1 Introduction

In today's digital era, ensuring the security of computer systems has become a critical concern. This project focuses on the evolving threat of malware, a significant and persistent challenge in cybersecurity. Malware, known for its diverse families and unique attack methodologies, continually threatens Windows machines, a widely used operating system. Our aim is to explore the complexities of these malware families and identify the vulnerabilities specific to Windows systems. By examining the unique properties of these systems, we intend to predict their susceptibility to certain malware types. This predictive analysis is crucial as it enables users and system administrators to take proactive measures to enhance their security.

The importance of predicting a Windows machine's vulnerability to malware is immense. As cyberattacks grow in sophistication and frequency, the risk to data integrity, privacy, and system functionality escalates. Windows, being a popular choice, often becomes a prime target for cyberattacks. Through this project, we aim to proactively identify and mitigate potential threats to Windows systems. This approach is vital not just for individual users but also for businesses, institutions, and essential infrastructures that rely on these systems. A proactive stance in this domain can significantly reduce disruptions, financial losses, and maintain user trust, thereby contributing to a more secure cyber environment.

2 Background

Over the years, malware detection on Windows machines has evolved from signature-based methods to more sophisticated techniques. Initially relying on identifying known malicious patterns, the field advanced with heuristic analysis to spot suspicious behaviors, and sandboxing to observe potential threats in controlled settings. The advent of machine learning and AI enabled predictive models based on extensive datasets. Concurrent developments included behavioral analysis, cloud-based threat intelligence, file reputation systems, Endpoint Detection and Response (EDR) tools, and network traffic analysis. Despite their effectiveness, these methods face challenges: signature-based detection is ineffective against new variants, heuristic analysis may produce false positives, sandboxing can be resource-heavy, machine learning needs updated data and is vulnerable to adversarial attacks, behavioral analysis might act too late, cloud solutions raise data privacy concerns, and network analysis might miss stealthy malware. Each technique, while adding a layer of defense, requires ongoing refinement to address its inherent limitations.

3 Approach

In our project, we utilized the Microsoft Malware Prediction dataset from Kaggle, applying a unified approach to data processing and modeling. Initially, the dataset underwent extensive cleaning to eliminate inconsistencies and missing values, ensuring data integrity. Subsequently, we implemented encoding strategies to transform categorical data into a machine-readable format, vital for effective algorithm application. Our team explored various predictive models: random forests, deep neural networks, and naïve Bayes. Rather than isolating these methods, we integrated their results to enhance our predictive accuracy. This collaborative approach allowed us to leverage the strengths of each method, leading to a more robust and comprehensive analysis. For instance, the random forests model's ability to handle non-linear data complemented the deep neural networks' proficiency in capturing complex patterns, while naïve Bayes offered fast and efficient baseline predictions.

3.1 Data and Data Analysis

3.1.1 Data Cleaning and Preparation

Our dataset from the Microsoft Malware Prediction competition on Kaggle consisted of a mixture of numerical and categorical variables, some with missing values and others with skewed distributions. Our data cleaning and preparation process involved the following steps:

- **Removal of NaN Values and Unbalanced Features:** Features with excessive NaN values or highly unbalanced dimensions were eliminated.
- **Feature Classification:** We categorized features into binary, numeric, and categorical groups for targeted processing.
- **Handling of Numerical Features:** NaN values in numerical features were replaced with "-1".
- **Processing of Binary Features:** NaN values in binary features were reassigned to the most frequent value.
- **Modification of Categorical Features:** Non-categorical features with NaN values were labeled as "-1", and other categorical features were processed individually, including standardization of value names and converting labels to numeric values.
- **Data Encoding:** Applied one-hot encoding and label encoding to categorical variables for compatibility with machine learning algorithms.

3.1.2 Data Encoding Process

In our data encoding stage, we implemented two primary strategies: frequency encoding and label encoding.

- **Frequency Encoding:**
  - Built a list `list_frequency_encoding` containing features such as 'AppVersion', 'AvSigVersion', etc.
  - Applied a custom function `frequency_encoding` to map each value of these features to its frequency count.
- **Label Encoding:**
  - Created a list `list_label_encoding` for features requiring label encoding, like 'Census_ActivationChannel', 'Census_ChassisTypeName', etc.
  - Used `preprocessing.LabelEncoder()` from scikit-learn for transforming categorical strings into numerical labels.
- **Final Dataset Preparation:**
  - After encoding, the dataset was split back into train and test sets.
  - The train set contained 8,921,483 rows and 72 features.

3.1.3 Data Analysis and Exploration Report

- **Distribution Analysis:**
  - Investigated distributions of 'Census_TotalPhysicalRAM', 'Census_ProcessorCoreCount', and 'Census_InternalPrimaryDiagonalDisplaySizeInInches'.
- **Feature Correlation with Label:**
  - Analyzed correlations of features with the label 'HasDetections' to identify key predictors.
- **Inter-feature Correlation:**
  - Explored relationships among features, such as the correlation between 'OsBuild' and 'Census_OSBuildNumber'.
- **Special Attention to Categorical Data:**
  - Assessed category distributions within 'ProductName', 'Platform', and 'SkuEdition' and considered encoding strategies.
- **Handling Missing Values:**
  - Identified features with missing values like 'RtpStateBitfield' and devised strategies for handling them.
- **Anomaly Detection:**
  - Searched for anomalies and outliers in continuous features to identify data irregularities.

