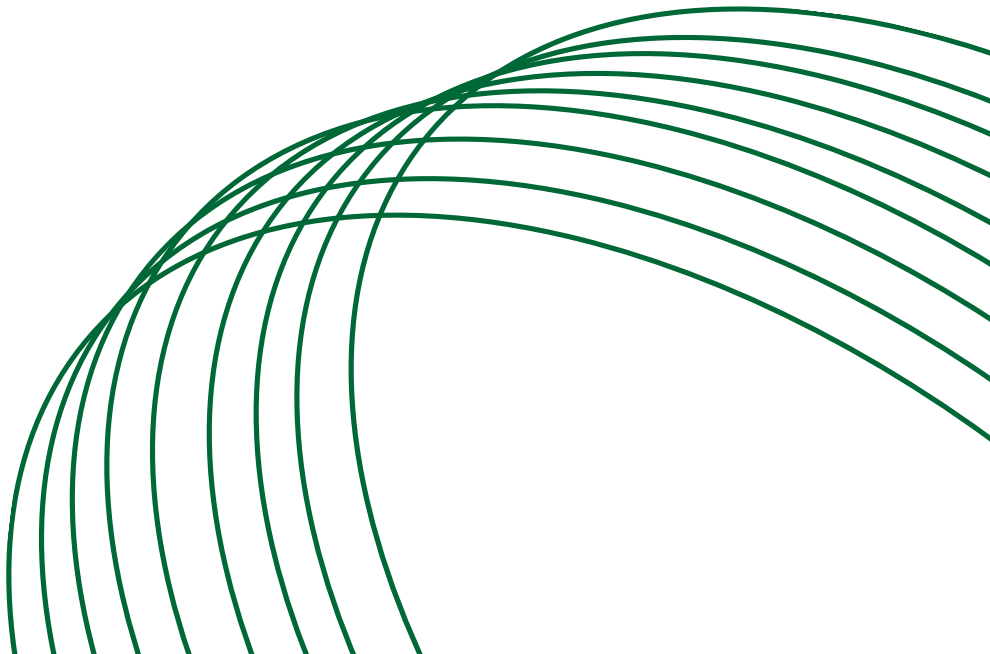
LWSM (Light-Weight Signature Matching)

June 7, 2023

Daniel Fadlon, Achia Rosin

IDs: 205984958, 304939820



Contents

1. Introduction

Malware detection is an essential component of modern cyber security solutions, and static malware detectors are critical classification algorithms that can identify whether a given executable file is benign or malicious without executing it. Static malware detection offers several advantages over other malware detection methods, including the ability to detect malware before execution, and it is faster and easier to scan files statically than other methods.

1.1 Problem Formulation

The rise of malware attacks has led to the development of various techniques and algorithms for malware detection. However, the lack of a centralized benchmark dataset for evaluating static malware scanning capabilities remains a significant challenge for researchers. Additionally, traditional signature matching services consume a large amount of memory and response time, resulting in customer loss and reduced profits for AV vendors such as Yoni& Ziv.

To address this problem, we propose the development of a Light-Weight Signature Matching (LWSM) framework for a resource-limited service using Probabilistic Data Structure (PDS) and AI-powered methods. Specifically, we utilize the EMBER dataset, a rich centralized benchmark dataset containing already extracted features from a large corpus of Windows portable executable (PE) malicious and benign files. By leveraging PDS and AI-powered methods, we may improve the performance of signature matching while reducing the amount of memory required, enabling Yoni & Ziv to expand their customer base by providing a service that can run efficiently even on low-memory devices.

However, there are potential limitations and challenges to be addressed during the development and implementation of the proposed LWSM framework. These include the need for more context to understand the significance of malware detection and signature matching in the AV industry, the technical details of the specific AI-powered methods and PDS algorithms that will be used, and the potential legal and security liabilities associated with sharing a large dataset of malicious files that could be mishandled.

Overall, in the swiftly evolving landscape of the Anti-Virus (AV) industry, meeting customer needs with limited resources remains a constant challenge. This problem is amplified for companies such as Yoni & Ziv AV, who strive to provide top-tier services while remaining competitive. The crux of the problem formulation lies in maintaining a careful balance: ensuring rapid and accurate detection of malicious files without excessive resource utilization, and keeping the false positive rate (FPR) at a minimal level - no more than 0.01%.

1.2 Objectives and Research Questions

The primary objective of this work is to enable to Yoni & Ziv to meet the needs of their customers with limited resources and remain competitive in the AV industry by developing a high-performance, lightweight model. This model must be versatile enough to operate efficiently across a diverse range of devices, underlining the importance of rapid response times. Thus, the framework needs effectively reduce memory consumption while maintaining high levels of accuracy and response time. A crucial criterion for the model's

effectiveness is its ability to accurately identify all the malicious files within the dataset while preserving false positive rate (FPR) – the proportion of benign files erroneously classified as malicious – that not exceed 0.01%. To achieve this objective, the following research questions will be addressed:

1. Which Probabilistic Data Structure (PDS) exhibits the most efficient performance in minimizing memory utilization and response time, while ensuring complete detection of all malicious samples and maintaining a false positive rate (FPR) of less than 0.01%?

2. How can AI-powered methods be integrated into the LWSM framework to enhance its performance?

3. In our LWSM, what is the best balance between the FPR of the AI model and the FPR of the PDS?

4. How does the LWSM framework compare to the current signature matching service in terms of resource usage, accuracy, and response time?

5. What is the impact of the LWSM framework on customer satisfaction and sales?

By developing a LWSM framework that uses the appropriate PDS and incorporates AI-powered methods, we aim to reduce memory usage and response time while ensuring minimal impact on accuracy performance. Ultimately, the proposed LWSM framework has the potential to improve the efficiency of Yoni & Ziv's signature matching service, leading to increased customer satisfaction and improved competitiveness in the AV industry.

