MINeD 2025

Team: not_so_smart

Track: track-4-crest-security-ai-ml

Presented by: Ayush, Aryan, Anmol, Jigar

Hosted by: Institute of Technology, Nirma University

Problem Statement

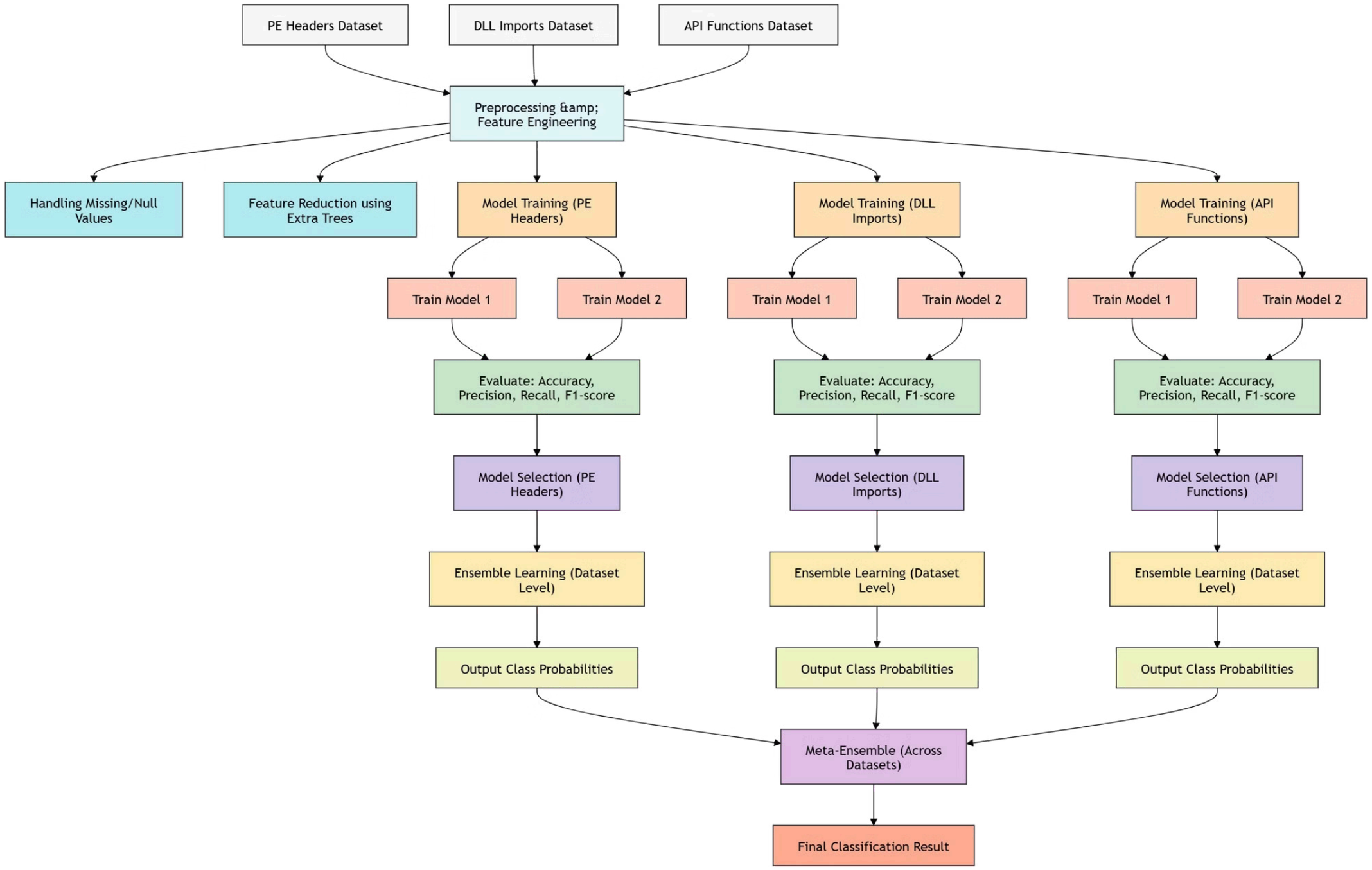
Cybersecurity threats are evolving rapidly, making traditional malware detection methods less effective against sophisticated attacks. This project aims to develop a machine learning-based malware classification system that accurately detects and classifies malware based on behavioral and structural attributes.

Participants must design, train, and evaluate a robust ML model for malware classification.

Objectives

- **Develop an efficient ML model** that generalizes well to unseen malware samples.
- **Enhance real-world malware detection and prevention** mechanisms using AI-driven techniques.
- **Optimize feature engineering** for improved model performance and scalability.

Proposed Approach for Malware Detection



- **Understanding the Datasets:**
  We have three different datasets, each capturing distinct characteristics of malware:
  - **Portable Executable (PE) Headers:** Contains 52 fields related to PE headers and 9 field values for each of 10 PE sections.
  - **DLL Imports:** Lists the DLLs imported by different malware families.
  - **API Functions:** Contains API function calls made by the malware. This dataset is particularly large (28,017 rows × 21,920 columns), requiring dimensionality reduction.

- **Modeling Each Dataset Separately:**
  Since each dataset contains unique types of features, we decided to process them individually before combining results:
  - **Preprocessing:** We handle missing values, null values, and other inconsistencies in each dataset.
  - **Feature Reduction:**
    - The API functions dataset is too high-dimensional for direct modeling, so we apply feature selection techniques (e.g., Extra Trees) to reduce the number of features while retaining important information.

