**Attack Classification**

**Overview**

This project focuses on building a machine learning model to classify network traffic as either a cyber attack or normal activity using a well-structured approach. The dataset consists of a training set with labels and a test set without a target column. The goal was to preprocess the data, train an optimized model, and generate accurate predictions.

**Solution Approach**

1. Dataset Exploration
   * Downloaded and explored the dataset.
   * Found that the test set was missing the target column.
   * Identified three object-type columns requiring conversion.
2. Data Preprocessing
   * Checked for null values.
   * Converted categorical variables to numerical values using OrdinalEncoder to streamline encoding.
   * Split the train dataset into features (X\_train) and target (y\_train).
   * Applied Standard Scaling to normalize numerical features for both train and test data.
3. Feature Selection
   * Trained a Random Forest Classifier to evaluate feature importance.
   * Used SelectFromModel to select the most important features in both train and test datasets.
4. Model Evaluation and Hyperparameter Tuning
   * Opened a new Colab notebook to fine-tune the model.
   * Split the training data into train and validation sets using train\_test\_split.
   * Initially observed overfitting (Train Accuracy: 0.99, Test Accuracy: 0.87).
   * Tuned the ccp\_alpha, max\_depth, and min\_samples\_leaf parameters.
   * Optimized the model to achieve:
     + Train F1 Score: 0.969
     + Test F1 Score: 0.967 (indicating no overfitting and a well-balanced model).
5. Final Model Training and Prediction
   * Retrained the optimized Decision Tree model on the entire training dataset.
   * Applied the trained model to the test set (without target column).
   * Generated predictions and saved them in a CSV file for submission.

**Challenges and Solutions**

1. Missing Target Column in Test Set

Challenge: Unable to evaluate performance directly on the test set. Solution: Used train\_test\_split on the train set to create a validation set for model tuning, then applying this model with the same parameters on the original train and test sets to get the predictions.

2. Encoding Categorical Variables

Challenge: Three object-type columns needed conversion. Solution: Used OrdinalEncoder for simplicity instead of LabelEncoder.

3. Overfitting in Initial Model

Challenge: Initial Decision Tree model overfit the training data. Solution: Tuned ccp\_alpha, max\_depth, and min\_samples\_leaf to balance performance.

**Insights Gained**

1. Importance of Feature Selection
   * Removing less relevant features improved model performance.
2. Effectiveness of Hyperparameter Tuning
   * Carefully tuning parameters prevented overfitting and improved generalization.

**Conclusion**

The final model demonstrated strong generalization with high F1 scores for both training and validation sets. The optimized Decision Tree model was used to generate predictions on the test set, and the results were successfully submitted as a CSV file