3.2 Implementation

- Logistic Regression Modeling

- **Objective Function:** In our project, the objective function of the Naive Bayes model is to estimate the probability of a data point belonging to a particular class (malware or non-malware) based on its features. It predicts the class with the highest posterior probability, effectively distinguishing between malicious and benign software.
- **Regularization Techniques:** In our implementation of Naive Bayes for malware detection, we employed Laplace smoothing as a regularization technique to address the zero-frequency problem and enhance model stability.
- **Model Performance:** In our Naive Bayes model for malware detection, the accuracy was 0.51. While the model showed a high recall (0.97) for class 1 (malware), its recall for class 0 (non-malware) was significantly lower (0.05), indicating a tendency to predict most samples as malware. This imbalance between precision and recall resulted in a lower overall F1-score, highlighting a need for further model tuning to better balance recall and precision, thereby improving its ability to correctly identify non-malware instances.

- Random Forest Modeling

- **Objective Function:** In our project, the Random Forest model's objective function is to classify each input accurately as malware or non-malware. This is achieved by aggregating predictions from multiple decision trees, each trained on a subset of the data. The model aims to maximize the overall accuracy, reducing the likelihood of false positives and negatives. By leveraging the collective decision-making of various trees, the Random Forest model ensures a robust and reliable classification, crucial in the context of malware detection.

- **Regularization Techniques:** In our Random Forest model for malware detection, we utilized its built-in regularization capabilities. By training each tree on data subsets and introducing randomness in feature selection, we effectively reduced the risk of overfitting, enhancing the model's generalization to new data.
- **Model Performance:** In our project, the Random Forest model demonstrated better performance with an accuracy of 0.6485 compared to the Naive Bayes model, which achieved a lower accuracy of 0.5083. The Random Forest model's feature importance analysis provided valuable insights, suggesting a more nuanced understanding of influential factors in malware detection than the Naive Bayes model.

- **LightGBM/Keras Modeling**

- **Objective Function:** The objective function for LightGBM/Keras in malware detection is to accurately classify instances as either malicious or benign, optimizing for a balance between precision and recall while minimizing the error rate.
- **Regularization Techniques:** In our LightGBM/Keras model for malware detection, we applied dropout and L1/L2 regularization to prevent overfitting and enhance the model's generalization capability.
- **Model Performance:** The LightGBM model's accuracy of 64.4% and the Keras model's 63.61% demonstrate their effectiveness in malware detection, with LightGBM slightly outperforming Keras. Both models show a notable improvement over the Naive Bayes model, which achieved an accuracy of 50.83

3.3 Handling Class Imbalance

In our project on malware detection, we initially dealt with a dataset consisting of 8,921,483 rows and 83 features. To tackle the challenge of class imbalance, a critical step in preparing our data for modeling, we adopted a two-pronged approach. First, we identified and removed features that exhibited extreme class imbalances, which could potentially skew our model's performance. This step was crucial for preventing the models from being biased toward the majority class, a common issue in imbalanced datasets. After this feature elimination, our dataset was reduced to 70 features, maintaining the same number of rows but offering a more balanced representation of the classes.

Furthermore, for features with moderate imbalances, we implemented resampling techniques. This method involved adjusting the representation of classes in our dataset to create a more equitable data structure. Such careful and strategic data preparation was pivotal in ensuring that our models were trained on data that reflected a more realistic scenario, thus enhancing their accuracy and reliability in real-world applications of malware detection. The combination of feature removal and resampling underscored our commitment to developing robust, effective models capable of accurately identifying malware threats.

4 Results and Evaluation

In evaluating our models for malware detection, we used accuracy as the primary metric, considering its relevance in classification tasks. The Naive Bayes model showed the lowest accuracy at 50.83%, indicating limited effectiveness, particularly in handling class imbalances and complex feature interactions. The Random Forest model, with 64.85% accuracy, demonstrated a notable improvement, benefiting from its inherent regularization and ensemble approach. LightGBM, achieving 64.4% accuracy, and Keras, at 63.61%, further indicated the potential of advanced machine learning techniques in this domain. Both models performed comparably to Random Forest, suggesting that while deep learning (Keras) and gradient boosting (LightGBM) are powerful, they do not significantly outperform traditional ensemble methods in this specific context.

Our confidence in these outcomes is moderated by the understanding that accuracy alone might not fully represent the models' predictive capabilities, especially in imbalanced datasets. Future evaluations could benefit from additional metrics like F1 score, precision, recall, and AUC-ROC, providing a more nuanced view of each model's performance. This comprehensive evaluation approach would enable a deeper understanding of the strengths and limitations of each model in malware detection.

5 Conclusions

Through this project, We gained insights into the significance of data cleaning in shaping model accuracy, particularly in handling class imbalances. We utilized various models like Naive Bayes, Random Forest, LightGBM, and Keras, each with its strengths. Techniques like feature importance analysis and balancing precision and recall were key to improving accuracy. Future directions could explore ensemble methods or advanced feature selection techniques, as suggested in recent literature on machine learning applications in cybersecurity.

6 Work Split

Data cleaning: Tian Ma, Wenyu Pan
Data encoding: Tian Ma
Logistic Regression Modeling: Tian Ma
Random Forest Modeling: Tian Ma
LightGBM Modeling: Wenyu Pan, Tian Ma
Keras Modeling: Wenyu Pan
Report: Wenyu Pan