- **Resampling:**
  - To address class imbalance, we apply SMOTE and compare model performance on both SMOTE-applied and original datasets.
- **Model Training & Evaluation:**
  - We trained two models on each dataset.
  - We evaluate models using key performance metrics such as accuracy, precision, recall, and F1-score.
- **Model Selection:**
  - We select models that exceed a predefined performance threshold for further ensemble learning.

- **Ensemble Learning within Each Dataset:**
  - Once we have trained multiple models on each dataset, we take their prediction probabilities and ensemble them using techniques such as stacking to improve robustness and generalization.
  - This step ensures that for each dataset (API, DLL, and PE Headers), we obtain the most reliable class probabilities.

- **Meta-Ensemble Across Datasets:**
  - After obtaining the final probability scores from API, DLL, and PE datasets (e.g., 7 probability values from each dataset), we merge them into a 21-column feature set.
  - We then train a meta-model (such as Logistic Regression) on this combined dataset to generate the final classification result.

- **How This Approach Aligns with Industry Standards:**
  - **Modular Processing:** Instead of merging raw features from different datasets into a single model, we process each dataset independently, ensuring that distinct patterns are captured effectively.
  - **Robust Feature Engineering:** By applying targeted feature selection methods, we avoid overfitting and improve interpretability.
  - **Adaptive Ensembling:** The use of stacked ensemble learning at both individual dataset and meta-level ensures improved classification accuracy and resilience against noisy data.
  - **Continuous Improvement:** This approach allows for periodic retraining and fine-tuning as new malware variants emerge.

Why Should One Choose Our Solution?

Our malware detection approach is designed to be **robust, scalable, and practical** for real-world cybersecurity applications. Here's why our solution stands out:

✅ 1. Multi-Perspective Analysis for Higher Accuracy

- Instead of relying on a single dataset, we leverage three distinct datasets (API calls, DLL imports, PE headers) to gain a **comprehensive** understanding of malware behavior.
- This approach ensures that even if a malware variant evades detection in one dataset, it can still be identified through patterns in the others.

✅ 2. Advanced Feature Selection for Better Performance

- The API dataset is extremely high-dimensional (21,920+ features), which can slow down training and introduce noise.
- We apply **Extra Trees-based feature selection** to retain only the most relevant attributes, leading to faster, more efficient, and more interpretable models.

✅ 3. Adaptive Handling of Imbalanced Data

- Cybersecurity datasets are often heavily imbalanced (i.e., some malware families are underrepresented).
- We use **SMOTE (Synthetic Minority Over-sampling Technique)** to balance the dataset, ensuring our models do not favor majority classes.
- We also compare results on **both SMOTEd and original data** to avoid unnecessary overfitting.

✅ 4. Ensemble Learning for Improved Generalization

- Instead of relying on a single model, we employ **stacking ensemble techniques** at two levels:
  a. **Within each dataset:** Multiple models are trained and ensembled for more reliable predictions.
  b. **Across datasets:** Predictions from the three datasets are merged into a **meta-classifier**, improving overall detection accuracy.
- This layered ensembling increases robustness and reduces false positives/false negatives.

✅ 5. Scalable and Modular Design

- Our approach allows for **independent improvements** in each dataset's processing.
- It can easily adapt to new malware families, additional datasets, or alternative ML techniques.
- Security teams can **fine-tune individual components** without overhauling the entire pipeline.

✅ 6. Practical Real-World Implementation

- Our methodology aligns with industry best practices:
  - **Segmented analysis** rather than blindly merging features.
  - **Stacked ensemble learning** for strong generalization.
  - **Feature selection** for computational efficiency.
- The final meta-model is lightweight and can be **deployed in real-world cybersecurity systems** without excessive computational cost.

🚀 The Bottom Line:

🔹 **Better generalization** using ensemble techniques.
🔹 **Scalability & adaptability** for real-world applications.
🔹 **Efficient and interpretable** through feature selection.

Limitations of Our Solution

- 🚧 **High Dimensionality Challenge** – Feature sets (21,920+ API features) remain computationally expensive.
- 🚧 **Zero-Day Malware Detection** – Struggles with attacks from unknown malware families.
- 🚧 **Feature Engineering Assumptions** – Extra Trees' effectiveness depends on assumptions that may not apply to all malware types.
- 🚧 **Synthetic Data Bias** – SMOTE may miss real-world variations in malware.
- 🚧 **Ensemble Complexity** – Increases accuracy but adds computational overhead and requires careful model selection.
- 🚧 **Limited Inter-Dataset Correlation** – Separate datasets (API, DLL, PE) limit the ability to capture comprehensive patterns.

Additional Information & References

Key References:

- **Ensemble Machine Learning and Feature Selection for Effective Malware Detection**: This study explores the combination of ensemble learning and feature selection to improve malware detection capabilities. (https://ieeexplore.ieee.org/abstract/document/10546128
- **Enhancing Malware Detection using Deep Learning with SMOTE and Noise Filtering**: This research investigates the use of deep learning models combined with SMOTE and noise filtering techniques to enhance malware detection accuracy. (https://ieeexplore.ieee.org/abstract/document/10851004
- **Enhancing Malware Detection with TabNetClassifier: A SMOTE-based Approach**: This work presents a method for improving malware detection using TabNetClassifier in conjunction with SMOTE. (https://www.manuscriptlink.com/society/kips/conference/ask2024/file/downloadSoConfManuscript/abs/KIPS_C2024A0197