trained a Gradient Boosting Machine (GBM) to create a straightforward

prediction model, which they subsequently adjusted for the test dataset. The authors identified the critical elements for domain adaptation, attempted to reduce the marginal distribution gap between the source and target domains, and modified the predictions' outcomes in accordance with the broad statistical regularities retrieved from the training set. 20 of the most crucial category elements were chosen for additional research after they conducted a GBM to gather each feature's relevance ratings. To make the issue simpler and cut down on computing expenses, this was done. The initial model, which employed 20 of the most crucial features, had a 63.7% accuracy rate. A second model with accuracy of 64.3% deleted the columns with the highest mean disagreement.

[2] The author of (Stephan Michaels, 2019) suggested a technique for virus prediction. Using LGBM, two models were trained and assessed. The dataset was cleaned and string values were encoded using one technique. A LightGBM was then trained. The second approach added new features to the first model's preprocessed data. Then, a LightGBM was trained using the relevant properties that were chosen. The results of both models' forecasts were then averaged. Because the author did not report findings using data from Microsoft Malware Prediction, we repeated the experiment (Microsoft, 2018). Large amounts of data that need to be examined for potential malicious intent are one of the major issues anti-malware solutions currently face.

[3] In (Rai and Mandoria, 2019), they chose to apply the Gradient Booster technique to a different malware challenge because it performed better than all other well-known algorithms at predicting malware.

[4] On 600 million computers throughout the world, Microsoft's real-time anti-malware detection application is installed (Caparas, 2020). The issue of anticipating malware attacks has been addressed by a variety of machine learning techniques.

[5] The most widely used classification method for finding malware right now is called Light Gradient Boosted Machine (LGBM). Some advantages of LGBM include simplicity, ease of creation, and ease of understanding (Vinayakumar et al., 2019). Additionally, many

large corporations employ AutoAI to safeguard their systems due to the rise in malware risks.

[6] More than 2.5 quintillion bytes of data are created and captured daily by human activity. Over 90% of the data, or 40 Zettabytes, or 40 trillion gigabytes, was created in the last two years (Dobre and Xhafa, 2014).

[8] (Choudhary and Kesswani, 2020) the authors tested classifiers for network intrusion using the NSL KDD dataset, including XG-Boost and LGBM. The dataset is composed of 41 features, including 21 classifications of assault, as well as fundamental features, traffic features, and content features. According to the authors' experimental findings, stacked ensembles of Gradient Boosting Decision Trees performed better than linear models and deep neural networks. Similar to prior relevant work, i would like to assess such methods on a more contemporary infection challenge because ensemble methods performed better than linear models and a deep neural network.

[9] In (Nov 2020) the author jugaloza worked with the same Microsoft data set to predict malware using LightGB, XGB algorithm and get 72% accuracy.

2.3 Research Summary

Basically, in our 'Web Based Malware & Virus Prediction System Using Machine Learning' Project have two parts. At first in our E-Commerce site customers/users will be able to order our advertised product like single/multiple file prediction, whole pc file prediction etc. Then in Second part, our created Malware & Virus Prediction System with Machine Learning will scan or predict those file that how much virus or benign present. And the result will be told to the customer by our E-Commerce site. We created our website with Django framework. Four algorithms were employed to determine the accuracy. The system will function better the higher the accuracy.

2.4 Scope of the Problem

In our E-commerce site users will maybe face to upload their file because we haven’t completed the site development yet. Also they will face to show the final result in the website. In this case we will send them the result through their E-mail address. To generate and find the result of virus-benign will take lengthy time because of using low configuration computer/laptop and lab studio.

2.5 Challenges

The two major obstacles to achieving our forecasting accuracy are target variable investigation and DF size reduction. Since reading enormous amounts of data as usual causes memory issues, we try to utilize the variable size. My DCL laptop's configuration is also inadequate for working with a large volume of data and implementing this project. My laptop takes a long time to load train/test csv files and examine data, which makes it difficult for me to focus on my assignment. During feature engineering, encoding, and model training, I encountered coding problems that took months to resolve. Then, with help from my manager and a few competent developers from GitHub and Kaggle, I finish my implementation. On my laptop, the complete program runs in about 10 to 11 hours. I'm less eager to conduct additional research for this topic as a result. If I get a supercomputer or laptop with a high configuration, I can boost the accuracy of this project. A variety of problems emerged as a result of the working method.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

We shall outline our study techniques and approach in this section. Additionally, this workshop will cover the research project's tools, data collecting, the research question, pre-processing, statistical analysis, and its application.
Figure 3.1 provides a quick overview of our E-commerce website process.

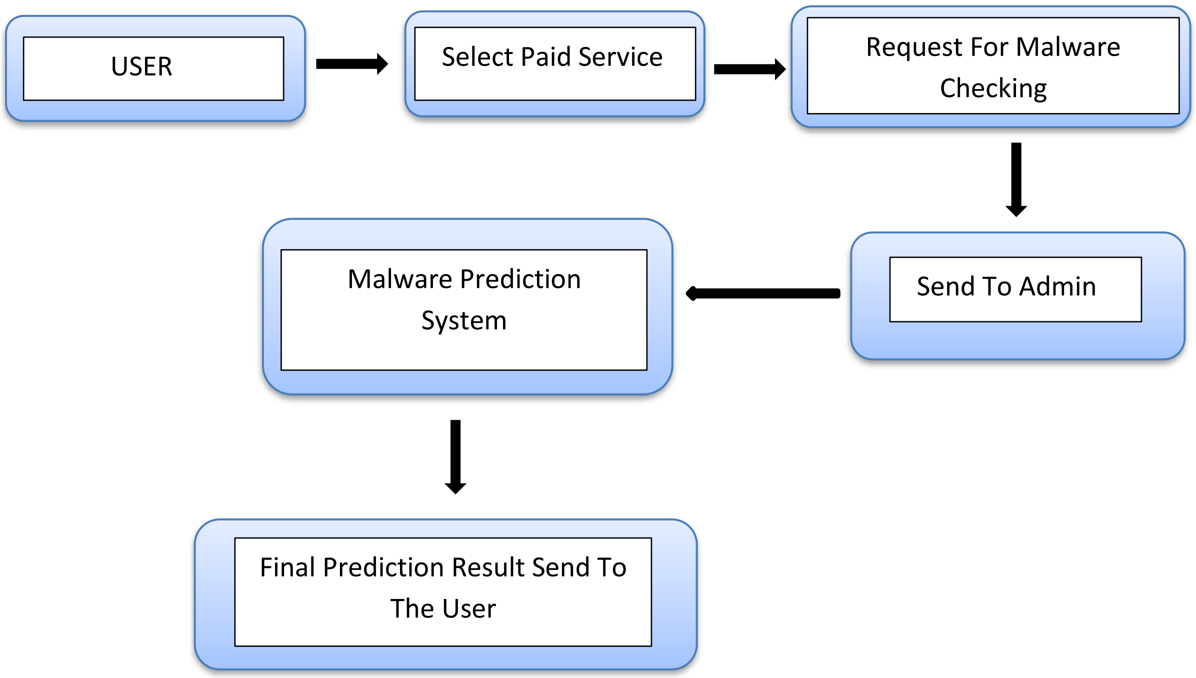

USER
Select Paid Service
Request For Malware Checking
Send To Admin
Malware Prediction System
Final Prediction Result Send To The User

Figure 3.1: Website Methodology at a Glance

Step 1 User Through internet anyone can visit our E-commerce site. Those who are suffering from malware/virus attack and want to check whether there is any malware in their file, can see our paid services list. Our website is user-friendly and all of the process is very easy. Everyone can visit our website.

Step 2 Select Service Customers can choose any service from a list of services. They can also cart a variety of services. They can add any service to the cart and check there order in order section. Then he will decide which service have to confirm.

Step 3 Request For Malware Checking Anyone with access to the internet can visit our e-commerce site. Those who have been the victim of a malware/virus attack and want to know if there is malware in their file can look at our paid services list. After that customer have to confirm the order and pay the service charge.

Step 4 Send To Admin After successfully paying for the services, the consumer will email our admin their file or data. By doing than that our team can access to the victim's files/data for malware checking.

Step 5 Malware Prediction System Now, our admin will use a malware prediction engine to predict the file or data and generate the results. We use the LGBM, XGB, Logistic Regression, and CatBoost Regressor models here. However, in order to save memory and time, I chose the LGBM algorithm for the model. To train my model, I used the tree-based gradient boosting framework lightgbm. There are 83 features in our training data set. among them, we chose five features for feature engineering, including "EngineVersion," "AppVersion," "AvSigVersion," "OsBuildLab," and "Census OSVersion." After performing exploratory data analysis and feature engineering, we can see in figure 3.2 Feature Importance that our selected features performed best among all features with LGBM.

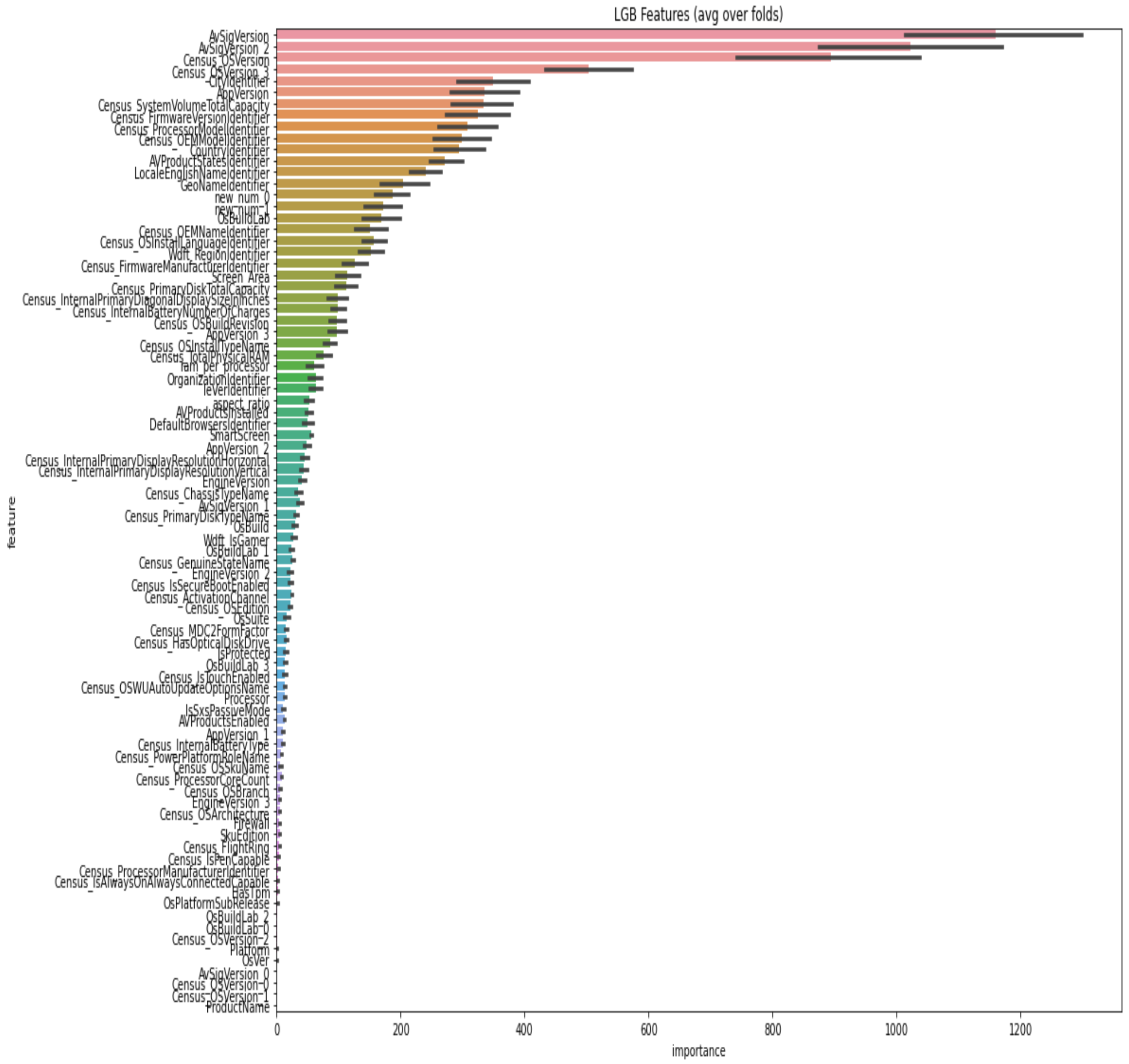


Figure 3.2: Feature Importance

Step 6 Final Prediction Result Send To Customer After generating final malware prediction result it will be sent to the paid customer through email.

