MALWARE DETECTION USING MACHINE LEARNING

**MACHINE LEARNING**



Table of Content

1. Abstract
2. Introduction

* Review of Literature
* Objective of the Study

1. Findings of the Study

* Discussion of Algorithms been used
* Results and Discussion

1. Conclusion
2. Reference

**Abstract**

Malware detection is an important factor in the security of the computer systems. However, currently utilized signature-based methods cannot provide accurate detection of zero-day attacks and polymorphic viruses. That is why the need for machine learning-based detection arises.

The purpose of this work was to determine the best feature extraction, feature representation, and classification methods that result in the best accuracy when used on the top of Cuckoo Sandbox. Specifically, Decision Trees, Support Vector Machines, Naive Bayes and Random Forest classifiers were evaluated.

This work presents recommended methods for machine learning based malware classification and detection, as well as the guidelines for its implementation. Moreover, the study performed can be useful as a base for further research in the field of malware analysis with machine learning methods.

**Keywords**

Malware, Machine Learning, Classification, Malware detection, Malware analysis, Decision Tree, Support Vector Machines, Random Forest, Naive Bayes

**Introduction**

With the rapid development of the Internet, malware became one of the major cyber threats nowadays. Any software performing malicious actions, including information stealing, espionage, etc. can be referred to as malware. Malware is been defined as “a type of computer program designed to infect a legitimate user's computer and inflict harm on it in multiple ways.”

While the diversity of malware is increasing, anti-virus scanners cannot fulfil the needs of protection, resulting in millions of hosts being attacked. In addition to that, there is a decrease in the skill level that is required for malware development, due to the high availability of attacking tools on the Internet nowadays. High availability of anti-detection techniques, as well as ability to buy malware on the black-market result in the opportunity to become an attacker for anyone, not depending on the skill level. Current studies show that more and more attacks are being issued by script-kiddies or are automated.

Therefore, malware protection of computer systems is one of the most important cybersecurity tasks for single users and businesses, since even a single attack can result in compromised data and sufficient losses. Massive losses and frequent attacks dictate the need for accurate and timely detection methods. Current static and dynamic methods do not provide efficient detection, especially when dealing with zero-day attacks. For this reason, machine learning-based techniques can be used. This paper discusses the main points and concerns of machine

learning-based malware detection, as well as looks for the best feature representation and classification methods.

The accuracy will be measured both for the case of detection of whether the file is malicious and for the case of classification of the file to the malware family. The accuracy of the obtained results will also be assessed in relation to current scoring implemented in Cuckoo Sandbox, and the decision of which method performs better will be made. The study conducted will allow building an additional detection module to Cuckoo Sandbox. However, the implementation of this module is beyond the scope of this project and will not be discussed in this paper.

**Pre-requisite topics**

1. **System Malware**

Malware is any software intentionally designed to cause damage to a computer, server, client, or computer network. Malware (short for “malicious software”) is a file or code, typically delivered over a network, that infects, explores, steals or conducts virtually any behaviour an attacker wants. And because malware comes in so many variants, there are numerous methods to infect computer systems. A wide variety of malware types exist, including computer viruses, worms, Trojan horses, ransomware, spyware, adware, rogue software, wiper and scareware.

**Types of Malwares**

* Trojan horse
* Virus
* Adware
* Bitware

**Machine Learning approach**

In Machine Learning, Data Analysis is the process of inspecting, cleansing, transforming, and modelling data with the goal of discovering useful information by informing conclusions and supporting decision making*.*

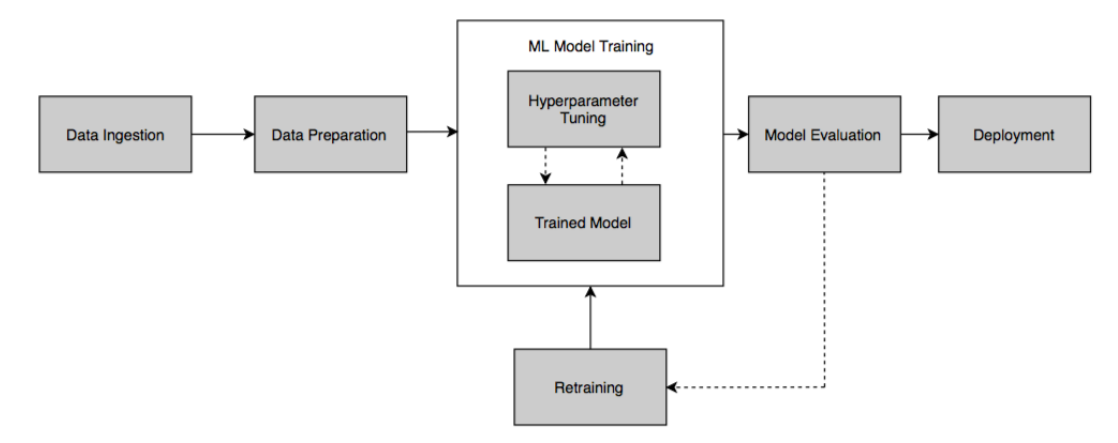


Fig., Machine Learning Pipeline

1. **General Methods to identify and rectify System Malware**

**Common Symptoms of a System Malware:**

* Onscreen Warnings about computer infection from a source other than your anti-virus software
* The browser redirects or a complete hijack of the browser
* You cannot open any EXE or Microsoft Installer (MSI) files
* The inability to change wallpaper or any desktop settings
* All entries under Start>Programs are empty and/or the C: drive is blank
* The anti-virus icon disappears from the system tray or cannot be started
* Random pop-ups show on-screen either in or out of the browser
* Unusual icons, erroneous start menu, or Device Manager entries appear

### **Malware Detection in our System:**

Look for unusual behaviour from your phone, tablet, or computer. Your device might have been infected with malware if it

* suddenly slows down, crashes, or displays repeated error messages
* won’t shut down or restart
* won’t let you remove software
* serves up lots of pop-ups, inappropriate ads, or ads that interfere with page content
* shows ads in places you typically wouldn’t see them, like government websites
* shows new and unexpected toolbars or icons in your browser or on your desktop
* uses a new default search engine, or displays new tabs or websites you didn’t open
* keeps changing your computer’s internet home page
* sends emails you didn’t write
* runs out of battery life more quickly than it should

If positive malware identification is made, you can use one of the options below. Remember if it does not work, we can take you through a clean OS reinstall to resolve the issue.

So, there are two main types of anti-virus.

1. Real-time anti-virus programs

They constantly watch for malware.

1. On-demand scanners

They search for malware infections when you open the program manually and run a scan.

**Rectification Process of System Malware:**

1. Stop shopping, banking, and doing other things online that involve usernames, passwords, or other sensitive information — until you get your device cleared of any malware.
2. Check to see if you have security software on your device — if not, download it. Find recommendations from independent review sites by doing a search online. Also ask friends and family for recommendations. Some software that claims to be security software to protect you from malware *is*malware, so it’s important to do your research.
3. Make sure your software is up to date. Check that all software — the operating system, security software, apps, and more — is up to date. Consider turning on automatic updates so your software always stays up to date.
4. Scan your device for malware. Run a malware or security Delete anything it identifies as a problem. You may have to restart your device for the changes to take effect. Run your scan again to make sure everything is clear. If the scan shows there are no more issues, you’ve likely removed the malware.

**Note:** If you’re not able to fix your device with the above steps, then upcoming steps may resolve the issue. When using either of these options, you risk losing data. If you’ve backed up your data regularly, you’ll minimize what you lose.

1. Recover your operating system. To find out how to recover your operating system (like Windows or Mac OS), visit your device manufacturer’s website. Recovering your system typically means you’ll get back a lot of the data stored on the device, so it’s a good alternative to reinstalling your operating system (step 6). That is, if it clears the malware problem. After recovering your operating system, you’ll want to go back to steps 2, 3 and 4 to ensure that you’ve removed the malware.
2. Reinstall your operating system. To find out how to reinstall your operating system (like Windows or Mac OS), visit your device manufacturer’s website. Reinstalling your system is the safest way to clean an infected device, but you’ll lose all of the data stored on the device that you haven’t backed up.

