Executive Portion

Executive Summary

Malware detection is an ever-increasing challenge in the field of cybersecurity and antivirus software. Current detection measures depend on extensive datasets of recognized malware which have been meticulously gathered and compiled by cybersecurity experts. This process takes cybersecurity analysts countless hours, results in antivirus scanners that are limited to detecting known malware, and often allows new malware to escape detection for days to months. To address these issues, we leveraged machine learning to create a software solution that analyzes software permissions and behaviors to categorize software into threats and non-threats. The software effectively identifies novel malware strains, demonstrates resilience against emerging threats, and potentially increases efficiency for cybersecurity analysts.

Dataset Overview

To ensure model validity, we based it on a dataset that was the product of extensive research in the computer science and cybersecurity field. This dataset was created to address the pressing need for robust cybersecurity measures and contains 4,465 records of software. Each software record in the dataset was classified as either malware or non-malware and included comprehensive documentation of all permissions requested and actions executed by the software. This provided a structured framework that we leveraged to analyze software behavior.

Modeling Effort Summary

Because malware detection is an essential part of our business functionality, we cannot tolerate a drop in performance or reliability. To prevent a drop in performance, we enacted several safeguards surrounding our model including eliminating errors and inconsistencies in the data, fixing imbalances in the data, and selecting appropriate modeling techniques. We did this by preparing and cleaning the data we were given. By eliminating errors and inconsistencies, we created a strong foundation for our model. This data cleaning process affirms the final outcome is of the highest quality. We ensured our data was balanced by performing various tests and applying functions where needed. To produce the most accurate model, we went through a variety of testing to find the optimal parameters. This helps to create a strong model that does not overfit the sample data we were given. Our model was tested with multiple combinations of new data and settings, evaluated based on statistical tests, and equipped with monitoring software. This ensures that our model is maintainable, precise, and reflective of our company's commitment to high-quality software.

This resulted in a model that can quickly classify software into threats and non-threats. The model's performance is comparable with current anti-malware threat detection methods when confronted with known malware. In addition, unlike current measures that have decreased performance when detecting novel malware, the model maintains its high-performance level.

This high performance is essential because the advent of AI and other automation tools has only increased the prevalence and ease of creating new malware. Without the model, internal teams are confronted with an overwhelming workload, and our external software products have their effectiveness reduced. This model levels the playing field both internally and externally because it maintains our product's performance levels and allows our analysts to leverage automation in their tasks. This will increase business revenue, decrease internal inefficiencies, and improve our reputation as a top antivirus provider.

Business Value Proposition

We estimate a large increase in revenue from deploying the model. If we charge an additional $5 on top of our normal subscription fee and half our customer base adopts the new software, we estimate a $500,000,000 increase in revenue per year. This is a conservative estimate because no other software on the market currently provides novel malware protection. This will likely result in higher customer conversion rates, bring an influx of new customers, and allow us to charge more than an additional $5. The potential savings for our customers are in the hundreds of thousands of dollars so we don't anticipate any difficulties entering the market.

A business could use our model for their employees so that they can ensure that all employees are downloading safe software. This would help maintain the integrity of the databases and websites involved with the software being downloaded. This would also decrease the amount of training needed for each employee to undergo in order to be safe when downloading content. However, it should be noted that our model is not perfect, and therefore some training should still be required to ensure the safety of the company.

Similarly individuals can use our model to know whether or not the content they are downloading is safe on their computer. While individuals don't have a lot of databases and websites that could potentially be hacked, they do have secure information such as passwords and bank information that could be exploited and used for evil. By using the model, we give individuals the ability to safeguard that information.