3.2 Research Subject and Instrumentation

We see that the examination's core component is information. Finding spectacular data and a fantastic method or model for our investigation work is a really hard task for a specialist. We also need to look at relevant exam papers. At that time, we must decide between a few options:

-Which information needs to be gathered?

-How can we make sure that everyone will hear about our project?

-How should an e-commerce website be set up?

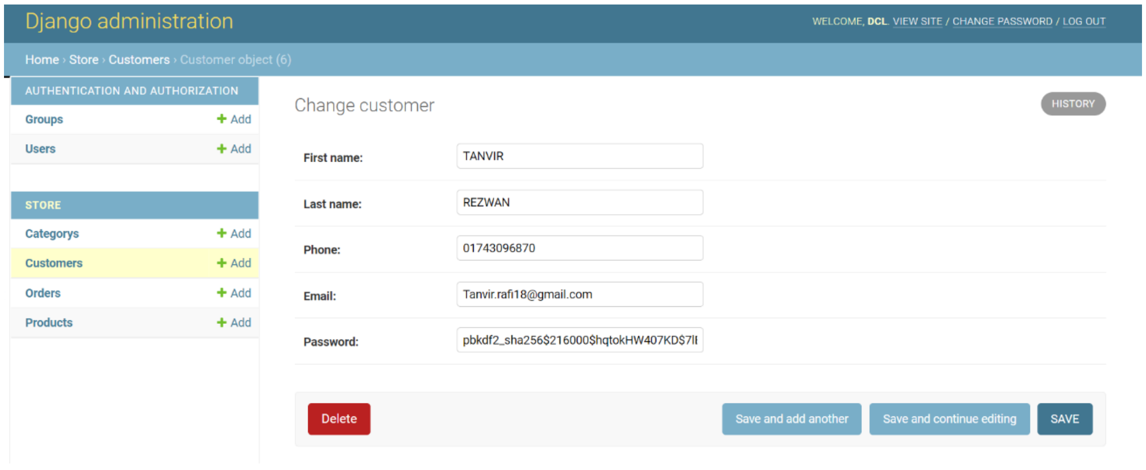
-How do I shrink a Data Frame?

3.3 Website Design and working Activities

In this section you can see our Trello sprint of website design and working activities. The key features of our E-commerce site are :

1. password hashing.
2. Session.
3. Cart.
4. Return to same service after adding cart.
5. Middleware.

Password Hashing Password hashing is the process of converting plaintext into an unintelligible series of numbers and letters by running it through a hashing algorithm (MD5, SHA, etc.). We use MD5 with the Django Framework in our e-commerce site. As we work on Windows security, we pay special attention to the security of our e-commerce site. Shown in figure 3.3.1


Django administration
WELCOME, DCL. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Store › Customers › Customer object (6)
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
STORE
Categorys + Add
Customers + Add
Orders + Add
Products + Add
HISTORY
Delete
Save and add another
Save and continue editing
SAVE

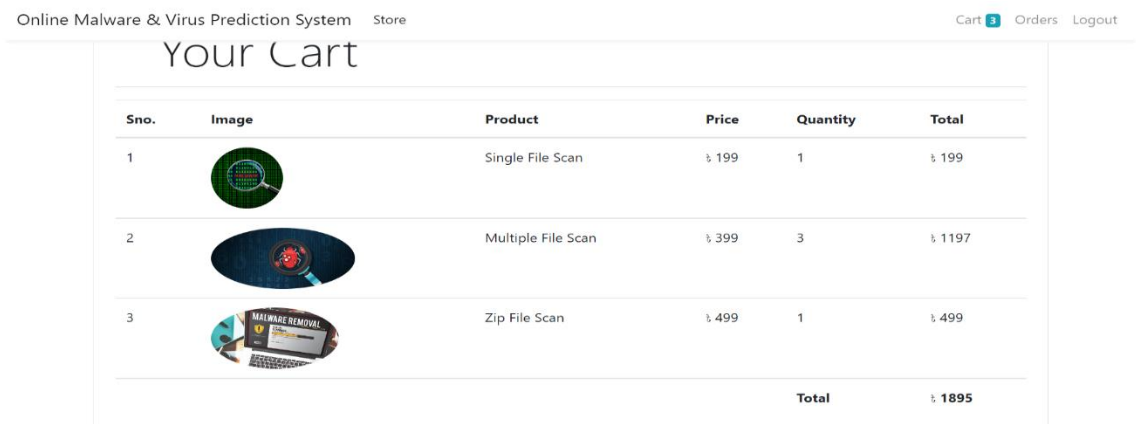
Change customer

First name:	TANVIR
Last name:	REZWAN
Phone:	01743096870
Email:	Tanvir.rafi18@gmail.com
Password:	pbkdf2_sha256$216000$hqtokHW407KD$7ll

Figure 3.3.1 Password Hashing

Session A session is a collection of user interactions with your website that happen over the course of a specific period of time. A single session, for instance, may include a number of website views, events, social interactions, and online purchases. In our E-commerce site session is used for to save information server-side while preserving user privacy in order to avoid keeping enormous volumes of data in-browser.

Cart A vital part of an ecommerce marketplace is the cart page. Users can easily check out by placing everything in their shopping cart and making an online payment on this page. To comprehend what a cart page does, picture a standard shopping cart in a store. In figure 3.3.1 shows our website cart page.



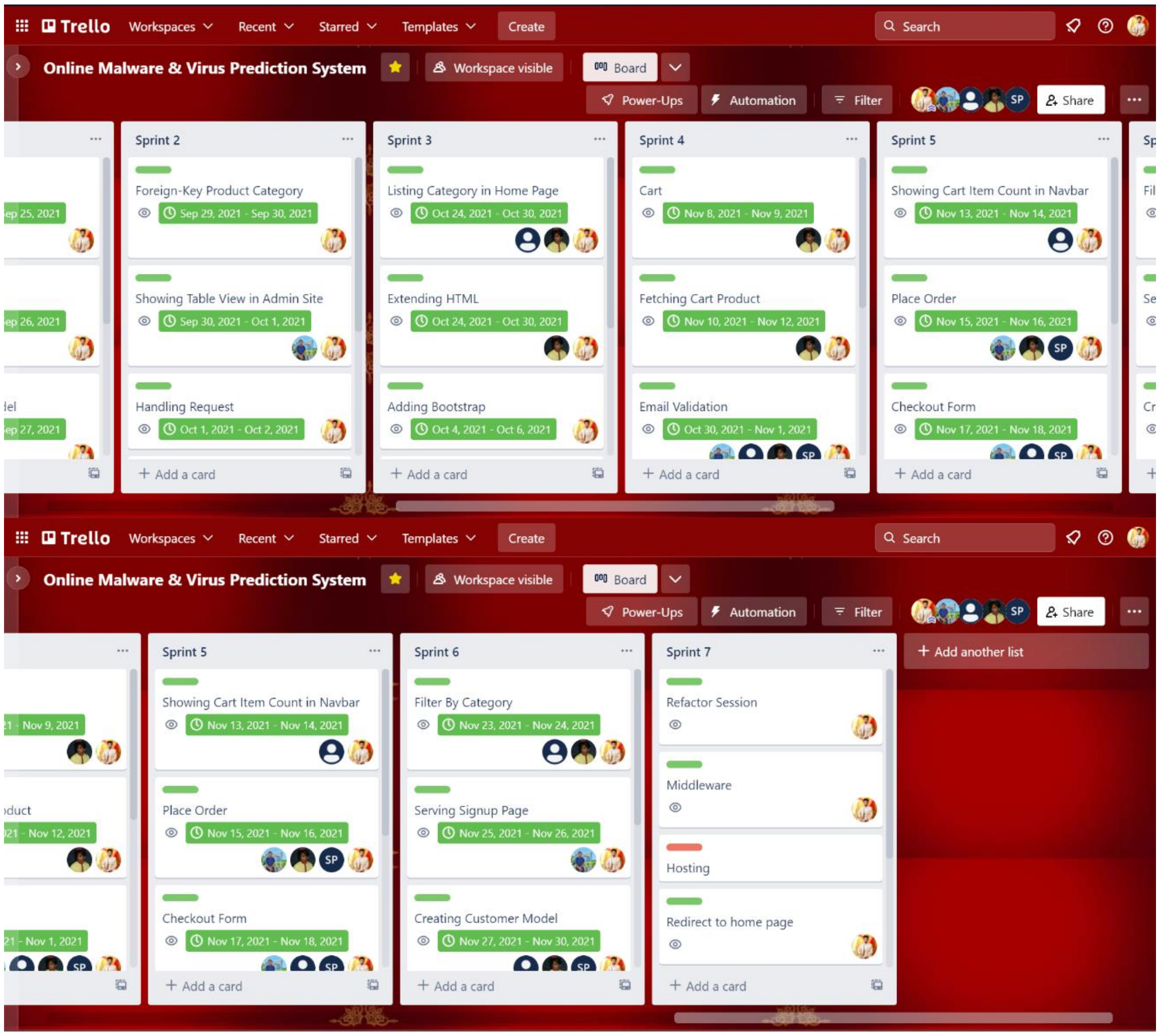
Your Cart

Sno.	Image	Product	Price	Quantity	Total
1		Single File Scan	৳ 199	1	৳ 199
2		Multiple File Scan	৳ 399	3	৳ 1197
3		Zip File Scan	৳ 499	1	৳ 499
				Total	৳ **1895**

Figure 3.3.2 Cart paid Services

Middleware Middleware is software and cloud services that provide common services and capabilities to applications and assist developers and operators in more efficiently building and deploying applications. Middleware serves as a layer between applications, data, and users. In our site middleware used as when user choose or request for any services the have to sign in or signup to our e-commerce site.

Trello Sprint of working Activities Trello is a list-making tool that runs on the web and is created by Trello Enterprise, an Atlassian subsidiary. Trello is a graphical work recording and project management tool that enables us to handle any kind of activity. Add paperwork, a to-do list, or even automation: Make everything specific to how we work best.



ep 25, 2021

ep 26, 2021

lel
ep 27, 2021

Sprint 2

Foreign-Key Product Category
Sep 29, 2021 - Sep 30, 2021

Showing Table View in Admin Site
Sep 30, 2021 - Oct 1, 2021

Handling Request
Oct 1, 2021 - Oct 2, 2021

Sprint 3

Listing Category in Home Page
Oct 24, 2021 - Oct 30, 2021

Extending HTML
Oct 24, 2021 - Oct 30, 2021

Adding Bootstrap
Oct 4, 2021 - Oct 6, 2021

Sprint 4

Cart
Nov 8, 2021 - Nov 9, 2021

Fetching Cart Product
Nov 10, 2021 - Nov 12, 2021

Email Validation
Oct 30, 2021 - Nov 1, 2021

Sprint 5

Showing Cart Item Count in Navbar
Nov 13, 2021 - Nov 14, 2021

Place Order
Nov 15, 2021 - Nov 16, 2021

Checkout Form
Nov 17, 2021 - Nov 18, 2021

Sp

Fil

Se

Cr

1 - Nov 9, 2021

oduct
21 - Nov 12, 2021

21 - Nov 1, 2021

Sprint 5

Showing Cart Item Count in Navbar
Nov 13, 2021 - Nov 14, 2021

Place Order
Nov 15, 2021 - Nov 16, 2021

Checkout Form
Nov 17, 2021 - Nov 18, 2021

Sprint 6

Filter By Category
Nov 23, 2021 - Nov 24, 2021

Serving Signup Page
Nov 25, 2021 - Nov 26, 2021

Creating Customer Model
Nov 27, 2021 - Nov 30, 2021

Sprint 7

Refactor Session

Middleware

Hosting

Redirect to home page

Figure 3.3.3: Trello sprint

3.4 Data Pre-Processing Procedure & Dataset

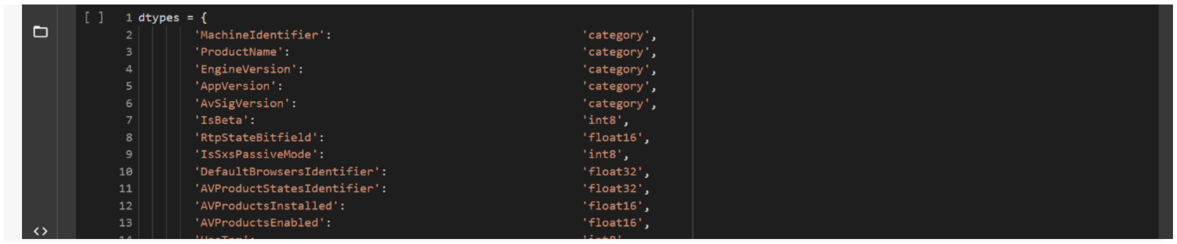
I chose the Microsoft Malware prediction problem statement from Kaggle, an online community of data scientists and machine learning practitioners that runs competitions to use machine learning and artificial intelligence to solve real-world problems. I obtained the files train.csv and test.csv directly from Kaggle, who held the competition.

