XGBoost Model Training Documentation

Dataset

dataset: nlp_combinedv4.csv (final cleaned dataset)

Target Variable: final_recall_level (values 1–5)

Feature Columns: name, name_manufacturer, implanted, type, action_summary, risk_class, classification, status, determined_cause

XGBoost Class Weighting approach :

The primary goal of this notebook is to predict the severity level of a medical device recall assigning it a numerical value from 1 to 3 and risk alerts as 4 and 5.

This approach was chosen to leverage both historical data and machine learning, creating a robust and practical solutions

Target Encoding

- Converted recall levels (1–5) → integers (0–4) using LabelEncoder.

Feature Transformation

- **Categorical columns:** Encoded using OneHotEncoder (with handle_unknown="ignore").
- **Numerical columns:** Passed through without modification.

Data Split

- **80/20 Train-Test split** with stratification to maintain class distribution.

Handling Class Imbalance

- Computed **class weights**:

$$wc = N/(k \cdot nc)$$

where N = total samples, k = number of classes, nc = samples in class.

Applied weights via sample_weight during model training.

Model Training

- Algorithm: XGBoost (XGBClassifier)
- Key Parameters:
  - n_estimators=300
  - learning_rate=0.1
  - max_depth=6
  - tree_method="hist"
  - eval_metric="mlogloss"

Evaluation Metrics

- **Metric Used:** Macro F1-score (to equally value all classes despite imbalance).
- **Results:**
  - Achieved Macro F1 score ≈ *0.7083416171618493*
  - Classification report generated with recall levels mapped back (1–5).

Model Persistence

- The trained model was serialized and stored as a .pkl file for later inference.

Model Persistence

Artifacts saved using joblib:

- xgb_model_hybrid.pkl → Trained XGBoost model
- preprocessor_hybrid.pkl → OneHotEncoder transformation
- label_encoder_hybrid.pkl → Target label encoder
- class_weights_hybrid.pkl → Dictionary of class weights
- full_dataset.pkl → Original dataset for historical lookup

Hybrid Prediction Strategy

Step 1: Historical Lookup (Fuzzy Matching)

- fuzzywuzzy used to match **device name** and **manufacturer** against historical dataset.
- If both matches ≥ 90% similarity:
  - Prediction taken directly from historical record.
  - Provides explanation: *"Exact/fuzzy historical match."*

Step 2: Machine Learning Prediction (XGBoost)

- If no good match:
  - New record is preprocessed using saved transformer.
  - XGBoost predicts recall level.
  - Provides explanation: *"No historical match; predicted using XGBoost model."*

Sample input

```
# Example usage
example_prediction = predict_final_recall_hybrid(
    name="Shear Valve Assembly",
    name_manufacturer="abbott diagnostics international ltd",
    implanted="Unknown",
    type_action="Recall"
)

print(example_prediction)
```

Output

```
...  Macro F1 Score: 0.7083416171618493
              precision    recall  f1-score   support

           1       0.77      0.67      0.72       811
           2       0.92      0.82      0.87      1624
           3       0.86      0.65      0.74      1658
           4       0.30      0.83      0.44       414
           5       0.87      0.70      0.77       210

    accuracy                           0.73      4717
   macro avg       0.75      0.73      0.71      4717
weighted avg       0.82      0.73      0.76      4717

{'Predicted_Final_Recall_Level': 4, 'Reason': 'Exact/fuzzy historical match. Type: Recall / Field Safety Notice
```