# Explanation of ANC Dropout Prediction Model Code and Results

This document provides a detailed explanation of the Python code used to build a machine learning model for predicting Antenatal Care (ANC) dropout using a LightGBM classifier, along with an analysis of the results. The code processes a large dataset, handles class imbalance, performs cross-validation, and evaluates the model using various metrics and SHAP analysis for feature importance.

## Code Overview

### 1. Imports and Setup

The code begins by importing necessary libraries for data processing, machine learning, and evaluation:

- **Data Handling**: `pandas`, `numpy`, `pyarrow.parquet` for data loading and manipulation.
- **Machine Learning**: `lightgbm` for the classifier, `sklearn` for metrics and cross-validation, `imblearn` for handling class imbalance.
- **Visualization and Interpretability**: `shap` and `matplotlib` for feature importance analysis.
- **Warnings**: Suppresses `UserWarning` to reduce console clutter.

A check ensures that `SMOTEENN` (Synthetic Minority Oversampling Technique combined with Edited Nearest Neighbors) is available, as it is critical for addressing class imbalance.

### 2. Data Loading and Preprocessing (`prepare_data_for_targets`)

The `prepare_data_for_targets` function loads data from a Parquet file in batches to handle large datasets efficiently. Key steps include:

- **Column Definitions**:
    - **Required Columns**: Ensures essential columns like `MOTHER_ID` and `GRAVIDA` are present.
    - **Numeric Columns**: Includes features like `GRAVIDA`, `PARITY`, `HEIGHT`, etc., converted to numeric types with NaNs handled by median imputation.

    - **Flag Columns**: Binary flags (e.g., `age_adolescent`, `anemia_mild`) are mapped from string values (`Y`, `N`, etc.) to 1/0 using a provided `flag_map`, with NaNs filled as 0.
    - **Categorical Columns**: Features like `FACILITY_TYPE`, `BLOOD_GRP`, and `SYS_DISEASE` are imputed with the mode, and `SYS_DISEASE` is limited to the top 10 categories to reduce dimensionality.
- **Feature Engineering**:
    - Creates a `gravida_parity_ratio` feature to capture the ratio of pregnancies to deliveries, avoiding division by zero with a small constant.
    - NaNs in new features are filled with the median.
- **Batch Processing**:
    - The Parquet file is read in chunks (default size: 10,000 rows) to manage memory.
    - Each batch is processed for type conversion, NaN imputation, and feature engineering, then concatenated into a single DataFrame.
- **NaN Checks**: After preprocessing, the code checks for remaining NaNs in critical columns and prints warnings if any are found.

## 3. Stratified Sampling (`create_stratified_sample`)

To address the computational burden of a large dataset and ensure balanced representation:

- The function creates a stratified sample of 10,000 rows, prioritizing all negative cases (`anc_dropout=0`) due to their rarity (130 negative vs. 4,029,441 positive cases).
- It samples positive cases to achieve a target ratio (up to 2:1 positive-to-negative) while ensuring at least 1,000 positive cases.
- The resulting sample has 9,870 positive and 130 negative cases, maintaining class representation for model training.

## 4. Feature Selection and Encoding

- **Features**:
    - **Numeric Features**: 18 features like `GRAVIDA`, `BMI`, `TOTAL_ANC_VISITS`, etc.
    - **Flag Features**: 16 binary indicators like `inadequate_anc`, `hypertension`, etc.
    - **Categorical Features**: `FACILITY_TYPE`, `BLOOD_GRP`, `SYS_DISEASE`.
- **Exclusion**: Columns related to outcomes or identifiers (e.g., `MOTHER_ID`, `MATERNAL_OUTCOME`) are excluded to prevent data leakage.

- **Encoding**: Categorical features are encoded using `TargetEncoder`, which replaces categories with the mean target value, preserving predictive power while handling high cardinality.

## 5. Model Training and Cross-Validation

- **Train-Test Split**: The sampled data is split into 80% training and 20% test sets with stratification to maintain class distribution.
- **Cross-Validation**: 5-fold stratified k-fold cross-validation ensures robust evaluation.
- **Class Imbalance Handling**:
  - A `Pipeline` with `SimpleImputer` (median strategy) and `SMOTEENN` (sampling_strategy=0.5) balances classes by oversampling the minority class and undersampling the majority class.
  - Post-SMOTEENN, each fold's training set has approximately 6,317 positive and 3,158 negative cases.
- **LightGBM Classifier**:
  - Parameters: `n_estimators=500`, `max_depth=7`, `learning_rate=0.05`, `scale_pos_weight=0.02` (adjusted for class imbalance), etc.
  - Early stopping (50 rounds) prevents overfitting based on validation AUC.
- **Metrics**:
  - Evaluates AUC, PR-AUC, F1 score, accuracy, precision, and recall at thresholds (0.1, 0.2, 0.3, 0.4).
  - Stores confusion matrices to analyze true positives, false positives, etc.
- **Training Time**: Tracks time per fold for performance monitoring.

