Pythonic Virus Anatomy & Computer Defense

Erez S. Sarousi

Bellevue University

DSC 680: Applied Data Science

Professor Williams

April 24, 2022

Topic

This topic focuses on understanding the composition of and the techniques utilized in recognizing, classifying and appropriately discerning a computer virus from a non-malicious file. This project aims to create an antivirus program with the highest accuracy possible.

Business Problem

As technology becomes more sophisticated, so too are the problems that technology pose in the wrong hands. In 2010, McAfee reported that lost work time due to computer viruses approximated $6.3 million dollars per day and an Information Week report from 2000 estimated a global cost of 1.6 trillion dollars (McKenzie, 2019).

People constantly use their personal computers for banking, shopping and other reasons. With the growing popularity of websites such as Amazon and TurboTax, users are increasingly storing their financial information online. This can potentially be a problem as computer viruses enter a machine by downloading infected files or even visiting infected websites (Hadhazy, 2019). This can be problematic as over 30% of American adults admit to being online almost constantly, which only increases their likelihood of being infected (Perrin & Atske, 2021).

The simplest answer to prevent malicious activity is to never interact with a computer, but this also seems to be the most unfeasible given today’s digital age. There is, however, a better solution. As the need for cyberdefense increases, the antivirus market has responded in kind. In an industry valued at over 132 billion dollars, the market works tirelessly to provide protection and can be measured with a daily registration of 450,000 new malicious programs (Lees, 2022).

It's also important to recognize that antiviruses aren't without fault; they often mistake benign programs as nefarious. This is what's called a false positive, and the repercussions can also be harmful (Guerrero, 2010). When an antivirus designates a file as harmful, it will respond. Two common responses include blocking the program or removing it. This results in important file deletion or interference with other programs.

Data Explanation

The datasets used are a collection of 57 virus and 100 non-malicious programs saved as .txt files to prevent accidental execution. The structure of the Antivirus program pores through all files and ultimately categorizes the files into four aspects: **Total Imports** - the amount of Python packages the file imports. **Bad Words** - text in the file that the antivirus function identifies as potentially malicious. **Safe Words** - text in the file that the function considers less likely to be malicious. **Threshold** - The barometer the function places to determine if the total points measuring suspicion level is enough for the function to consider if the file is malware.

The function itself takes in seven required and one optional arguments. This includes:

- **t_import_weight** - the weight used to determine how much total imports would be factored.
- **b_word_weight** - The weight determining the bad word importance.
- **s_word_weight** - The weight determining safe word importance.

- **threshold_weight** - The weight determining threshold importance.
- **threshold -** The numerical barometer that, if exceeded, will designate a program as malicious.
- **file_list_lines** is an argument as that is the list of all lines of all files.
- **classification** is an array to distinguish which files are viruses and which are non-malicious.
- **debug.** This last argument is optional, debug. If toggled true, this will list all files to show each individual grading.

Methods

A count vectorizer was used to measure the frequency of words used, A sparse matrix was then transformed and densed into a data frame so it can be sorted by its frequency. Each frequency was normalized by dividing it from the amount of files in its classification. A column was created to determine how associated the keywords are with each classification. Both safe and virus keywords were formed into lists by the data frame accordingly. Logistic regression and decision tree classification were then utilized to test the accuracy of the file designation. This was fed into the antivirus program where the model wass assessed using Numpy's XOR logic gate. In order to determine the best data weights, where a brute force algorithm was implemented.

Analysis

The key column in the data frame used is the Difference % column, which subtracts the scaled virus association from the good likelihood. The list of keywords most associated with non-malicious files are: “len, and, return, utils, assert, dict, __name__, all, from”, and “none”. Conversely, the list for viruses are: “os, while, windows, subprocess, sleep, py, write, time, socket”, and “pass”.

During the logistic regression on both keywords, there were no keywords that were seen as statistically significant to be a safe keyword, whereas there were three keywords statistically seen to be associated with that of viruses. Those keywords were: “windows” (coef ~ 0.817, p ~ 0.028), “subprocess” (coef ~ 0.637, p ~ 0.013), and “time” (coef ~ -0.300, p ~ 0.028).

Conclusion

The cross_val_score, used in conjunction with the decision tree classifier, resulted in an average accuracy rating of the model of 76.5%. The accuracy rating of the antivirus’ ability to correctly designate a file as malicious or non-malicious as ~80.128%, with the optimal weights of total imports weight as 1, bad word weight of 4, safe word weight of 5, threshold weight of 1 and threshold of 5.

Assumptions

The program here is created to test files with a Python structure. It is unknown how effective this program would be to test files with a different coding structure.

Limitations

The files used for this project were wrangled manually as there were not any datasets found that contained these files as they would with an established antivirus company. Many websites that collect data do not want to disseminate actual malware. Therefore, it wasn’t feasible to ascertain a sufficient viral dataset. Furthermore, it was a large risk to download a large amount of viruses in an executable format, considering the potential for damage.

Challenges

There were challenges in finding out how to properly determine the accuracy and determine which keywords were more likely to increase the accuracy. A lot of the challenge also included how to run a lot of the cells while having the time to adjust as necessary as the brute force analysis can take up to days at a time in order to properly complete the ranking for the weight allocation. There was also difficulty in removing some commented out statements, mostly when the program uses triple quotes as their method of commenting out. This was done as the commented out statements adversely impacted the accuracy of the antivirus function.

Future Uses / Additional Applications

There are a lot of additional applications that could be used from this antivirus program. Mainly, as there are a multitude of different tactics that go into an antivirus software in a way that often turns antivirus approaches into somewhat of an art, this could be used to detect viruses in a way that other approaches fail to capture.

Recommendations

Compared to the many professional antivirus programs out there such as Norton, McAfee and AVG, the main recommendation would be to improve this program for use.

Implementation Plan

One of the best ways to improve this would be to gain access to a larger dataset of both non-malicious and viral files. Another method in order to optimize the algorithm to result in more accurate antivirus ratings is a behavior-based model, that tests based on virus actions as well as a sandbox method, where files are run in a virtual environment and its effects measured. However, the latter method is seen as a heavier and slower method of virus detection (GeeksForGeeks, 2018).

Ethical Assessment

The ethical considerations for this project are immense. It’s important to remember that this project is interacting with real computer viruses, which carry the potential to cause real harm. With that in mind, all files are saved as .txt files to prevent accidental activation and are saved in a folder, which is staged to act as a documents folder. All personal information related to any of the files utilized has been redacted.

Ten Questions From The Audience

1. Why couldn't this accuracy rating reach 100% accuracy?
    a. **No antivirus rating can reach 100% accuracy because it's virtually impossible for a computer program to understand intention; it can only pick up keywords and behaviors that are associated with viruses, but good programs can also use these same keywords and behaviors as well.**
2. Why couldn't the model rating reach 100% accuracy?
    a. **Antivirus programs are, in many ways, seen as an art form. As much as being able to accurately designate a file isn't possible to reach 100% accuracy, so too is the likelihood of being able to create a model that will be accurate all the time isn't likely to happen.**
3. Was the stop words method considered in order to reduce filler words in the code?
    a. **Yes. The method of removing stop words was considered, but this idea was seen as likely to reduce the accuracy rating considering that many stop words such as "in, and, with, is", and many other stop words are Python keywords that perform a specific key role in how the program operates, which is likely to indicate if the file was malicious or not.**
4. Why was a brute force algorithm used to determine the most accurate weight allocation for the antivirus function arguments?

   a. **The reason why a brute force algorithm was utilized is that despite its numerous flaws such as its time-expensive processing time, it is a scientific approach to ensure that all weights have been processed and run through the program to determine the weight allocation that results in the highest accuracy ratings.**

5. Why was the heuristic model chosen to form this antivirus program?
   a. **The heuristic model was chosen to form this antivirus program for many reasons. A few of them are that there aren't a large number of antiviruses stored that could be used to form behavior-based detection methods and that it's more likely to generate a false positive rather than that of a false negative, which would produce more disastrous effects.**
6. Why was a behavior-based model not considered?
   a. **Beyond what was stated before, other detection models such as a behavior-based detection allow the malicious file to execute to wait for actions such as program setting changes, other files modified or deleted or any other sort of suspicious activity.**
7. What's the cause for the large disparity between the antivirus accuracy ratings from companies such as Norton or Bit Defender against the accuracy from this project's model?
   a. **There are many reasons why many large companies can get accuracy ratings between 95-99% whereas I could only get approximately 80%. Large cyber defense companies have large**

teams of data scientists much more experienced than myself. Furthermore, they have access to a much more extensive collection of virus files that they could analyze and otherwise gain deeper insights that they could use to increase their accuracy rating.

8. Which websites were used to manually wrangle the files used for this project?
    a. **The main website used to wrangle the files needed came from Github. A simple search from Google using the keywords "Github virus" were most likely to result in viruses. For non-malicious files, keywords such as "Useful Python programs github" and links such as https://github.com/mahmoud/awesome-python-applications/blob/master/BY_PLATFORM.md were used.**
9. Is it possible for a Python program to catch viruses in real time?
    a. **Likely not. Python is not considered a Real-Time Programming Language that can act in real time as other languages can, namely those in the C-family, much like C, C++, and C#. The concern with this model is that a "while True" phrase for constantly searching for new files would still perform the same action and not provide real action, it would also be memory-intensive and detrimental to computer functionality.**
10. Can viruses use antivirus algorithms to evade detection?
    a. **Yes. Many antiviruses are designed to seek out keywords such as, "virus", and "hack". Many antiviruses also seek out specific imports**

such as socket, which many antivirus companies see as indicative of maliciousness. By using different imports or phrases, viruses are more likely to evade detection.

