Project Report

Microsoft Cybersecurity Incidents Classification with Machine Learning

By

Sripathi V. R

1. Introduction

This project aims to develop a machine learning model to assist Security Operation Centers (SOCs) by automating the triage process of cybersecurity incidents. The model is trained to classify incidents into True Positive (TP), Benign Positive (BP), or False Positive (FP) using the comprehensive GUIDE dataset. The ultimate goal is to provide SOC analysts with precise, context-rich recommendations, enhancing the security posture of enterprise environments.

2. Problem Statement

In SOC environments, efficiently triaging incidents is critical. With a vast number of alerts generated daily, manually categorizing incidents is time-consuming and prone to errors. The machine learning model developed in this project will predict the triage grade of incidents based on historical data and customer responses, ensuring that real threats are promptly addressed.

3. Workflow Overview

The project follows a structured workflow to ensure a comprehensive approach to model development:

3.1 Basic Overview

The initial step involves understanding the dataset structure and defining the project scope, including the evaluation metrics like macro-F1 score, precision, and recall, which are crucial for assessing model performance.

3.2 Data Exploration

3.2.1 Objective

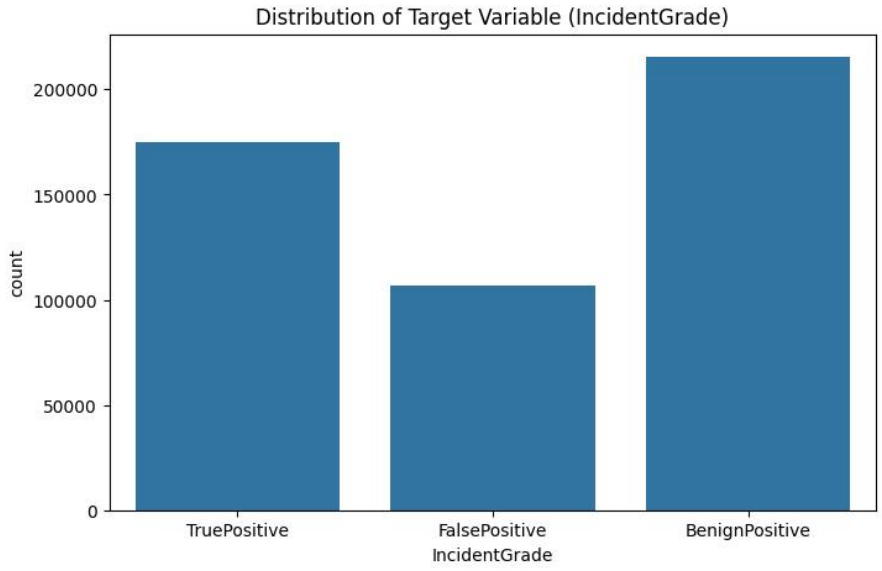
To gain insights into the dataset's structure, identify patterns, correlations, and any anomalies that might affect the model's performance.

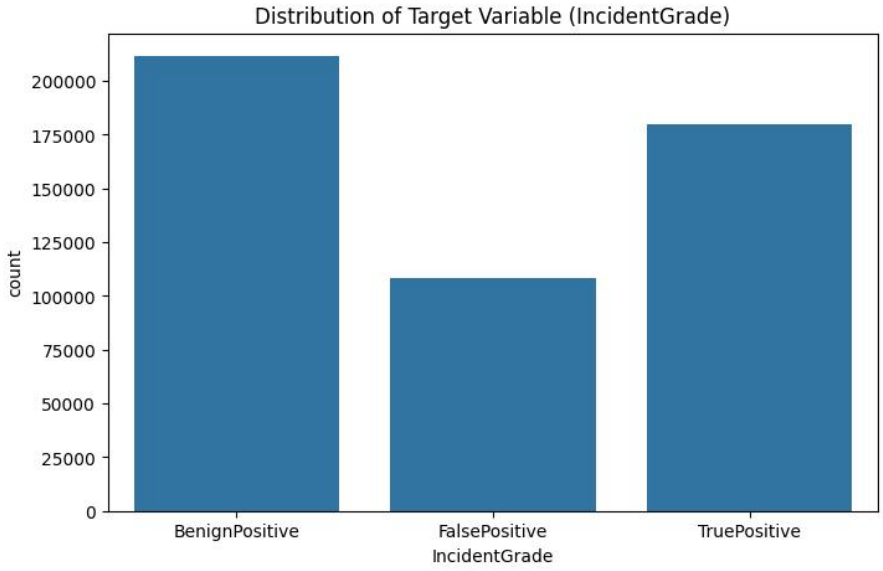
3.2.2 Process

- **Data Summary**: Key statistics, such as the shape of the dataset, data types, and missing values, are computed.
- **Visualization**: The distribution of the target variable IncidentGrade is visualized, revealing class imbalances that must be addressed during model training.

