A Project Report on

Detecting Malicious Activity to Stop Attacks using Machine Learning

Submitted By :

Akshay Bura

Tanmay Doshi

Sakshi Deshpande

TABLE OF CONTENTS

Abstract

The purpose of this project is to introduce the functionality and accuracy of five different machine learning algorithms to detect if an executable is infected or clean. There are various software or code elements that are intended to hijack computer systems to steal information or destroy it. We will somehow understand these malware programs. After a quick introduction to the phenomenon, we will show how malware has evolved over time. The following is a presentation of various protection techniques. Next, we will discuss the importance of machine learning in solving this situation. The algorithms used in this article and their advantages will be introduced. Machine learning is widely used in this area by antivirus and antimalware programs, as well as these malicious programs. Polymorphing malware, for example, uses machine learning algorithms to encode itself differently each time it infiltrates a new population, making it increasingly difficult to detect. Chapter 3 describes how to use the pefile module to understand PE files and their structure.

Introduction

Due to the usage of signature-based techniques, conventional detection technologies like antivirus software and intrusion detection/prevention systems are unable to identify malware that has not yet been observed.

Therefore, it's important to precisely identify this type of malware in Windows environments. This project introduces a machine learning (ML)-based malware detection system that extracts characteristics from the header of a portable executable file to determine whether the executable is dangerous or not.

To deal with the virus, a number of ML models are employed once the data has been preprocessed, including Random Forest, Support Vector Machine (SVM), Decision Tree, AdaBoost, Gaussian Naive Bayes (GNB), and Gradient Boosting. Additionally, a comparison of ML models is done to determine which is best for the specified issue.

Objectives

- To perform data pre-processing that will prepare appropriate format to be input to Machine Learning classifiers.
- To develop two representatives supervised machine learning models; Support vector machine and artificial neural network.
- To evaluate the performance of Support vector machine and artificial neural network to classify for new malicious executables programs.

Scope

- Malware detection and prevention technologies are widely useful for servers, gateways, user workstations, and mobile devices, with some tools offering the capability to centrally monitor malware detection software installed on multiple systems or computers.
- Can be used by data scientists, CS-IT engineers & students for malware analysis.
- Anti-virus companies can use this application for testing files foreign to their existing dataset.

Literature Survey

Here a flexible architecture is proposed that enables the effective application of various machine learning techniques to discriminate between malware files and clean files while attempting to reduce the incidence of false positives. By first working with cascade one-sided perceptrons and then with cascade kernelized one-sided perceptrons, the concepts underlying our system are shown in this research. The concepts underlying this framework were put through a scaling-up procedure that enables it to work with very big datasets of malware and clean files after being successfully tested on medium-sized datasets of malware and clean files [1].

Created a malware detection website with the front end utilising Flask, HTML, Bootstrap, and CSS. To develop the model, entropy calculations using PE files were used. Applied a number of decision-making algorithms and obtained a Random Forest accuracy of 99%. To upload and deliver files for analysis, API queries were used. PE files have been classified as "Legitimate" or "Malicious" [2].

The purpose of this study is to discuss how data science is used to detect malware as well as analyse how machine learning techniques are used to analyse malware. Training assault detection systems enables the creation of better defences that can recognise novel tactics. This study demonstrates that a variety of models can be used to assess their detectability. The demonstration results show how it is possible to compare different machine learning (ML) techniques to evaluate malware [3].

