COLORADO MESA UNIVERSITY

COMPUTER SCIENCE
&
DATA SCIENCE

TOPICS: PYTHON MACHINE LEARNING - SPRING 2022

Malware Detection using Machine Learning

Joseph Edwards,
George Hufford

supervised by
Dr. Ram Basnet

Acknowledgements

"Thank you to all of my fellow students, professors, friends, and family for making this project a reality, as well as my dogs Riley and Max."
- *Joseph Edwards*
"I would like to thank my partner/pal Joseph for not only his expert knowledge in Cyber Security, but also for his charisma to carry our project to fruition."
- *George (Riley) Hufford*

Abstract

Today's technology is essential in nearly every aspect of how we function as a society. However, security is becoming increasingly important as dependence on technology grows and our personal data becomes susceptible to malicious attacks. Due to this need, more and more resources are being allocated toward businesses and individuals with the goal of ensuring a highly secure environment. Our project aims to make a difference in this area by creating a program that uses Machine Learning techniques to classify software as malicious or benign in a given dataset. We propose to build and deploy a classification model using a malware dataset, as well as Python and its sci-kit learn ecosystem to improve the prevention of detrimental malware-related attacks.

Keywords

Capstone; Computer Science; Data Science; Colorado Mesa University

Contents

1 Introduction

Due to the importance of security in today's digitally oriented world, the need for greater levels of research and understanding of how malicious software works and can be detected is increasing. Our project explores a particular method for detecting Malware that utilizes machine learning techniques, implemented using the Python programming language, to create a model that automatically detects if files on a computer are malicious or benign. The data used for this project comes from the Portable Executable format (PE for short) for files such as object code, executables, and DLLs (Dynamic Link Library). The information inside this file format is used by the operating system to handle the executable files properly.

Our dataset was provided by a security blogger name Prateek Lalwani, which contains 41323 malware files, and 96724 clean files collected from the website VirusShare. In total, the dataset contains 138048 files.

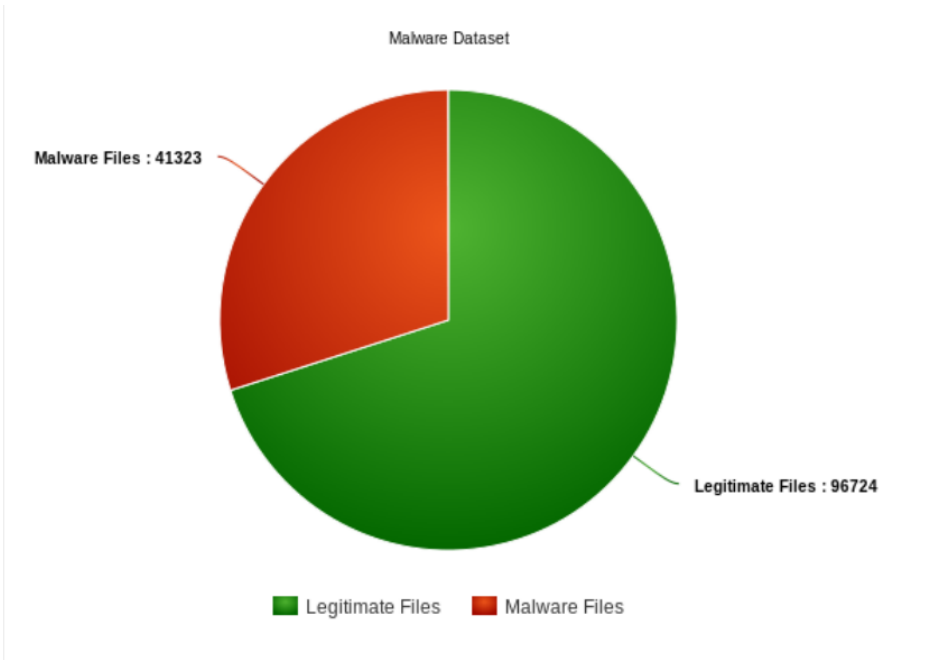


Figure 1: Representation of the malware data set. Provided by Prateek Lalwani

2 Methods

We began by loading our dataset and removing any elements that contained non-numerical data. This included elements such as the names of the files, as well as an md5 hash attached to each file. After cleaning the data, we began the feature selection process. We used the Random Forest Classifier as an estimator to fit the data and used the SelectFromModel function to create a separate instance of the data set that only contained the most significant features as determined by the fit. This method allowed us to compare the results of our classifiers after the feature selection process and observe any significant differences. We then created a list of the best selected features and ranked them in order of importance.

We utilized 6 different classifiers: Random Forest, Gradient Boosting, Ada Boost, Decision Tree, Complement Naive Bayes, and XGBoost. The accuracy of each were compared, as well as the false positive and false negative results using confusion matrices.

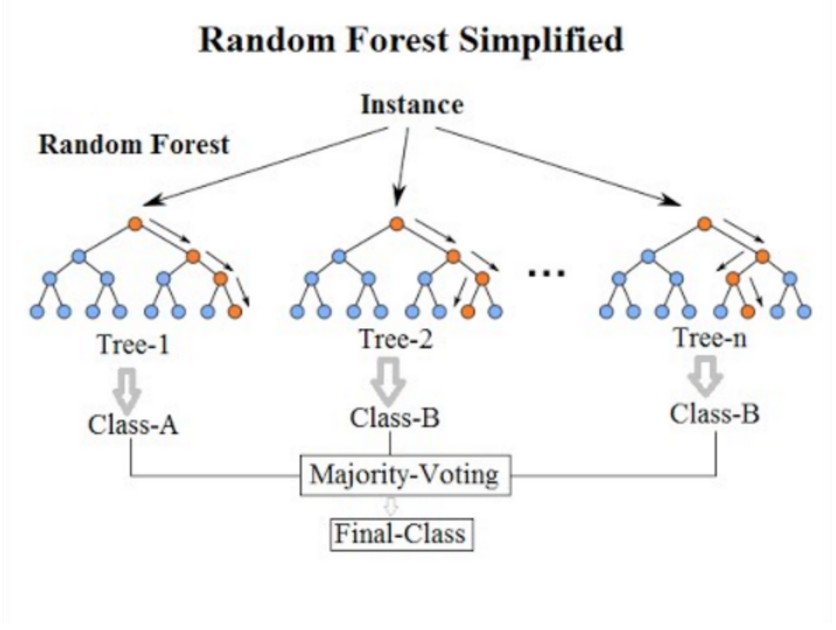
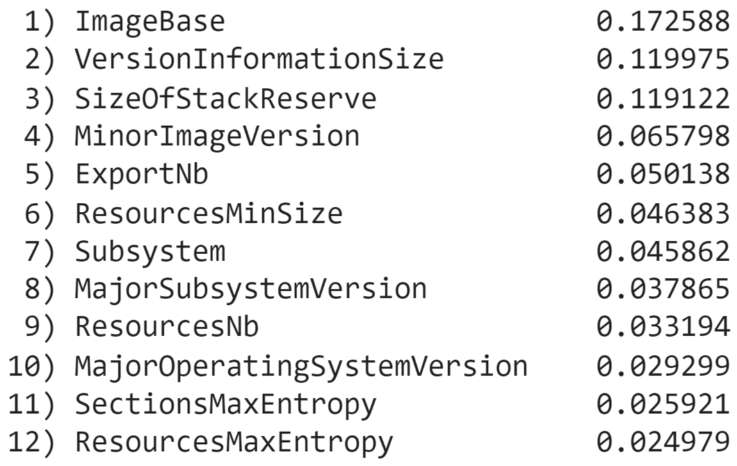


Figure 2: Simple diagram of a Random Forest decision tree

3 Results

3.1 Feature Selection

After feature selection, the new data contained only 12 features, almost a 78% decrease in size from the original data set. The most important feature was the ImageBase, which specifies the address that the executable should be loaded into memory, and the VersionInformation details where files should be installed on the computer. Finally, the SizeOfStackReserve feature is used to dictate the size of the stack used by the executable. More documentation on each of the features in the dataset can be found here: https://docs.microsoft.com/en-us/dotnet/api/system.reflection.portableexecutable.peheader?view=net-6.0



```
 1) ImageBase                      0.172588
 2) VersionInformationSize         0.119975
 3) SizeOfStackReserve             0.119122
 4) MinorImageVersion              0.065798
 5) ExportNb                       0.050138
 6) ResourcesMinSize               0.046383
 7) Subsystem                      0.045862
 8) MajorSubsystemVersion          0.037865
 9) ResourcesNb                    0.033194
10) MajorOperatingSystemVersion    0.029299
11) SectionsMaxEntropy             0.025921
12) ResourcesMaxEntropy            0.024979
```

Figure 3: List of most important features.

3.2 Classifier Results

Each classifier returned fantastic results, with the lowest accuracy score being 95.01%, and the highest being 99.45%. The lowest false positive rate was .46%, and the highest was .85%. Out of all the classifiers, the Random Forest classifier performed the best overall, yielding both the highest accuracy score(99.51%) and lowest false positive rate(0.44%) post-feature selection.