Bias and Fairness Considerations

As with any software, the model does have limitations and biases. Because the model uses software permissions and actions without surrounding context, it may occasionally mistakes legitimate software for malware. The more complicated a software becomes, or the more like malware it behaves, the more likely our model is to mistake it for malware because of the extensive permissions required by these softwares. One example of a potential mis-classification is a company's security monitoring software. Both malware and security monitoring software need access to a wide variety of computer resources, access to send reports outside the device, and access to sensitive computer data. If the software is placed on the computer by the company, a security expert would consider it legitimate software. However, if it was not intentionally installed by the company the same security expert would classify the software as spyware (a type of malware). Because our model classifies software based solely on permissions and actions, it will likely perform poorly on software that is classified as threat or non-threat based solely on the circumstances of its installation. We recommend mitigating this with a two tier approach involving a second model that classifies based on install circumstances, software download numbers, source company, and other related data. Using both models in tandem should reduce mis-classfication of highly complex softwares and softwares that behave like malware.

Conclusion and Recommendations

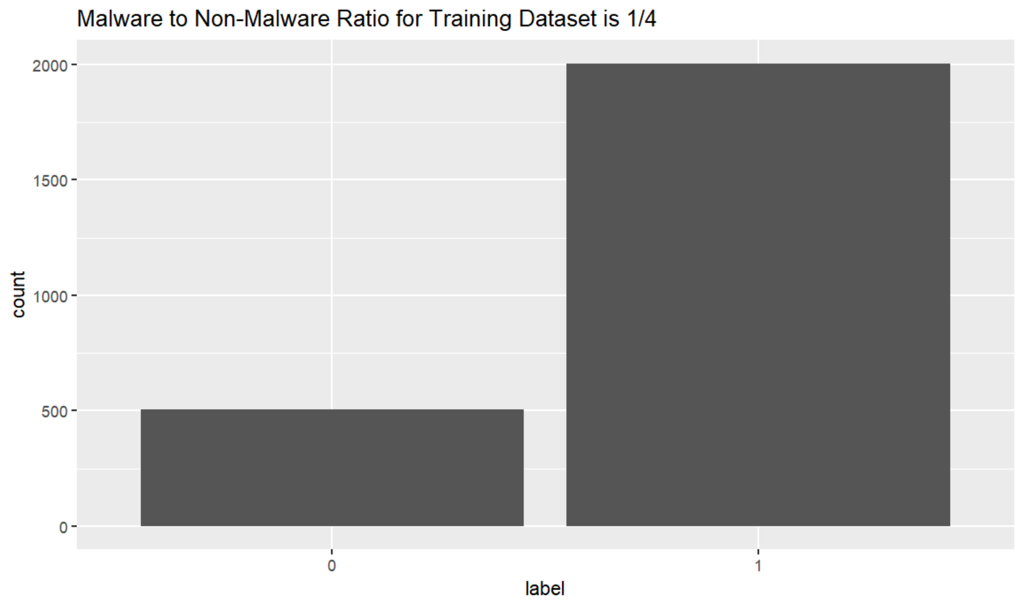
As mentioned previously our model was equipped with monitoring software after it was deployed. This software reports that the model continued to correctly detect upwards of 90% of malware. This performance is comparable to the industry average, when detecting known malware, and approximately 41% above the industry average for unknown malware. With those benchmarks in mind, we recommend retraining the model if performance ever drops below 85%. While 85% is a higher performance on unknown malware, it compares unfavorably with competing software when classifying known malware. Because long-term poor performance on known malware may eventually reflect poorly on our company's standards for anti-virus software, we recommend retraining the model as soon as it falls below that threshold.