Train.csv: - This file has 8921483 items, each of which corresponds to a computer that is uniquely identified by a MachineIdentifier. The ground truth of this file is the HasDetections field, which shows whether malware was found on the system. It has 83 columns, among them MachineIdentifier and HasDetections, which are used to train the model. Features, columns, and variables are comparable to one another in this context, so I'll use them all interchangeably in this article.

Test.csv: - There are 7853253 entries in this file. and, with the exception of HasDetections, has 82 columns, including MachineIdentifier.

3.4.1 Utilize Variable Size

We'll try to load the variables size can show in figure 3.4.1 because reading enormous amounts of data as usual leads to memory errors. We'll load the following variables: -Objects can be loaded as categories. -Binary data is converted to int8. -Binary values with missing values are converted to float16 (int does not comprehend nan); category may also be used in this situation. -The entire 64-bit encoding is altered to 32 bits, or 16 of the possible.

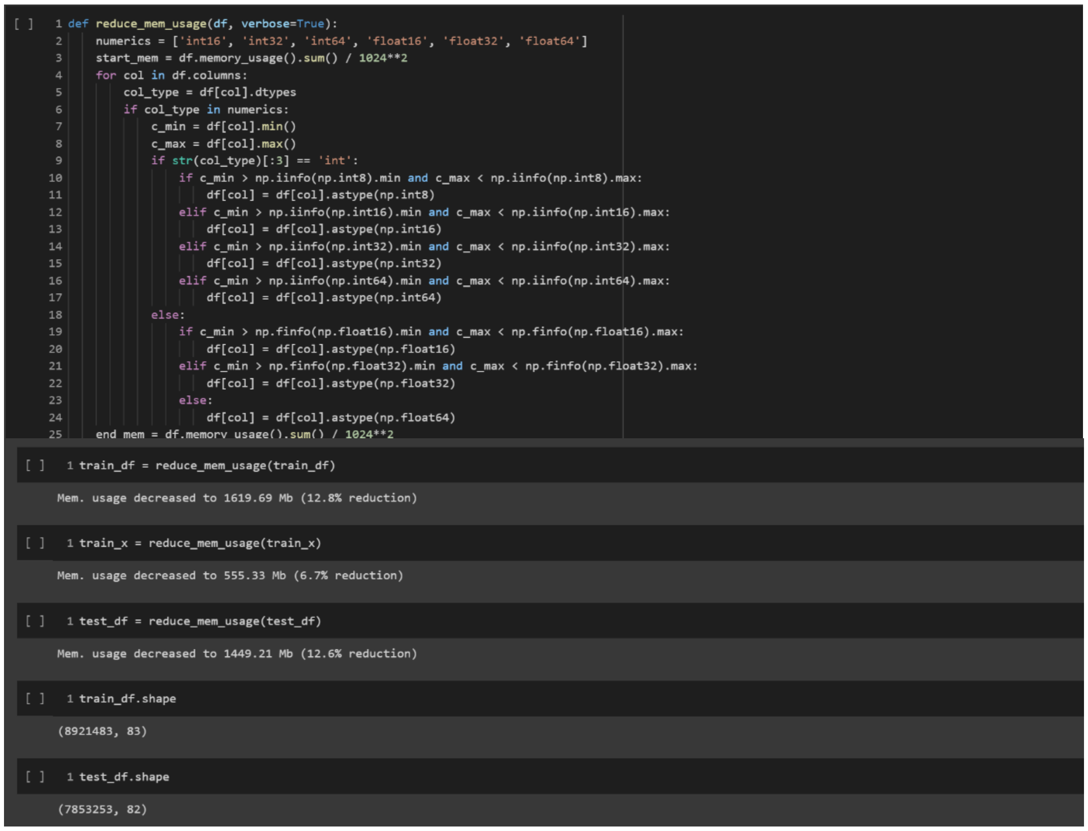


```
dtypes = {
        'MachineIdentifier':                            'category',
        'ProductName':                                  'category',
        'EngineVersion':                                'category',
        'AppVersion':                                   'category',
        'AvSigVersion':                                 'category',
        'IsBeta':                                       'int8',
        'RtpStateBitfield':                             'float16',
        'IsSxsPassiveMode':                             'int8',
        'DefaultBrowsersIdentifier':                    'float32',
        'AVProductStatesIdentifier':                    'float32',
        'AVProductsInstalled':                          'float16',
        'AVProductsEnabled':                            'float16',
```

Figure 3.4.1: Utilize variable size And Loaded data

3.4.2 Function to Reduce Data Frame Size

To avoid memory usage error in Google CoLab we use a function that help us to reduce our data frame size. This is a much needed function for those who have low configuration laptops and computer.



```
def reduce_mem_usage(df, verbose=True):
    numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
    start_mem = df.memory_usage().sum() / 1024**2
    for col in df.columns:
        col_type = df[col].dtypes
        if col_type in numerics:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
    end_mem = df.memory_usage().sum() / 1024**2
```

```
train_df = reduce_mem_usage(train_df)
```

```
Mem. usage decreased to 1619.69 Mb (12.8% reduction)
```

```
train_x = reduce_mem_usage(train_x)
```

```
Mem. usage decreased to 555.33 Mb (6.7% reduction)
```

```
test_df = reduce_mem_usage(test_df)
```

```
Mem. usage decreased to 1449.21 Mb (12.6% reduction)
```

```
train_df.shape
```

```
(8921483, 83)
```

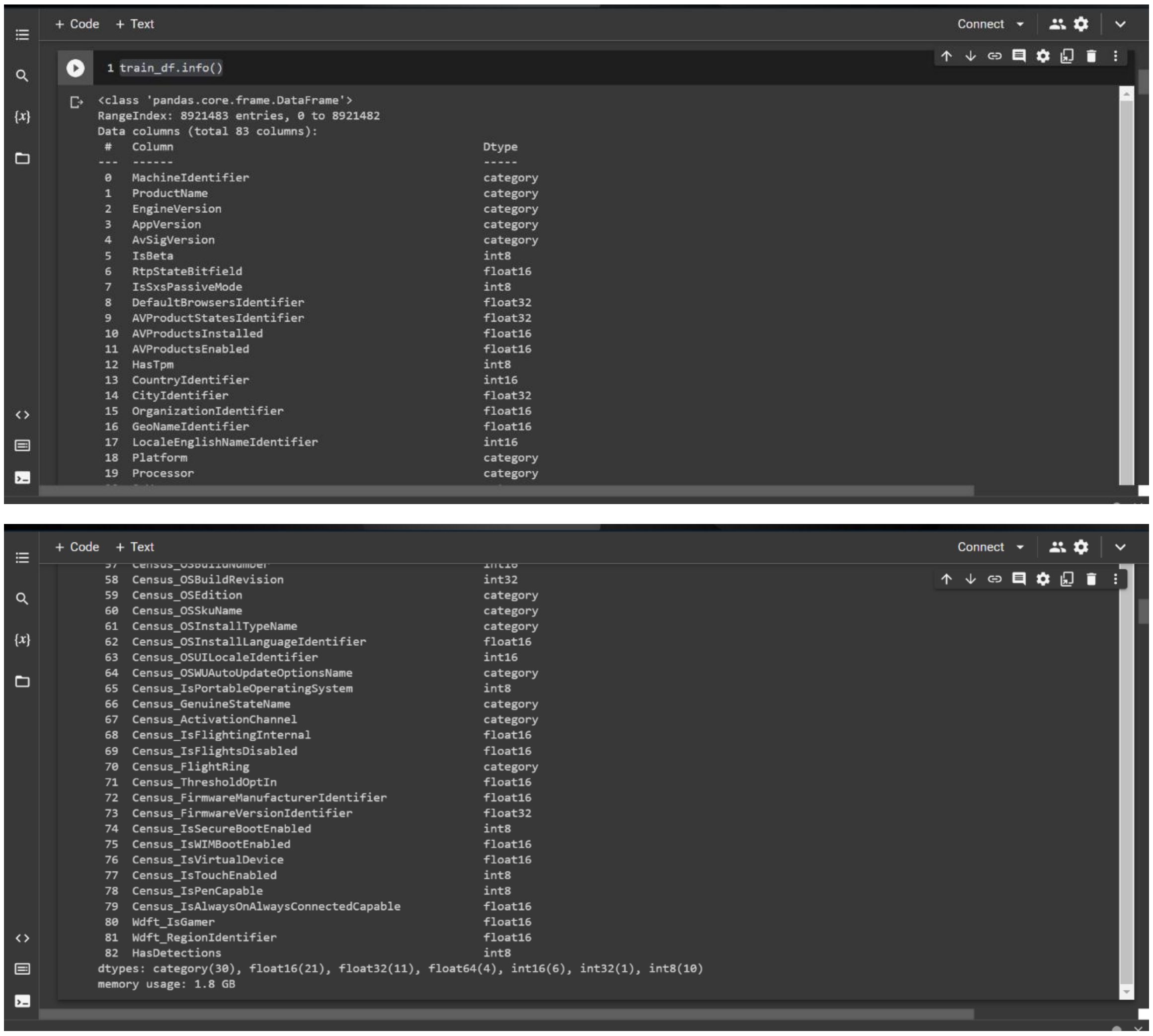
```
test_df.shape
```

```
(7853253, 82)
```

Figure 3.4.2: Reduce data frame size

3.4.3 Features

A feature is a part of something that makes it special or able to work better. The dataset that was obtained has 83 features. The HashDetections variable is the target. We have 83 features with a variety of data types, therefore feature selection is necessary before we begin working with the data.



```python
train_df.info()
```