## 6. Evaluation on Test Set

- The best model (highest average F1 score across thresholds) is selected from cross-validation.
- Evaluates the test set using the same metrics and thresholds, reporting AUC, PR-AUC, and confusion matrices.

## 7. SHAP Analysis

- Uses SHAP (SHapley Additive exPlanations) to interpret feature importance.
- Samples 1,000 test set rows for computational efficiency.
- Generates two plots:
  - **Summary Plot**: Shows the impact of each feature on predictions.
  - **Bar Plot**: Displays mean absolute SHAP values for feature importance.
- Outputs a DataFrame ranking features by SHAP importance.

## 8. Alternative Model (Commented)

- Suggests using `HistGradientBoostingClassifier` as an alternative, which natively handles NaNs, but it is not used in the main execution.

# Results Analysis

## Data Characteristics

- **Dataset Size**: 4,029,571 rows (4,029,441 positive, 130 negative for `anc_dropout`).
- **Sampled Data**: 10,000 rows (9,870 positive, 130 negative).
- **Class Imbalance**: Severe imbalance (99.997% positive cases), addressed via stratified sampling and SMOTEENN.

## Cross-Validation Results

- **AUC**: 1.0000 ± 0.0000 across all folds, indicating perfect discrimination (likely due to model overfitting or data issues).
- **Metrics Across Thresholds (0.1, 0.2, 0.3, 0.4)**:
  - **F1 Score**: ~0.9935 ± 0.0001
  - **Accuracy**: ~0.9870 ± 0.0003
  - **Precision**: ~0.9870 ± 0.0003
  - **Recall**: 1.0000 ± 0.0000
  - **PR-AUC**: 1.0000 ± 0.0000
- **Confusion Matrices**:
  - True Negatives (TN): 0 (all negative cases misclassified).
  - False Positives (FP): 20–21 per fold.
  - False Negatives (FN): 0 (all positive cases correctly classified).
  - True Positives (TP): ~1,579–1,580 per fold.
- **Training Time**: ~0.06–0.10 seconds per fold, indicating efficient training.

## Test Set Results (Best Model: Fold 1)

- **AUC**: 1.0000
- **PR-AUC**: 1.0000
- **Metrics at Thresholds (0.1, 0.2, 0.3, 0.4)**:
  - **F1 Score**: 0.9935
  - **Accuracy**: 0.9870
  - **Precision**: 0.9870
  - **Recall**: 1.0000
  - **Confusion Matrix**:
    - TN: 0
    - FP: 26
    - FN: 0

        - TP: 1,974
- **Best Threshold**: 0.1 (F1 Score: 0.9935)

### SHAP Feature Importance

- **Top Feature**: `inadequate_anc` (SHAP Importance: 0.075975), indicating strong influence on predicting ANC dropout.
- **Other Notable Features**: `systolic_bp`, `BMI`, `FACILITY_TYPE` (very low importance, ~1e-17 to 2e-18).
- **Most Features**: Zero importance, suggesting minimal contribution to predictions.
- **Plots**: Saved as `shap_summary_plot_anc_dropout.png` and `shap_importance_bar_anc_dropout.png`.

### Issues and Observations

- **Overfitting/Perfect Scores**: AUC and PR-AUC of 1.0000 suggest the model may be overfitting, possibly due to:
    - **Data Leakage**: Despite excluding outcome-related columns, some features may indirectly correlate with `anc_dropout`.
    - **Class Imbalance**: Even after SMOTEENN, the model struggles to classify negative cases (TN=0).
    - **Feature Distribution**: The dataset may lack sufficient variability in negative cases, making them indistinguishable.
- **Confusion Matrix**: The model fails to predict negative cases (all classified as positive), indicating poor generalization for the minority class.
- **SHAP Analysis**: Only `inadequate_anc` has significant importance, suggesting the model relies heavily on this feature, which may be a proxy for dropout behavior.

## Recommendations

1. **Address Class Imbalance**:
    - Explore alternative sampling techniques (e.g., ADASYN, Tomek Links) or adjust SMOTEENN parameters.
    - Collect more negative cases if possible.
2. **Feature Engineering**:
    - Investigate additional features or interactions to better distinguish negative cases.
    - Remove or re-evaluate low-importance features to reduce noise.
3. **Model Tuning**:
    - Adjust `scale_pos_weight` or use cost-sensitive learning to penalize misclassifying negative cases.
    - Try simpler models (e.g., Logistic Regression) or ensemble methods to reduce overfitting.

4. **Data Validation**:
    - Verify `anc_dropout` labeling accuracy, as the imbalance and perfect scores suggest potential data issues.
    - Check for leakage in features like `inadequate_anc`.
5. **Alternative Models**:
    - Test the commented `HistGradientBoostingClassifier` for robustness to NaNs and potentially better handling of imbalance.

# Conclusion

The code implements a robust pipeline for predicting ANC dropout, with careful preprocessing, imbalance handling, and model evaluation. However, the perfect AUC and failure to predict negative cases indicate overfitting or data limitations. Further investigation into data quality, feature selection, and model tuning is necessary to improve generalization, particularly for the minority class.