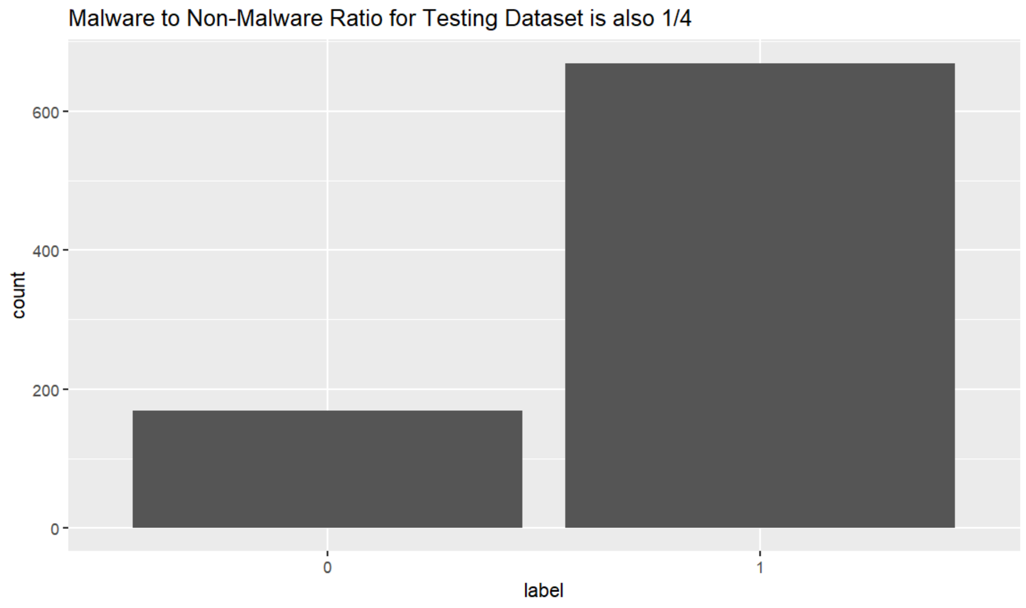
Considering our model's performance in both the test environment and production environments, the extensive validation, the academic rigor behind the generation of the base data, comparable (or better) performance than current software, and the potential revenue, we recommend implementing this model in the company’s current software product line.

Appendix

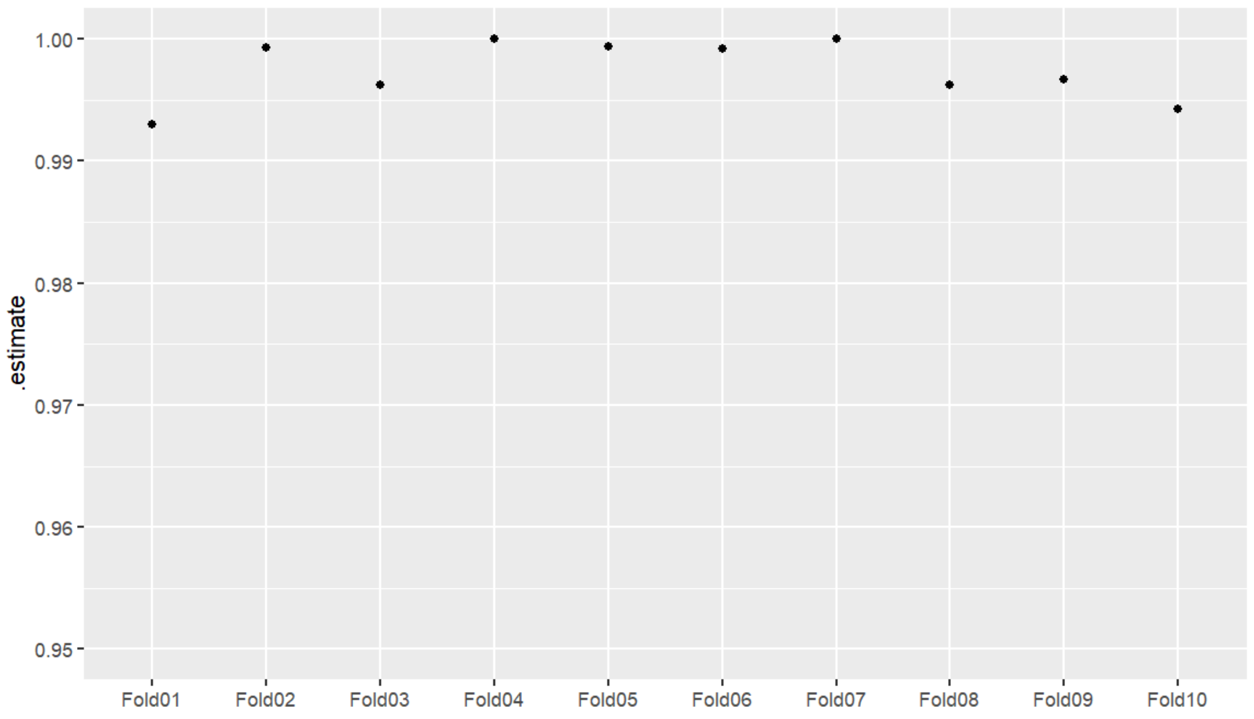
Evaluation of Data Split and Model Validation

