Microsoft: Classifying Cybersecurity Incidents with Machine Learning

Developed by: Kadar Meeran

1. Objective:

The objective of this project is to classify cybersecurity incidents by predicting their triage grade using machine learning techniques. This classification aims to assist Security Operation Centers (SOCs) in managing and prioritizing incidents more efficiently, enabling quicker responses to critical threats.

---

2. Methodology:

1. Data Exploration & Preprocessing:

- **Data Source:** The training and test datasets contained millions of rows with features like 'AlertTitle', 'Category', 'EntityType', 'EvidenceRole', and others. The target variable for prediction is 'IncidentGrade.'
- **Handling Missing Values:** Significant missing values were found in columns such as 'MitreTechniques', 'IncidentGrade', and others. These were managed by imputing values where possible or dropping irrelevant columns.
- **Feature Extraction:** From timestamp data, features such as Year, Month, Day, and Hour were extracted. Irrelevant timestamp columns were dropped.
- **Feature Encoding:** Categorical columns were label encoded to make them interpretable by machine learning models, ensuring more efficient training.

2. Data Balancing:

- **SMOTE (Synthetic Minority Over-sampling Technique):** Applied to tackle class imbalance in the target variable 'IncidentGrade,' which had a skewed distribution of incident classes.

3. Model Selection & Training:

Several models were tested on a downsampled version of the dataset to speed up processing, including:

- Logistic Regression
- Random Forest
- Decision Tree
- XGBoost

- LightGBM
- Gradient Boosting

4. Model Evaluation:

Model performance was evaluated using the following metrics:

- **Accuracy:** Measures the overall correctness of the model.
- **Precision:** Measures the accuracy of the positive predictions.
- **Recall:** Measures the model's ability to capture all positive instances.
- **F1 Score:** The harmonic mean of precision and recall.
- **Macro-F1 Score:** Averages the F1 scores across all classes, treating each class equally regardless of size.

---

3. Model Performance Summary:

Model	Accuracy	Macro-F1 Score	Precision	Recall
Logistic Regression	0.63	0.54	0.64	0.55
Decision Tree	0.70	0.67	0.70	0.66
Random Forest	0.70	0.67	0.70	0.66
XGBoost	0.68	0.62	0.71	0.61
LightGBM	0.68	0.61	0.72	0.61
Gradient Boosting	0.64	0.55	0.68	0.56

---

4. Findings:

1. Best Performing Model:

- **Random Forest** had the best performance based on the Macro-F1 Score (0.67), indicating a good balance between precision and recall across all incident categories.
- **Decision Tree** also performed well, but Random Forest typically produces more stable results due to its ensemble approach.

2. Accuracy vs Macro-F1 Score:

- Random Forest, Decision Tree, and XGBoost showed similar accuracy (~0.70), but Random Forest's superior Macro-F1 score indicated better performance in predicting all incident grades fairly.

3. Error Analysis:

- **Logistic Regression** had a lower recall for certain classes, particularly in predicting incidents in the minority class, leading to misclassification in critical instances.

---

5. Rationale for Model Selection:

- **Random Forest** was selected as the final model due to its balanced performance across key metrics (accuracy, F1 score, precision, recall). Its ability to handle class imbalance and prevent overfitting through ensembling makes it highly suitable for this cybersecurity use case.
- While **XGBoost** and **LightGBM** achieved higher precision scores, their lower recall for minority classes made them less optimal for this task, where accurately identifying all incident types is crucial.

---

6. Model Improvement:

1. Hyperparameter Tuning:

- Further improvements can be made by using Grid Search to optimize parameters such as `n_estimators`, `max_depth`, and `min_samples_split` for Random Forest.

2. Cross-Validation:

- Using k-fold cross-validation ensures robustness by evaluating the model on different data splits, reducing the risk of overfitting and improving generalization to unseen data.

3. Feature Engineering:

- Additional exploration of feature interactions or the creation of new features, such as contextual information related to entities (e.g., 'EntityType', 'Category'), could further enhance the model's understanding of the incidents.

---

7. Conclusion:

This project demonstrated the application of machine learning in classifying cybersecurity incidents based on historical data. The **Random Forest** model provided the best performance, offering a balanced trade-off between accuracy and fairness across all classes. Further refinements such as hyperparameter tuning and feature engineering could lead to even better results, making this approach a valuable tool for SOCs in prioritizing and addressing cybersecurity threats more effectively.