**Existing Systems**

* Malware detection by using window API sequence and machine learning
* Detecting unknown malicious code by applying classification techniques on oppose patterns
* Detecting scareware by mining variable length instructions sequence
* Accurate adware detection using oppose sequence extraction
* Detection of spyware by mining executable files
* Detection by using neural networks on the malware

**Review of Literature**

1. **Machine Learning for Android Malware Detection Using Permission and API Calls**

The Google Android mobile phone platform is one of the most anticipated smartphone operating systems on the market. The open-source Android platform allows developers to take full advantage of the mobile operation system, but also raises significant issues related to malicious applications. On one hand, the popularity of Android absorbs attention of most developers for producing their applications on this platform. The increased numbers of applications, on the other hand, prepares a suitable prone for some users to develop different kinds of malware and insert them in Google Android market or other third-party markets as safe applications. In this paper, we propose to combine permission and API

(Application Program Interface) calls and use machine learning methods to detect malicious Android Apps. In our design, the permission is extracted from each App's profile information and the APIs are extracted from the packed App file by using packages and classes to represent API calls. By using permissions and API calls as features to characterize each Apps, we can learn a classifier to identify whether an App is potentially malicious or not. An inherent advantage of our method is that it does not need to involve any dynamical tracing of the system calls but only uses simple static analysis to find system functions involved in each App. In addition, because permission settings and APIs are always available for each App, our method can be generalized to all mobile applications. Experiments on real-world Apps with more than 1200 malware and 1200 benign samples validate the algorithm performance.

# Automatic analysis of malware behaviour using machine learning

Malicious software – so called malware – poses a major threat to the security of computer systems. The amount and diversity of its variants render classic security defences ineffective, such that millions of hosts in the Internet are infected with malware in the form of computer viruses, Internet worms and Trojan horses. While obfuscation and polymorphism employed by malware largely impede detection at file level, the dynamic analysis of malware binaries during run-time provides an instrument for characterizing and defending against the threat of malicious software. In this article, we propose a framework for the automatic analysis of malware behavior using machine learning. The framework allows for automatically identifying novel classes of malware with similar behaviour (clustering) and assigning unknown malware to these discovered classes (classification). Based on both, clustering and classification, we propose an incremental approach for behavior-based analysis, capable of processing the behavior of thousands of malware binaries on a daily basis. The incremental analysis significantly reduces the run-time overhead of current analysis methods, while providing accurate discovery and discrimination of novel malware variants.

# Malware detection using machine learning

We propose a versatile framework in which one can employ different machine learning algorithms to successfully distinguish between malware ﬁles and clean ﬁles, while aiming to minimize the number of false positives. In this paper we present the ideas behind our framework by working ﬁrstly with cascade one-sided perceptron’s and secondly with cascade kernelized one-sided perceptron’s. After having been successfully tested on medium-size datasets of malware and clean ﬁles, the ideas behind this framework were submitted to a scaling-up process that enable us to work with very large datasets of malware and clean ﬁles.

1. **Automatic Analysis of Malware Behavior using Machine Learning**

Malicious software—so called malware—poses a major threat to the security of computer systems. The amount and diversity of its variants render classic security defences ineffective, such that millions of hosts in the Internet are infected with malware in form of computer viruses, Internet worms and Trojan horses. While obfuscation and polymorphism employed by malware largely impede detection at file level, the dynamic analysis of malware binaries during run-time provides an instrument for characterizing and defending against the threat of malicious software. In this article, we propose a framework for automatic analysis of malware behavior using machine learning. The framework allows for automatically identifying novel classes of malware with similar behavior (clustering) and assigning unknown malware to these discovered classes (classification). Based on both, clustering and classification, we propose an incremental approach for behavior-based analysis, capable to process the behavior of thousands of malware binaries on a daily basis. The incremental analysis significantly reduces the run-time overhead of current analysis methods, while providing an accurate discovery and discrimination of novel malware variants.

# A Review of Malware Classification Methods using Machine Learning

This paper talks about methods, problems and solutions to Malware Classification using Machine Learning. It is believed that the amount of malicious software being released might surpass the release of authoritative software. Since malware gets more sophisticated every year, this results in a need to shift from traditional methods and make the system automatically learn. The main focus here is studying machine learning methods along with their problems for detection and classification. Feature selection and high false-positive problems are explained and solutions to them are presented. Opcode, n-gram opcode, image-based classification techniques and are then compared. These methods will help cleaning malware and to classify them into their families. The results based on accuracy are better when using n-gram opcode classification as compared to normal opcode and image-based classifiers but using an ensemble method combines the benefits of both methods like less overfitting and low FPR, eventually the results display increased classification accuracy and provide overall better classification of malware.

**Objective of our Study**

The goal is to determine the best feature representation method and how the features should be extracted, the most accurate algorithm that can distinguish the malware families with the lowest error rate.

The goal of this project is to develop the proof of concept for the machine learning based malware classification. This representation will be utilized for the extraction of the behaviour of the malware samples, which will be used as an input to the machine learning algorithms.

**Problem identification:**

* Detecting unknown malicious code by applying classifications techniques on oppose pattern: Evaluated number of experiments and found that setting of 2 grams, TF, using 300 features selected by Def measured outperform the perform lacks ML specific techniques
* Detecting scareware by Mining variable length instructions sequence: This paper present the static analysis method based on data mining which extends the general heuristic detection techniques using a variable length instructions sequence mining approach for purpose of scareware detection but metrics specific and unsupervised techniques un included can be broken

**Discussion of Algorithms been used**

Different kinds of Machine Learning Algorithms been applied as follows

* DECISION TREE
* SVM
* Random forest
* Ada boost

**Decision Trees (Supervised Learning – Classification/Regression)**

A decision tree is a flow-chart-like tree structure that uses a branching method to illustrate every possible outcome of a decision. Each node within the tree represents a test on a specific variable – and each branch is the outcome of that test.

**Support Vector Machine Algorithm (Supervised Learning - Classification)**

Support Vector Machine algorithms are supervised learning models that analyse data used for classification and regression analysis. They essentially filter data into categories, which is achieved by providing a set of training examples, each set marked as belonging to one or the other of the two categories. The algorithm then works to build a model that assigns new values to one category or the other.

**Random Forests (Supervised Learning – Classification/Regression)**

Random forests or ‘random decision forests’ is an ensemble learning method, combining multiple algorithms to generate better results for classification, regression and other tasks. Each individual classifier is weak, but when combined with others, can produce excellent results. The algorithm starts with a ‘decision tree’ (a tree-like graph or model of decisions) and an input is entered at the top. It then travels down the tree, with data being segmented into smaller and smaller sets, based on specific variables.

**Ada Boost (Supervised Learning)**

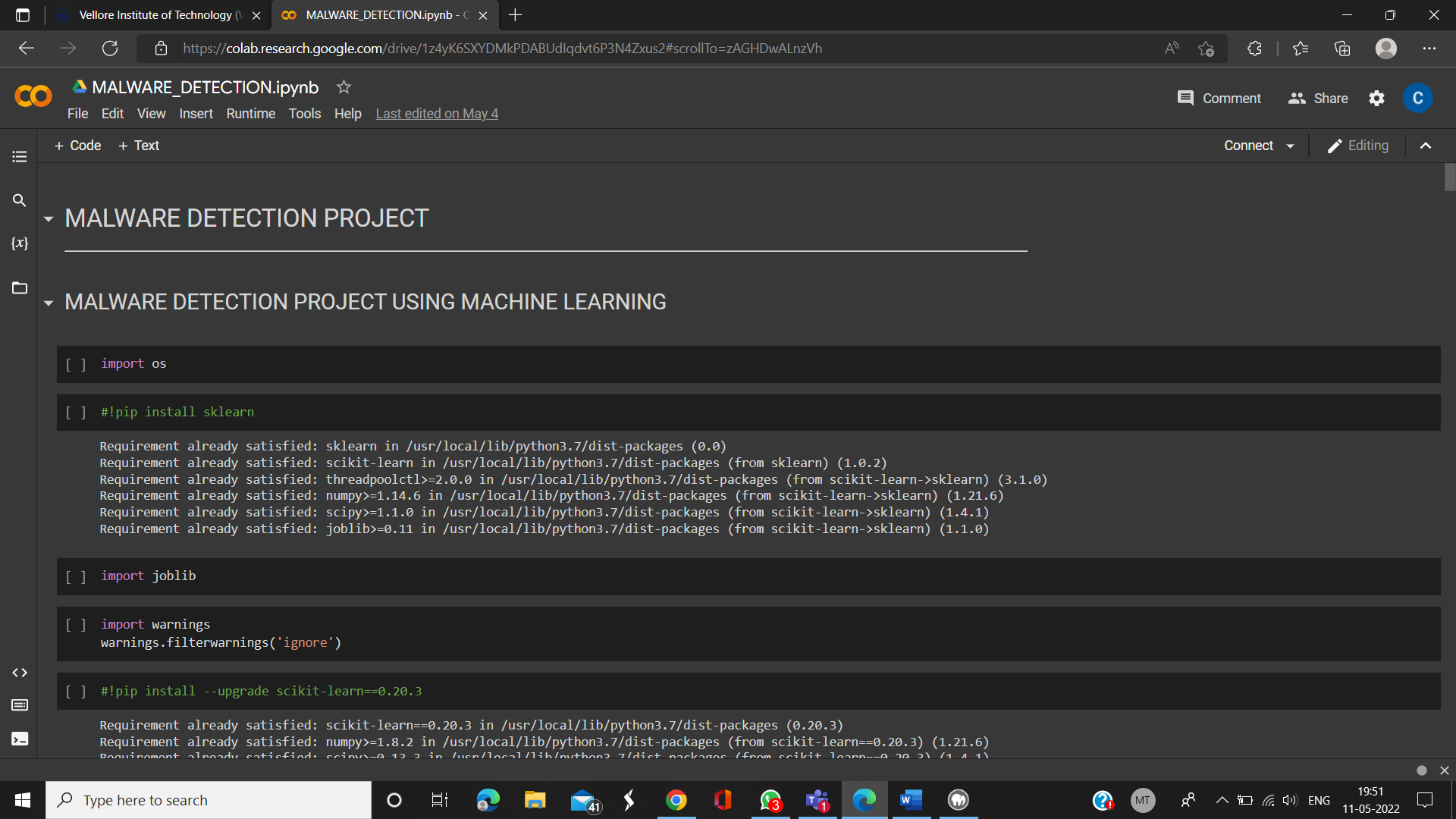
AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps. Ada boost the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is used to reduce bias as well as variance for [supervised learning](https://www.mygreatlearning.com/academy/learn-for-free/courses/supervised-machine-learning-tutorial?gl_blog_id=15100). It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones.