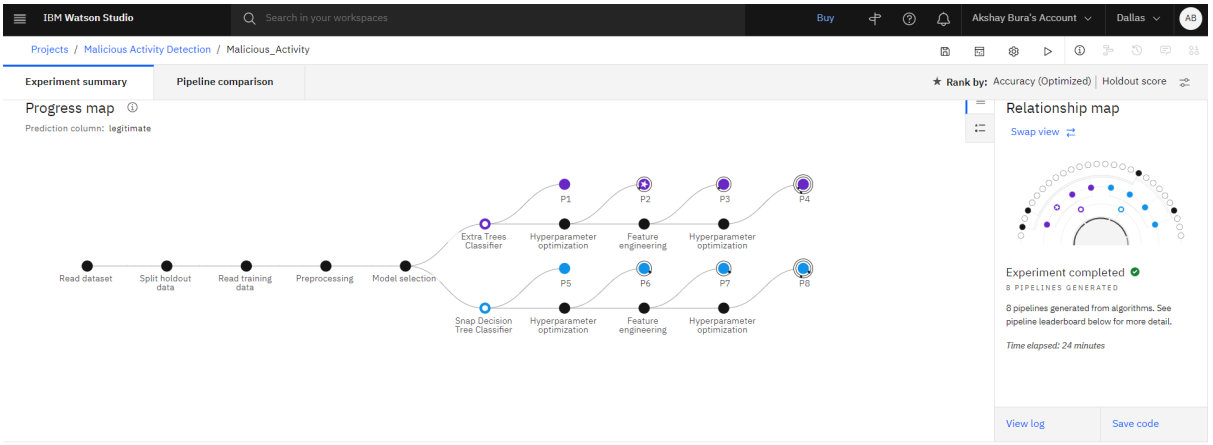
Simple pre-execution rules that had been manually set in the early days of the cyber era were frequently sufficient to identify dangers because there weren't many malware threats at the time. Manually constructed detection rules were no longer practicable due to the Internet's quick development and the resulting rise in malware, necessitating the development of new, cutting-edge protection solutions. In order to improve their malware detection and categorization, anti-malware organisations turned to machine learning, an area of computer science that has previously been employed successfully in picture recognition, searching, and decision-making. Utilising multiple types of data from host, network, and cloud-based anti-malware components, machine learning today improves malware detection [4].

Problem Definition

Windows is a well-known operating system with a graphical user interface that offers features like storage, the ability to execute third-party software, the ability to play films, network connectivity, etc. By focusing on their accessibility, these services' intended function can be destroyed. One of the main security issues for the Windows platform is malware. Any sort of computer programme known as malware interferes with the accessibility of computer services. Due to the usage of signature-based techniques, conventional detection technologies like antivirus software and intrusion detection/prevention systems are unable to identify malware that has not yet been observed. Therefore, it's important to precisely identify this type of malware in Windows environments.

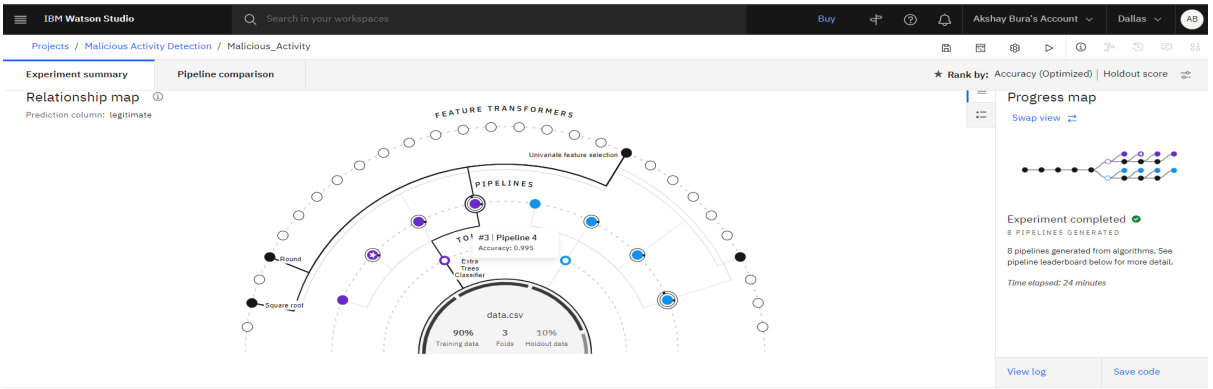
Implementation

Implemented different classification algorithms on IBM Service (Auto AI).