1.3 Literature review

In the face of increasing cybersecurity threats, traditional malware detection techniques, such as signature matching, are proving limited due to their static nature and dependence on known malware signatures. Probabilistic Data Structures (PDS), particularly Bloom Filters, have emerged as a solution to overcome these limitations. Bloom Filters are a space-saving data structure, designed for membership queries, but they present a trade-off between space and error in the form of false positives.

Learned Bloom Filters (LBFs) have been introduced to address these challenges. LBFs merge a machine learning model with a traditional Bloom filter to predict and handle false positives, improving both accuracy and efficiency based on data distribution.

The application of machine learning in cybersecurity, especially in malware detection, is gaining significant interest. The emergence of LBFs offers an innovative opportunity to boost the accuracy and efficiency of malware detection methods. Key research initiatives like the EMBER project and the MalConv model have employed machine learning models to predict file 'membership' - distinguishing between benign and malicious files. The MalConv model uses a deep neural network trained to classify files as benign or malicious based on their binary representation. This approach allows MalConv to accurately predict file 'membership' without relying on predefined malware signatures, thus enhancing its adaptability and effectiveness. The EMBER project also employs machine learning to predict the nature of portable executable files, and provides a vast dataset

for cybersecurity research (elaborated in the baseline). Both projects indicate promising future applications of machine learning in cybersecurity solutions.

However, transitioning to LBFs from traditional machine learning models involves several critical considerations, such as the choice of machine learning model, handling of false positives and negatives, and the overall system performance. Addressing these challenges effectively is crucial to leverage the potential of LBFs in malware detection.

1.4 Baseline Analysis

1. **SHA256 Hashset Signature Matching**:

The existing signature matching service of Yoni & Ziv, which uses a SHA256 hashset, serves as our first baseline. This approach offers high coverage but encounters challenges related to memory usage and response time. In essence, it successfully identifies all malicious files, ensuring zero collisions and maintaining a low false positive rate. However, this comes at the expense of potentially significant memory usage. For N malicious SHA256 hashes, each comprising 32 bytes, the resultant memory usage would amount to 32*N bytes. This level of memory usage could be substantial, particularly for systems with limited resources. As the volume of malware instances continues to proliferate, the size of the hashset, and in turn, the memory requirements, will grow proportionally. Such an expansion could potentially impede the operational effectiveness of the client's system.

2. **Gradient-Boosted Decision Tree Model (EMBER 2018 Baseline)**:

The second baseline is derived from the Endgame Malware BEnchmark for Research (EMBER) 2018 dataset. The creators of EMBER developed a baseline model that employs a gradient-boosted decision tree algorithm implemented through LightGBM.

This gradient-boosted decision tree model is trained on static features extracted from PE (Portable Executable) files. The static features encompass a broad spectrum, ranging from size and entropy statistics to byte-level and string-based properties. Notably, this model does not rely on dynamic or behavioral features, meaning it doesn't need to execute the files for analysis. This aspect reduces the resource requirements to a certain extent, making it an appealing strategy for malware detection.

The training process is performed on a high-performance server, with an Intel Xeon CPU E5-2690 v4 at 2.60GHz and 256GB of RAM, and took approximately 10 days. (25 hours for each Epoch, and they trained for 10 epochs)

While gradient-boosted decision tree models can offer high predictive accuracy due to their ability to model complex patterns in the data, they can be computationally intensive. A model of this kind, trained on a large dataset like EMBER, could reach a size of several hundreds of megabytes to a few gigabytes, especially considering the model's complexity and the number of trees used (1000 trees). The size of this particular LightGBM model is specifically 127MB.

One particular challenge that the EMBER model faces is the 50/50 split in class distribution (malicious/benign) in the data. While this balanced distribution simplifies the model training process, it is not representative of real-world scenarios, where malicious instances are considerably less frequent. This discrepancy could potentially affect the model's performance when deployed in real-world settings, where it would have to deal with highly imbalanced data.

In summary, while the EMBER baseline model showcases strong performance, its high computational and memory demands, coupled with the potential misalignment with

real-world class distribution, highlight the necessity for more efficient, adaptable, and resource-conscious alternatives.