**Linear Regression (Supervised learning – Classification)**

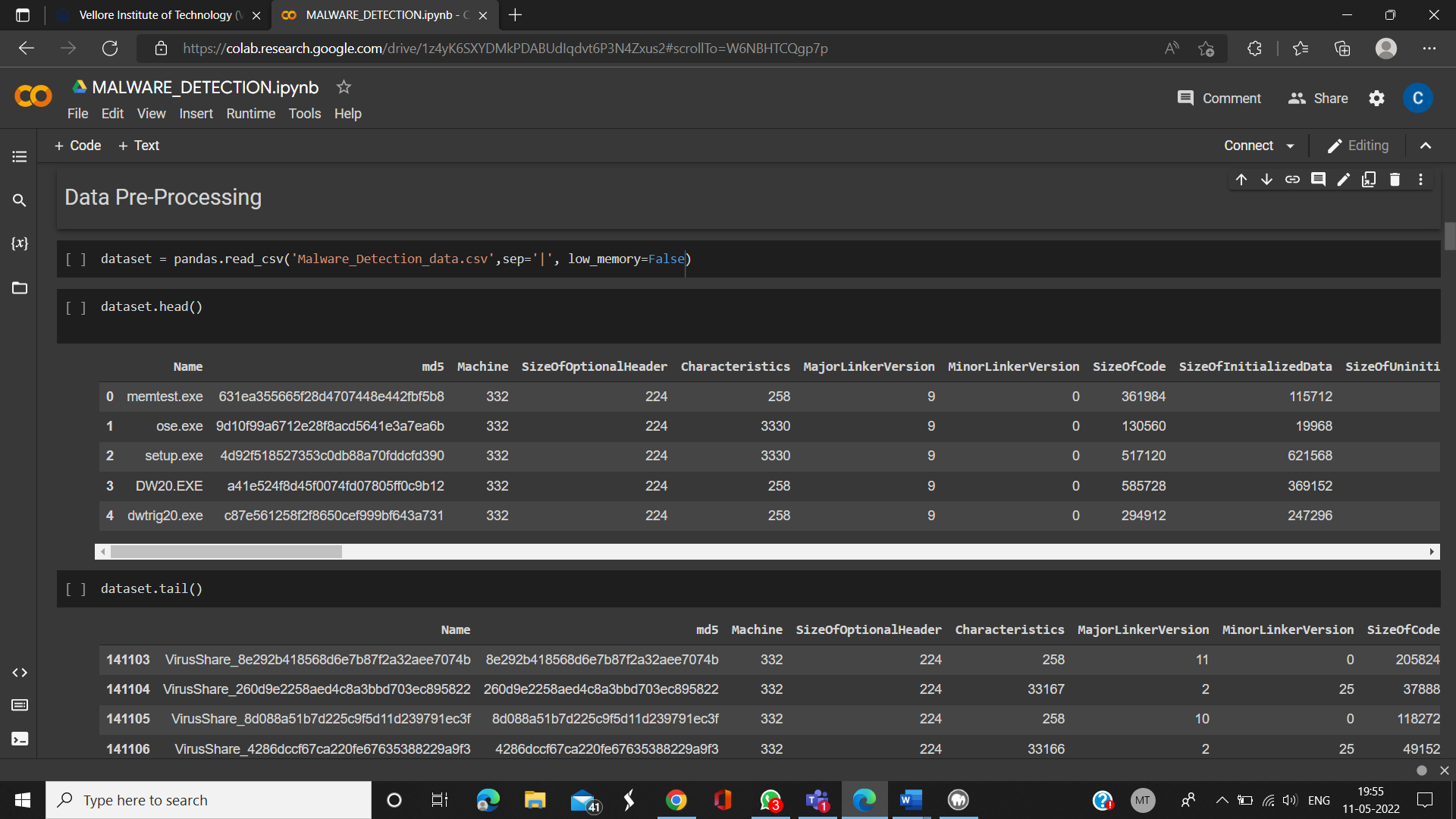
Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