The data was randomly split into training and testing datasets. The ratios of Malware and Non-Malware in the datasets was kept roughly proportionate between testing and training datasets:





Our folds were also randomly created using the label for stratification and Roc_Auc remained consistent across folds:



Data Processing and Preparation

Processes	**Operations and Reasoning**
Null Value Removal	We removed all the null values in the dataset. There was only one row that had null values so dropping them only benefited the model. This helped ensure the model trained on complete and accurate data.

Factorizing Data	We converted the categorical variables to factors as this made the model run smoothly and without errors.
Calculations for Group Features	We aggregated features into columns based on prefixes like 'READ', 'WRITE', and 'ACCESS'. This reduced dimensionality and captured underlying patterns across multiple related features.
Normalization Feature Engineering	We normalized all the numeric data. This helped to standardize the scale of the data, ensuring that the model was not biased towards variables with larger scales.
Fixed Imbalanced Data	We used upsampling to fix the imbalanced data. This helped to balance the data set to allow the model to train with better accuracy and to be more robust.
Feature Selection	We removed features with near zero variance and high correlation. This helped to reduce the redundant data and complexity of the model and increase training efficiency.
Dimensionality Reduction	We used PCA (Principal Component Analysis) to reduce the number of features. This helped in capturing the most

	important features, reducing the noise and improving the speed and performance of the model.

Model Algorithm and Justification

We chose XGBoost because it performed on a higher level than the logistical regression model when evaluated with ROC_AUC. Our data was thoroughly filled out with no missing values and XGBoost performs better with full datasets rather than sparse datasets.

Another reason for selecting XGBoost was its ability to efficiently process the complex and large amount of data associated with our project. The scalability of XGBoost is essential for adapting to the increasing data demands as cybersecurity threats become more sophisticated. However, XGBoost does have some limitations, including its complexity which can lead to overfitting. Overfitting is problematic as it causes the model to perform well on training data but poorly on unseen data. Additionally, the computational intensity of XGBoost could potentially increase operational costs due to the high resource requirements.

To address these issues, we implemented several strategies. We utilized cross-validation methods to prevent overfitting, ensuring that the model generalizes well to new data by validating its performance across different subsets of our dataset. We also continually adjusted and optimized the model's hyperparameters, seeking settings that maximize performance without leading to overfitting. This ongoing tuning process is crucial to maintaining the model's effectiveness in practical scenarios.

Model Tuning and Optimization

In order to tune the model and optimize it, we started by setting up a boosted tree model where the number of trees and the minimum number of data points in a node are set up to be tuned. We used ‘XGBoost’ as the engine and configured this for classification tasks.

We then created a workflow which included our tunable model and a recipe that prepares the input data for modeling.

Next, we chose the settings to test using latin hypercube sampling and decided to use a size of ten to make sure that the model could run efficiently. We set a seed of 42 to ensure consistency in our results.

