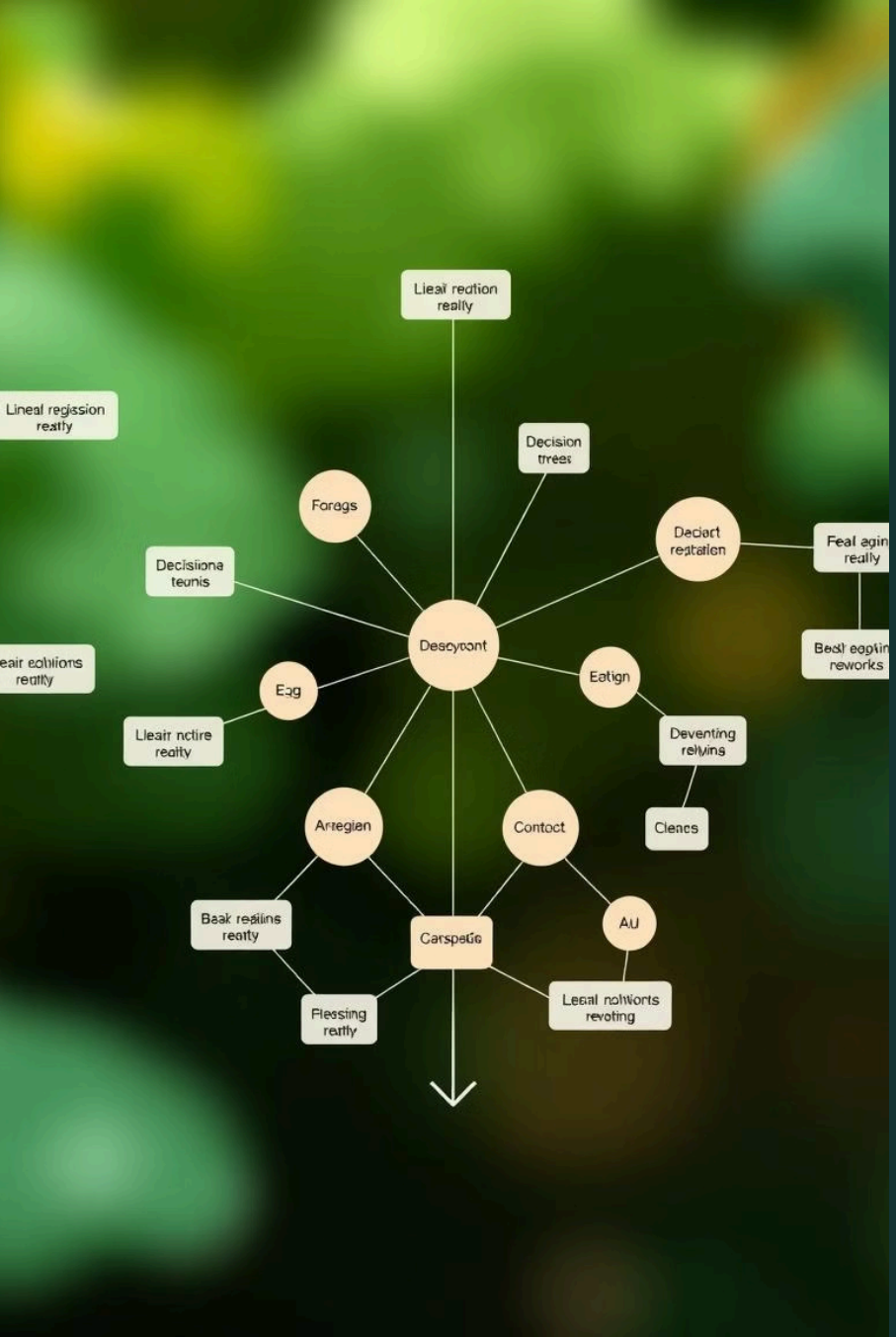
Classifying Cybersecurity Incidents with Machine Learning

by KIRUBANITHI R

Problem Statement:

SOCs face the challenge of managing an overwhelming number of alerts daily. Most of these alerts are not critical, but manually identifying which ones require attention is time consuming and prone to errors. This project seeks to solve this problem by creating a machine learning model that automatically classifies incidents as True Positive, False Positive, or Benign Positive. The model will reduce manual efforts, improve response times, and help SOC teams focus on actual threats.



Data Exploration:

· Data Loading: The dataset was loaded in chunks to handle its large size. ·

Summary Statistics: The data was analyzed to check its structure, data types, and missing values. ·

Visualizations: The distribution of key features, including the target variable (Incident Grade), was visualized to understand class imbalances.

Data Preprocessing:

Handling Missing Data: Missing values were imputed using forward fill and mean imputation.

Columns with more than 50% missing values were dropped. • Feature Engineering:

Derived timestamp-based features and removed redundant columns. •

Encoding Categorical Variables: Categorical features were converted into numerical formats using encoding technique. •

Scaling: Standardized numerical features to ensure equal contribution during model training.

Data Splitting:

Split the data into training and validation sets to evaluate model performance.

