CASSANDRA PS2 Report

TEAM NAME : BIZLI

Names : Mansi , Ishanika , Sameera

APPROACH TOWARDS THE PROBLEM

Data Loading and Preprocessing

1. Dataset Loading

- The training and test datasets were loaded into structured DataFrames.
- The training data contained both features and the target label (`traffic_label`), while the test data only had features.

2. Initial Exploration

- Dataset structure, column data types, and null values were examined.
- The distribution of `traffic_label` classes was analyzed to check for class imbalance (which can heavily affect classification performance).

3. Handling Categorical Variables

- Certain features (`proto_label`, `svc_type`, `conn_state`) were identified as categorical.
- These were converted into string format to standardize the data and avoid datatype mismatches.
- Label encoding was applied to convert each unique category into a numeric value, using a consistent mapping across both train and test sets.

4. Target Label Encoding

- The output column (`traffic_label`) was encoded into numerical class labels using label encoding.
- This is necessary because most machine learning models require numeric input for classification tasks.
- A mapping of class names to numeric values was stored for interpretation of final model predictions.

5. Feature and Target Selection

- Irrelevant identifiers like `Id` were removed from the input features.
- The features and the encoded label were separated into `X` (input) and `y` (output), preparing them for model training.

6. Train-Test Splitting

- A stratified train-test split was performed to preserve the original class distribution in both training and validation datasets.
- This approach helps ensure fair evaluation of the model, especially in cases of class imbalance.

XGBoost Modeling Phase

1. Feature and Label Preparation

- Input features were extracted from the dataset by removing identifiers and the target label.
- The `traffic_label` column, which contains the classification target, was separated for supervised learning.

2. Label Encoding

- The output classes (`traffic_label`) were encoded into numerical format using Label Encoding to make them compatible with XGBoost.

- The mapping was preserved to later decode predictions back to original class names.

3. Stratified Train-Test Split

- Data was split into training and testing subsets using a **stratified approach**, maintaining class distribution across both sets.
- This ensures fair model evaluation and guards against skewed learning.

4. Handling Class Imbalance with Class Weights

- Class imbalance was handled by calculating **class weights** based on inverse class frequency.
- These weights were passed to the model to penalize the majority classes less and boost learning for underrepresented classes.

5. XGBoost Classifier Setup

- The **XGBoost algorithm** was chosen for its power in multi-class classification and ability to handle imbalanced data well.
- Key hyperparameters tuned:
  - `max_depth`, `learning_rate`, `n_estimators` to control tree growth and learning speed.
  - `subsample` and `colsample_bytree` for regularization and diversity among trees.
  - `scale_pos_weight` to address class imbalance.
  - `reg_lambda` and `reg_alpha` for L2 and L1 regularization to prevent overfitting.
- Objective function was set to **multi:softprob** for multi-class probability output.

6. Training with Early Stopping

- Training was monitored using the validation set (`eval_set`) and **early stopping** to halt training when performance stopped improving — preventing overfitting.
- Performance metric used was `mlogloss` (multi-class log loss), a common loss function for probability-based classification.

7. Prediction and Evaluation

- Model predictions were obtained as class probabilities, which were converted to final class predictions by selecting the class with highest probability.
- Evaluation metrics used:
    - **Classification Report**: Precision, recall, F1-score for each class.
    - **Weighted F1-Score**: Balanced performance measure accounting for class imbalance.
    - **Multi-class ROC AUC Score**: Evaluates how well the model distinguishes between classes in a one-vs-rest strategy.

Error Analysis & Class-wise Refinement

1. Confusion Matrix Insights

- After evaluating the multi-class XGBoost model, a **confusion matrix** was plotted to visualize misclassifications.
- A significant overlap was observed between **Class 2 (DoS attacks)** and **Class 3 (Exploits)**.
- The model was consistently misclassifying many DoS instances as Exploits, indicating **semantic or feature-level similarity** between these two classes.

2. SHAP-based Feature Importance Analysis

- **SHAP (SHapley Additive exPlanations)** values were utilized to understand **feature influence per class**.
- It was revealed that the most influential features for Classes 2 and 3 were **highly overlapping**, suggesting that the model struggled to distinguish them due to **shared feature space**.
- Additionally, the SHAP plots highlighted that the model had a **bias toward Class 3**, further compounding the misclassification issue.

3. Initial Mitigation Idea & Challenges

- The first idea was to **remove overlapping features** between Class 2 and Class 3 in hopes of improving class separability.

- However, this approach was discarded after realizing that these overlapping features were **crucial for accurately identifying Class 2**.
- Removing them would have weakened the model's ability to detect DoS attacks, which was not acceptable.

4. Refined Strategy: Targeted Feature Removal

- The ideation was revised: instead of removing **common** features, we focused on removing **features unique to Class 3**, as:
    - The model was already **biased in favor of Class 3**.
    - Removing its unique features would potentially balance the prediction tendency and reduce false positives for Class 3.

5. Binary Classification Between Conflicting Classes

- A **binary classifier** was trained using XGBoost to **specifically distinguish between Class 2 and Class 3**.
- This targeted approach allowed the model to **learn finer discriminative boundaries** between the two confusing classes.
- The result was a **significant improvement** in:
    - **Precision** and **Recall** of both Class 2 and Class 3.
    - **Overall weighted F1 score ,** reflecting better balanced classification across all classes.