**Microsoft: Classifying Cybersecurity Incidents with Machine Learning**

**Problem Statement:**

To develop a machine learning model capable of accurately predicting the triage grade of cybersecurity incidents that can enhance the efficiency of Security Operations Centers (SOCs).

**Objective:**

To build a robust classification model that categorizes incidents into three categories: **True Positive (TP)**, **Benign Positive (BP)**, or **False Positive (FP)**, based on historical data and customer feedback. The model will support with guided response to SOC analysts with precise, context-aware recommendations.

**Business Use Cases:**

* Security Operation Centers (SOCs): Automates the triage process by accurately classifying incidents, enabling SOC analysts to prioritize and respond to critical threats more effectively.
* Incident Response Automation: Supports guided response systems that suggest appropriate actions for different incidents, speeding up threat mitigation.
* Threat Intelligence: Improves threat detection by integrating historical data and customer feedback into the triage process, enhancing the identification of true and false positives.

Enterprise Security Management: Strengthens overall security by reducing false positives and ensuring timely responses to actual threats, improving the enterprise's security posture.

**Methodology, models used, and evaluation results:**

1. **Data Exploration and Understanding:**
   1. **Initial Inspection:**
      * Load the train.csv dataset and
      * Understand the structure of the data, including the number of features, types of variables (categorical, numerical), and the distribution of the target variable (TP, BP, FP).
   2. **Exploratory Data Analysis (EDA):**
      * Understand the statistical summaries
      * Visualize the distribution of data for the different classes.
      * Perform crosstab and Chi2 test to see the correlation between the columnsi and ‘IncidentGrade’ column.
2. **Data Preprocessing:**
   1. **Handling Missing Data:**

* Identify any missing values in the dataset.
* Find the percentage of null values and drop columns having exceeding considerable range.
* Dropna for other columns
  1. **Feature Engineering:**
* Create new feature- day of the week and drop timestamp column.
  1. **Encoding Categorical Variables:**
* Convert categorical features into numerical representations using one-hot encoding techniques if the number of categories as not sufficiently large. As the training data is enormously large -choose top 3 categories and rename the rest as ‘Others’
* Label encoding is done for other columns

1. **Data Splitting:**

* Prepare the independent X and dependent y data
  1. **Train-Validation Split:**
* The train.csv data is split into training and validation sets with 70-30 ratio.
  1. **Stratification:**
* Stratify sampling such that both the training and validation sets have similar class distributions.

1. **Model Selection and Training & Model Evaluation:**
   1. **Baseline Model:**

* Logistic regression and decision tree and decision tree models are built to understand how complex the model needs to be.
  1. **Performance Metrics:**
* Evaluate the model using the validation set, focusing on macro-F1 score, precision, and recall. Analyze these metrics across different classes (TP, BP, FP) to ensure balanced performance.

 Benign Positive (BP) → 0

 False Positive (FP) → 1

 True Positive (TP) → 2

* 1. **Cross-Validation:**
* Implement cross-validation (e.g., k-fold cross-validation) to ensure the model's performance is consistent across different subsets of the data.

|  |  |  |
| --- | --- | --- |
|  | **LogisticRegression** | **DecisionTreeClassifier** |
| **Cross validation score** | [0.56441657 0.56293804 0.56214176 0.56022849 0.56422353] | [0.99648724 0.99646717 0.99635465 0.99634462 0.99625376] |
| **Train** | Accuracy: 0.5579611053144731  Precision: 0.5384783536282706  Recall: 0.49187611540465354  F1 Score: 0.4726871073897092 | Accuracy: 1.0  Precision: 1.0  Recall: 1.0  F1 Score: 1.0 |
| **Validation** | Accuracy: 0.5577233109714226  Precision: 0.5385902566007681  Recall: 0.49175425580800497  F1 Score: 0.47272175286448387 | Accuracy: 0.9964444561829803 Precision: 0.9960795986917588  Recall: 0.996130548569481  F1 Score: 0.9961050585294067 |

* 1. **Advanced Models:** Experiment with more sophisticated models such as Random Forests, Gradient Boosting Machines (e.g., XGBoost, LightGBM), and Neural Networks.

