Microsoft : Classifying Cybersecurity Incidents with Machine Learning

Objective

1. **Preprocess and Clean Data**
   Preprocess and clean large-scale incident data (1.3M rows).

2. **Feature Engineering**
   Engineer features for better model performance.

3. **Train and Evaluate Model**
   Train and evaluate a high-performing classification model.

4. **Provide Insights**
   Provide interpretability and actionable insights from predictions.

Dataset Description

Key Details

Size: 1,297,443 rows × 39 columns

Key Features: Category, IncidentGrade, EntityType, Hour, DayOfWeek, etc.

Target Variable Distribution

BenignPositive: 2,054,774

TruePositive: 1,662,087

FalsePositive: 1,015,782

Missing Values Summary

Columns dropped (missing >50%): ActionGrouped, ResourceType, etc.

Imputed numerical and categorical columns with median/mode.

Preprocessing Steps

Data Cleaning

Removed duplicates (0 rows).

Handled missing values by imputation.

Outlier Removal

Used IQR method for numerical features.

Feature Engineering

Extracted temporal features: Year, Month, Hour, DayOfWeek.

Encoded categorical features with Label Encoding and One-Hot Encoding.

Scaling

Applied Min-Max Scaling to numerical features.

Choosing the Right Model

Model **Chosen**

After careful evaluation, we selected the **Random Forest Classifier** as our machine learning model.

Why Random Forest?

- Handles large, complex datasets efficiently
- Robust to overfitting through ensemble learning
- Provides valuable feature importance insights
- Performs exceptionally well on imbalanced data with SMOTE

Training Process

1. Split data into train and test sets.
2. Addressed class imbalance using SMOTE.
3. Conducted hyperparameter tuning via GridSearchCV.
4. Evaluated model with cross-validation (F1 score metric).
5. Saved the best-performing model for testing.

Test Dataset Workflow

1. Data Cleaning and Preprocessing (same as training).
2. Feature alignment with training dataset.
3. Loaded saved model for predictions.
4. Evaluated test performance.

Challenges Faced

Class Imbalance

Solved using SMOTE.

High Missing Values

Dropped columns (>50%) and imputed remaining.

Overfitting

Addressed with cross-validation and hyperparameter tuning.

Temporal Data Handling

Extracted features like Hour, DayOfWeek, etc.

Final Results

96%

Train F1 Score

59%

Test F1 Score

Reason for Difference:

Despite performing **cross-validation and hyperparameter tuning** with RandomSearch to prevent overfitting, the **model consistently overfit**. Running adjustments **took 3–4 hours due to the large dataset**, making local execution challenging. Applying **SMOTE further increased runtime** but did not resolve the overfitting issue.

Conclusion and Future Work

1. Conclusion
2. Successfully Implemented Pipeline
3. High Training Accuracy
4. Areas for Improvement