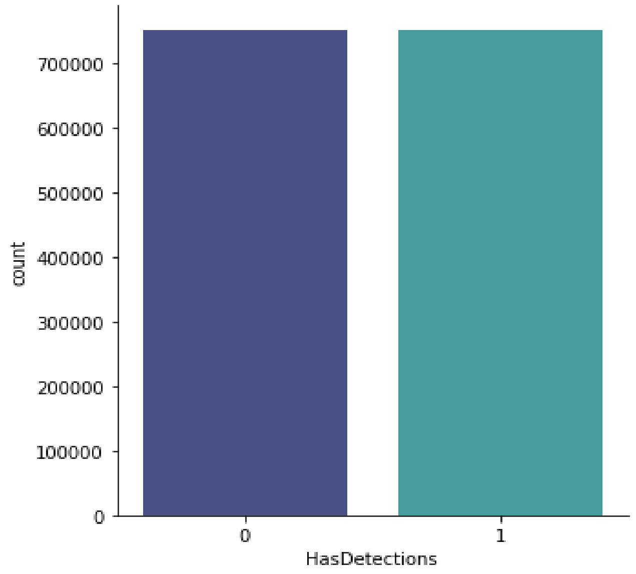
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8921483 entries, 0 to 8921482
Data columns (total 83 columns):
 #   Column                                   Dtype
---  ------                                   -----
 0   MachineIdentifier                        category
 1   ProductName                              category
 2   EngineVersion                            category
 3   AppVersion                               category
 4   AvSigVersion                             category
 5   IsBeta                                   int8
 6   RtpStateBitfield                         float16
 7   IsSxsPassiveMode                         int8
 8   DefaultBrowsersIdentifier                float32
 9   AVProductStatesIdentifier                float32
 10  AVProductsInstalled                      float16
 11  AVProductsEnabled                        float16
 12  HasTpm                                   int8
 13  CountryIdentifier                        int16
 14  CityIdentifier                           float32
 15  OrganizationIdentifier                   float16
 16  GeoNameIdentifier                        float16
 17  LocaleEnglishNameIdentifier              int16
 18  Platform                                 category
 19  Processor                                category
```

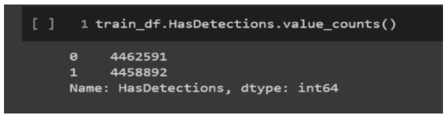
```
 57  Census_OSBuildNumber                     int16
 58  Census_OSBuildRevision                   int32
 59  Census_OSEdition                         category
 60  Census_OSSkuName                         category
 61  Census_OSInstallTypeName                 category
 62  Census_OSInstallLanguageIdentifier       float16
 63  Census_OSUILocaleIdentifier              int16
 64  Census_OSWUAutoUpdateOptionsName         category
 65  Census_IsPortableOperatingSystem         int8
 66  Census_GenuineStateName                  category
 67  Census_ActivationChannel                 category
 68  Census_IsFlightingInternal               float16
 69  Census_IsFlightsDisabled                 float16
 70  Census_FlightRing                        category
 71  Census_ThresholdOptIn                    float16
 72  Census_FirmwareManufacturerIdentifier    float16
 73  Census_FirmwareVersionIdentifier         float32
 74  Census_IsSecureBootEnabled               int8
 75  Census_IsWIMBootEnabled                  float16
 76  Census_IsVirtualDevice                   float16
 77  Census_IsTouchEnabled                    int8
 78  Census_IsPenCapable                      int8
 79  Census_IsAlwaysOnAlwaysConnectedCapable  float16
 80  Wdft_IsGamer                             float16
 81  Wdft_RegionIdentifier                    float16
 82  HasDetections                            int8
dtypes: category(30), float16(21), float32(11), float64(4), int16(6), int32(1), int8(10)
memory usage: 1.8 GB
```

Figure 3.4.3: Features List

3..4.4 Target Variable Exploration

The target variable in a dataset is the piece you want to better understand. The user would like to assess this variable in light of the rest of the data. The majority of the time, standard supervised learning is applied to derive the target variable. Such an algorithm takes use of past data to spot trends and find links between the target and various pieces of your data. According to the aim and the available information, the target variables could change. HasDetections is our target variable in this case. We shall forecast malware in the base of this variable.



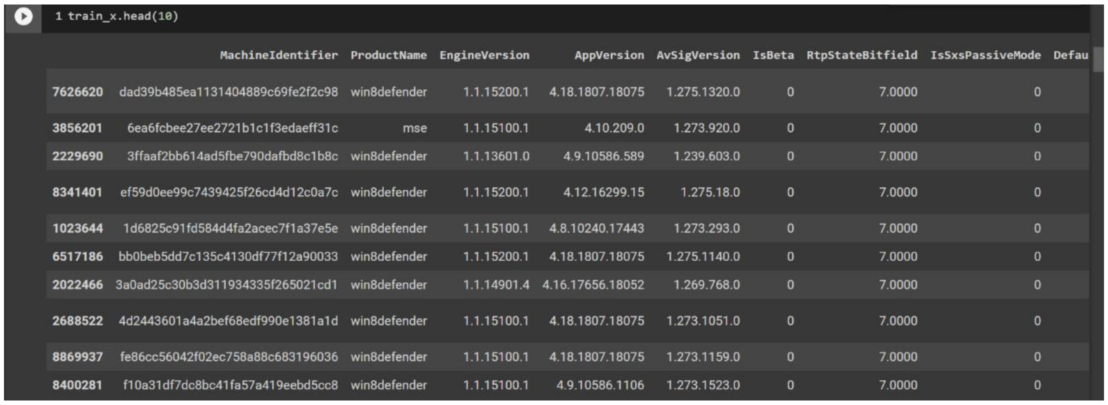


```
[ ]   1 train_df.HasDetections.value_counts()

0    4462591
1    4458892
Name: HasDetections, dtype: int64
```

Figure 3.4.4: Target Variable

3.4.5 EDA & Data Exploration

Data scientists use exploratory data analysis (EDA) to analyze and investigate data sets and summarize their main characteristics, often using data visualization methods. It aids in determining how to best manipulate data sources to obtain the answers required, making it easier for data scientists to discover patterns, detect anomalies, test hypotheses, and validate assumptions. We have 82 characteristics with a variety of data formats; some of these features may have highly correlated values, skewed distributions, or null values, rendering them unusable. In order to begin working with the data, feature selection is mandatory.



```
1 train_x.head(10)
```

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion	IsBeta	RtpStateBitfield	IsSxsPassiveMode	Defau
7626620	dad39b485ea1131404889c69fe2f2c98	win8defender	1.1.15200.1	4.18.1807.18075	1.275.1320.0	0	7.0000	0	
3856201	6ea6fcbee27ee2721b1c1f3edaeff31c	mse	1.1.15100.1	4.10.209.0	1.273.920.0	0	7.0000	0	
2229690	3ffaaf2bb614ad5fbe790dafbd8c1b8c	win8defender	1.1.13601.0	4.9.10586.589	1.239.603.0	0	7.0000	0	
8341401	ef59d0ee99c7439425f26cd4d12c0a7c	win8defender	1.1.15200.1	4.12.16299.15	1.275.18.0	0	7.0000	0	
1023644	1d6825c91fd584d4fa2acec7f1a37e5e	win8defender	1.1.15100.1	4.8.10240.17443	1.273.293.0	0	7.0000	0	
6517186	bb0beb5dd7c135c4130df77f12a90033	win8defender	1.1.15200.1	4.18.1807.18075	1.275.1140.0	0	7.0000	0	
2022466	3a0ad25c30b3d311934335f265021cd1	win8defender	1.1.14901.4	4.16.17656.18052	1.269.768.0	0	7.0000	0	
2688522	4d2443601a4a2bef68edf990e1381a1d	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1051.0	0	7.0000	0	
8869937	fe86cc56042f02ec758a88c683196036	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1159.0	0	7.0000	0	
8400281	f10a31df7dc8bc41fa57a419eebd5cc8	win8defender	1.1.15100.1	4.9.10586.1106	1.273.1523.0	0	7.0000	0	

Figure 3.4.5: Data Exploration

As there is only one record for each system and the Machine Identifier says nothing about the infection, it has no purpose in this context. ProductName, EngineVersion, and AppVersion are not mentioned in the data description, but we may be able to gather useful information from them that suggests the device has been infected with malware.

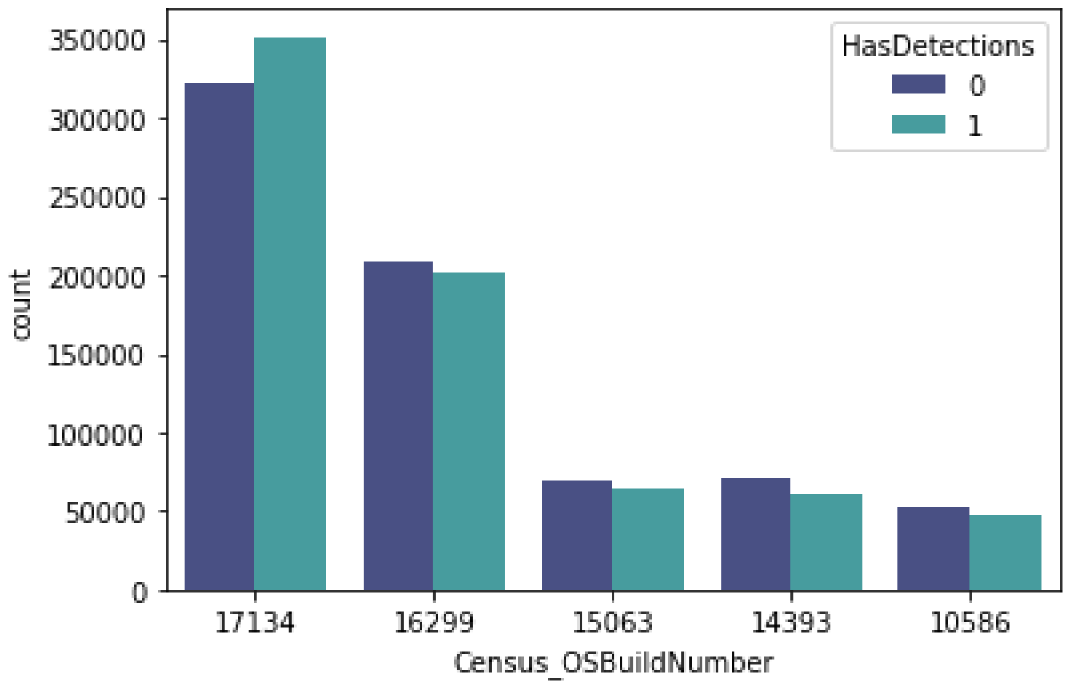
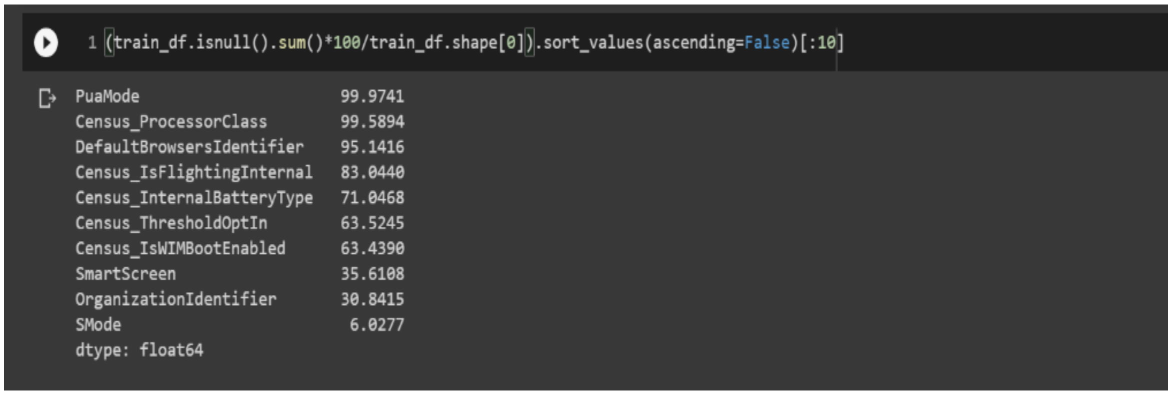


Figure 3.4.5.2: Data Exploration

3.4.6 Missing Values Pre-processing

Missing data, also known as missing values, occurs in statistics. The conclusions that can be derived from the data can be significantly impacted by missing data, which is a typical occurrence. Missing data can make the model less accurate. Because null values are present during preprocessing, the visualization we obtain for a certain feature may be deceptive. The final model that is produced might be skewed. In actual life, these null values may lead to issues. First, we find the missing values: -

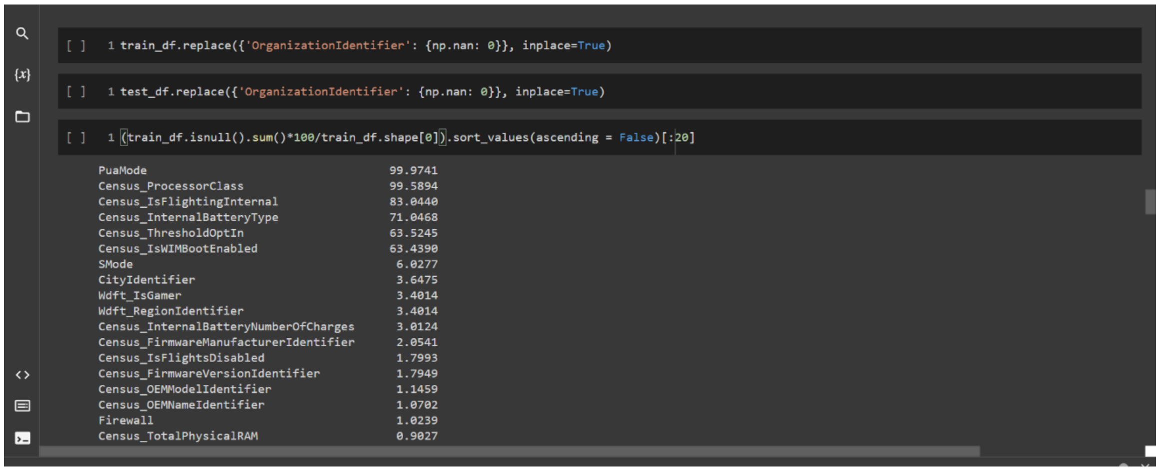


```
(train_df.isnull().sum()*100/train_df.shape[0]).sort_values(ascending=False)[:10]
```

```
PuaMode                      99.9741
Census_ProcessorClass        99.5894
DefaultBrowsersIdentifier    95.1416
Census_IsFlightingInternal   83.0440
Census_InternalBatteryType   71.0468
Census_ThresholdOptIn        63.5245
Census_IsWIMBootEnabled      63.4390
SmartScreen                  35.6108
OrganizationIdentifier       30.8415
SMode                         6.0277
dtype: float64
```

Figure 3.4.6.1: Missing Values

We will think about deleting the two features that have 99% missing values because they are essentially meaningless. We will then use correlation or feature impotence to check the 95% and 83% values.