**Findings of our Study**

**Results and Discussion**

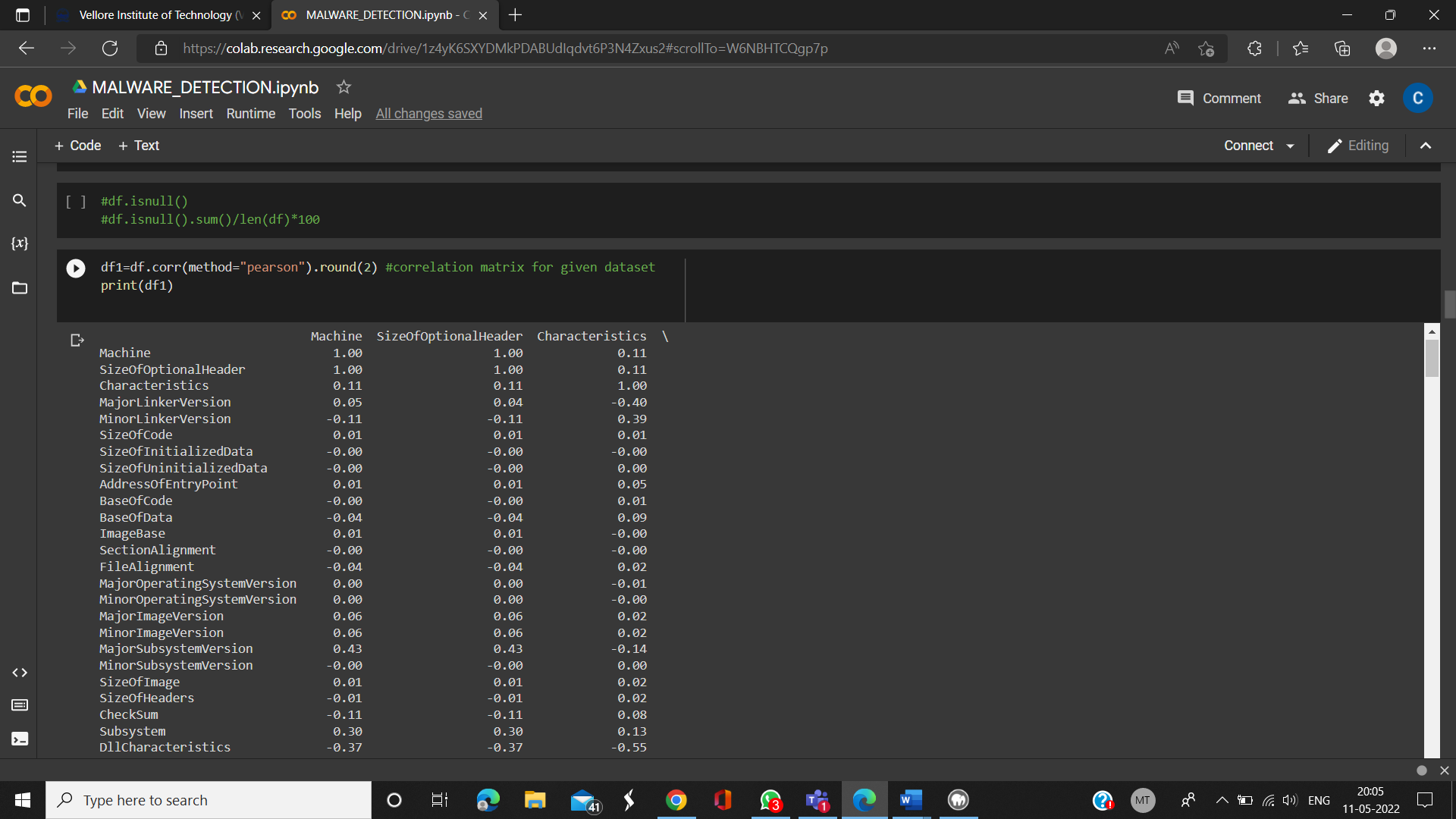


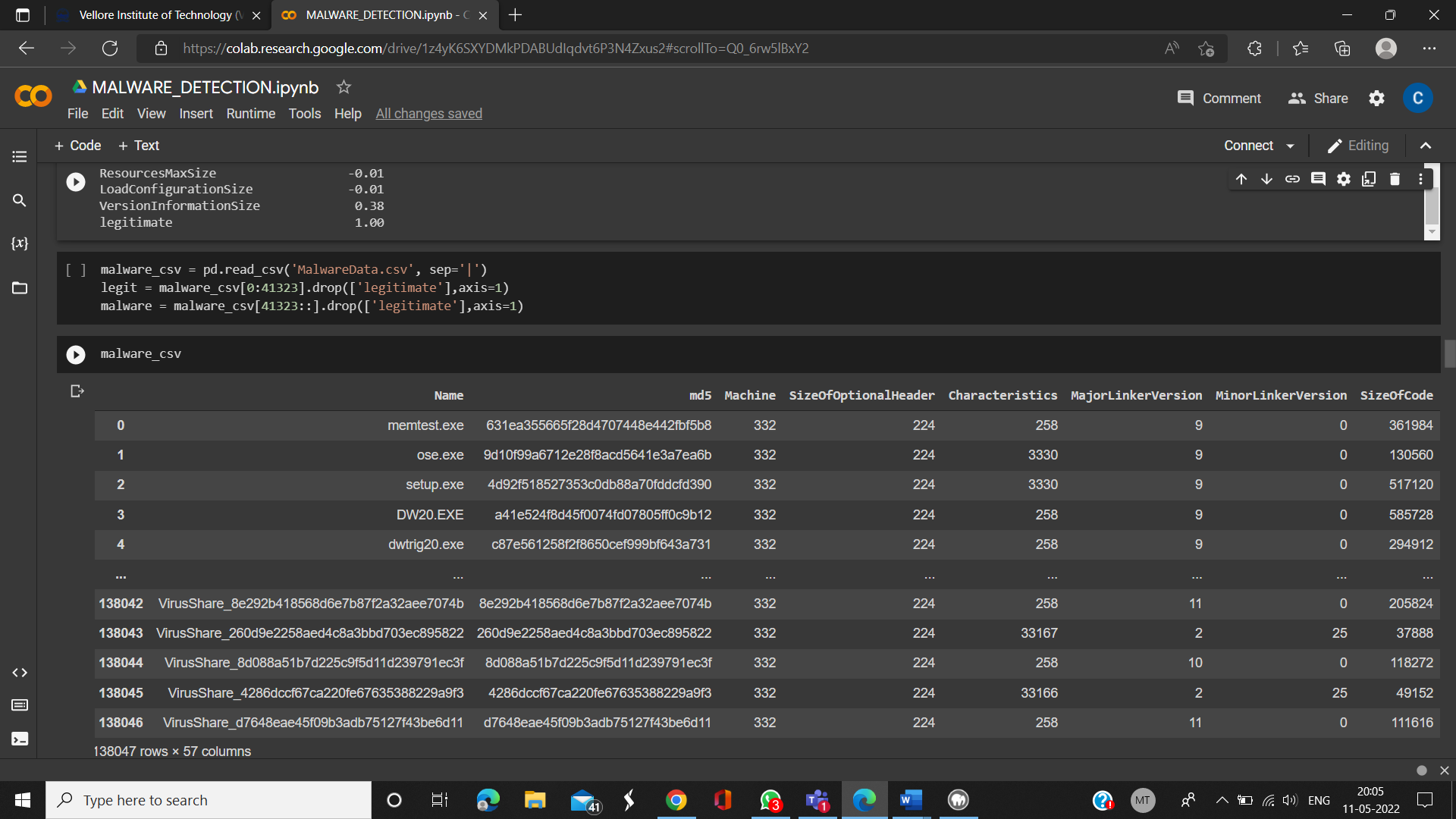
In the first few lines of the code, we have added all the required header files along the necessary modules that needed to be downloaded