Train-Validation Split: Data was split into 80% for training and 20% for validation, while maintaining the balance between classes.

Model Selection and Training:

• Logistic Regression: A simple model used as a baseline for comparison. •

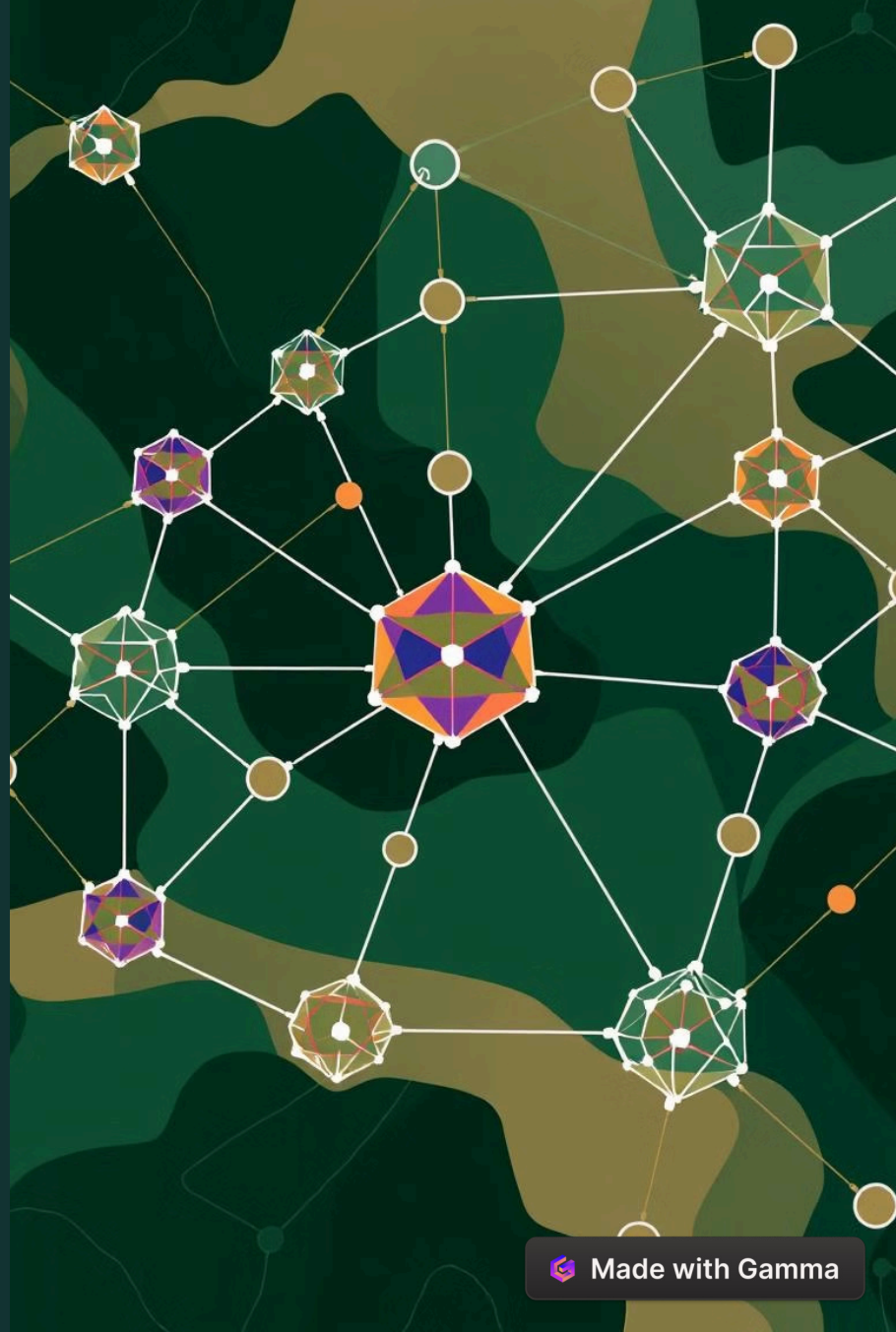
Decision Tree: A non-linear model that works well for small datasets and easy interpretability. •

Random Forest: An ensemble of decision trees that provides more accuracy and stability. •

XGBoost: A powerful algorithm that handles large datasets efficiently.

✓ Random Forest was the best-performing model, achieving high accuracy and macroF1 scores.

✓ XGBoost also performed well, though slightly lower than Random Forest.



Model Evaluation and Tuning:

Evaluate the model's performance using cross-validation and optimize it using hyperparameter tuning.

Metrics Used · Accuracy: Measures overall correctness.

· Precision: Measures how many positive predictions were correct. ·

Recall: Measures how well the model identifies actual positives. ·

Macro-F1 Score: A balanced metric that treats all classes equally.

Evaluation on Test Set

• Test the final model on unseen data to ensure it generalizes well. • The Random Forest model was evaluated on the test set, achieving high precision, recall, and macro-F1 scores.

Macro-F1 Score: 0.79

Macro Precision: 0.74

Macro Recall: 0.87