```
train_df.replace({'OrganizationIdentifier': {np.nan: 0}}, inplace=True)
```

```
test_df.replace({'OrganizationIdentifier': {np.nan: 0}}, inplace=True)
```

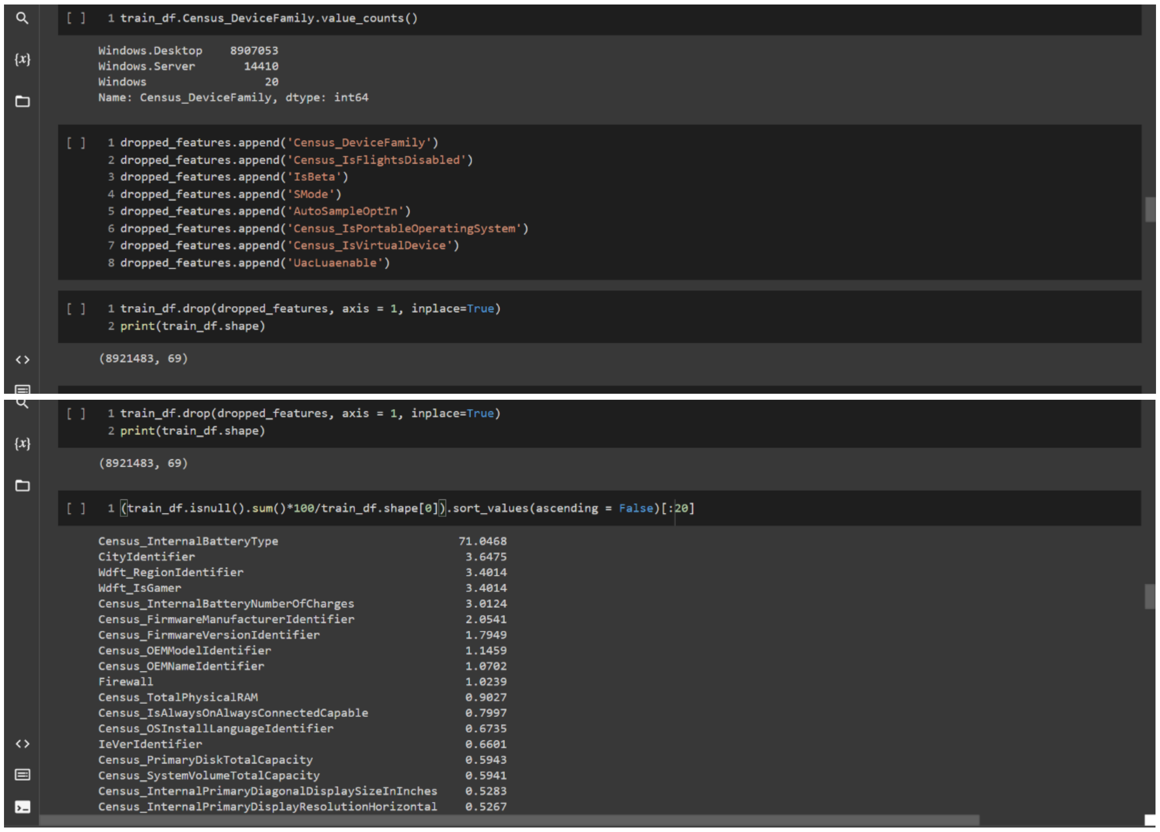
```
(train_df.isnull().sum()*100/train_df.shape[0]).sort_values(ascending = False)[:20]
```

```
PuaMode                                  99.9741
Census_ProcessorClass                    99.5894
Census_IsFlightingInternal               83.0440
Census_InternalBatteryType               71.0468
Census_ThresholdOptIn                    63.5245
Census_IsWIMBootEnabled                  63.4390
SMode                                     6.0277
CityIdentifier                            3.6475
Wdft_IsGamer                              3.4014
Wdft_RegionIdentifier                     3.4014
Census_InternalBatteryNumberOfCharges     3.0124
Census_FirmwareManufacturerIdentifier     2.0541
Census_IsFlightsDisabled                  1.7993
Census_FirmwareVersionIdentifier          1.7949
Census_OEMModelIdentifier                 1.1459
Census_OEMNameIdentifier                  1.0702
Firewall                                  1.0239
Census_TotalPhysicalRAM                   0.9027
```

Figure 3.4.6.2: Missing Data imputation

3.4.7 Skewness Check

The absence of symmetry is what is referred to as a distribution's skewness. The Mean, Median, and Mode are equivalent in a symmetrical distribution. The skewness of the normal distribution is 0. Skewness reveals information about our data's distribution. When the skewness value is larger than 1 or lower than -1, the distribution is strongly skewed. A mildly skewed value is one between 0.5 and 1 or -0.5 and -1. When the value falls between -0.5 and 0.5, the distribution is considered to be fairly symmetrical. After skewness check train_df shape was (8921483, 69) and test df shape was (7667789, 69). means 69 features and 8921483 entries in train dataset. Shown in figure Figure 3.6.5.2.



```
[ ]  train_df.Census_DeviceFamily.value_counts()

     Windows.Desktop    8907053
     Windows.Server       14410
     Windows                 20
     Name: Census_DeviceFamily, dtype: int64

[ ]  dropped_features.append('Census_DeviceFamily')
     dropped_features.append('Census_IsFlightsDisabled')
     dropped_features.append('IsBeta')
     dropped_features.append('SMode')
     dropped_features.append('AutoSampleOptIn')
     dropped_features.append('Census_IsPortableOperatingSystem')
     dropped_features.append('Census_IsVirtualDevice')
     dropped_features.append('UacLuaenable')

[ ]  train_df.drop(dropped_features, axis = 1, inplace=True)
     print(train_df.shape)

     (8921483, 69)
```

```
[ ]  train_df.drop(dropped_features, axis = 1, inplace=True)
     print(train_df.shape)

     (8921483, 69)

[ ]  (train_df.isnull().sum()*100/train_df.shape[0]).sort_values(ascending = False)[:20]