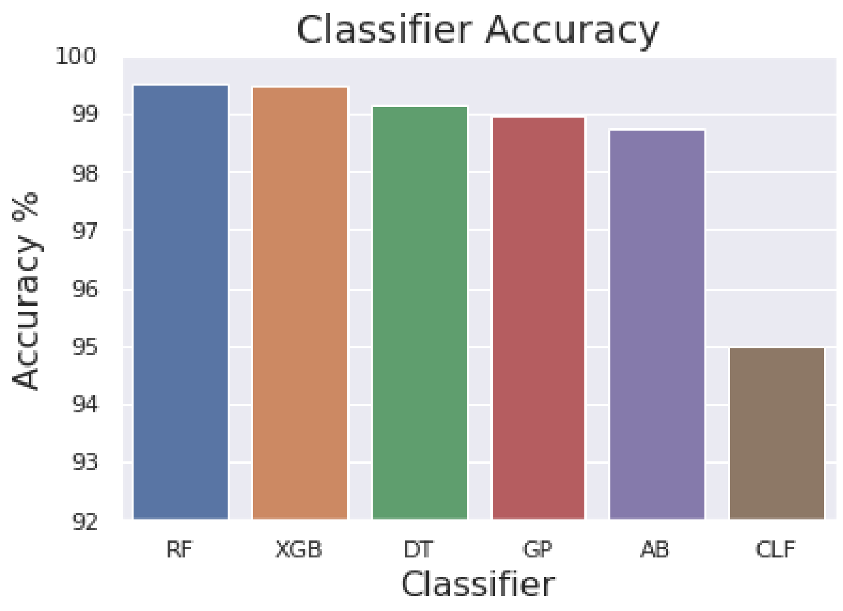


Figure 4: Accuracy's of each classification algorithm.

3.3 Differences in Accuracy and Confusion Matrices

We found that the differences between accuracy scores and confusion matrix results before and after feature selection were not incredibly significant. However, the XGBoost classifier had a 99.8% accuracy result when applied to the data with all the elements intact.

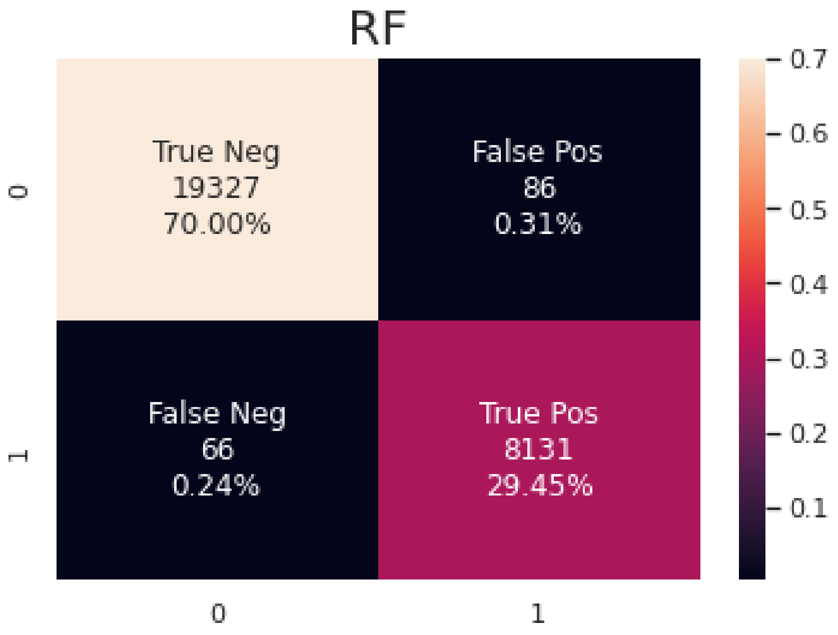


Figure 5: Confusion Matrix for Random Forest Classifier.

4 Discussion/Conclusion

Our results show that using machine learning for malware detection is extremely effective when using data from the portable executable file format. We concluded that the differences in accuracy before and after feature selection were not significantly hampered enough to only rely on the original data. By reducing the total number of features, the speed at which the classifiers were able to run increased significantly while still retaining close to the same performance.

While this research shows promising results for malware analysis, there were still many limitations we had when performing this research. Given the many forms that malicious software can come in, as well as all the new forms being developed constantly, it's impossible to have a solution that could cover all forms of malware detection. Models such as these could become easily outdated in the face of new technologies and countering new forms of malware is an ongoing battle.

However, this research shows promising results that could be extremely useful for malware analysts to understand the underlying elements that strongly indicate if a file is malicious.

5 References

Chebbi, Chiheb. Mastering Machine Learning for Penetration Testing: Develop an Extensive Skill Set to Break Self-Learning Systems Using Python. Packt Publishing Ltd, 2018.

Kath, Randy. “The Portable Executable File Format from Top to Bottom.” Csn, http://www.csn.ul.ie/ cao lan/pub/winresdump/winresdump/doc/pefile2.html.

Monnappa, K. A. Learning Malware Analysis: Explore the Concepts, Tools, and Techniques to Analyze and Investigate Windows Malware. Packt Publishing, 2018.

Pietrek, Matt. “Peering inside the PE: A Tour of the win32 Portable Executable File Format.” Microsoft Docs, 30 June 2010, https://docs.microsoft.com/en-us/previous-versions/ms809762(v=msdn.10) ?redirectedfrom=MSDN.

Revers3r. “Malware Researcher’s Handbook (Demystifying Pe File).” Infosec Resources, 10 Aug. 2021, https://resources.infosecinstitute.com/topic/2-malware-researchers-handbook-demystifying-pe-file/.