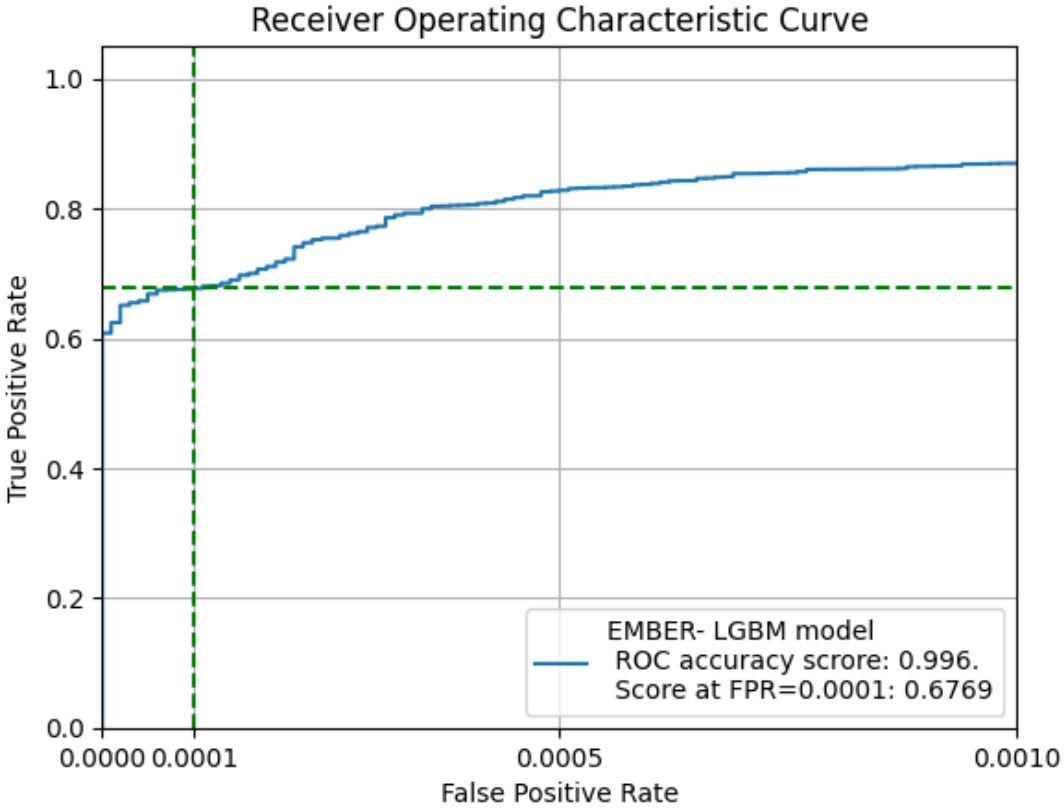


Figure 1.1: LGBM Model - ROC Curve Results

2. Data Management

2.1 Data set description

The EMBER dataset is structured as a collection of JSON lines files, each containing a unique JSON object. The dataset was created considering use cases such as comparing machine learning models for malware detection, researching interpretable machine learning, comparing features for malware classification, and researching adversarial attacks against machine learning malware.

- **Dataset Layout**: The dataset is composed of various data types:
  - The sha256 hash of the original file as a unique identifier.
  - Coarse time information (month resolution) estimating when the file was first seen.
  - A label, which may be 0 for benign, 1 for malicious or -1 for unlabeled.
  - Eight groups of raw features that include both parsed values as well as format-agnostic histograms
- **Parsed Features**:
  - **General file information**: Includes data such as file size, virtual size, the number of imported/exported functions, and whether the file has sections such as debug, thread local storage, resources, relocations, or a signature.
  - **Header information:** Details from the COFF header and the optional header, including the timestamp, target machine, image characteristics, target subsystem, DLL characteristics, file magic, major and minor image versions, linker versions, system versions, subsystem versions, and the sizes of code, headers, and commit.
  - **Imported functions**: Represents functions imported by the library, processed into model features using a feature hashing trick.
  - **Exported functions**: Similar to imported functions, this contains a list of functions that the PE file exports, also processed using a feature hashing trick for model features.
  - **Section information**: Provides properties of each PE file section, including the name, size, entropy, virtual size, and characteristics.
- **Format-Agnostic Features**:
  - **Byte histogram**: Contains 256 integers, each representing the count of each byte value within the file.
  - **Byte-entropy histogram**: Approximates the joint distribution of entropy and byte value, computed for a fixed-length window sliding across the input bytes.

– **String information**: Includes statistics about printable strings (at least five printable characters long), such as the number of strings, average length, a histogram of printable characters within those strings, entropy of characters across all printable strings, number of strings that begin with 'C:\', occurrences of 'http://' or 'https://', 'HKEY_', and 'MZ'.

The dataset also contains unlabeled samples to facilitate research in semi-supervised learning approaches. Training/test sets are temporally split to mimic generational dependencies of both malicious and benign software. The sha256 hash of the original file allows linking features to raw binaries and other metadata available through file-sharing sites. The dataset has been curated to ensure the benignity or maliciousness of files, according to VirusTotal reports.

The raw features provided are human-readable and can be transformed into a numeric feature vector using the code provided with the dataset. This allows researchers to study explainable machine learning and feature importance. The feature matrix, however, is not explicitly provided but can be derived using the provided code.

2.2 Data cleaning and preparation

1. **Unlabeled Data Exclusion**: In our preprocessing pipeline, we first eliminate all unlabeled instances from our dataset. As we intend to utilize a supervised learning approach for our study, unlabeled samples provide no value. Moreover, by omitting these entries at the outset, we ensure more efficient allocation of computational resources.

2. **Handling of Missing and Infinite Values**: We've verified our dataset and confirmed that it contains no missing or infinite values. This ensures a smooth, disruption-free machine learning process.

3. Exploratory Data Analysis

3.1 Data visualization and summary statistics

In the initial phase of our exploratory data analysis (EDA), we aimed to extract meaningful insights from the data. However, due to resource limitations on our machine, we faced challenges when attempting to compute statistics such as the mean, standard deviation, and identify outliers for each feature and between features. Consequently, we opted for a more efficient approach (using chucks), and recognizing the need for dimensionality reduction.

Firstly, we utilized the standard deviation-based feature selection technique to assess the variability and dispersion within each feature. Features with low standard deviation were deemed less informative as their values were closely clustered, offering minimal potential for our model.

Secondly, we performed a correlation analysis to evaluate the linear relationship between each feature and the target variable. Our findings revealed that approximately 400 features exhibited a Pearson correlation coefficient below 0.005 with the target variable. And none of the features exhibit a Pearson correlation greater than 0.44 with the target variable.

We also intended to explore collinear features that might provide redundant information, capturing insights already present in other features. Due to limited computing units, we divided the features into groups and calculated the correlation in each group. It is evident that it is less informative, but we find it sufficient. We found that there were 848 pairs of features with a correlation greater than 0.9 within our groups only.

Given the high-dimensional nature of the data (2,381 features), we relied on dimensionality reduction techniques.

Principal Component Analysis (PCA):

To gain an overview of the data structure, we applied Principal Component Analysis (PCA) to the dataset. This approach allowed us to project the data from a high-dimensional space to a more manageable lower-dimensional space, while preserving as much variance as possible. Remarkably, it was found that about 82% of the variance in the data could be captured using 700 principal components, and 90% captured by using 1000 principal components. (3.1)

t-Distributed Stochastic Neighbor Embedding (t-SNE) We proceeded to utilize t-distributed Stochastic Neighbor Embedding (t-SNE) as an additional method for gaining insights.