     Census_InternalBatteryType                        71.0468
     CityIdentifier                                     3.6475
     Wdft_RegionIdentifier                              3.4014
     Wdft_IsGamer                                       3.4014
     Census_InternalBatteryNumberOfCharges              3.0124
     Census_FirmwareManufacturerIdentifier              2.0541
     Census_FirmwareVersionIdentifier                   1.7949
     Census_OEMModelIdentifier                          1.1459
     Census_OEMNameIdentifier                           1.0702
     Firewall                                           1.0239
     Census_TotalPhysicalRAM                            0.9027
     Census_IsAlwaysOnAlwaysConnectedCapable            0.7997
     Census_OSInstallLanguageIdentifier                 0.6735
     IeVerIdentifier                                    0.6601
     Census_PrimaryDiskTotalCapacity                    0.5943
     Census_SystemVolumeTotalCapacity                   0.5941
     Census_InternalPrimaryDiagonalDisplaySizeInInches  0.5283
     Census_InternalPrimaryDisplayResolutionHorizontal  0.5267
```

Figure 3.4.7: After Skewness check train_df shape & Feature list

3.4.8 Correlation Test

Correlation analysis assesses the strength of a link between two variables. Through correlation analysis, you may find the correlation coefficient, which shows how much one variable change when the other one does. You can determine the linear relationship between two variables using correlation analysis. The correlation of features with the target variable allows us to determine which features will affect the model's ability to predict toward the ground truth or the target variable in this case, "HasDetections." This aspect of exploratory data analysis greatly aided me in improving the performance of my model. To reduce the dimensions, look into the correlation between the different factors. We will divide each of the ten columns for better viewing. Keep the variables with higher value variants and remove the others. We will eliminate RtpStateBitfield because it has more values. Now let’s check the correlation between all features in figure 3.6.4

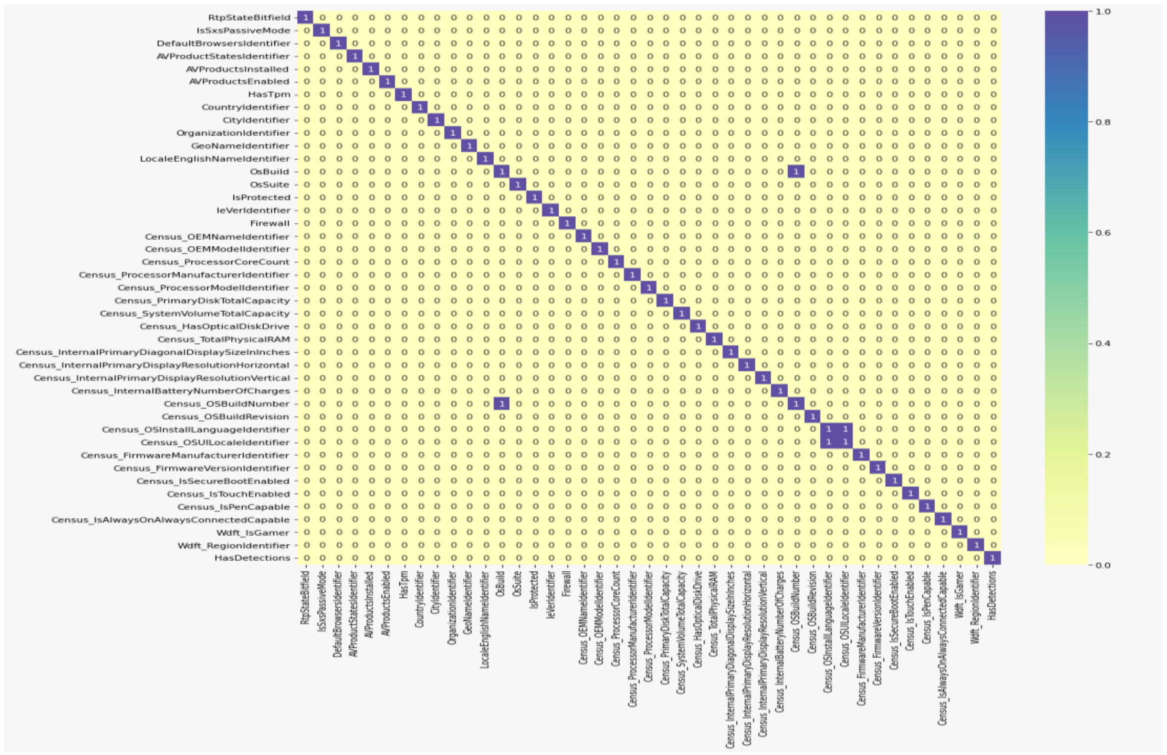
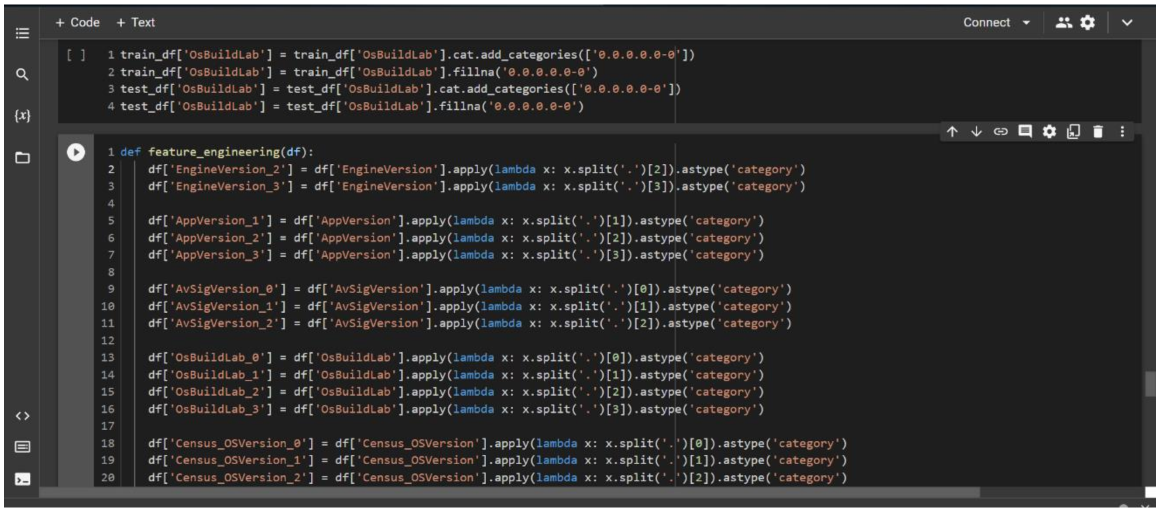


Figure 3.6.6.4: Correlation Between All Feature

3.5 Feature Engineering

Feature engineering is the process of modifying your data set, including addition, deletion, combination, and mutation, in order to enhance the training of your machine learning model and achieve improved accuracy and performance. A solid understanding of the business issue and the data sources at hand is the foundation for effective feature engineering. Feature creation, transformation, extraction, and selection are the four processes. As follows is a description of these procedures: Finding the most beneficial variables to include in a predictive model is the process of creating features. Following exploratory data analysis, feature engineering is regarded as the most important aspect of developing a good machine learning system. We chose five features as part of feature engineering: 'EngineVersion,' 'AppVersion,' 'AvSigVersion,' 'OsBuildLab,' and 'Census OSVersion.'



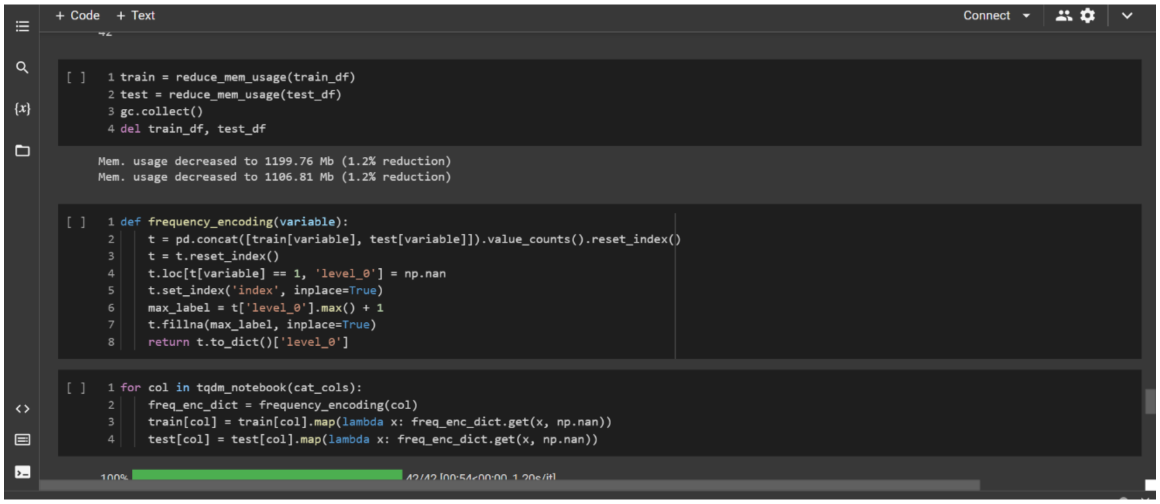
```
train_df['OsBuildLab'] = train_df['OsBuildLab'].cat.add_categories(['0.0.0.0.0-0'])
train_df['OsBuildLab'] = train_df['OsBuildLab'].fillna('0.0.0.0.0-0')
test_df['OsBuildLab'] = test_df['OsBuildLab'].cat.add_categories(['0.0.0.0.0-0'])
test_df['OsBuildLab'] = test_df['OsBuildLab'].fillna('0.0.0.0.0-0')
```

```
def feature_engineering(df):
    df['EngineVersion_2'] = df['EngineVersion'].apply(lambda x: x.split('.')[2]).astype('category')
    df['EngineVersion_3'] = df['EngineVersion'].apply(lambda x: x.split('.')[3]).astype('category')

    df['AppVersion_1'] = df['AppVersion'].apply(lambda x: x.split('.')[1]).astype('category')
    df['AppVersion_2'] = df['AppVersion'].apply(lambda x: x.split('.')[2]).astype('category')
    df['AppVersion_3'] = df['AppVersion'].apply(lambda x: x.split('.')[3]).astype('category')

    df['AvSigVersion_0'] = df['AvSigVersion'].apply(lambda x: x.split('.')[0]).astype('category')
    df['AvSigVersion_1'] = df['AvSigVersion'].apply(lambda x: x.split('.')[1]).astype('category')
    df['AvSigVersion_2'] = df['AvSigVersion'].apply(lambda x: x.split('.')[2]).astype('category')

    df['OsBuildLab_0'] = df['OsBuildLab'].apply(lambda x: x.split('.')[0]).astype('category')
    df['OsBuildLab_1'] = df['OsBuildLab'].apply(lambda x: x.split('.')[1]).astype('category')
    df['OsBuildLab_2'] = df['OsBuildLab'].apply(lambda x: x.split('.')[2]).astype('category')
    df['OsBuildLab_3'] = df['OsBuildLab'].apply(lambda x: x.split('.')[3]).astype('category')

    df['Census_OSVersion_0'] = df['Census_OSVersion'].apply(lambda x: x.split('.')[0]).astype('category')
    df['Census_OSVersion_1'] = df['Census_OSVersion'].apply(lambda x: x.split('.')[1]).astype('category')
    df['Census_OSVersion_2'] = df['Census_OSVersion'].apply(lambda x: x.split('.')[2]).astype('category')
```

Figure 3.5: Feature Engineering

3.6 Encoding

Because the dataset contains categorical features but the model performs well when numerical values are fed, I encoded the categorical features first before implementing the first cut approach. There are several encoding techniques for encoding categorical features, but I used frequency encoding for this problem statement.



```
train = reduce_mem_usage(train_df)
test = reduce_mem_usage(test_df)
gc.collect()
del train_df, test_df
```

```
Mem. usage decreased to 1199.76 Mb (1.2% reduction)
Mem. usage decreased to 1106.81 Mb (1.2% reduction)
```

```
def frequency_encoding(variable):
    t = pd.concat([train[variable], test[variable]]).value_counts().reset_index()
    t = t.reset_index()
    t.loc[t[variable] == 1, 'level_0'] = np.nan
    t.set_index('index', inplace=True)
    max_label = t['level_0'].max() + 1
    t.fillna(max_label, inplace=True)
    return t.to_dict()['level_0']
```

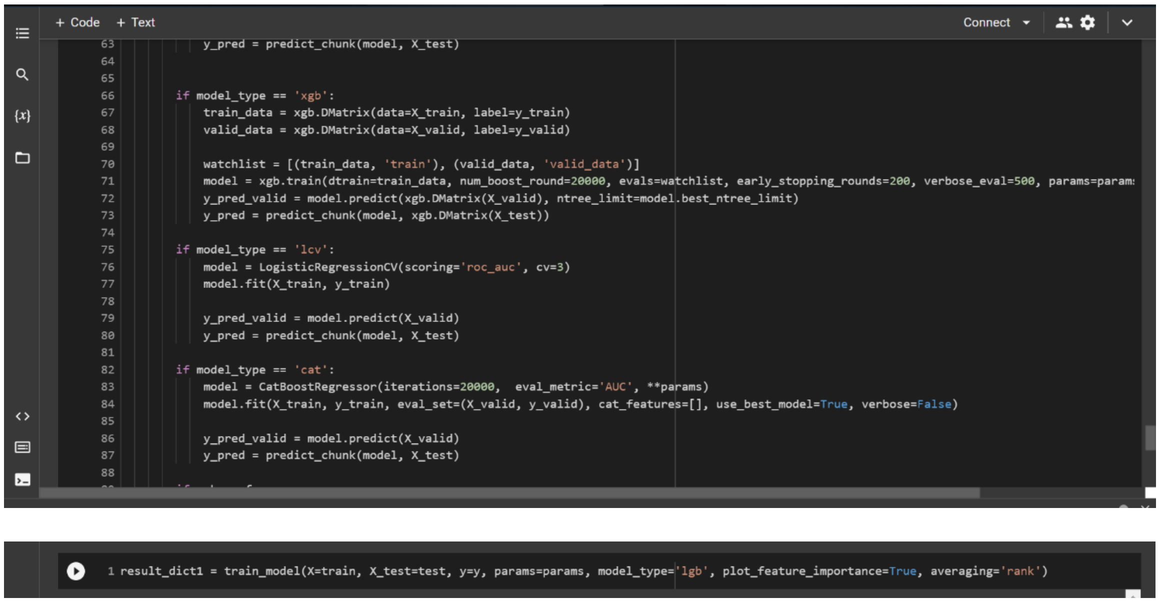
```
for col in tqdm_notebook(cat_cols):
    freq_enc_dict = frequency_encoding(col)
    train[col] = train[col].map(lambda x: freq_enc_dict.get(x, np.nan))
    test[col] = test[col].map(lambda x: freq_enc_dict.get(x, np.nan))
```

```
100% 42/42 [00:54<00:00, 1.20s/it]
```

Figure 3.6: Encoding

3.7 Model Select

Here we implement the LGBM, XGB, Logistic Regression, CatBoost Regressor model. But for insufficient memory usage and reduce time I select LGBM algorithm for the model. I used the tree-based gradient boosting framework lightgbm to train my model. Due to the size of the dataset and the fact that lightgbm is well renowned for its high speed, low memory usage, and excellent handling of data with a high number of categorical characteristics, I chose this technique as a first cut option. By separating the data into folds and then using each fold as a testing set, I had employed stratified K-Fold cross validation to validate the evaluation measure. Model accuracy was at its highest with the application of stratified 5-fold cross validation.