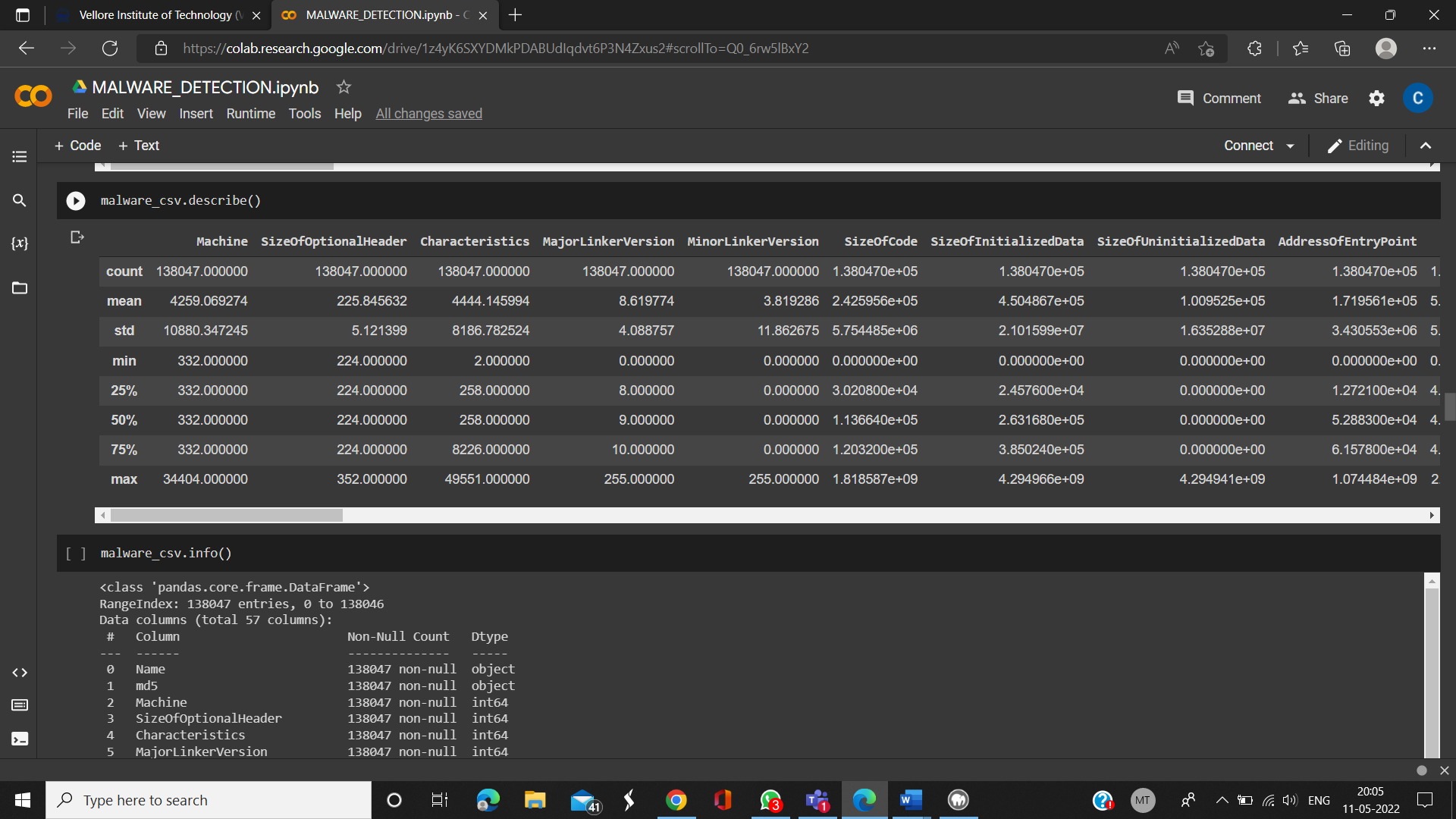


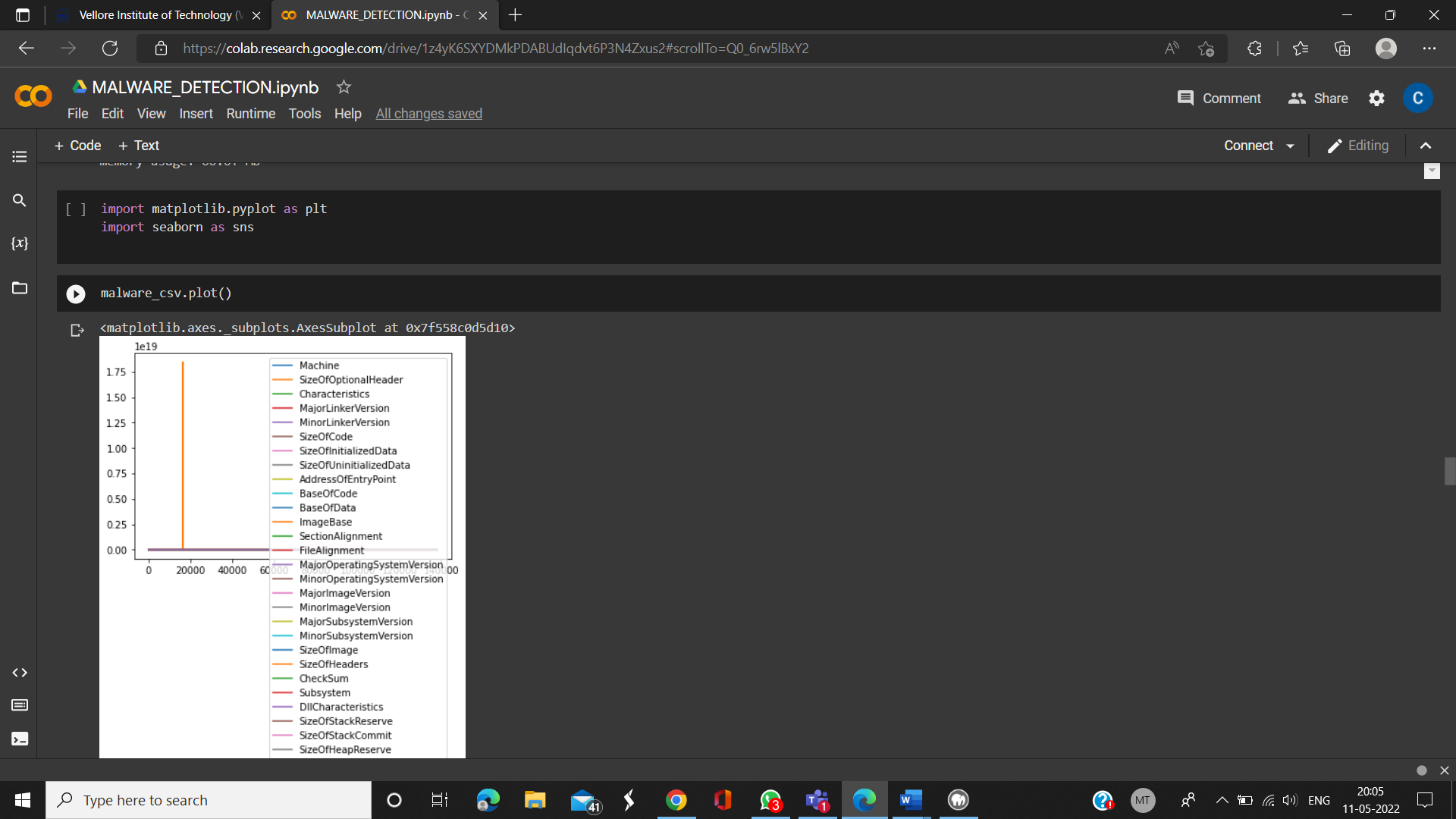
In the data pre-processing stage, we have followed all the necessary and required pre-

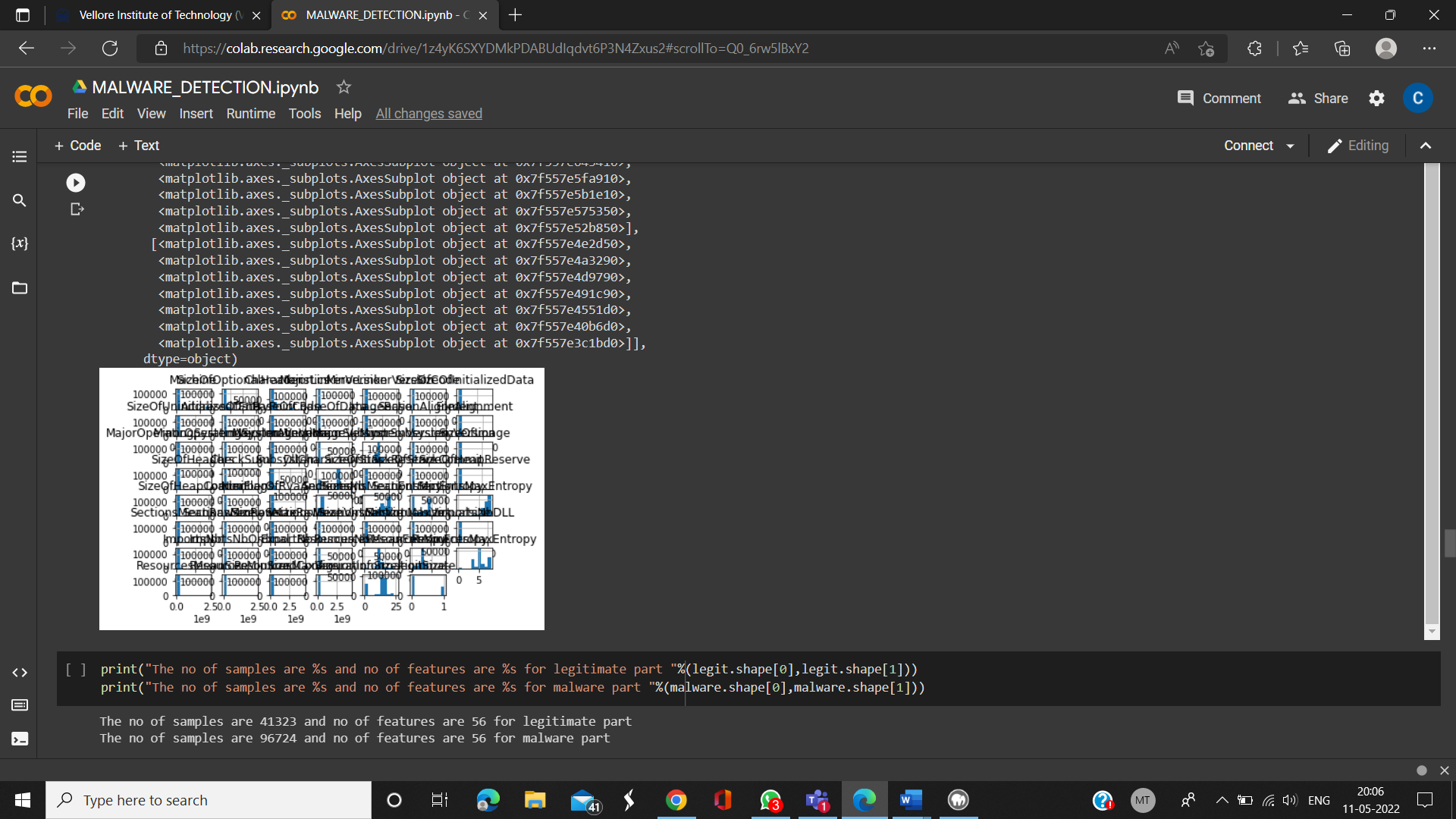
processing of the dataset. Since we have 2 different datasets, with different data, we have selected one of the datasets based on which we have carried out the whole projects. So in the first stage we have done dataset display using head (), tail (), correlation matrix to display which attributes has more importance on deciding the response output variable, followed by null entry checking, and dataset splitting. Since our dataset has the class label in the form of 0 and 1’s hence we avoid the class label encoding method along with the display and visualization of the dataset.