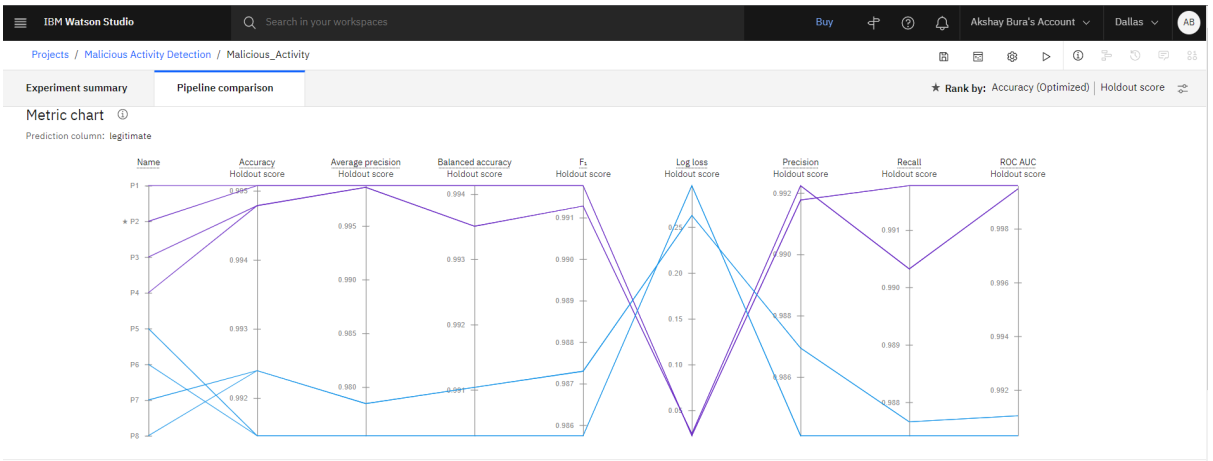
Pipeline leaderboard

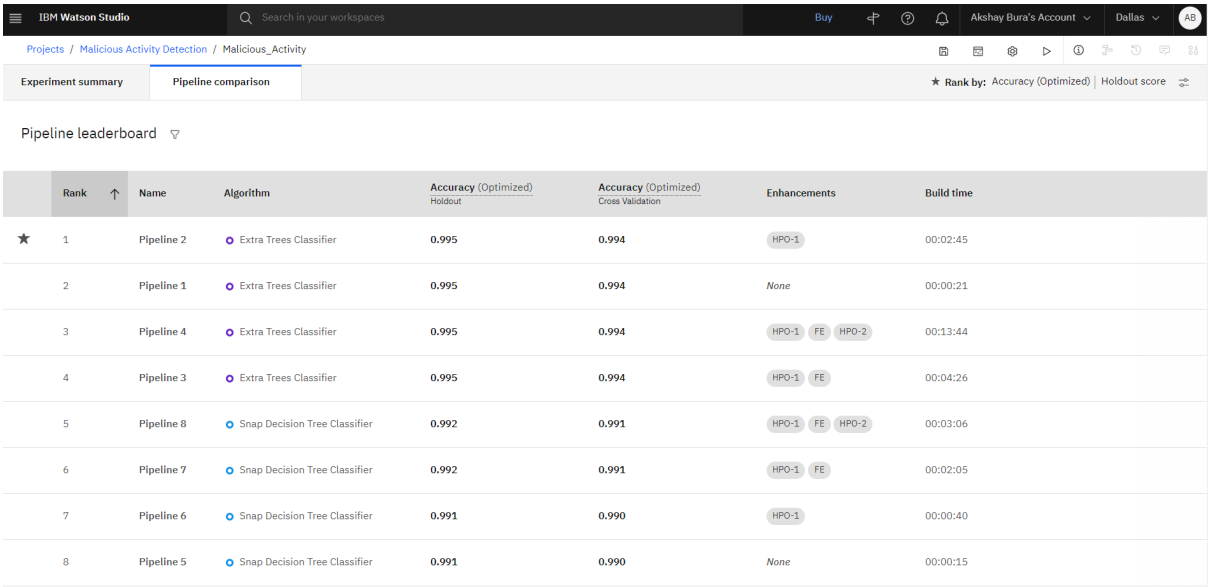
	Rank	Name	Algorithm	Accuracy (Optimized) Holdout	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 2	Extra Trees Classifier	0.995	0.994	HPO-1	00:02:45



Pipeline leaderboard

	Rank	Name	Algorithm	Accuracy (Optimized) Holdout	Accuracy (Optimized) Cross Validation	Enhancements	Build time





	Rank	Name	Algorithm	Accuracy (Optimized) Holdout	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 2	Extra Trees Classifier	0.995	0.994	HPO-1	00:02:45
	2	Pipeline 1	Extra Trees Classifier	0.995	0.994	*None*	00:00:21
	3	Pipeline 4	Extra Trees Classifier	0.995	0.994	HPO-1 FE HPO-2	00:13:44
	4	Pipeline 3	Extra Trees Classifier	0.995	0.994	HPO-1 FE	00:04:26
	5	Pipeline 8	Snap Decision Tree Classifier	0.992	0.991	HPO-1 FE HPO-2	00:03:06
	6	Pipeline 7	Snap Decision Tree Classifier	0.992	0.991	HPO-1 FE	00:02:05
	7	Pipeline 6	Snap Decision Tree Classifier	0.991	0.990	HPO-1	00:00:40
	8	Pipeline 5	Snap Decision Tree Classifier	0.991	0.990	*None*	00:00:15

Using Auto AI service we have found that ExtraTrees Classifier is giving more accurate scores and we will be using this model to train and test our dataset.