```
        y_pred = predict_chunk(model, X_test)


    if model_type == 'xgb':
        train_data = xgb.DMatrix(data=X_train, label=y_train)
        valid_data = xgb.DMatrix(data=X_valid, label=y_valid)

        watchlist = [(train_data, 'train'), (valid_data, 'valid_data')]
        model = xgb.train(dtrain=train_data, num_boost_round=20000, evals=watchlist, early_stopping_rounds=200, verbose_eval=500, params=param:
        y_pred_valid = model.predict(xgb.DMatrix(X_valid), ntree_limit=model.best_ntree_limit)
        y_pred = predict_chunk(model, xgb.DMatrix(X_test))

    if model_type == 'lcv':
        model = LogisticRegressionCV(scoring='roc_auc', cv=3)
        model.fit(X_train, y_train)

        y_pred_valid = model.predict(X_valid)
        y_pred = predict_chunk(model, X_test)

    if model_type == 'cat':
        model = CatBoostRegressor(iterations=20000,  eval_metric='AUC', **params)
        model.fit(X_train, y_train, eval_set=(X_valid, y_valid), cat_features=[], use_best_model=True, verbose=False)

        y_pred_valid = model.predict(X_valid)
        y_pred = predict_chunk(model, X_test)

```

```
result_dict1 = train_model(X=train, X_test=test, y=y, params=params, model_type='lgb', plot_feature_importance=True, averaging='rank')
```

Figure 3.7: LGB Model select

3.8 Statistical Analysis

To identify patterns and trends, data is gathered and evaluated for statistical analysis. It is a part of data analytics. Statistical analysis may be used for tasks including getting research interpretations, statistical modeling, or developing surveys and studies. Businesses that deal with massive amounts of data may also find it useful in their business intelligence operations. Statistical analysis in the context of business intelligence entails gathering and examining each data sample in a set of things from which samples can be drawn. In our dataset, it contains 8921483 entries in train data file and 7853253 entries in test data file data. To apply machine learning algorithm, we have tried to find out better accuracy in our model. We have briefly described how we conducted our research in this figure. Figure 3.6 shows us how to proceed step-by-step toward our goal.

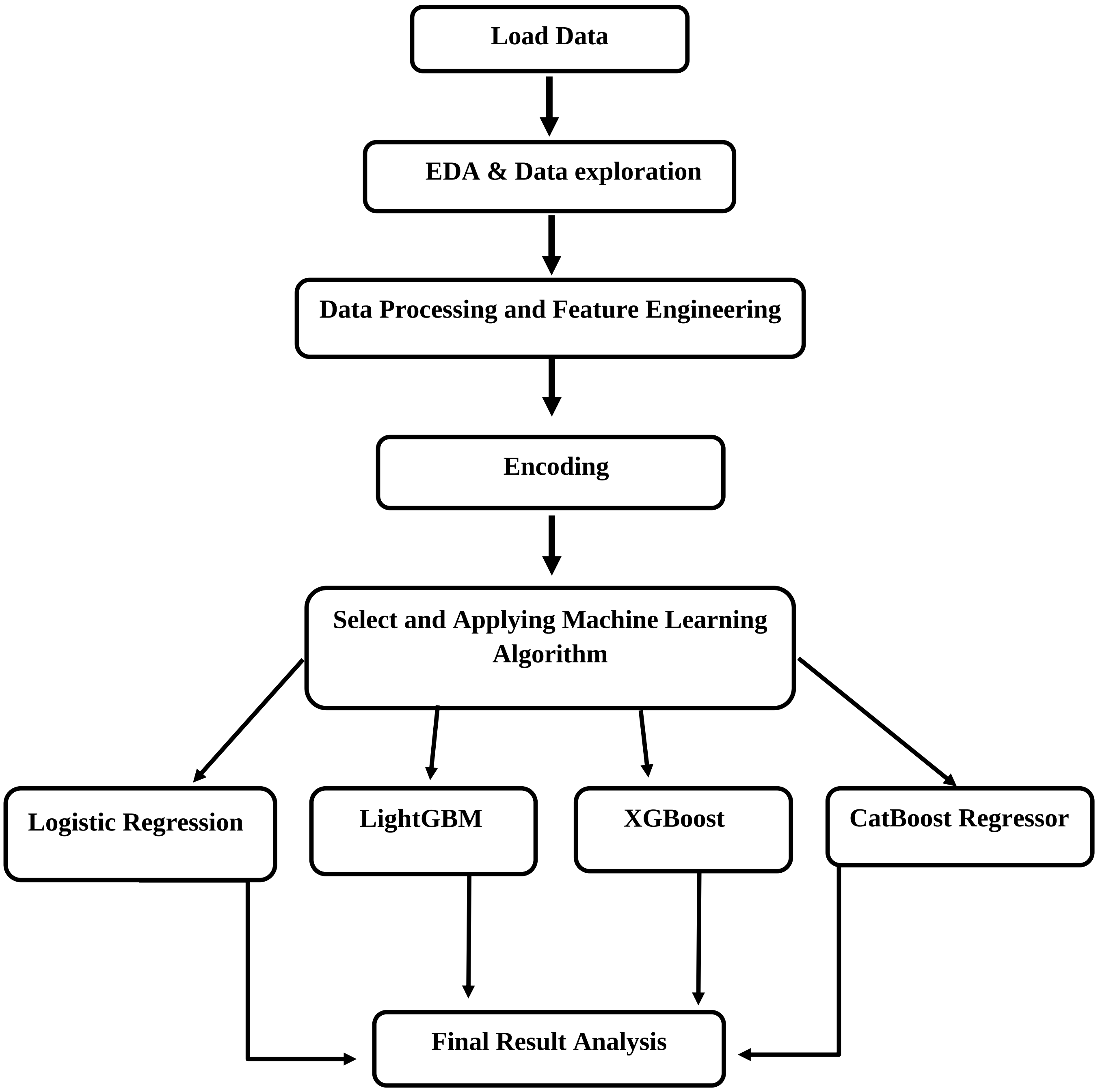


Figure 3.6: Proposed Model Structure

3.9 Implementation Requirements

Python 3.8

There is Python 3.8 available. The programming language used is sophisticated. Most scholars use it in their studies. It is a highly recommended programming language for AI-based work and is especially well-liked by the upcoming generation of programmers due to how easy it is to learn and understand.

Google Collab

Python is made available as free to use open-source software through Google Collab. It is possible to work here online using both the Jupiter notebook and our browser. The major benefit of Google Collab is that it provides us with free internet access to virtual GPUs.

Hardware & Software Requirements

Operating System (Windows 8 or higher),
Web browser (Chrome is preferred), SSD (Minimum 4 GB),
and RAM are requirements (More than 4 GB)

CHAPTER 4
EXPERIMENTAL RESULT AND DISCUSSION

4.1 Experimental Setup

We first gathered the data for use in our model and code execution. The system is described below: -We gathered information from numerous official websites while we worked on the forecast of malware and viruses in the Microsoft Windows operating system.

-The majority of the time allotted to us for the exploratory project was used to generate the model and pre-process the variables and missing values.

-We also got information from the online Kaggle website.

At that moment, the information has been completed and standardized, allowing us to begin the implementation process.

4.2 Model Summary

After performing exploratory data analysis and feature engineering, I trained a model using the lightgbm first-cut procedure, which is a tree-based gradient boosting framework. Due to the size of the dataset and the fact that lightgbm is well renowned for its high speed, low memory usage, and excellent handling of data with a high number of categorical characteristics, I chose this technique as a first cut option.

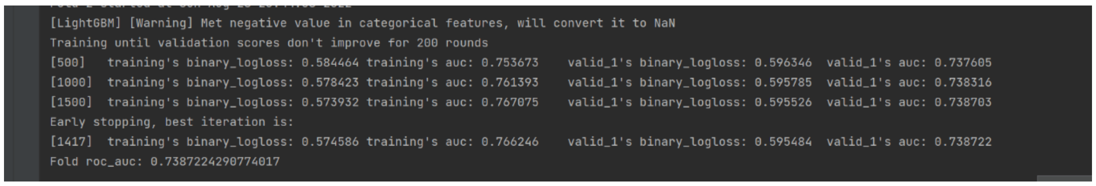
4.3 Experimental Result and Analysis

After pre-processing and data exploration, we used a variety of calculations. We'll talk about the results of several calculations shortly. We used our overall highlights, which we extracted by reading numerous articles, from the beginning and performed various AI computations to determine the accuracy of our model. The average accuracy we obtained was quite close to 74%.

We have got almost same type of accuracy by using LGBM, we have got 0.7387 valid auc and by using XGB we have got 0.7115 valid auc, using LCV we have got 0.72393 valid and lastly by using CatBoost we have got 0.728781 valid accuracy that has been shown.

Table 4.1: Accuracy Table

Algorithm Name	**Accuracy**
LightGBM	74%
XGB	71%
Logistic Regression	72%
CatBoost Regressor	73%