By mapping high-dimensional data into a two or three-dimensional space, t-SNE allows us to visualize complex patterns and relationships between our variables that may not be discernible with PCA alone. However, it's important to interpret t-SNE plots with caution. The specific layout or positioning of clusters doesn't hold a meaningful interpretation in terms of the original dimensions of the data. The received t-SNE results provided compelling confirmation of our existing hypotheses regarding the potential clustering of samples. These findings successfully delineated distinct malware hotspots, enhancing the visualization and comprehension of the data. (3.2)

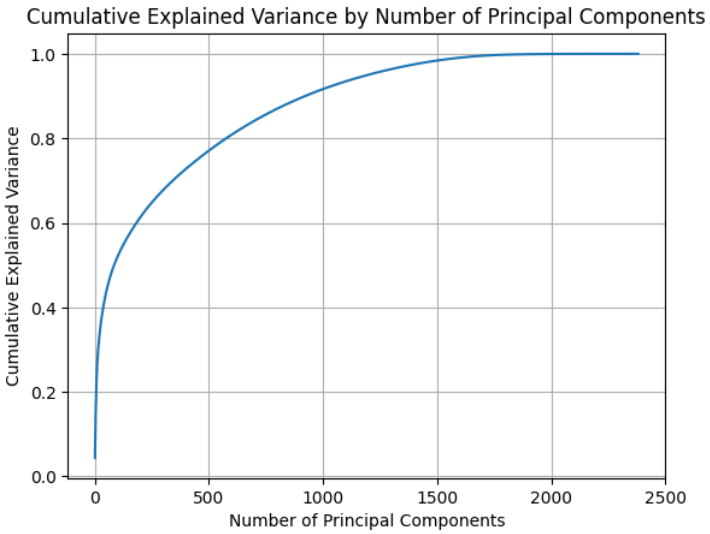


Figure 3.1: Cumulative Explained Variance by Number of PCs

Figure 3.2: t-SNE

3.2 Data Distribution

Distribution of Target Variable: In our dataset, both benign and malicious classes constitute 50% of the data. While this balance aids model training, it contrasts with the real-world scenario. A model trained on our balanced data may therefore perform differently in the field due to this class distribution discrepancy.

If we drastically adjust our dataset to mirror the real-world proportion of 8% malicious to 92% benign, we might risk information loss or potential overfitting. Nonetheless, adjusting our dataset towards a ratio that more closely approximates real-world conditions could enhance our model's effectiveness in practical scenarios.

To potentially navigate this challenge, we can consider the implementation of certain techniques that can appropriately handle class imbalance. These include Anomaly Detection or One-Class Classification, Cost-Sensitive Training, and Resampling Techniques.

It is our objective to develop a robustness model that performs well across a variety

of traget distributions. The alternative would be to experiment with these strategies and select the one which is most aligned with the goals of our project.

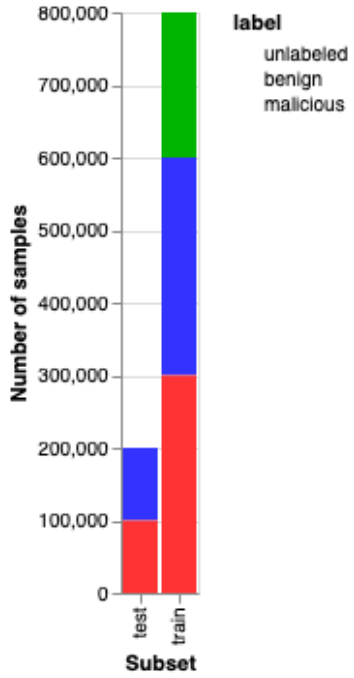


Figure 3.3: EMBER - Train and Test Dataset Separation with Labels

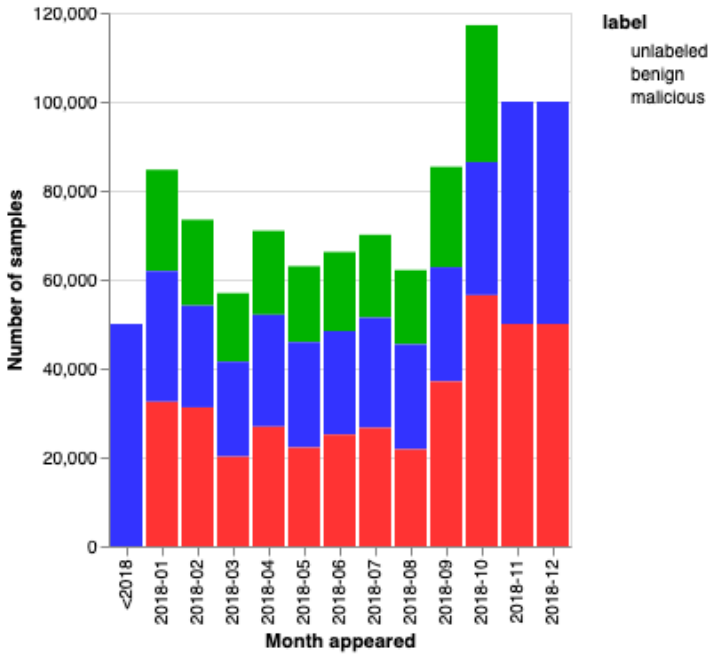


Figure 3.4: EMBER - Monthly Distribution of Sample Labels

Distribution of Principal Components: To gain further insights into the data, we also examined the distribution of the first few principal components. These components are those that explain the highest variance in the dataset.

