**Rejection Model Documentation: -**

**Random Forest Classifier to predict the status of the rejection (Closed or Open). - sklearn**

**1. Explanation of the Prediction Model Used:**

The prediction model used in the code is a **Random Forest Classifier**, a supervised machine learning algorithm based on the ensemble learning technique. Here's an overview:

* **What is a Random Forest?**
  + It builds multiple decision trees (a "forest") during training and aggregates their predictions.
  + Each decision tree in the forest votes on the outcome, and the majority vote determines the final classification.
* **Why Random Forest?**
  + It's robust to overfitting because it averages multiple decision trees.
  + Handles mixed data types (numerical and categorical) well.
  + Works well for classification problems like predicting whether a return case is "Closed" (1) or "Open" (0).
* **How was it trained?**
  + The features used include:
    - Aging from(Months): Numerical.
    - TPN, Product Family, Location: Categorical, one-hot encoded.
  + The target variable is Final Status:
    - 1: Closed.
    - 0: Open.
  + A train-test split divided the dataset into training (70%) and testing (30%) sets.
  + The classifier was trained on the training data and evaluated on the testing data.
* **Evaluation Metrics:**
  + The performance is measured using metrics like **precision**, **recall**, **F1-score**, and **accuracy**, which are derived from the confusion matrix.

**2. Explanation of the Results Shown in the Classification Report:**

The classification report summarizes the performance of the Random Forest model:

| **Metric** | **Explanation** |
| --- | --- |
| **Precision** | Out of all predictions made for a class, how many were correct? (High precision means fewer false positives.) |
| **Recall** | Out of all actual instances of a class, how many were predicted correctly? (High recall means fewer false negatives.) |
| **F1-Score** | The harmonic mean of precision and recall, balancing both metrics. |
| **Support** | The number of actual instances for each class in the test set. |

**Breakdown of Results:**

* **Class 0 (Open cases):**
  + **Precision = 0.799**: 79.9% of the cases predicted as "Open" were correct.
  + **Recall = 0.8419**: 84.19% of all actual "Open" cases were correctly predicted.
  + **F1-Score = 0.8199**: This shows a good balance between precision and recall for this class.
  + **Support = 949**: There were 949 actual "Open" cases in the test set.
* **Class 1 (Closed cases):**
  + **Precision = 0.5482**: 54.82% of the cases predicted as "Closed" were correct.
  + **Recall = 0.4752**: 47.52% of all actual "Closed" cases were correctly predicted.
  + **F1-Score = 0.5091**: Indicates that the model struggles with "Closed" cases due to lower precision and recall.
  + **Support = 383**: There were 383 actual "Closed" cases in the test set.

**Accuracy:**

* The **accuracy = 73.65%** reflects the proportion of all correctly predicted cases (both classes combined).

**Macro Average:**

* This is the unweighted average of precision, recall, and F1-score across classes. Since class 1 has lower scores, the macro average is lower than the weighted average.

**Weighted Average:**

* The weighted average takes into account the support (number of instances) for each class. It emphasizes the performance for the majority class (0).

**Key Observations:**

1. **Better Performance for Class 0 (Open Cases):**
   * The model performs better at predicting "Open" cases, which is likely due to class imbalance (more "Open" cases than "Closed" cases).
2. **Struggles with Class 1 (Closed Cases):**
   * The lower precision and recall for "Closed" cases indicate that the model often predicts "Closed" cases incorrectly (either as false positives or false negatives).
3. **Class Imbalance Issue:**
   * With 949 "Open" cases and only 383 "Closed" cases, the model is biased toward predicting the majority class (0).

**Suggestions to Improve the Model:**

1. **Handle Class Imbalance:**
   * Use techniques like oversampling (e.g., SMOTE) or undersampling to balance the dataset.
2. **Feature Engineering:**
   * Explore additional features or interactions between features that might better distinguish "Open" and "Closed" cases.
3. **Hyperparameter Tuning:**
   * Perform grid search or random search to optimize Random Forest parameters, such as the number of trees, max depth, etc.
4. **Try Other Models:**
   * Experiment with other algorithms, such as Gradient Boosting (e.g., XGBoost, LightGBM), which might perform better on imbalanced datasets.