```
[LightGBM] [Warning] Met negative value in categorical features, will convert it to NaN
Training until validation scores don't improve for 200 rounds
[500]   training's binary_logloss: 0.584464 training's auc: 0.753673    valid_1's binary_logloss: 0.596346  valid_1's auc: 0.737605
[1000]  training's binary_logloss: 0.578423 training's auc: 0.761393    valid_1's binary_logloss: 0.595785  valid_1's auc: 0.738316
[1500]  training's binary_logloss: 0.573932 training's auc: 0.767075    valid_1's binary_logloss: 0.595526  valid_1's auc: 0.738703
Early stopping, best iteration is:
[1417]  training's binary_logloss: 0.574586 training's auc: 0.766246    valid_1's binary_logloss: 0.595484  valid_1's auc: 0.738722
Fold roc_auc: 0.7387224290774017
```

Figure 4.3: LightGBM Accuracy

4.4 Model Accuracy compare

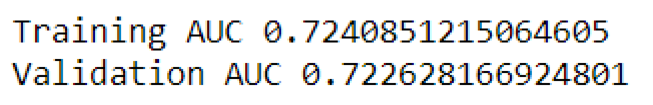
Author Jugaloza Model	**Proposed Model**
 Training AUC 0.7240851215064605 Validation AUC 0.722628166924801	Training AUC 0.767875 Validation AUC 0.738783

Figure 4.4.1: LightGBM Model Accuracy Compare with Author Jugaloza Model

4.5 Discussion

Our dataset and model have been modified. According to the update, this model can be used for a variety of purposes. We have made progress in characterizing the precision of effect expectations. The model makes it possible to consider as well as discover the appropriate result. Also, we are able to reach everyone with our E-commerce site. Where anyone through internet can take our paid services to predict their windows file. however, we are working on to build the model in our website. If we successfully do then it will be a great windows security assault.

CHAPTER 5

IMPACT ON SOCIETY, ENVIRONMENT AND SUSTAINABILITY

5.1 Impact on Society

A computer virus has the power to infect and endanger life much like a biological virus does. Malicious software encompasses more than simply straightforward viruses; it also contains worms, Trojan horses, and ransomware. Since the late 1980s, there has been malware. Hacking used to be a way for people to get attention online, but these days it's more of a business. Cyber attacks come from all across the world, and not just one person can profit from them. In the past 20 years, a large number of cybercrime syndicates have emerged, and governments from all over the world support hacker communities. Malware has evolved over the past 25 years to focus less on testing the integrity of computer security and more on being well-known in the shadowy online community.

5.2 Impact on Environment

Malicious malware, sometimes known as malware, has a big impact on our civilization. In order to make money, sophisticated malware is increasingly being used against consumers, utility software, and user apps. Some of the dangers include copyright violations, the theft of private information like credit card details, spamming, impersonation, and financial extortion (ransomware). It is a hacker-operated cybercrime. Indeed, those who hack into computers and networks for illegal purposes are known as crackers. As a result, computing systems are very concerned about cyber security.

5.3 Ethical Aspects

There are several ethical aspects to consider when discussing malware predict and the use of machine learning to detect it.

First, it is important to ensure that the privacy and confidentiality of individuals are protected. This includes ensuring that personal data is collected, stored, and used in a waythat is in accordance with relevant laws and regulations. It is also important to ensure thatindividuals are informed about how their data will be used and have the opportunity to opt-out or withdraw their consent if they wish.

Second, it is important to ensure that the use of machine learning to detect malware does not discriminate against certain individuals or groups. This could includeissues such as biased training data or algorithms that are more likely to misdiagnose certain groups of people.

Third, it is important to consider the potential consequences of a misdiagnosis or a false positive or negative result. It is important to ensure that any machine learning approach is accurate and reliable in order to minimize these risks.
It is important to consider the potential ethical implications of these developments in order to ensure that the technology is used in a responsible and ethical manner.

5.4 Sustainability Plan

We will attempt to employ additional machine learning algorithms and raise the scores now that we have a lot better grasp of the ways that are frequently used to accomplish the Microsoft Malware Prediction challenge.
-We will work with Artifical Intelligent.
-We will build proposed model into our E-commerce site.
-We'll also perform hyperparameter adjustment to get better outcomes.

CHAPTER 6

SUMMARY, CONCLUSION, RECOMMENDATION, IMPLICATION FOR FUTURE RESEARCH

6.1 Summary of the Study

To forecast malware and viruses, we have developed this study project. LightGBM Algorithm has been shown by the project to be the best algorithm for its analysis. To evaluate the accuracy of several author models with the suggested model, data has been gathered. In our studies, we pre-process the Microsoft malware prediction dataset in the best way possible to obtain the highest possible valid accuracy.

6.2 Conclusion

When we select various machine learning algorithms to conduct our training and testing, there is always a compromise. It is clear that machine learning algorithms are not without their difficulties. Memory use, processing time, and output accuracy are the primary difficulties or so-called issues. The project's findings attest to that as well. This is due to the fact that the runtime of the earlier method, which uses LightGBM and has a shorter runtime than the other algorithm, has increased when it attempts to prioritize memory. In general, we may state that longer-running algorithms were able to perform more effectively. One very key finding is that every algorithm requires some method of preparing the data; hence, it is highly vital to know what the data is beforehand to preprocess the data because most algorithms would have performed worse if that very critical step was skipped.

6.3 Recommendations

Because of the development of malware and virus detection, the digital age has witnessed a number of significant developments. Artificial intelligence will ultimately decide the future of this planet. An important component of that innovation is AI. Therefore, developing AI-based frameworks to solve our problems is essential for the advancement of the current advances. Therefore, technologists and experts need to concentrate on this field. The information classifier is perhaps one of the core components of AI. It might completely refute previous theories regarding how the device worked.

6.4 Implication for Further Study

We will attempt to employ additional machine learning algorithms and raise the scores now that we have a lot better grasp of the ways that are frequently used to accomplish the Microsoft Malware Prediction challenge.

-We will work with Artifical Intelligent.

-We will build proposed model into our E-commerce site.

-We'll also perform hyperparameter adjustment to get better outcomes.

-Apart from that, we would examine the impact that increasing iteration would have on accuracy. -Takeaways from the project include: - Frequency/OH Encoding for handling categories during preprocessing. - Making sensible and effective use of training and testing datasets. - How to use ML algorithms to solve issues.

REFERENCES

[1] Malware, as of May 2019. [Online]. You can find viruses at https://www.veracode.com/security/.

[2] What are spyware, malware, viruses, and cookies, and how are they distinct from one another? Connecting is Symantec. [Online]. What are malware viruses, spyware, and cookies, and what distinguishes them? is available at https://connect.symantec.com.

[3] Cybercrimemag, 6 trillion dollars in damages from cybercrime by 2021.The Hackerpocalypse Cybercrime Report 2016 is accessible at https://cybersecurityventures.com.Lincolnshire County Council's complete it system was shut down due to a zero-day attack, according to G. Burton in Computing, May 2017. [Online]. Available: total-it-shut-down-at-lincolnshire-county-council-overzeroday-attack (https://www.computing.co.uk/ctg/news/2443531/).

[4] "Malware detection using machine learning," in 2009 International Multiconference on Computer Science and Information Technology, IEEE, pp. 735-741. D. Gavrilut, M. Cimpoe, D. Anton, and L. Ciortuz.Early-stage malware prediction using recurrent neural networks, computers & security, vol. 77, pp. 578-594, 2018.

[5] M. Baset, "Machine learning for malware detection," PhD thesis, MSc dissertation, Heriot-Watt School of Mathematical and Computer Sciences, 2016.

[6] "Flow-based malware detection using convolutional neural network", in 2018 International Conference on Information Networking (ICOIN), IEEE, pp. 910913. M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song, and C. Park.

[7] "Ensemble models for data-driven prediction of malware infections," in Proceedings of the Ninth ACM International Conference on They Search and Data Mining, ACM, 2016, pp. 583-592. C. Kang, N. Park, B. A. Prakash, E. Serra, and V. Subrahmanian.

[8] [Online]. The following URL is available: https://lightgbm. readthedocs.io/en/latest/Features.html.

[9] "Lightgbm: A very efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems, 2017, pp. 3146-3154, by G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. M. J. Caparas (2020). docs.microsoft.com/en-us/windows/security/threat-protection/, Threat Protection.

[10] S. Choudhary and N. Kesswani (2020). Deep learning in IoT is used to analyze the kdd-cup'99, nsl-kdd, and unswnb15 datasets. 167:1561–1573. Procedia Computer Science.

[11] Data science and intelligence. C. Dobre and F. Xhafa (2014). Intelligent Big Data science services. Systems for next computers,

[12] 37:267 – 281. Innovative Techniques and Algorithms for Modern Data-Intensive Computing Special Section: Big Data Services, Semantics, and Intelligent Processing Advanced Data-Intensive Modeling and Simulation in the Special Section Special\sSection

[13] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W.,Ye, Q., and Liu, T.-Y. (2017).

[14] Efficient Gradient Boosting Decision Tree. In Proceedings of the 31st International
Conference on Neural Information Processing Systems, NIPS'17, page

[15] 3149–3157, Red Hook, NY, USA. Curran Associates Inc.LIN, C. (2019). Naive Transfer Learning Approaches for Suspicious Event Prediction. In 2019 IEEE Inter- national Conference on Big Data (Big Data), pages

[16] 5897–5901. Predicting Malware Attacks using Machine Learning and AutoAI (2021)Mark Sokolov a and Nic Herndon b East Carolina University, Greenville, NC,

[17] Analysis of Malware Prediction Based on Infection Rate Using Machine Learning Techniques by Saffir Zawad 2019 Brac University

[18] Machine Learning and AutoAI for Malware Attack Prediction
Sokolov, Mark East Carolina University, Greenville, North Carolina, USA, and Nic Herndon 2021

[19] N. S. Netanyahu and O. E. David. Deepsign: Deep learning for the automatic generation and classification of malware signatures. Page 1 through page 8 of the International Joint Conference on Neural Networks (IJCNN) 2015 publication J. Saxe and K. Berlin

[20] malware detection utilizing two-dimensional binary program characteristics and deep neural networks. 2015, pages 11–20, in 10th International Conference on Malicious and Unwanted Software (MALWARE).

[21] A. Thomas, H. Sanossian, M. Marinescu, J. W. Stokes, and R. Pascanu. classifying malware using recurrent networks. Pages 1916–1920 in the April 2015 issue of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[22] George Webster, Bojan Kolosnjaji, Apostolis Zarras, and Claudia Eckert. The classification of malware system call sequences using deep learning. Pages 137–149 of AI 2016: Advances in Artificial Intelligence, edited by Byeong Ho Kang and Quan Bai, Cham, 2016. International Springer Publishing

[23] Virustotal. Available at virustotal.com as of 2015 Last accessed on July 25, 2014.

[24] Adasyn: Adaptive synthetic sampling strategy for unbalanced learning, by Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. IEEE World Congress on Computational Intelligence, 2008 IEEE International Joint Conference on Neural Networks, pp 1322-1328, June 2008.

[25]. a scaling rule for the ratio of training to validation sets. 1997 at AT & T Bell Laboratories.Scikit-learn is a Python library for machine learning. In November 2011, J. Machine Learning Research, 12:2825–2830.

[26] Franc¸ois Chollet et al. Keras. https://keras.io, 2015, Date last accessed 23-March-2017. [

[27] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Date last accessed 27-March-2017. Software available from tensorflow.org.

[28] Djork-Arne Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast ´ and accurate deep network learning by exponential linear units (elus). CoRR, abs/1511.07289, 2015.

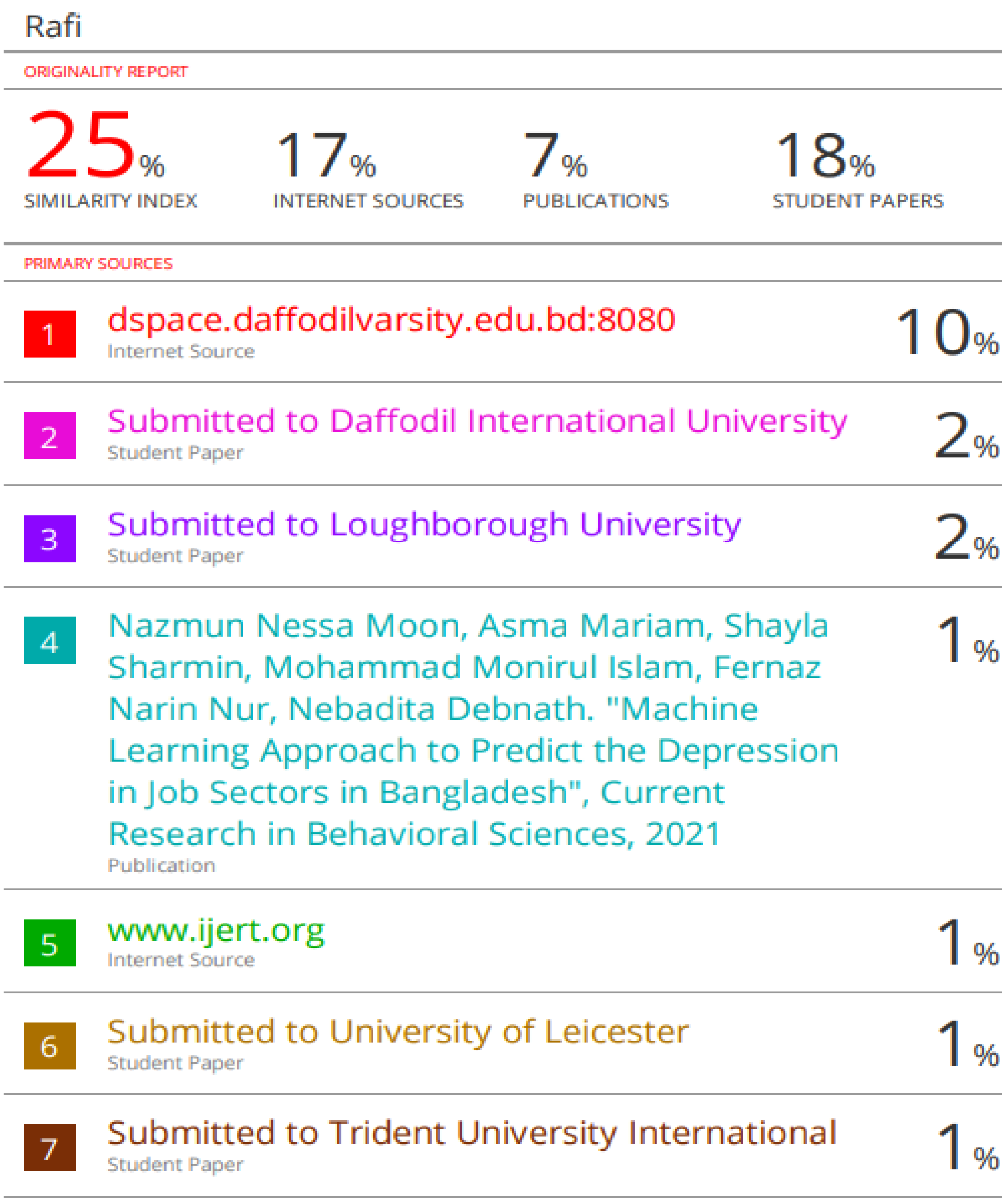
[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.

[30] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. Annals of Operations Research, 134(1):19–67, February 2005.

APPENDIX RESEARCH REFLECTION

We have encountered a variety of issues while working on the project. But among them, three issues were really significant. Data pre-processing, choosing the algorithm query, and managing memory consumption are three examples. We attempted numerous approaches to solving our difficulties before using the LightGBM method, but we were unable to achieve the optimum results. Memory use, processing time, and output accuracy are the primary difficulties or so-called issues.

Plagiarism Report



Rafi

ORIGINALITY REPORT

25%	17%	7%	18%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	10%
2	Submitted to Daffodil International University Student Paper	2%
3	Submitted to Loughborough University Student Paper	2%
4	Nazmun Nessa Moon, Asma Mariam, Shayla Sharmin, Mohammad Monirul Islam, Fernaz Narin Nur, Nebadita Debnath. "Machine Learning Approach to Predict the Depression in Job Sectors in Bangladesh", Current Research in Behavioral Sciences, 2021 Publication	1%
5	www.ijert.org Internet Source	1%
6	Submitted to University of Leicester Student Paper	1%
7	Submitted to Trident University International Student Paper	1%