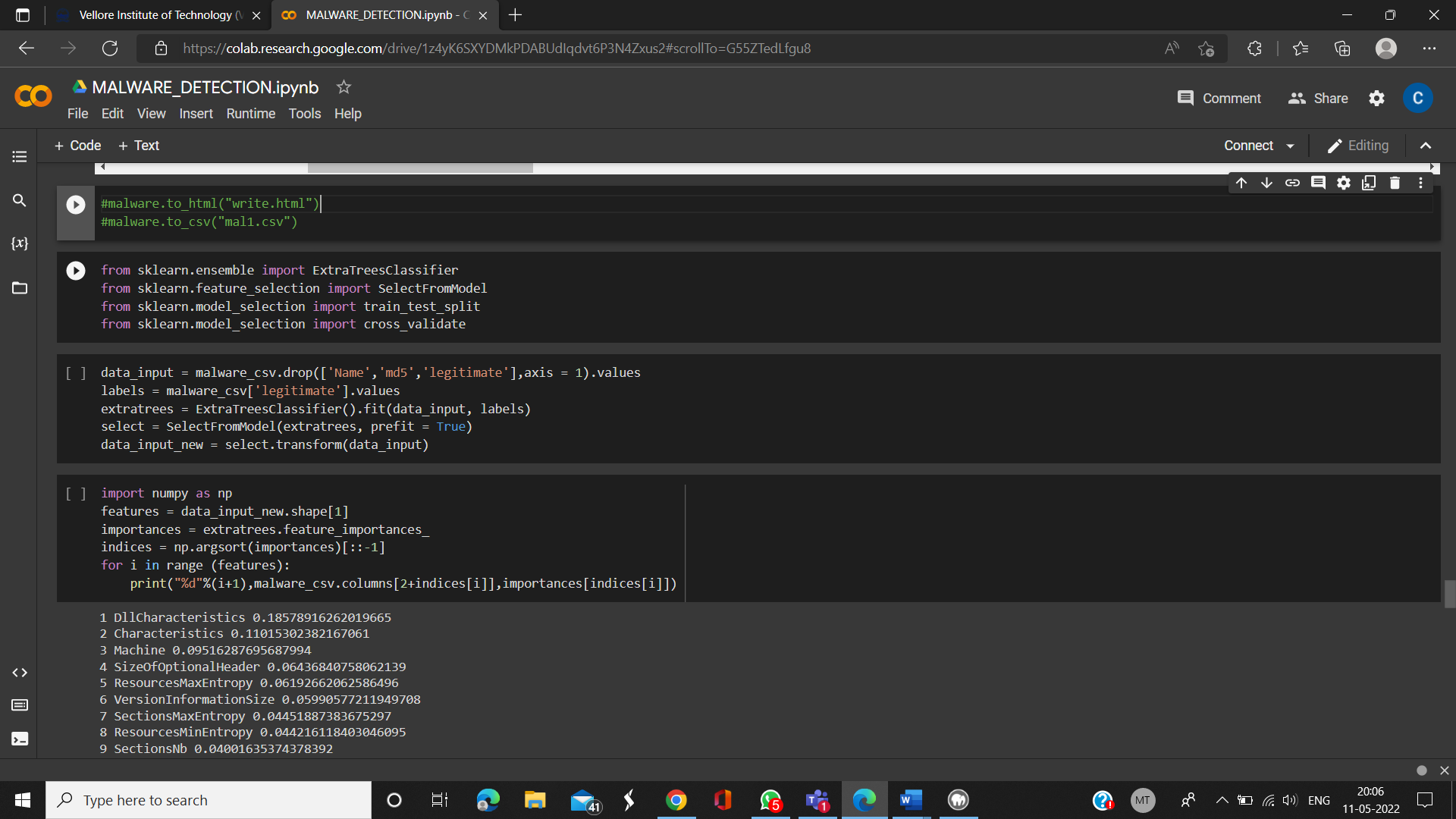




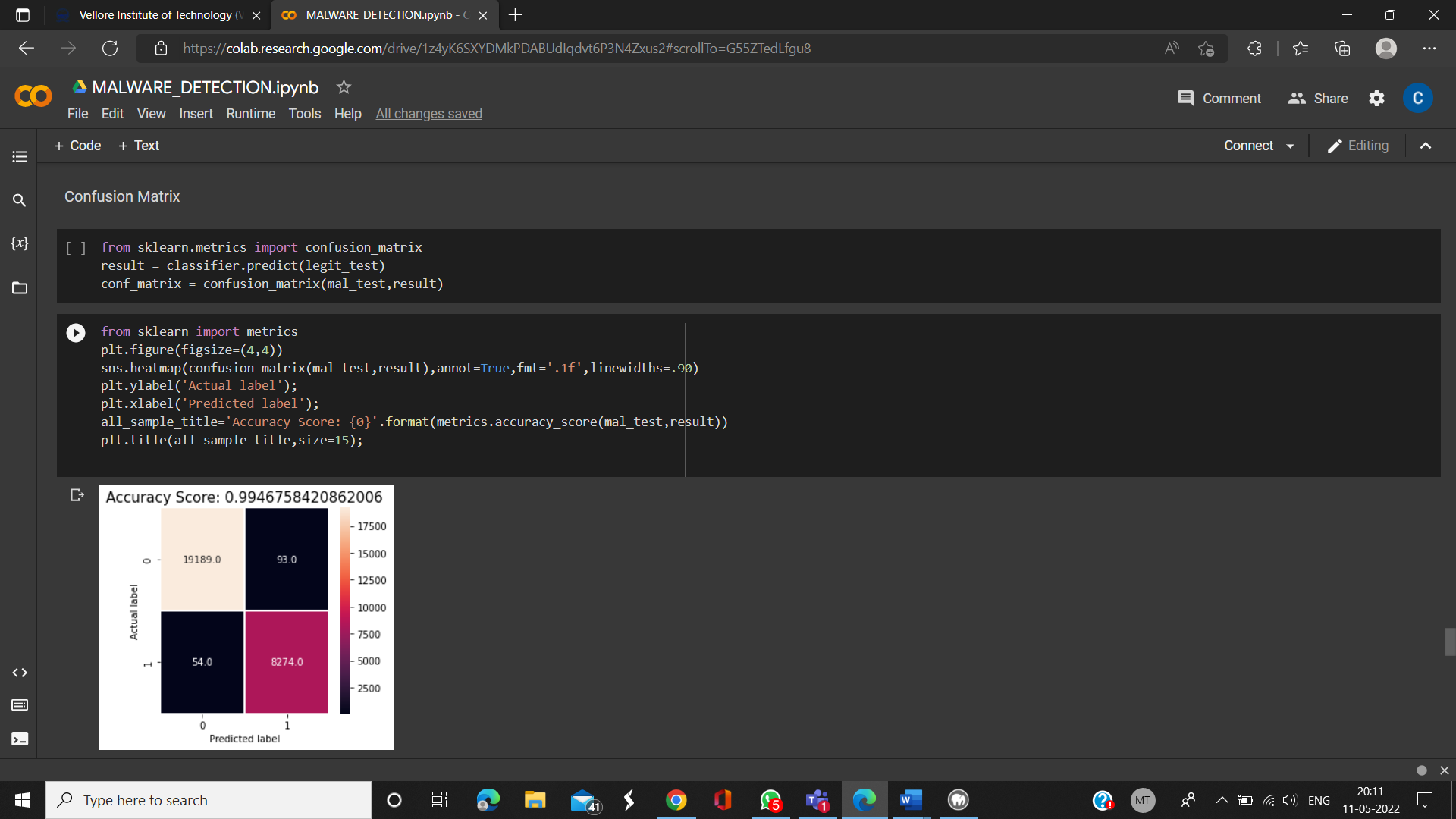


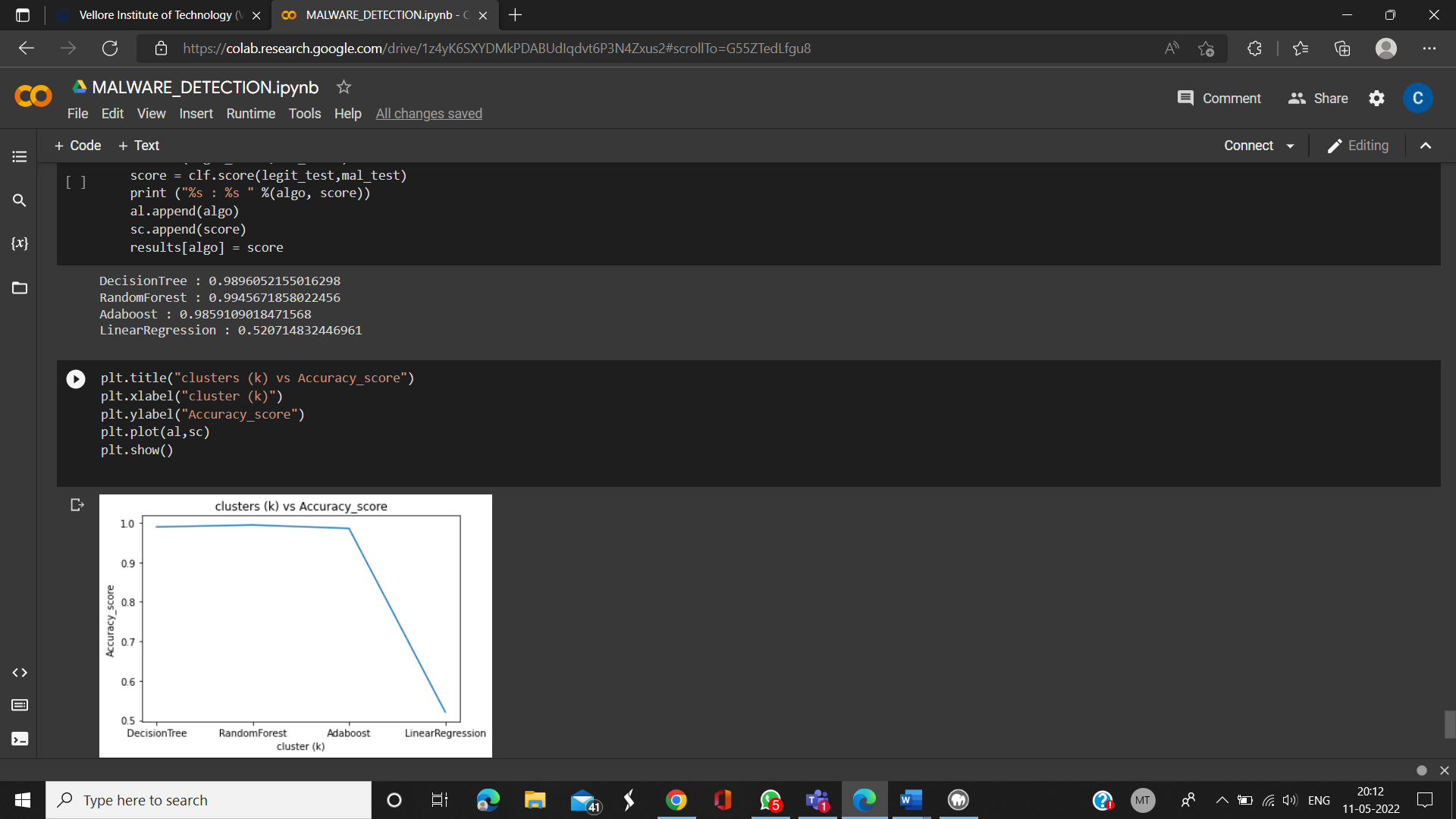


Once all the dataset pre-processing is done, we export the dataset to the cloud along with the in-built to\_html() where the dataset is converted to the table format for the html use.

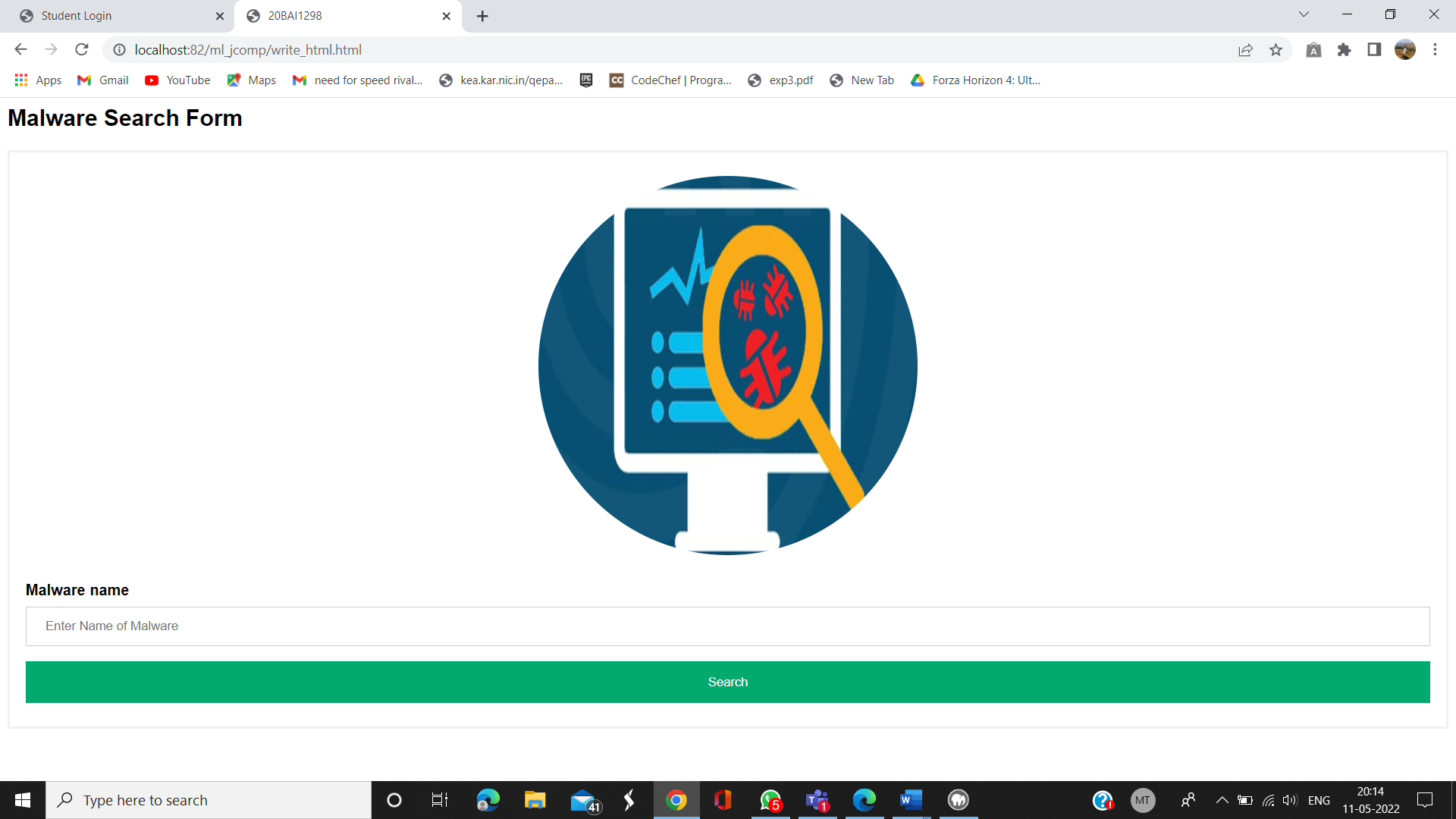


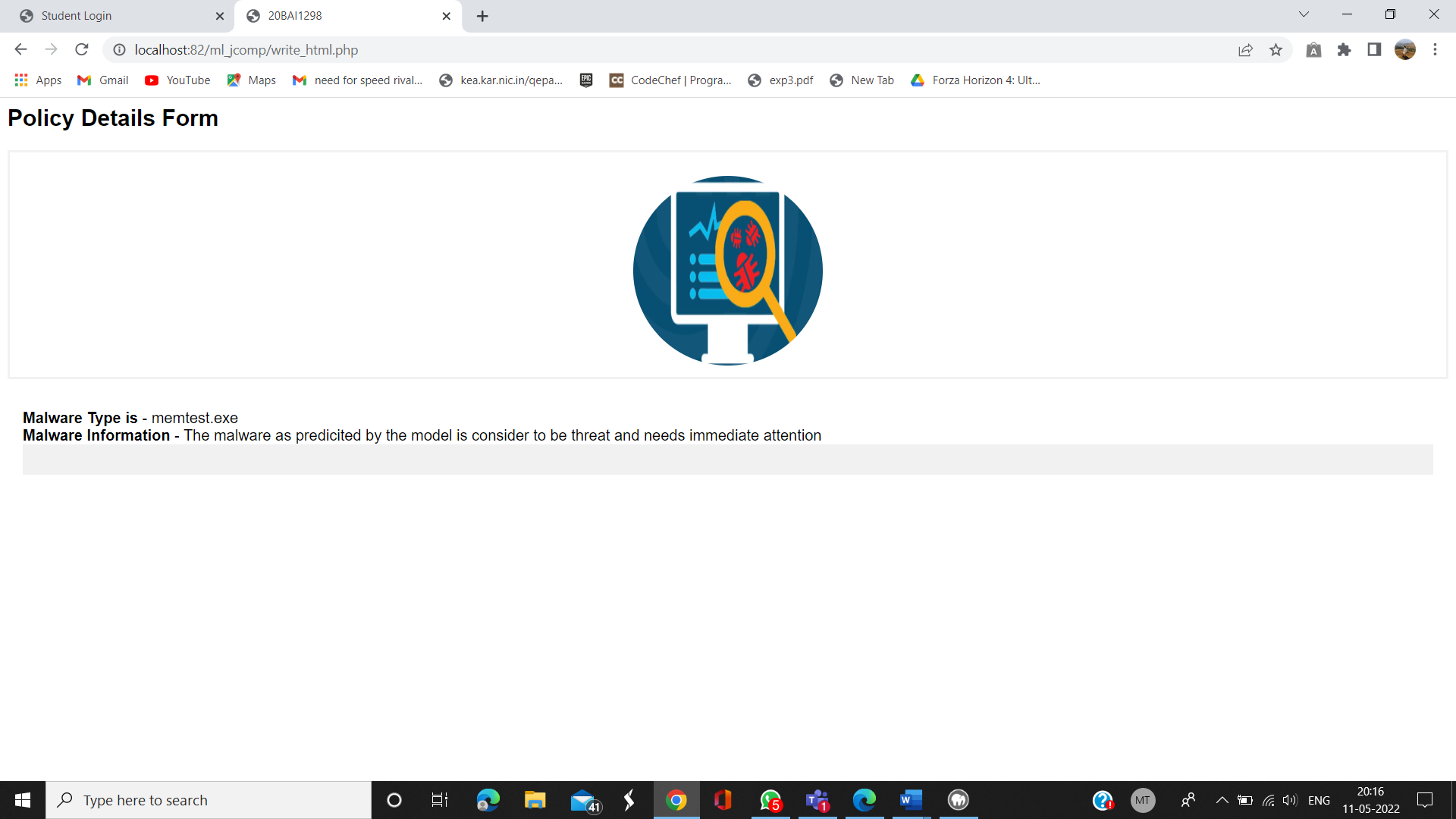
Now we have used different algorithm and analysed along with the display of the accuracy score.





**Front-End**





In our website we have included the features where any anti-virus developing company can view the whole information and details but few to constraints, we will be displaying the short description whether the virus is harmful or not based on the prediction done by the algorithm.

In classification problems, different models gave different results. The lowest accuracy was achieved by Linear Regression (52.07%), followed by Decision Tree (98.96%). The highest accuracy was achieved with Random Forest model and Adaboost model, and it was equal to for multi-class classification 98.59% and 99.45% for binary classification respectively.

**Conclusion**

Overall, the goals defined for this study were achieved. The desired feature extraction and representation methods were selected and the selected machine learning algorithms were applied and evaluated.

**Future works**

The study performed in this project was a proof-of-concept. We’d thought of implementing it in several future plans and further improvements related to the practical implementation of this project can be identified.

**Online interaction of user with our setup**

Since, defining malware in our home pc system is somewhat a hard process for normal users. If they need to confirm that means, they need to look for a computer analysist or should hand over their pc to any of the nearby service centre. To overcome all such difficulties in our day-to-day routine life, we’d thought of implementing this model as an online tool where the user can identify whether their pc is been corrupted with malware or not. So, we’d done a basic interface for this one and will try to make it more dynamic with the user.

**Use a wider dataset**

Although the dataset that was used in this study is broad, covering most of the malware types that are relevant to the modern world, it does not cover all possible types. Collecting a malware dataset is a tedious task that requires a lot of time and effort. For more accurate evaluation of the predictors, it is advised to test the models on all the possible types of malwares: spyware, adware, rootkits, backdoor, banking malware, etc. In addition to that, it is important to understand that the model will only be able to predict the samples of the families that it has seen earlier. In other words, in a real-world application, the maximum number of possible families should be used before the launch of the project for real-world environments.

**Use pre-selected APIs**

In this work, the big overhead in the data processing was created by the need of selecting the relevant API calls and removing the redundant ones. For further implementation, only the APIs that were identified as relevant in this study can be used. This will decrease the amount of time required for data pre-processing, reduce the performance requirements of the machine on which the analysis is being done and decrease the level of feature selection to be made. However, it should be noted, that for more accurate description, the relevant APIs should be extracted from the biggest possible dataset. Also, it is advised to select the relevant APIs per malware family, as this will result in another level of flexibility and accuracy.