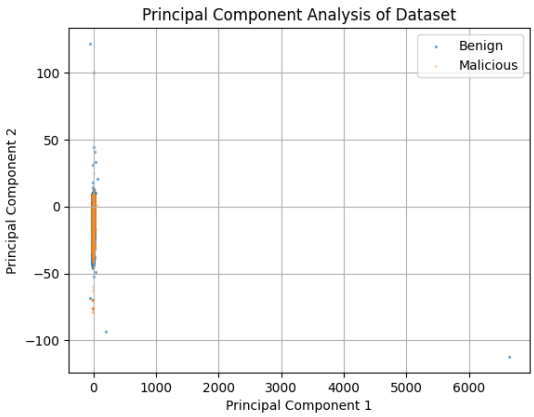


Figure 3.5: Comparison between the First and Second Principal Components

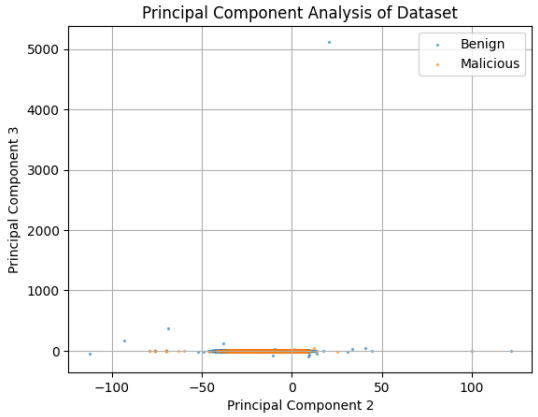


Figure 3.6: Comparison between the Second and Third Principal Components

4. Proposed Methodology

4.1 Overview of the proposed methods for PDS and AI models

Our proposed method integrates two principal components: Probabilistic Data Structures (PDS), specifically the Bloom Filter, and Artificial Intelligence (AI) models, designed for feature extraction and data interpretation. The Bloom Filter, a space-efficient PDS, is employed to perform the signature matching. Its properties of high speed and space efficiency make it well suited to the task. However, the raw Bloom Filter is static and can lead to suboptimal performance. To overcome this, we propose using a Learned Bloom Filter, a hybrid system that uses a machine learning model as a first-level predictor and a traditional Bloom Filter as a backup.

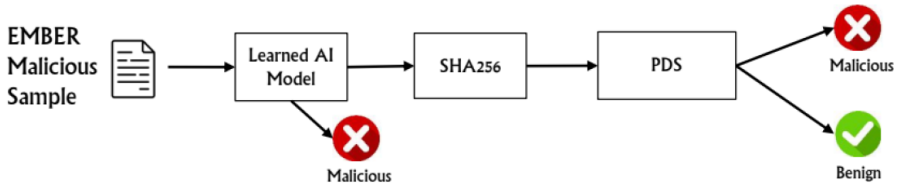


Figure 4.1: Model Architecture

Our Bloom Filter design consists of a bit-array and multiple hash functions, where the size of the array and the number of hash functions are calibrated according to the desired False Positive Rate (FPR) of 0.01 percent and the number of estimated elements in the bloom filter. This dynamic configuration enables optimal memory usage while maintaining the required FPR.

The Bloom Filter's primary advantage is its capability to indicate if an element might be in the set or is definitively not in the set, providing an essential balance between memory usage and operation speed. This trait aligns perfectly with our requirement for quick and efficient detection of malicious files in a high-volume data environment.

Our proposed AI methodology places significant emphasis on harnessing the power of XGBoost, a gradient-boosted decision tree model, acclaimed for its adeptness at handling high-dimensional data. The rationale behind this choice is its proven track record of high performance, and the inherent efficiency and optimization of the algorithm.

However, our primary focus lies in ensuring that we provide XGBoost with the best possible representation of our data. Our efforts are directed towards diligent data preprocessing and sophisticated feature extraction and selection, with the aim of retaining the maximum informational value. This robust approach to data handling allows us to reduce computational complexity while preserving the intrinsic variability and richness of our dataset.

4.2 Feature extraction and engineering

In the process of building our model, a key focus is placed on feature extraction and selection, given their crucial role in the overall predictive performance and computational efficiency of the machine learning model. The primary objective is to retain a compact,

yet representative, set of features that encapsulate the critical patterns and variations in the data, while reducing the dimensional complexity.

Firstly, based on our EDA, features with low variance are deemed as potential candidates for removal. Low variance indicates that a feature's values do not vary significantly, and thus likely do not contribute much informational value for our model. Removing such features could potentially enhance model performance by reducing noise and overfitting, and improving training speed.

Secondly, we investigate the correlation of each feature with the target variable. Features with a negligible correlation suggest a weak linear relationship with the target, implying limited predictive utility. Excluding these uncorrelated features further refines our feature set, ensuring it comprises variables that are most pertinent to our prediction task.

Furthermore, we examine the linear relationship between the features. High correlation between pairs of features implies redundancy, as they could provide similar information for our predictive model. By removing one feature from each highly correlated pair, we ensure the uniqueness of information carried by each feature, preventing any single feature from being inferred from another. This approach helps to eliminate redundant data and enhance the performance of our model.

Lastly, given the high dimensionality of our dataset, we employ Principal Component Analysis (PCA), a powerful dimensionality reduction technique. Interestingly, we observe that a substantial portion of the dataset's variance—approximately 90%—is captured within the first 1000 Principal Components (PCs) and 85% is captured by the first 700 PCs. This observation suggests that these PCs retain most of the dataset's informational value but encapsulate it in a significantly lower-dimensional space. Leveraging these PCs instead of the original features can drastically reduce the computational complexity of our model without sacrificing much of the predictive information.

Through these comprehensive feature extraction and selection processes, we aim to equip our machine learning model with a robust, yet computationally efficient, set of features, fostering a balance between performance and complexity.

