Project Title	**Microsoft : Classifying Cybersecurity Incidents with Machine Learning**
Skills take away From This Project	• **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**
Domain	**Cybersecurity and Machine Learning**

Problem Statement:

Imagine you are working as a data scientist at Microsoft, tasked with enhancing the efficiency of Security Operation Centers (SOCs) by developing a machine learning model that can accurately predict the triage grade of cybersecurity incidents. Utilizing the comprehensive GUIDE dataset, your goal is to create a classification model that categorizes incidents as true positive (TP), benign positive (BP), or false positive (FP) based on historical evidence and customer responses. The model should be robust enough to support guided response systems in providing SOC analysts with precise, context-rich recommendations, ultimately improving the overall security posture of enterprise environments.

You need to train the model using the **train.csv** dataset and provide evaluation metrics—macro-F1 score, precision, and recall—based on the model's performance on the **test.csv** dataset. This ensures that the model is not only well-trained but also generalizes effectively to unseen data, making it reliable for real-world applications.

Business Use Cases:

The solution developed in this project can be implemented in various business scenarios, particularly in the field of cybersecurity. Some potential applications include:

- **Security Operation Centers (SOCs):** Automating the triage process by accurately classifying cybersecurity incidents, thereby allowing SOC analysts to prioritize their efforts and respond to critical threats more efficiently.
- **Incident Response Automation:** Enabling guided response systems to automatically suggest appropriate actions for different types of incidents, leading to quicker mitigation of potential threats.
- **Threat Intelligence:** Enhancing threat detection capabilities by incorporating historical evidence and customer responses into the triage process, which can lead to more accurate identification of true and false positives.
- **Enterprise Security Management:** Improving the overall security posture of enterprise environments by reducing the number of false positives and ensuring that true threats are addressed promptly.

Approach:

1. **Data Exploration and Understanding:**
    a. **Initial Inspection:** Start by loading the `train.csv` dataset and perform an initial inspection to 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).
    b. **Exploratory Data Analysis (EDA):** Use visualizations and statistical summaries to identify patterns, correlations, and potential anomalies in the data. Pay special attention to class imbalances, as they may require specific handling strategies later on.
2. **Data Preprocessing:**
    a. **Handling Missing Data:** Identify any missing values in the dataset and decide on an appropriate strategy, such as imputation, removing affected rows, or using models that can handle missing data inherently.
    b. **Feature Engineering:** Create new features or modify existing ones to improve model performance. For example, combining related features, deriving new features from timestamps (like hour of the day or day of the week), or normalizing numerical variables.
    c. **Encoding Categorical Variables:** Convert categorical features into numerical representations using techniques like one-hot encoding, label

encoding, or target encoding, depending on the nature of the feature and its relationship with the target variable.

3. **Data Splitting:**
    a. **Train-Validation Split:** Before diving into model training, split the `train.csv` data into training and validation sets. This allows for tuning and evaluating the model before final testing on `test.csv`. Typically, a 70-30 or 80-20 split is used, but this can vary depending on the dataset's size.
    b. **Stratification:** If the target variable is imbalanced, consider using stratified sampling to ensure that both the training and validation sets have similar class distributions.
4. **Model Selection and Training:**
    a. **Baseline Model:** Start with a simple baseline model, such as a logistic regression or decision tree, to establish a performance benchmark. This helps in understanding how complex the model needs to be.
    b. **Advanced Models:** Experiment with more sophisticated models such as Random Forests, Gradient Boosting Machines (e.g., XGBoost, LightGBM), and Neural Networks. Each model should be tuned using techniques like grid search or random search over hyperparameters.
    c. **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.
5. **Model Evaluation and Tuning:**
    a. **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.
    b. **Hyperparameter Tuning:** Based on the initial evaluation, fine-tune hyperparameters to optimize model performance. This may involve adjusting learning rates, regularization parameters, tree depths, or the number of estimators, depending on the model type.
    c. **Handling Class Imbalance:** If class imbalance is a significant issue, consider techniques such as SMOTE (Synthetic Minority Over-sampling Technique), adjusting class weights, or using ensemble methods to boost the model's ability to handle minority classes effectively.
6. **Model Interpretation:**
    a. **Feature Importance:** After selecting the best model, analyze feature importance to understand which features contribute most to the predictions. This can be done using methods like SHAP values, permutation importance, or model-specific feature importance measures.

    b. **Error Analysis:** Perform an error analysis to identify common misclassifications. This can provide insights into potential improvements, such as additional feature engineering or refining the model's complexity.
7. **Final Evaluation on Test Set:**
    a. **Testing:** Once the model is finalized and optimized, evaluate it on the `test.csv` dataset. Report the final macro-F1 score, precision, and recall to assess how well the model generalizes to unseen data.
    b. **Comparison to Baseline:** Compare the performance on the test set to the baseline model and initial validation results to ensure consistency and improvement.
8. **Documentation and Reporting:**
    a. **Model Documentation:** Thoroughly document the entire process, including the rationale behind chosen methods, challenges faced, and how they were addressed. Include a summary of key findings and model performance.
    b. **Recommendations:** Provide recommendations on how the model can be integrated into SOC workflows, potential areas for future improvement, and considerations for deployment in a real-world setting.