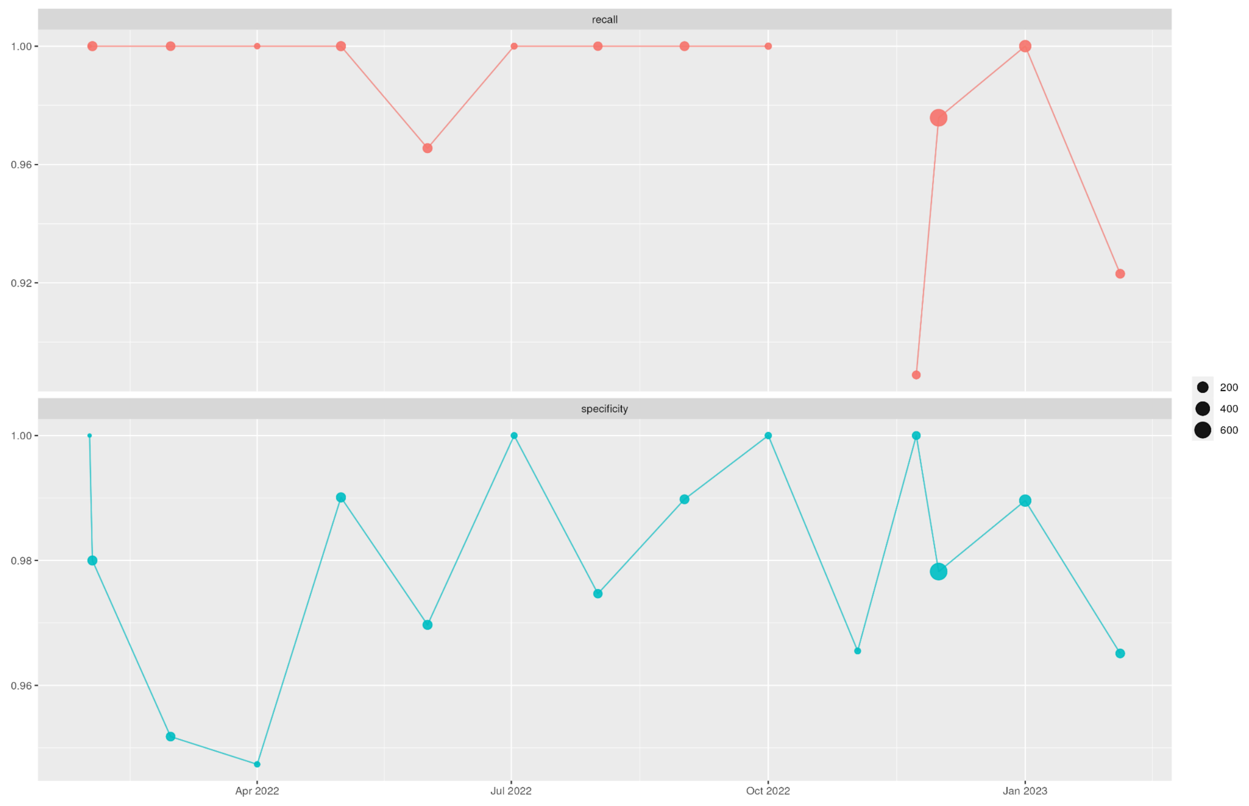
Finally, we used the tune_grid function to tune the model based on different hyperparameter combinations specified in our tuning grid. It was conducted using predefined data folds for cross-validation, which can help us assess the model’s performance more reliably.

While our tuning process resulted in a high performance model, we are aware of some improvements that could be implemented for the future. One area for improvement is in regards to our sampling. We used a limited size for our latin hypercube sampling for speed and memory purposes but the future model would benefit from a larger size.

Post-Deployment Evaluation

The model performed relatively well and we would be confident in using it for the future purposes mentioned earlier. However, we would like to test, or retrain, our model with more variety of data so we can determine if there is anything in our model that needs to be fixed before being used by the general public.

Date	**Recall**	**Specificity**
Jan 2022	1	1
Mar 2022	1	0.953
May 2022	0.98	0.99
July 2022	1	1
Sept 2023	1	0.98
Nov 2022	0.89	0.965
Jan 2023	1	0.98


recall
specificity
1.00
0.96
0.92
0.98
Apr 2022
Jul 2022
Oct 2022
Jan 2023
200
400
600

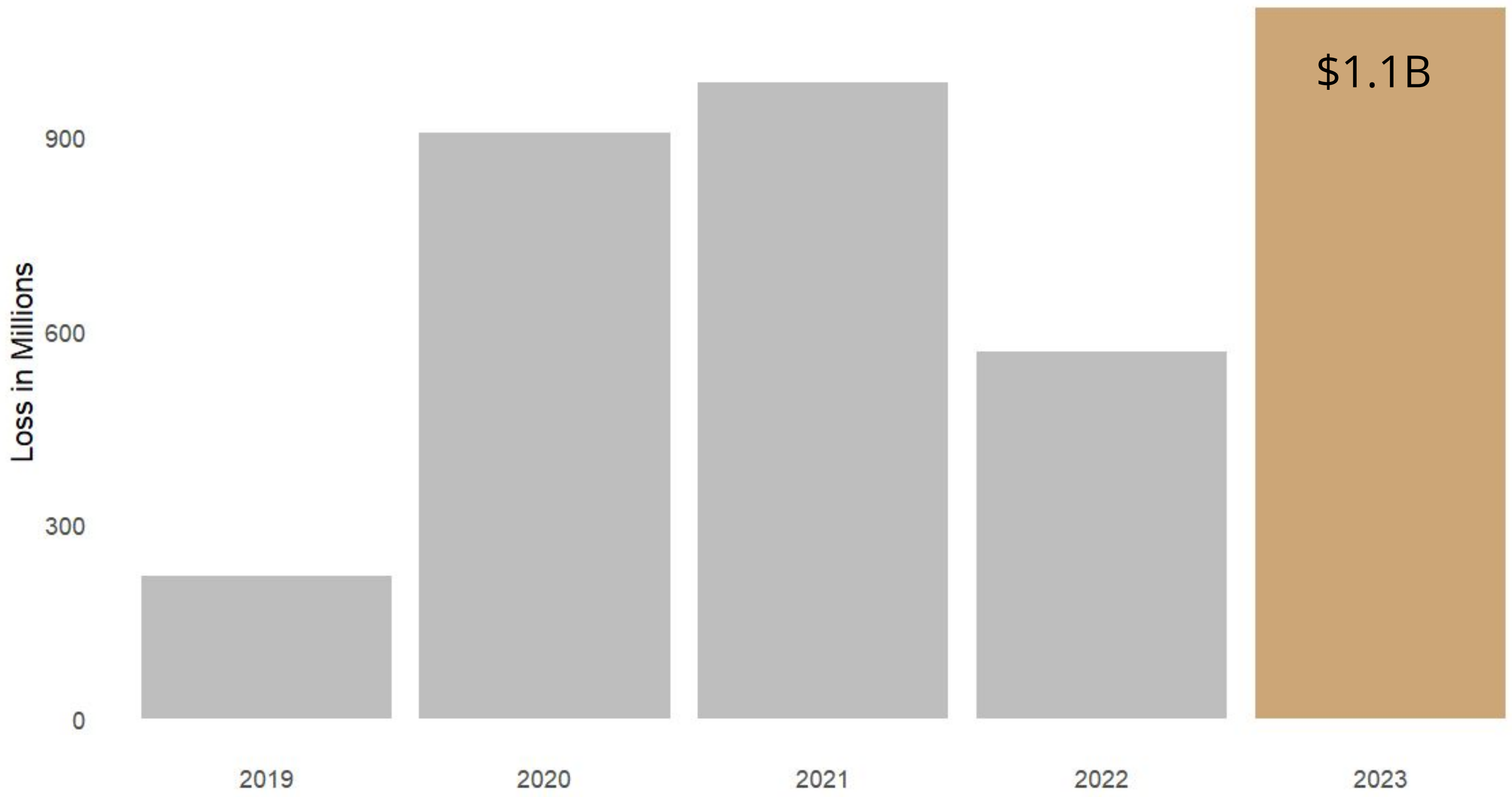
Malware Detection

Toa Pita
Lillie Heath
Bruyn Decker
Riley Marshall

49%

AV Scanners Fail to Detect New Malware

Financial loss from Malware has increased since 2019



Malware Detection Model

Detect New Malware

Optimal Parameters & Algorithm

Integration

Internal / External API

Continual Monitoring

Business Impact

Internally

- Increase revenue and sales
- Reduce breaches and security incidents (internal and external)
- Decrease response time and financial loss

Externally

- Penetrate into new market
- Enhance public image and confidence
- Assist in compliance reporting

Reliability and Adaptability

Diverse Data

Validation and Evaluation

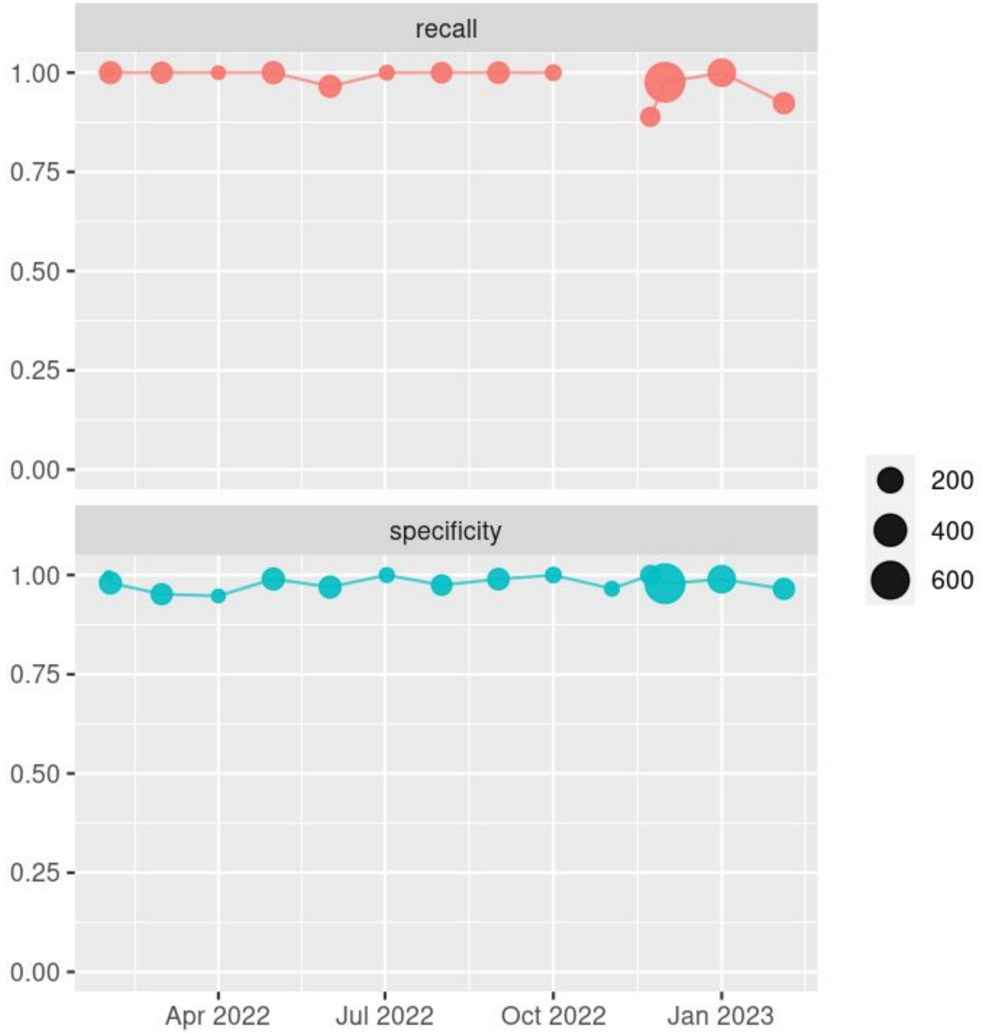
Large Data

Selected Specific Columns in Data

Performance Evaluation

Average Recall: 98%

Average Specificity: 98%



Recommendations and Future Steps

"Offering our enhanced security service at just $5 more per year to half of a worldwide customer base* we could conservatively add $500 million to our annual revenue."

We recommend that we lease access to our API via a subscription method

*assuming a customer base similar to Avast, a multinational security firm

Questions

References

https://www.chainalysis.com/blog/ransomware-2024/

https://blog.knowbe4.com/antivirus-isnt-dead-it-just-cant-keep-up