4.3 Model Selection and Evaluation Metrics

Our system is formulated from a machine learning model that serves as the initial predictor, supplemented by a Bloom Filter acting as a fail-safe mechanism.

The Bloom filter we selected was designed with considerations for both the estimated quantity of malicious files that could potentially be added and the mandatory False Positive Rate (FPR) of 0.0001. From this, we calculated the optimal size (which is also the smallest size possible) and the number of hash functions to manage collisions, thereby ensuring an FPR less than 0.01%. This strategy provided us with the advantage of minimal memory usage and response time, given that we utilized an optimally sized array and the least number of hash functions possible. Nonetheless, it should be acknowledged that this approach has the potential to hover close to the maximum allowable FPR, which could be considered a potential drawback. Upon the successful execution of feature extraction and Principal Component Analysis (PCA), we contemplated three potential AI models for our designated task:

- Extracted Features Models: With the assistance of XGBoost and the Exploratory Data Analysis (EDA) elaborated above, we embarked on training utilizing three

distinct feature extractions. The differences among these models lie within the extraction thresholds:

Dataset	TH. VAR	TH. Target Corr	TH. Fetures Corr	Num. Features
#1	0.95	0.012	0.005	**978**
#2	0.90	0.005	0.004	**1060**
#3	0.97	0.003	0.003	**1253**

Table 4.1: Threshold values for extracted models datasets

- PCA Model: Another approach was to leverage XGBoost to train a model using 700 or 1000 principal components derived from PCA.
- Real-world Contingency: Acknowledging the potential for the aforementioned models to falter in real-world scenarios, we prepared to implement alternative data separation techniques and initiate retraining, if necessary.

This selection process aimed to ascertain the model that would offer the most reliable performance under the diverse and unpredictable circumstances that it may encounter in practical applications.

Initially, a grid search was conducted on the 'Extracted Models'. Unfortunately, the computational demands of this process exceeded the capacity of our system. Consequently, we resorted to manual tuning, iteratively running each model with varying parameters.

Subsequently, we employed PCA to generate 700 and 1000 principal components, which served as inputs for our model. This approach significantly expedited the training process, albeit at the cost of direct access to the raw data.

Finally, we selected a model based on the validation performance. Then, evaluated the performance of our models on two imbalanced distinct validation sets to examin the model robustness. (23/77 and 9/91) Consider our goal to produced the best model for malicious detection that has FPR of 0.0001. We evaluated the models by ROC curve and focus on the threshold that gives the required FPR.

Finally, we adjusted the bloom filter size according to the selected model performance while considering the required FPR of 0.01%. At this point, we integrate the bloom filter and selected AI model to a Lear Bloom Filter Model. In order to find the optimal false positive rate (FPR) for the AI model, while still maintaining a good true positive rate (TPR), we studied the ROC curve of the AI model, which provided us with a better understanding of the model's performance (illustrates the relationship between the FPR and the TPR). Thus, we were able to create an efficient model with an optimal size.

5. Experimental Results

5.1 Description of the Experimental Setup and Parameters

Each AI model was trained using their respective methodologies. For evaluation, the models were tested on an unseen data set. Different class separations were enforced on the test set to check the robustness of the model.

In the course of our grid search examination on the validation set, we selected the following parameters for our models:

- Extracted Model: The model employed was an XGBoost model, which was configured with these specific parameters:
  - **Maximum tree depth** was set to **13**, allowing the model to learn more complex relationships.
  - The **learning rate** (eta) was set at **0.3**, balancing the speed and accuracy of learning.
  - The objective was defined as 'binary:logistic', aligning with our binary classification task.
  - The **number of rounds** for boosting was specified as **300**, setting an upper limit for the number of iterations in the learning process. We also use an early stop technique to prevent from overfitting.
- PCA Model: We also utilized an XGBoost model for this model, with the parameters established as follows:
  - The **maximum depth** of each tree was defined as **8**, enabling the model to learn more complex patterns.
  - The **boosting iterations** were confined to **400** rounds, marking a limit on the number of learning iterations.
  - All remaining parameters were consistent with those from the Extracted Model. A surprising finding was the poor performance of the PCA models.

Then, to create the Learned Bloom Filter model, we adjust the bloom filter according to the selected model performance while considering the required FPR of 0.01% on the EMBER2018 dataset. We examined several separations of the FPR between the AI model and the bloom filter by using the AI model ROC curve results and considering the trade-off between accuracy performance to memory usage.

5.2 Results and Analysis of Model Performance

The extract models showed better performance and faster response times. The **best extracted model** was trained on the **first dataset** (#1 - described above) with **978 features**. Nevertheless, we were surprised by the poor performance of the PCA models.

Model	TH ACC	TPR	ROC ACC
Extracted Model #1	0.84	0.67	0.99
PCA 1000 Model	0.76	0.53	0.98

Table 5.1: Performance measures for Extracted Model #1 and PCA 1000 Model at FPR=0.01% on the test dataset

To assess the robustness of our models, we conducted an additional analysis using two modified versions of the test dataset. Each variant was carefully crafted to mimic scenarios with varying concentrations of malicious samples. These modified datasets provide a rigorous testing environment that closely emulates potential real-world situations with diverse threat densities. Our models demonstrated substantial robustness to varying distributions of malicious and benign samples. Indicating their potential effectiveness in other distributions as well. In particularly, those that will be found in AV Yoni and Ziv vendors.

