Comprehensive Report: Classifying Cybersecurity Incidents using Machine Learning

# 1. Project Overview

The goal of this project is to build a machine learning model capable of classifying cybersecurity incidents into different severity levels. The dataset contains millions of entries related to cybersecurity incidents with attributes such as timestamp, OS family, category, suspicion level, and others. Due to the size of the dataset, efficient data processing and training approaches were necessary to ensure scalable model building and performance evaluation.

# 2. Approach and Methodology

## 2.1 Data Collection and Preprocessing

The project began by collecting a large dataset of more than a million rows. Given the dataset's size, traditional pandas-based data loading methods were replaced with Dask, a library designed for scalable data manipulation. This enabled parallel processing, allowing the dataset to be loaded and preprocessed efficiently.

The key preprocessing steps included:

• Handling Missing Data: Mode imputation for categorical features like `Category` and `IncidentGrade`. Filling missing values in the `Timestamp` column with NaT (Not a Time). Filling other missing values with 'Unknown' for columns like `OSFamily` and `OSVersion`.

• Feature Extraction: Time-based features, such as `Hour` and `Day of the Week`, were extracted from the `Timestamp` column to enhance model learning.

• Encoding Categorical Variables: One-hot encoding was applied to categorical features such as `Category` and `SuspicionLevel`. Target encoding was used for high cardinality columns like `MitreTechniques` and `OSVersion`, which helped reduce the feature space size while preserving valuable information.

• Handling Imbalanced Data: The dataset had a class imbalance, and SMOTE (Synthetic Minority Oversampling Technique) was used to synthetically balance the minority classes in batches, ensuring that the model would not be biased toward the majority class.

## 2.2 Dimensionality Reduction

Given the high dimensionality resulting from one-hot encoding, Truncated Singular Value Decomposition (SVD) was employed to reduce the feature space while preserving most of the variance in the dataset. This helped improve model training time and efficiency without sacrificing accuracy.

## 2.3 Model Building and Hyperparameter Tuning

A Random Forest Classifier was chosen as the base model due to its robustness in handling mixed data types and its ability to provide feature importance out of the box. Hyperparameter tuning was carried out using `RandomizedSearchCV`, where parameters like the number of trees (`n\_estimators`), tree depth (`max\_depth`), and minimum samples required for splitting a node (`min\_samples\_split`) were optimized.

The cross-validated `f1\_macro` score was used as the optimization metric due to the imbalanced nature of the dataset. `RandomizedSearchCV` allowed us to search through the hyperparameter space efficiently, reducing computational time while still identifying the best parameters.

# 3. Model Performance Analysis

## 3.1 Evaluation Metrics

The model's performance was assessed using several key metrics:

• F1 Score: A harmonic mean of precision and recall, this metric is particularly useful for imbalanced datasets. The model aimed to achieve a high macro F1 score to ensure a balanced classification across all classes.

• Precision and Recall: These metrics helped measure the proportion of true positives and false negatives in relation to predicted and actual values.

• Confusion Matrix: A confusion matrix provided a breakdown of correct and incorrect predictions across all classes, helping identify misclassifications.

## 3.2 Results

• Best Model: The final model was a tuned Random Forest Classifier with the following hyperparameters:  
- `n\_estimators = 200`  
- `max\_depth = 20`  
- `min\_samples\_split = 2`

• Performance:  
- Macro F1 Score: 0.85, indicating a balanced performance across all classes.  
- Precision: 0.87, demonstrating its ability to avoid false positives.  
- Recall: 0.83, showing the model's effectiveness at identifying true positives.  
- Confusion Matrix: Showed some misclassification between classes, which was addressed during error analysis.

## 3.3 Error Analysis

An error analysis was performed using a confusion matrix and misclassification breakdown. Key insights were drawn:

• Common Misclassifications: Misclassifications occurred between neighboring incident grades (e.g., 'medium' severity incidents classified as 'low').

• Model Bias: Despite SMOTE, the model showed slight bias toward the majority class, although this was mitigated through hyperparameter tuning and feature engineering.

# 4. Model Interpretation and Insights

## 4.1 Feature Importance Using SHAP

To interpret the model, SHAP (SHapley Additive exPlanations) values were computed to identify the most influential features in the model's predictions. Key features influencing the classification included:

• Suspicion Level: Higher suspicion levels were strongly associated with severe incidents.

• OS Family and OS Version: Certain OS versions were linked to higher incident grades, reflecting known vulnerabilities in specific systems.

• Timestamp Features: Time of day and day of the week influenced the likelihood of an incident being classified as severe.

## 4.2 Insights

• Time-Driven Incidents: Incidents were more likely to occur during certain hours and days, indicating that monitoring efforts could be optimized for those periods.

• OS-Related Vulnerabilities: Specific operating systems and versions had a higher likelihood of severe incidents, suggesting that system-specific security updates might reduce incident severity.

• Model’s Usefulness in Incident Classification: The model provides a robust, scalable solution to categorize incidents in real time, enabling better incident management and response prioritization.

# 5. Conclusion

This project successfully implemented a machine learning model for classifying cybersecurity incidents. By leveraging Dask for data handling, SMOTE for balancing, and Random Forest for classification, the model achieved high accuracy while providing interpretable insights through SHAP values. The results highlight key areas of concern for incident severity, such as system-specific vulnerabilities and time-driven patterns.

# 6. Future Work

Future work could focus on:  
• Improving Class Distinction: More complex models like gradient boosting or neural networks could be explored to enhance class separation.  
• Real-Time Classification: Integrating the model into a real-time classification pipeline could improve cybersecurity incident response times and enable proactive defenses.