Microsoft: Classifying Cybersecurity Incidents with Machine Learning

Objective

The project aims to assist Security Operation Centers (SOCs) by creating a machine learning model to predict the triage grade (true positive, benign positive, false positive) of cybersecurity incidents. The classification will enable SOC analysts to prioritize incident response, reducing response time to critical threats.

Skills Acquired

- **Data Preprocessing and Feature Engineering**
- **Machine Learning Classification Techniques**
- **Model Evaluation Metrics (Macro-F1 Score, Precision, Recall)**
- **Cybersecurity Concepts and Frameworks (MITRE ATT&CK)**
- **Handling Imbalanced Datasets**
- **Model Benchmarking and Optimization**

Problem Statement

- As a data scientist at Microsoft, the task is to create a model that classifies incidents based on historical evidence and customer feedback. Using the GUIDE dataset, the model should accurately predict triage grades (TP, BP, FP) to support SOCs in guided response systems, aiming to improve enterprise security.

Business Use Cases

1. **SOC Automation**: Streamline incident triage, allowing SOC analysts to respond more efficiently to threats.
2. **Incident Response**: Automated guidance on action steps for different incidents.
3. **Threat Intelligence**: Enhanced threat detection by leveraging historical and customer-provided evidence.
4. **Enterprise Security**: Minimized false positives and prioritized response to real threats.

Approach

1. **Data Exploration and Understanding**
    - Inspect and analyze the structure of the train.csv data.
    - Perform EDA, focusing on class imbalance and correlations.
2. **Data Preprocessing**
    - Handle missing values through imputation or exclusion.
    - Perform feature engineering and categorical encoding.
3. **Data Splitting**
    - Split the data into training and validation sets, using stratification if there's class imbalance.
4. **Model Selection and Training**
    - Begin with a baseline model for benchmarking.
    - Experiment with advanced models (Random Forest, XGBoost, LightGBM, etc.), utilizing cross-validation and hyperparameter tuning.

5. **Model Evaluation and Tuning**
    - Focus on macro-F1, precision, and recall.
6. **Model Interpretation**
    - Analyze feature importance and conduct error analysis.
7. **Final Evaluation on Test Set**
    - Evaluate the model's performance on the test.csv, comparing it with baseline results

2. Methodology

1. Data Exploration & Preprocessing

- **Data Source**: The datasets used for training and testing contained millions of rows with features such as 'AlertTitle,' 'Category,' 'EntityType,' 'EvidenceRole,' among others. The target variable for prediction was 'IncidentGrade.'
- **Handling Missing Values**: Columns with significant missing values, including 'MitreTechniques' and 'IncidentGrade,' were addressed by imputing values where feasible and removing irrelevant columns.
- **Feature Extraction**: Features like Year, Month, Day, and Hour were extracted from timestamp data, and unnecessary timestamp columns were dropped.
- **Feature Encoding**: Categorical columns were label-encoded, allowing machine learning models to interpret them effectively and enhancing training efficiency.

2. Model Selection & Training

Multiple models were tested on a downsampled dataset to improve processing efficiency. The models tested included:

- **Logistic Regression**
- **Decision Tree**
- **Random Forest**
- **XGBoost**
- **LightGBM**
- **Gradient Boosting**

3. Model Evaluation

The models were evaluated based on the following metrics:

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

3. Model Performance Summary

The following table compares the performance of each model based on key metrics:

Model	Accuracy	Macro-F1 Score	Precision	Recall
Logistic Regression	**0.6460**	**0.55**	**0.67**	**0.65**
Random Forest	**0.7093**	**0.68**	**0.71**	**0.71**
Decision Tree	**0.7083**	**0.68**	**0.71**	**0.71**
XGBoost	**0.6860**	**0.62**	**0.73**	**0.69**
LightGBM	**0.6849**	**0.62**	**0.72**	**0.68**
Gradient Boosting	**0.6533**	**0.56**	**0.70**	**0.65**

4. Findings

1. Best Performing Model

- **Random Forest**: Achieved the best overall performance with an accuracy of 70.93% and a Macro-F1 Score of 0.68. Its precision and recall scores (0.71 each) indicate a balanced ability to predict all incident classes fairly. Random Forest outperformed other models by maintaining a balance between accuracy, precision, recall, and the Macro-F1 Score.

2. Accuracy vs. Macro-F1 Score

- Both Random Forest and Decision Tree performed similarly in terms of accuracy (~0.71), but Random Forest had a slight advantage in Macro-F1 Score, indicating better performance across different incident grades. XGBoost and LightGBM exhibited slightly higher precision scores but lower recall, making them less suitable for situations where capturing all incidents accurately is critical.

3. Error Analysis

- **Logistic Regression**: Had a lower recall for minority classes, leading to misclassifications in critical instances. This makes it less suitable for applications where high recall across all incident types is essential.
- **XGBoost and LightGBM**: Showed higher precision but slightly lower recall, indicating these models excel in cases where positive predictions are more likely to be correct, but they may miss some positive instances in the minority classes.

5. Rationale for Model Selection

- **Final Model**: Random Forest was selected as the final model due to its superior balance across key metrics (accuracy, F1 score, precision, recall). Its ensemble nature allows it to handle class imbalance effectively and reduce overfitting, making it well-suited to this cybersecurity classification task.
- **Comparison with Other Models**: While XGBoost and LightGBM achieved higher precision, their relatively lower recall scores indicated they might miss some critical incidents, which could be a limitation in a SOC environment where catching all types of incidents is crucial.

6. Model Improvement

1. Hyperparameter Tuning

- Using techniques like Grid Search or Random Search to optimize parameters such as n_estimators, max_depth, and min_samples_split for the Random Forest model could improve accuracy and generalizability.

2. Cross-Validation

- Implementing k-fold cross-validation can further ensure robustness by evaluating the model across different data splits, helping to reduce overfitting and improving the model’s performance on unseen data.

3. Feature Engineering

- Additional feature engineering, such as exploring interactions or deriving contextual information from features like 'EntityType' and 'Category,' could further enhance the model's ability to understand incident characteristics.

7. Conclusion

This project successfully demonstrated the application of machine learning for classifying cybersecurity incidents. The Random Forest model emerged as the most balanced performer, making it a valuable tool for SOCs to prioritize and address cybersecurity threats more effectively. By implementing further improvements in hyperparameter tuning and feature engineering, this approach could offer even greater support to SOC analysts in real-time threat management.

THANK YOU