Model—Dataset	9%/91%		23%/77%	
	TPR	ACC	TPR	ACC
Extracted #1 Model	0.67	0.92	0.66	0.97
PCA 1000 Model	0.52	0.81	0.51	0.82

Table 5.2: Validation and Test scores for Extracted Model and PCA Model

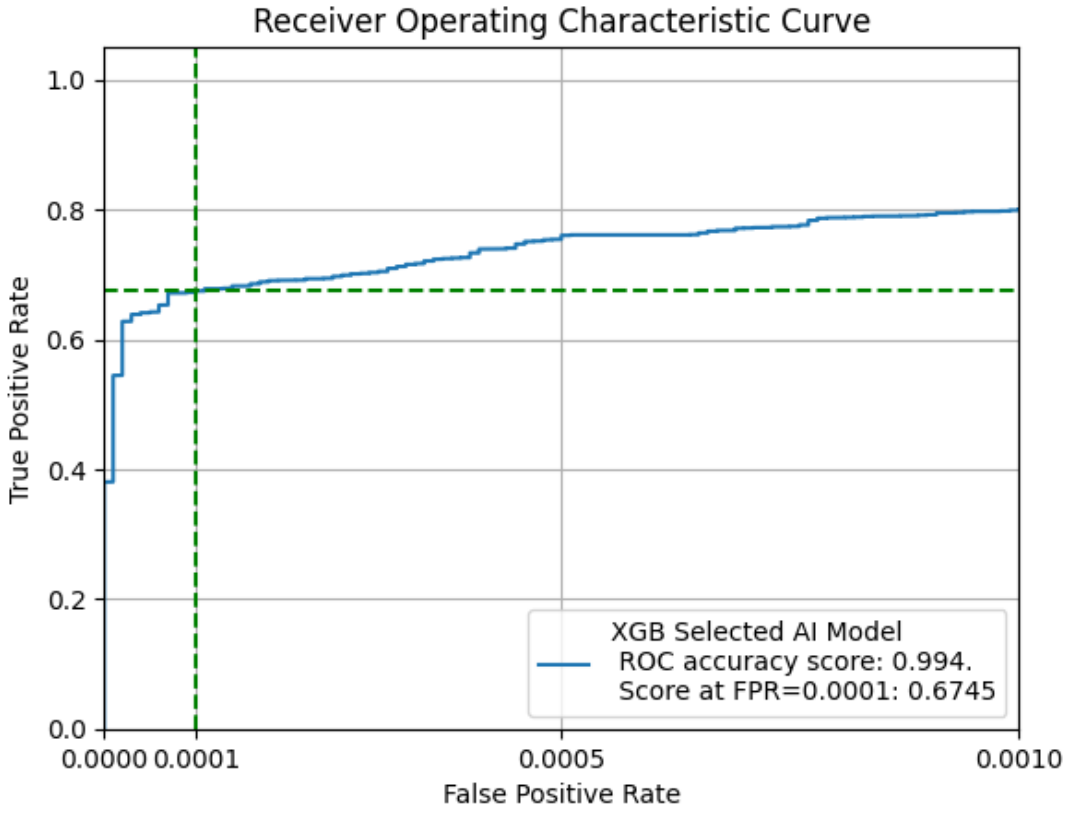


Figure 5.1: XGB Model - ROC Curve Results on Test dataset

After selecting the model, we fitted our Learned Bloom filter model by evaluating the

AI model on the dataset when inserting into the bloom filter only the malicious files that the AI model was unsure about.

Considering the requirement of FPR of 0.01% on the EMBER2018 dataset, we re-infer the ROC curve for the selected AI model on the entire dataset. In the FPR range of 0.00007 to 0.00009, TPR scores gradually decline, but in the case of 0.00006, they have fallen precipitously. The AI model is then selected to achieve 0.00007 FPR, while the learn bloom filter is selected to achieve 0.00003. Therefore, we have an overall FPR of 0.0001 on the EMBER2018 dataset. By having a high TPR, we can prevent the bloom filter from having to handle 67% of the malicious files.