References

GeeksforGeeks. (2018, July 31). How an Antivirus Works? https://www.geeksforgeeks.org/how-an-antivirus-works/

Guerrero, J. (2010, September 8). False positives - What are they? Panda Security Mediacenter. https://www.pandasecurity.com/en/mediacenter/security/false-positives-what-are-they/

Hadhazy, A. (2010, June 2). How Does a Virus Infect Your Computer? Livescience.Com. https://www.livescience.com/32619-how-does-a-virus-infect-your-computer.html

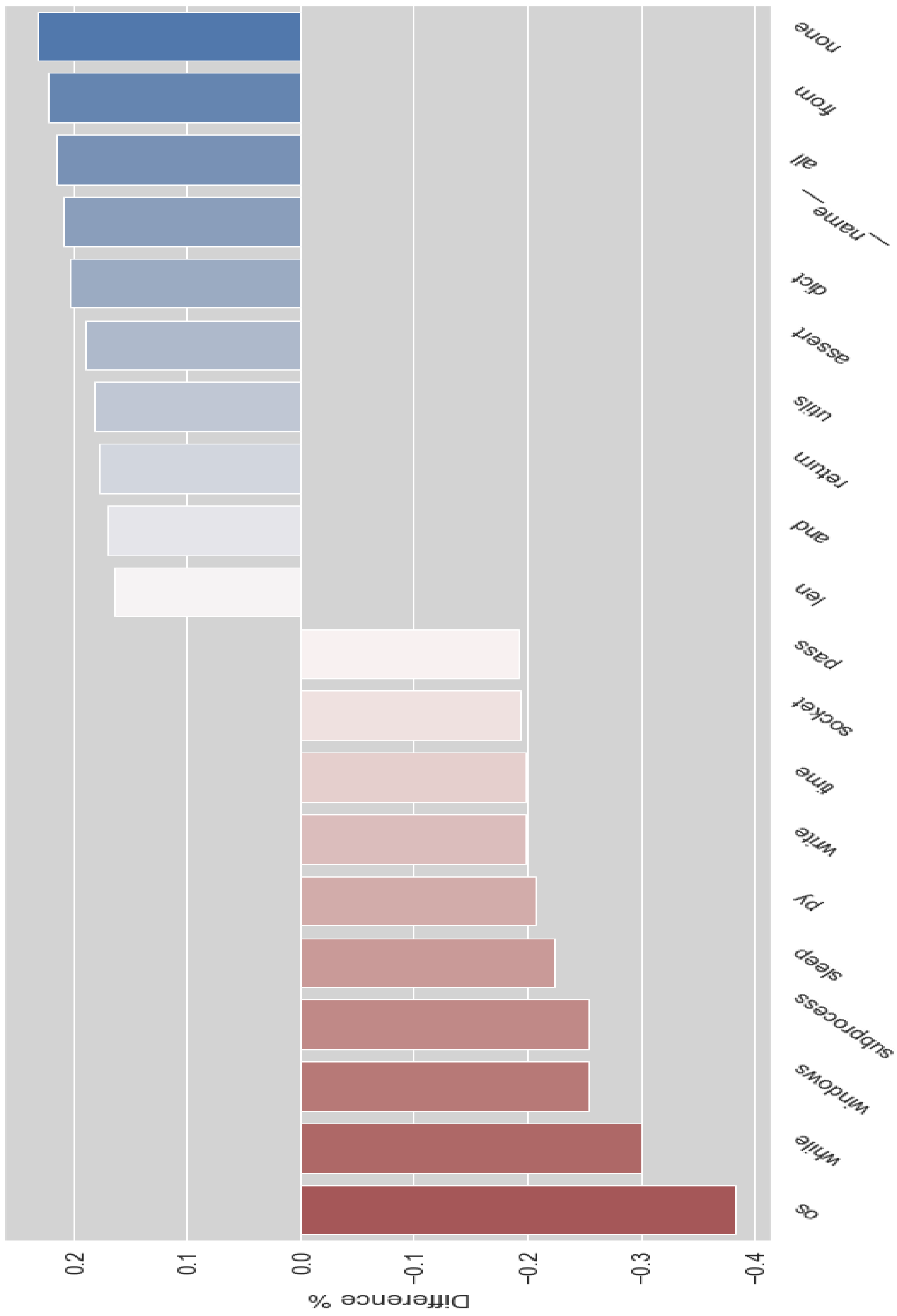
Lees, H. (2022, March 14). The Hottest Antivirus Statistics for 2022. TrustRadius Blog. https://www.trustradius.com/buyer-blog/antivirus-statistics-trends

McKenzie, E. (2019, January 10). Computer Viruses & How They Affect Our Economy. It Still Works. https://itstillworks.com/computer-viruses-affect-economy-8739209.html

Perrin, A., & Atske, S. (2021, June 5). About three-in-ten U.S. adults say they are 'almost constantly' online. Pew Research Center. https://www.pewresearch.org/fact-tank/2021/03/26/about-three-in-ten-u-s-adults-say-they-are-almost-constantly-online/

Vendator, D. (2021, December 12). Real-Time Systems And Ada Programming Language - Dav Vendator. Medium. https://medium.com/@eruditoguerrero/real-time-systems-and-ada-programming-language-74b73b30310d

Appendix


none
from
all
__name__
dict
assert
utils
return
and
len
pass
socket
time
write
py
sleep
subprocess
windows
while
os
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Difference %