**Reference**

1. [**https://ieeexplore.ieee.org/abstract/document/6735264**](https://ieeexplore.ieee.org/abstract/document/6735264)
2. [**https://content.iospress.com/articles/journal-of-computer-security/jcs410**](https://content.iospress.com/articles/journal-of-computer-security/jcs410)
3. [**https://www.researchgate.net/publication/224089748\_Malware\_detection\_using\_machine\_learning**](https://www.researchgate.net/publication/224089748_Malware_detection_using_machine_learning)
4. [**https://do.tu-berlin.de/bitstream/11303/11388/1/tr-2009-18.pdf**](https://do.tu-berlin.de/bitstream/11303/11388/1/tr-2009-18.pdf)
5. [**https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=3769906**](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3769906)
6. [**https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/**](https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/)
7. [**https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/**](https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/)
8. [**https://blog.paperspace.com/adaboost-optimizer/**](https://blog.paperspace.com/adaboost-optimizer/)
9. [**http://theprofessionalspoint.blogspot.com/2019/03/difference-between-random-forest-and.html#:~:text=Random%20Forest%20uses%20parallel%20ensembling,instead%20uses%20a%20sequential%20approach**](http://theprofessionalspoint.blogspot.com/2019/03/difference-between-random-forest-and.html#:~:text=Random%20Forest%20uses%20parallel%20ensembling,instead%20uses%20a%20sequential%20approach)**.**
10. [**https://vitalflux.com/differences-between-random-forest-vs-adaboost/**](https://vitalflux.com/differences-between-random-forest-vs-adaboost/)
11. [**https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725**](https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725)
12. [**https://www.analyticsvidhya.com/blog/2021/05/logistic-regression-supervised-learning-algorithm-for-classification/**](https://www.analyticsvidhya.com/blog/2021/05/logistic-regression-supervised-learning-algorithm-for-classification/)
13. [**https://www.geeksforgeeks.org/understanding-logistic-regression/**](https://www.geeksforgeeks.org/understanding-logistic-regression/)
14. [**https://www.tutorialspoint.com/machine\_learning\_with\_python/machine\_learning\_with\_python\_classification\_algorithms\_logistic\_regression.htm**](https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_logistic_regression.htm)
15. [**https://colab.research.google.com/github/goodboychan/chans\_jupyter/blob/master/\_notebooks/2020-06-03-01-Decision-tree-for-classification.ipynb**](https://colab.research.google.com/github/goodboychan/chans_jupyter/blob/master/_notebooks/2020-06-03-01-Decision-tree-for-classification.ipynb)
16. [**https://colab.research.google.com/github/daniyal9538/GeneralProjects/blob/master/Decision\_Tree\_Tutorial.ipynb**](https://colab.research.google.com/github/daniyal9538/GeneralProjects/blob/master/Decision_Tree_Tutorial.ipynb)
17. [**https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.08-Random-Forests.ipynb**](https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.08-Random-Forests.ipynb)
18. [**https://colab.research.google.com/github/lewtun/hepml/blob/master/notebooks/lesson03\_random-forest-from-scratch.ipynb**](https://colab.research.google.com/github/lewtun/hepml/blob/master/notebooks/lesson03_random-forest-from-scratch.ipynb)