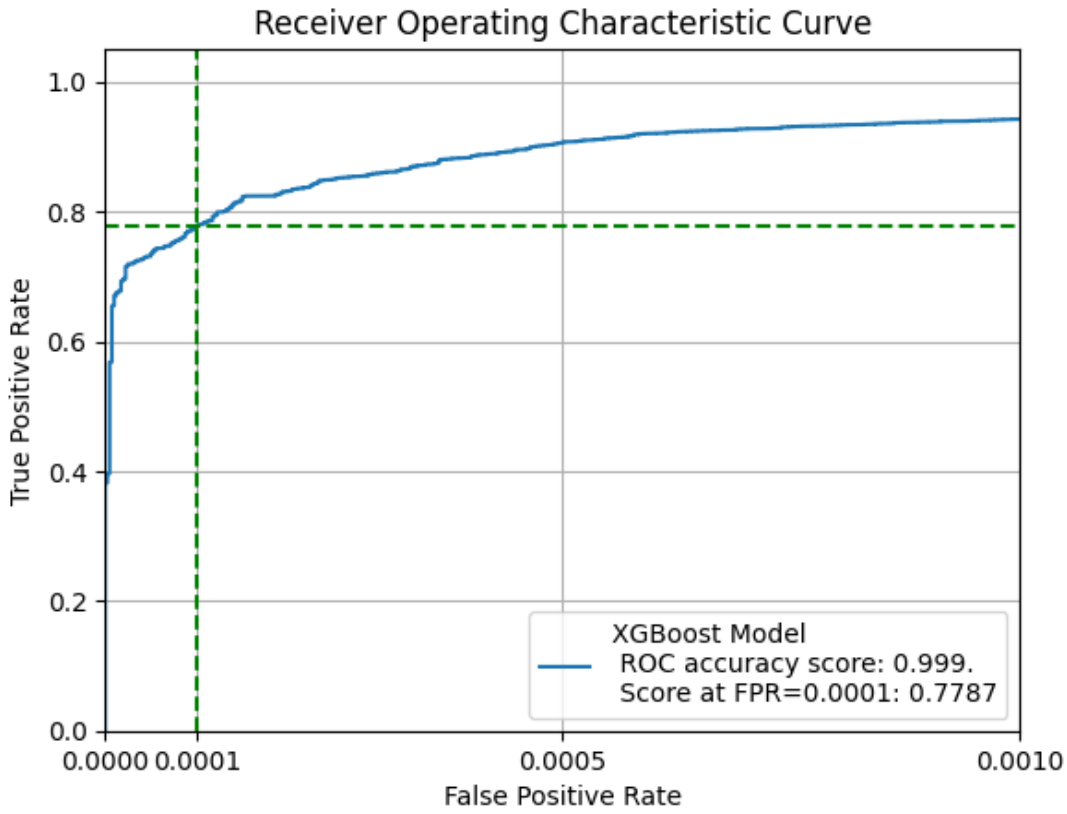


Figure 5.2: XGB Model - ROC Curve Results on EMBER2018 (entire) dataset

We have successfully developed a Learned Bloom Filter (LBF) model that accurately identifies all malicious files in the dataset while maintaining a FPR of 0.01%. The model has a compact size of 5.95 MB, subdivided into 5.69 MB for the AI-model and 0.26 MB for the Bloom Filter. Furthermore, our AI-model demonstrated robust performance on a new dataset, achieving an accuracy rate of 0.856 (after fitted to the LBF requirements).

5.3 Comparison with Baseline Methods

A comparative assessment was conducted to gauge the efficacy of our models vis-à-vis prevailing baseline methodologies. Our models outperformed the benchmarks in terms of memory efficiency and response time.

Specifically, our AI model matched the performance of the Baseline EMBER model at an FPR of 0.01% on the test (unseen) dataset, achieving an accuracy of 0.84, a True Positive Rate (TPR) of 0.67, and an ROC curve score of 0.99. Notably, we achieved

this with only 978 extracted features using a significantly more compact and expedient model. Whereas the EMBER baseline model is 127MB in size and takes 15.7 seconds to make predictions on the test set, our AI model is merely 5.69MB in size and responds in 5.3 seconds.

Furthermore, our approach offers Yoni Ziv AV the opportunity to cater to a broader clientele by drastically reducing memory consumption in comparison to the conventional bloom filter.

In the context of the 400K malicious files where each SHA256 is 32 bytes (400,000 x 32 bytes = 12,800,000 bytes or 12.8 Megabytes), the existing signature machine currently uses approximately 0.9MB for a naive model. In contrast, our Learned Bloom Filter weighs in at just 0.26MB. More impressively, the complete Learned Bloom Filter is less than half the size of the existing signature machine at 5.96MB. Yet, it retains the capacity to predict new malicious files based on their features and maintains an FPR of 0.01% on the EMBER2018 dataset.

6. Limitations and Discussion

This chapter explores the constraints and potential considerations of the proposed Light Weight Signature Matching (LWSM) methodology, implemented via Learned Bloom Filter.

6.1 Limitations of the Proposed Approach

While our proposed Light-Weight Signature Matching (LWSM) approach yielded encouraging results, certain limitations did come to light. Firstly, our operations were hampered by a constraint on Random Access Memory (RAM), impacting our capacity to effectively understand the dataset and utilize desired models or techniques. Additionally, despite the comprehensive description of raw features provided by EMBER, the vectorized features were merely numeric data without any detailed explanations or potential matches to the raw descriptions. This limitation restricted us to statistical analyses for feature understanding, as opposed to drawing upon domain knowledge.

Moreover, our understanding of Yoni and Ziv's clientele is limited. Detailed insights into their client distribution, the specific file types they handle, and the prevalence of malicious files within their operations could significantly aid in further enhancing our solution. Hence, this dearth of client-related information presents a substantial limitation to our current approach.

6.2 Potential Future Work and Directions

To address the limitations and further enhance the proposed approach, several future directions could be explored.

- **Infrastructure Up-gradation:** Allocating resources to upgrade the computational hardware, particularly augmenting RAM, could significantly enhance the performance and efficiency of the process. In addition, to give us abilities for exploring and improve our understanding of the dataset.

- **Adaptive Probabilistic Data Structures (PDS):** Given the stringent FPR requirement of 0.01%, ensuring the accuracy of file classification is paramount for Yoni & Ziv AV. The memory efficiency of our Bloom Filter can potentially be leveraged to create multiple PDSs of varying sizes. These size-varied Bloom Filters can then be adapted to match the memory capabilities of different client endpoints, optimizing memory usage while minimizing collision and maintaining quick response times.

- **Integrating Ensemble AI:** Taking into account the significantly more compact model we propose, our suggestion is to explore the addition of another AI model to form an ensemble prior to employing the Bloom filter. This approach could potentially decrease the size of both the Bloom filter and the overall model, while also possibly enhancing our performance on new files.

- **Threshold Optimization for Benign File Classification:** Our findings suggest a feasible approach to expedite the process of benign file identification by setting

a confidence threshold. If the score of a file, predicted as benign, exceeds this threshold, it could be classified as benign without the need to utilize the Bloom Filter backup, thereby delivering a **faster response**. In scenarios where the score is below the threshold, the Bloom Filter backup would be utilized to ensure accurate classification.

6.3 Conclusion

In conclusion, this study presented a Light Weight Signature Matching method, leveraging the Learned Bloom Filter, aimed at enhancing the efficiency of Yoni & Ziv AV's current signature matching approach.

Despite certain limitations, the proposed approach shows significant potential to improve Yoni & Ziv AV's clients satisfaction and establishes a foundation for future studies in the field.

We look forward to further exploring the realm of cybersecurity with you. Our investigations into Learned Bloom Filters have opened exciting avenues for malware detection, and there is much more to uncover. Let's continue this journey together, making strides towards enhanced safety and security.