|  |  |  |
| --- | --- | --- |
|  | **RandomForestClassifier** | **XGBClassifier** |
| **Cross validation score** |  | [0.91206011 0.91216523 0.91417987 0.9127716 0.91061851] |
| **Train** | Accuracy: 0.9999990755899354 Precision: 0.9999988276199981  Recall: 0.9999989869316153  F1 Score: 0.9999989072756335 | Accuracy: 0.914963650215381 Precision: 0.9209997429848246  Recall: 0.9017913557555817  F1 Score: 0.9100184623356014 |
| **Validation** | Accuracy: 0.9828123184195235  Precision: 0.9828287114835147  Recall: 0.9805056682407938  F1 Score: 0.9816360624104764 | Accuracy: 0.9147382599968306 Precision: 0.9207625697483933  Recall: 0.9015620342160814  F1 Score: 0.9097794679941752 |

|  |  |  |
| --- | --- | --- |
|  | **LGBMClassifier** | **NeuralNetwork** |
| **Cross validation score** | [0.89475041 0.89381227 0.89426121 0.89540061 0.89385183] |  |
| **Train** | Accuracy: 0.8939349059485392  Precision: 0.9050638953719998  Recall: 0.8769508437774941  F1 Score: 0.8882059824524147 | Accuracy: 0.4343  Precision: 0.1448  Recall: 0.3333  F1 Score: 0.2019 |
| Validation | Accuracy: 0.8937462363319423  Precision: 0.9049366415001087  Recall: 0.8767747218441385  F1 Score: 0.8880369987427361 | Accuracy: 0.4343  Precision: 0.1448  Recall: 0.3333  F1 Score: 0.2019 |

1. **Model Interpretation:**

* Analyzing the scores RamdomForestClassifier seems to perform well than the other models.
  1. **Feature Importance:**
* Model-specific feature importance measures of RandomForestClassifier is used to analyze feature importance to understand which features contribute most to the predictions. And top 20 columns has been picked to further fine cue the model training.
  1. **Cross-Validation:**
* Implement cross-validation (e.g., k-fold cross-validation) to ensure the model's performance is consistent across different subsets of the data. This reduces the risk of overfitting and provides a more reliable estimate of the model's performance.

Random Forest Classifier :[0.98991443 0.98998679 0.98999418 0.99011251 0.98989382]

* 1. **Pickling:**
* Store the trained model in pickle format

1. **Final Evaluation on Test Set:**
   1. **Testing:**

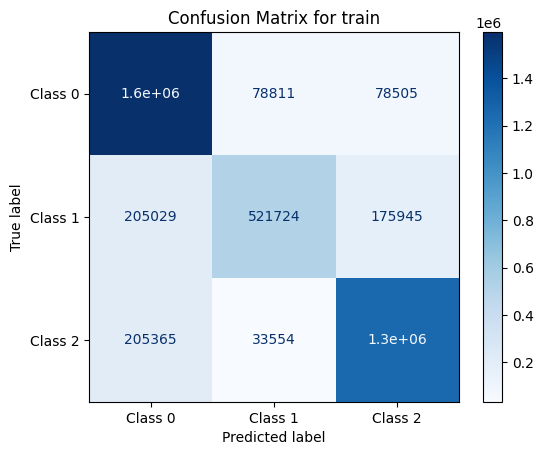
* Evaluate the loaded model on the test.csv dataset. Assess the macro-F1 score, precision, and recall to assess how well the model generalizes to unseen data.

Accuracy: 0.8126300629316546

Precision: 0.8164884366677265

Recall : 0.7760404184971149

F1 Score: 0.7878361490102618



1. Recommendations for Deployment

* **SOC Integration**: Integrate the model with SOC tools to automate triage and provide context-rich recommendations for analysts.
* **Scalability**: Ensure real-time predictions via microservices architecture and support batch processing for historical data.
* **Monitoring & Maintenance**: Continuously track model performance and set up an automated retraining pipeline to adapt to evolving threats.
* **Handling Misclassifications**: Implement a feedback loop for analysts to correct errors and use confidence scores to flag low-confidence predictions for human review.
* **Security & Compliance**: Ensure data privacy, security, and compliance (e.g., GDPR, HIPAA) with role-based access and audit trails for transparency.
* **Analyst Interaction**: Provide an intuitive dashboard with explainable AI features (e.g., SHAP, LIME) to help analysts understand predictions.
* **Future Improvements**: Incorporate external threat intelligence, explore multi-model systems, and allow customization for different enterprise environments.