3.2.3 Key Findings

- **Class Imbalance**: Significant imbalance in the target variable, with BenignPositive being the most frequent class.
- **Missing Values**: Several features, including MitreTechniques and ActionGrouped, have a high proportion of missing values.





3.3 Data Preprocessing

3.3.1 Objective

To clean and transform the data into a format suitable for model training, addressing issues such as missing values, encoding categorical variables, and scaling numerical features.

3.3.2 Process

- **Handling Missing Data**: Missing values were imputed using strategies like forward fill and mean imputation, depending on the nature of the feature.
- **Feature Engineering**: New features were derived, such as timestamp-based features, and redundant features were removed.
- **Scaling**: Numerical features were standardized to ensure that all features contribute equally to the model training.
- **Class Cardinality Handling:** Categorical variables were limited to their top 19 unique values, with the rest grouped as 'Other'.

3.3.3 Key Outcomes

- **Cleaned Dataset**: A fully processed dataset ready for model training.

3.4 Data Splitting and Sampling

3.4.1 Objective

To split the dataset into training and validation sets to facilitate model training and evaluation.

3.4.2 Process

- **Train-Validation Split**: The data was split into training and validation sets with an 80-20 ratio, ensuring that the class distribution remains consistent.
- **Stratified Sampling**: This technique was used to maintain the balance of classes in both sets, preventing skewed results during model training.

3.4.3 Key Outcomes

- **Balanced Training and Validation Sets**: Ensures reliable model training and evaluation.

3.5 Model Selection and Training

3.5.1 Objective

To select and train various machine learning models to find the best-performing model for the classification task.

3.5.2 Models Used

1) Logistic Regression: Baseline model achieving 59% accuracy with a macro-F1 score of 0.55.

2) Decision Tree: Achieved 81% accuracy and outperformed Logistic Regression in handling class imbalance.

3) Random Forest: Best performing model with an accuracy of 80% and macro-F1 score of 0.65.

4) XGBoost: Achieved 75% accuracy but slightly lower macro-F1 score of 0.58.

5) LightGBM: Similar performance to XGBoost, with an accuracy of 75% and macro-F1 score of 0.60.

6) Neural Network: Slightly lower macro-F1 score of 0.57, but performed robustly on complex patterns.

3.5.3 Key Insights from Outputs

Best Performance: Random Forest demonstrated the best overall performance, balancing precision, recall, and F1-score.

Model Complexity: Neural Networks, while promising, require more tuning and additional data to match the tree-based models' performance.

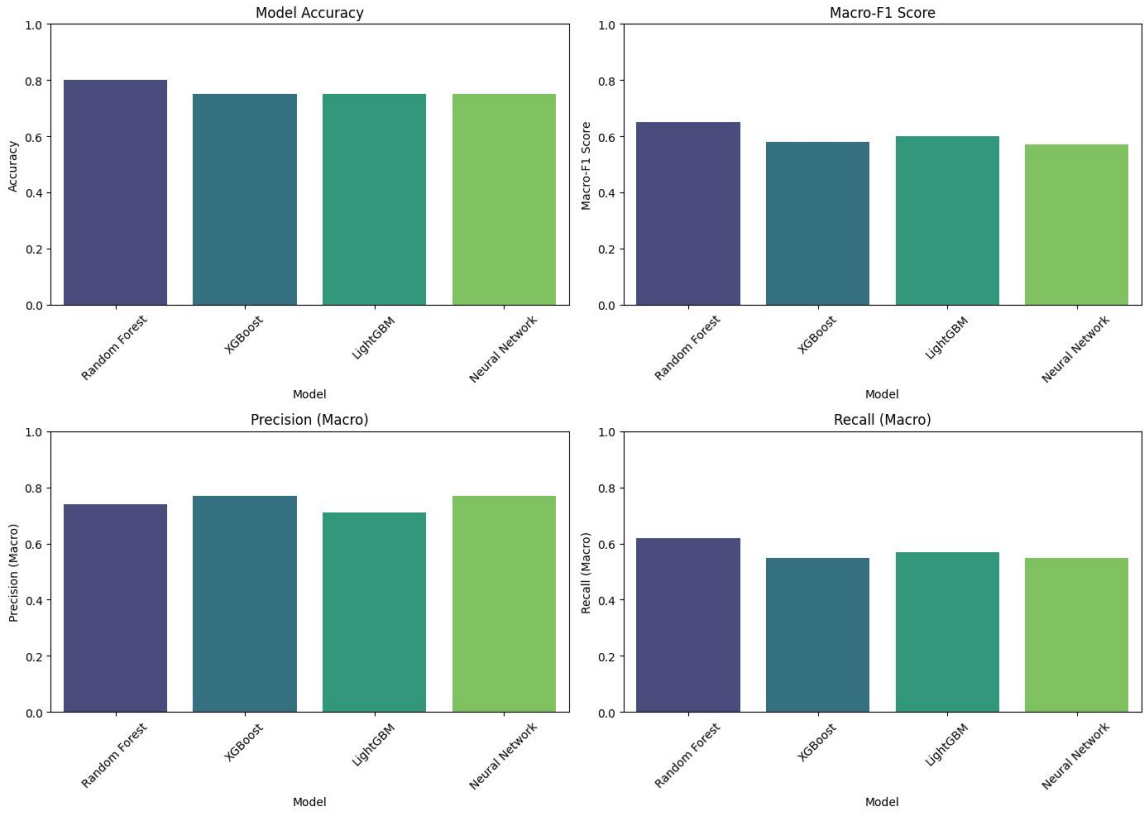
3.6 Model Evaluation and Tuning

3.6.1 Objective

To evaluate the models on the validation set using metrics like macro-F1 score, precision, and recall, and to fine-tune the models for optimal performance.

3.6.2 Evaluation Metrics

- **Macro-F1 Score**: Assesses the model's performance across all classes, treating each class equally.
- **Precision and Recall**: Precision measures the accuracy of the model's positive predictions, while recall measures its ability to identify all relevant instances.



3.6.3 Process

- RandomizedSearchCV: Used for hyperparameter tuning, optimizing parameters like n_estimators, max_depth, min_samples_split, and min_samples_leaf.
- Class Imbalance Handling: Implemented Balanced Random Forest for improved performance on minority classes.

3.6.3 Outputs

Random Forest (tuned): Achieved a macro-F1 score of 0.74 and improved recall for False Positive class from 70% to 77%.

3.7 Model Interpretation

3.7.1 Objective

To interpret the model's predictions and identify important features.

3.7.2 Techniques

Feature Importance: The most influential features included OrgId, DetectorId, EntityType, and AlertTitle.

3.7.3 Outputs

Top Features: OrgId had the highest importance (0.215), followed by EntityType (0.158), and DetectorId (0.081).

3.8 Final Evaluation on Test Set

3.8.1 Objective

To evaluate the performance of the final model on the test set.

3.8.2 Process

Test Set Results: The Random Forest model achieved a macro-F1 score of 0.513, with macro-precision of 0.547 and macro-recall of 0.4899.

3.9 Comparison with Baseline Models

3.9.1 Findings

Significant Improvement: The Random Forest model significantly outperformed baseline models like Logistic Regression and Decision Tree, particularly in handling class imbalances.

3.9.2 Insights

Model Complexity: Random Forest and XGBoost demonstrated their ability to handle complex feature interactions and high-dimensional data better than simpler models.

4. Key Findings and Recommendations

4.1 Key Findings

- **Random Forest Outperformance:** The Random Forest model was the top performer across all metrics.
- **Feature Importance:** Features such as OrgId, DetectorId, and AlertTitle were consistently important across models.

4.2 Recommendations

- SOC Integration: The Random Forest model can be integrated into SOC workflows to automate incident triaging.
- Continuous Monitoring: Regular updates and monitoring of the model are necessary to maintain accuracy as new incidents emerge.