Result

```
In [13]: model = { "DecisionTree":tree.DecisionTreeClassifier(max_depth=10),
                   "RandomForest":ek.RandomForestClassifier(n_estimators=50),
                   "ExtraTrees":ek.ExtraTreesClassifier(),
                   "GNB":GaussianNB(),
                   "LogisticRegression":LogisticRegression()
                 }

In [14]: results = {}
         for algo in model:
             clf = model[algo]
             clf.fit(X_train,y_train)
             score = clf.score(X_test,y_test)
             print ("%s : %s " %(algo, score))
             results[algo] = score

         DecisionTree : 0.9908366533864542
         RandomForest : 0.9944585295182905
         ExtraTrees : 0.99380659181456
         GNB : 0.6979355306048534
         LogisticRegression : 0.6978993118435349
```

This is the snippet of accuracy on the Jupyter Notebook.

```
[19]: clf = model[winner]
      res = clf.predict(X_new)
      mt = confusion_matrix(y, res)
      print("False positive rate : %f %%" % ((mt[0][1] / float(sum(mt[0])))*100))
      print('False negative rate : %f %%' % ( (mt[1][0] / float(sum(mt[1]))*100)))

      False positive rate : 0.100285 %
      False negative rate : 0.171817 %

[20]: # Load classifier
      clf = joblib.load('C:/Users/Acer/Documents/ML based Malicious Activity Detection/classifier/classifier.pkl')
      #load features
      features = pickle.loads(open(os.path.join('C:/Users/Acer/Documents/ML based Malicious Activity Detection/classifier/features.pkl'),'rb').read())

[21]: %run "C:\Users\Acer\Documents\ML based Malicious Activity Detection\malware_test.py" "C:/Users/Acer/Documents/ML based Malicious Activity Detection/msedge.

      The file msedge.exe is legitimate
```

```
# Load classifier
clf = joblib.load('C:/Users/Acer/Documents/ML based Malicious Activity Detection/classifier/classifier.pkl')
#load features
features = pickle.loads(open(os.path.join('C:/Users/Acer/Documents/ML based Malicious Activity Detection/classifier/features.pkl'),'rb').read())

%run "C:\Users\Acer\Documents\ML based Malicious Activity Detection\malware_test.py" "C:/Users/Acer/Documents/ML based Malicious Activity Detection/Ikea-8.89.0.403.

The file Ikea-8.89.0.403.exe is malicious
```

This is the final output which shows whether the .exe file is legitimate or malicious.

Conclusion

In this project, We have successfully created an ML model which will help in finding whether the downloaded .exe file is malicious or legitimate. This project can reach the application level with the help of a library called pickle, to save what the algorithm has learned and then we can test a new file to see if it is clean or infected. Static analysis has also proven to be safer and free from the overhead of execution time.

Future Scope

- This model can be made more accurate by increasing the size of datasets.
- Using different algorithms with better performance can increase accuracy.
- This model can be hosted on the web for real time analysis of exe files on the cloud.

References

[1] Gavriluț, Dragoş & Cimpoesu, Mihai & Anton, D. & Ciortuz, Liviu. (2009). Malware detection using machine learning. 4. 735 - 741. 10.1109/IMCSIT.2009.5352759.

[2] Parashar, Rakshit. (2020). Project report Malware analysis. 10.13140/RG.2.2.32274.48321.

[3] Moubarak, J. and Feghali, T. (2022). Comparing Machine Learning Techniques for Malware Detection. DOI: 10.5220/0009373708440851

[4] Abrar Hussain, Muhammad Asif, Maaz Bin Ahmad, Toqeer Mahmood & M. Arslan Raza. (2022). Malware Detection Using Machine Learning Algorithms for Windows Platform. DOI: 10.1007/978-981-16-7618-5_53