Results:

By the end of the project, learners should aim to achieve the following outcomes:

- A machine learning model capable of accurately predicting the triage grade of cybersecurity incidents (TP, BP, FP) with high macro-F1 score, precision, and recall.
- A comprehensive analysis of model performance, including insights into which features are most influential in the prediction process.
- Documentation that details the model development process, including data preprocessing, model selection, evaluation, and potential deployment strategies.

Project Evaluation metrics:

The success and effectiveness of the project will be evaluated based on the following metrics:

- **Macro-F1 Score:** A balanced metric that accounts for the performance across all classes (TP, BP, FP), ensuring that each class is treated equally.

- **Precision:** Measures the accuracy of the positive predictions made by the model, which is crucial for minimizing false positives.
- **Recall:** Measures the model's ability to correctly identify all relevant instances (true positives), which is important for ensuring that real threats are not missed.

.

Technical Tags:

- Machine Learning
- Classification
- Cybersecurity
- Data Science
- Model Evaluation
- Feature Engineering
- SOC
- Threat Detection

Data Set Overview:

We provide three hierarchies of data: (1) evidence, (2) alert, and (3) incident. At the bottom level, evidence supports an alert. For example, an alert may be associated with multiple pieces of evidence such as an IP address, email, and user details, each containing specific supporting metadata. Above that, we have alerts that consolidate multiple pieces of evidence to signify a potential security incident. These alerts provide a broader context by aggregating related evidences to present a more comprehensive picture of the potential threat. At the highest level, incidents encompass one or more alerts, representing a cohesive narrative of a security breach or threat scenario.

The primary objective of the dataset is to accurately predict incident triage grades—true positive (TP), benign positive (BP), and false positive (FP)—based on historical customer responses. To support this, we provide a training dataset containing 45 features, labels, and unique identifiers across 1M triage-annotated incidents. We divide the dataset into a train set containing 70% of the data and a test set with 30%, stratified based on triage grade ground-truth, OrgId, and DetectorId. We ensure that incidents are stratified together within the train and test sets to ensure the relevance of evidence and alert rows.

Data Set Explanation:

The GUIDE dataset contains records of cybersecurity incidents along with their corresponding triage grades (TP, BP, FP) based on historical evidence and customer responses. Preprocessing steps may include:

- **Handling Missing Data:** Identifying and addressing any missing values in the dataset.
- **Feature Engineering:** Creating new features or modifying existing ones to improve model performance, such as combining related features or encoding categorical variables.
- **Normalization/Standardization:** Scaling numerical features to ensure that all input data is on a similar scale, which can be important for certain machine learning models.

Dataset Link : Dataset

Project Deliverables:

Learners are expected to submit the following upon project completion:

- **Source Code:** Well-documented code that includes all steps from data preprocessing to model evaluation.
- **Model File:** The trained machine learning model ready for deployment.
- **Documentation:** A comprehensive report that includes the approach, methodology, model performance analysis, and insights drawn from the project.
- **Presentation:** A brief presentation summarizing the project outcomes, challenges faced, and potential business impact.

Project Guidelines:

- **Coding Standards:** Ensure that the code is clean, well-commented, and adheres to best practices, such as PEP 8 for Python.
- **Version Control:** Use version control systems like Git to manage changes and collaborate effectively. Regular commits with meaningful messages are encouraged.
- **Modularity:** Structure the code into reusable modules or functions to enhance readability and maintainability.

- **Documentation:** Document each step clearly, including any assumptions made, to make the project easy to understand for others.
- **Model Iteration:** Regularly iterate on the model based on evaluation results and document each iteration's impact on performance metrics.

References:

- **EDA Guide - Exploratory Data Analysis (EDA) Guide**
- **Project Live Evaluation - Project Live Evaluation**
- **Capstone Explanation Guideline - Capstone Explanation Guideline**
- **Project Orientation Link (Tamil) - Tamil_recording**
- **Project Orientation Link (English) - English_recording**

Timeline:

1 week

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.
Note: Book the slot at least before 12:00 Pm on the same day

Timing: Tuesday, Thursday, Saturday (5:00PM to 7:00PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.
Note: This form will Open on Saturday and Sunday Only on Every Week

Timing: Monday-Saturday (11:30PM to 12:30PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurRA