**Project Report: Microsoft - Classifying Cybersecurity Incidents**

Name: Dinesh V

Date: 20-10-2025

Project: End-to-End Cybersecurity Incident Classification Model

**1. Executive Summary**

This report details the development of an end-to-end machine learning solution for Microsoft, designed to enhance the efficiency of its Security Operation Centers (SOCs). The primary business objective was to reduce analyst workload and improve response times by automatically predicting the triage grade of new cybersecurity incidents.

A machine learning pipeline was successfully built, beginning with data ingestion, extensive preprocessing, and feature engineering on the Microsoft GUIDE dataset. An **XGBoost Classifier** model was trained and selected for its high performance, achieving a final **accuracy of 89.3%** and a **Macro-F1 Score of 0.89** on the unseen test dataset.

The final model and all necessary preprocessing objects (label encoders) were saved as .pkl files, making them ready for deployment in an automated SOC workflow.

**2. Introduction & Problem Statement**

**2.1. Business Problem**

Security Operation Centers (SOCs) are overwhelmed by the sheer volume of daily security alerts. Analysts must manually triage these incidents, classifying them as True Positives (real threats), Benign Positives (legitimate but noisy activity), or False Positives (errors). This manual process is time-consuming, leads to analyst fatigue, and can delay the response to critical, genuine threats.

**2.2. Project Objectives**

The project's goal was to leverage historical data to build a classification model to solve this problem. The key objectives were:

* Analyze the Microsoft GUIDE dataset to identify key factors that predict an incident's grade.
* Build, train, and benchmark several machine learning models to find the top performer.
* Develop a final model capable of achieving high accuracy and a high macro-F1 score.
* Ensure the final model generalizes well to new, unseen data (test.csv).

**3. Data & Methodology**

**3.1. Data Source**

The project utilized two primary datasets:

* train.csv: 4.7 million incident records used for training and validation.
* test.csv: 4.1 million incident records used for the final, blind evaluation.

**3.2. Data Preprocessing**

To prepare the data for modeling, several key steps were taken:

* **Handling Missing Data:** Columns with over 50% missing values (e.g., MitreTechniques, ActionGrouped) were dropped to avoid imputing unreliable data.
* **Correlation Analysis:** A correlation matrix was used to identify and remove highly-correlated numerical features, which prevents multicollinearity and simplifies the model.

**3.3. Feature Engineering**

To extract the maximum predictive value from the data, the following features were engineered:

* **Timestamp Conversion:** The Timestamp column was converted into three new numerical features: Day, Month, and Hour. This allows the model to capture time-based patterns in security incidents.

**3.4. Preprocessing Pipeline**

A reproducible pipeline was built for model deployment:

* **Label Encoding:** LabelEncoder was applied to all categorical features (Category, EntityType, etc.).
* **Saving Encoders:** Critically, each fitted encoder was **saved as a .pkl file**. This is a best practice that ensures the unseen test data (and future production data) is transformed using the *exact same* integer mapping as the training data, preventing errors.
* **Feature Selection:** SelectKBest was used to identify and select the **top 15 most influential features** for the model.

**4. Model Development & Evaluation**

**4.1. Model Benchmarking**

The processed training data was split (80% train / 20% validation) to benchmark several models. The XGBoost Classifier was selected as the final model due to its superior performance.

| **Model** | **Validation Accuracy** |
| --- | --- |
| **XGBoost (Selected Model)** | **91.7%** |
| LGBM | 89.5% |
| Random Forest | 77.6% |
| Decision Tree | 77.4% |
| Logistic Regression (Baseline) | 52.6% |

**4.2. Final Model Evaluation**

The saved xgboost\_model.pkl was loaded and evaluated on the fully independent test.csv dataset. The model's performance was strong and consistent with the validation results, indicating it generalized well.

* **Final Accuracy:** **89.3%**
* **Macro-F1 Score:** **0.89**

**Final Classification Report (Test Set)**

| **Class** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- |
| 0 (BenignPositive) | 0.87 | 0.93 | 0.90 |
| 1 (FalsePositive) | 0.88 | 0.81 | 0.85 |
| 2 (TruePositive) | 0.93 | 0.90 | 0.91 |
|  |  |  |  |
| **Macro Avg** | **0.89** | **0.88** | **0.89** |
| **Weighted Avg** | **0.89** | **0.89** | **0.89** |

**5. Deployment Strategy**

The trained xgboost\_model.pkl and all preprocessing objects (the .pkl label encoders) were saved for production use.

* **Application:** The model is designed to be integrated into an automated SOC pipeline.
* **Functionality:**
  1. New incident data is ingested by the system.
  2. The pipeline loads the saved encoders and SelectKBest object to transform the new data.
  3. The saved xgboost\_model.pkl model predicts the triage grade (TP, BP, or FP).
  4. The application uses this prediction to automatically route the incident. For example, **True Positives (TP)** are escalated to a high-priority queue, while **False Positives (FP)** can be automatically suppressed, freeing up analyst time.

**6. Conclusion & Future Work**

**6.1. Conclusion**

This project successfully delivered an end-to-end, data-driven solution for classifying cybersecurity incidents. The final XGBoost model (89.3% accuracy) provides an accurate and reliable tool that solves the core business problem, empowering SOC analysts to prioritize real threats more effectively.

**6.2. Future Work**

* **Model Optimization:** Implement GridSearchCV or RandomizedSearchCV to perform hyperparameter tuning on the XGBoost model. This was a key recommendation in the project brief and could provide a significant performance boost.
* **Handle Class Imbalance:** Implement advanced techniques to further improve recall for the minority "FalsePositive" class, such as using the stratify=y parameter in the train\_test\_split or applying the scale\_pos\_weight parameter in XGBoost.
* **Automated Retraining:** Create a pipeline to automatically monitor and retrain the model on new incident data. This will prevent model drift as attacker techniques and behaviors evolve over time.