Explanation of Code and Results for Premature Birth Risk Prediction

This document explains the provided Python code, which performs data preprocessing and trains a Random Forest Classifier to predict premature birth risk using a dataset stored in a Parquet file. It also interprets the cross-validation results to assess model performance.

Code Overview

The code consists of two main components:

1. **Data Preprocessing** (`prepare_data_for_targets` function):
    - Loads and preprocesses a dataset from a Parquet file for machine learning.
    - Handles missing values, converts data types, and maps categorical flags to numeric values.
    - Ensures data is ready for training by selecting numeric features and excluding columns that may introduce data leakage.
2. **Model Training and Evaluation**:
    - Uses a Random Forest Classifier with 5-fold stratified cross-validation to predict the target variable `premature_birth_risk`.
    - Evaluates the model across multiple classification thresholds (0.1 to 0.7) using metrics like F1 score, accuracy, precision, recall, and AUC.
    - Outputs performance metrics and confusion matrices for each fold and threshold.

Detailed Code Explanation

1. Data Preprocessing (`prepare_data_for_targets`)

This section loads and cleans the dataset from a Parquet file.

Key Steps:

- **Loading Data**:
    - The dataset is read from `/kaggle/input/fullfinal/telangana_data_with_features_and_targets (1).parquet` using `pyarrow.parquet`.
    - Data is processed in batches (default size: 10,000 rows) to handle large datasets efficiently.
    - The Parquet file is assumed to contain health-related data, including features like `AGE`, `GRAVIDA`, `HEMOGLOBIN_mean`, and target variables like `premature_birth_risk`.
- **Required and Numeric Columns**:
    - **Required Columns**: Ensures critical columns (`MOTHER_ID`, `GRAVIDA`) are present.
    - **Numeric Columns**: Defines a list of expected numeric columns (e.g., `AGE`, `HEMOGLOBIN_mean`, `WEIGHT_max`) for conversion to appropriate data types.
- **Data Cleaning**:
    - **GRAVIDA Cleaning**: Converts `GRAVIDA` (number of pregnancies) to numeric, replacing non-numeric values like `'nan'` with `NaN`. Missing values are filled with 0.
    - **Numeric Conversion**: Converts other numeric columns to `float64` or similar, handling non-numeric values by setting them to `NaN`.
    - **Flag Mapping**: Maps categorical flag columns (`IS_CHILD_DEATH`, `IS_DEFECTIVE_BIRTH`) to binary values (e.g., `Y`/`YES` → 1, `N`/`NO` → 0, `None`/`nan` → `NaN`) using a predefined `flag_map`.
    - **Missing Value Imputation**: Fills `NaN` values in numeric columns with 0 to ensure compatibility with machine learning models.
    - **Non-Numeric Debugging**: Checks for non-numeric values in numeric columns and prints them for debugging.
- **Output**:
    - Returns a concatenated `pandas.DataFrame` with cleaned data, ready for feature selection and model training.

Key Output from Preprocessing:

- The code outputs messages indicating that `GRAVIDA` had non-numeric values (`'nan'`) which were replaced, and no missing `GRAVIDA` values remained after filling with 0.
- The resulting DataFrame (`df`) contains cleaned numeric and binary columns.

2. Feature Selection

The code defines a list of columns (`target_columns`) to exclude from features to prevent **data leakage**. These include:

- Post-delivery outcomes (e.g., `DEL_COMPLICATIONS`, `CHILD_NAME`).
- Administrative or logging data (e.g., `REGISTRATION_DT`, `CURRENT_USR`).
- Screening/test results (e.g., `VDRL_RESULT`, `HIV_STATUS`).
- Derived risk or score columns (e.g., `HIGH_RISKS`, `hemoglobin_trend`).
- Features like `AGE_preg`, `TOTAL_ANC_VISITS`, which may be redundant or post-hoc.

Feature Selection Logic:

- Features are selected as columns with numeric data types (`float64`, `float32`, `int64`, `int32`, `int8`) that are not in the `target_columns` list.
- If no valid features are found, a message is printed: `Skipping {target_columns}: No valid features available.`

Final Features Used:

- The features used for training include:
  - `GRAVIDA`, `AGE`, `PARITY`, `ABORTIONS`, `HEIGHT`
  - `HEMOGLOBIN_mean`, `HEMOGLOBIN_min`, `HEMOGLOBIN_max`
  - `WEIGHT_anc_mean`, `WEIGHT_anc_min`, `WEIGHT_anc_max`
  - Binary flags: `age_adolescent`, `age_elderly`, `age_very_young`, `multigravida`, `grand_multipara`, `previous_loss`, `recurrent_loss`
  - Derived metrics: `gravida_parity_ratio`, `inadequate_anc`, `no_anc`, `irregular_anc`, `missed_first_anc`, `consecutive_missed`
  - Health indicators: `anemia_mild`, `anemia_moderate`, `anemia_severe`, `ever_severe_anemia`, `systolic_bp`, `diastolic_bp`, `hypertension`, `severe_hypertension`
  - BMI-related: `BMI`, `underweight`, `obese`, `normal_weight`
  - Mental health: `depression`, `severe_depression`, `anxiety`, `severe_anxiety`
  - Weight metrics: `weight_gain`, `weight_gain_per_week`, `inadequate_weight_gain`

3. Model Training and Evaluation

The code trains a Random Forest Classifier using 5-fold stratified cross-validation and evaluates performance across multiple thresholds.

Setup:

- **Data Split**:
    - The dataset is split into training (80%) and test (20%) sets using `train_test_split` with stratification to maintain class balance.
    - The training set is further divided into 5 folds using `StratifiedKFold` to ensure consistent class distribution.
- **Random Forest Parameters**:
    - `n_estimators`: 100 (number of trees).
    - `max_depth`: 10 (limits overfitting).
    - `min_samples_split`: 50 (minimum samples to split a node).
    - `min_samples_leaf`: 25 (minimum samples per leaf).
    - `class_weight`: `'balanced'` (adjusts for class imbalance).
    - `random_state`: 42 (ensures reproducibility).
    - `n_jobs`: -1 (uses all CPU cores).
- **Metrics Tracked**:
    - For thresholds 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7:
        - F1 score, accuracy, precision, recall.
        - Confusion matrix (TN, FP, FN, TP).
    - AUC (threshold-independent).
    - Average F1 score across thresholds per fold.
    - Training time per fold.

Training Loop:

- For each fold (1 to 5):
    - Splits training data into training and validation sets.
    - Prints class distribution for training and validation sets (e.g., ~760,444 negative and ~519,556 positive samples in training; ~190,111 negative and ~129,889 positive in validation).
    - Trains a Random Forest model and measures training time.
    - Predicts probabilities for the positive class (`y_pred_proba`).
    - Applies thresholds (0.1 to 0.7) to convert probabilities to binary predictions.
    - Computes metrics (F1, accuracy, precision, recall, AUC) and confusion matrices.
    - Stores metrics and the trained model.

Output:

- For each fold, prints:
    - AUC and training time.
    - Metrics and confusion matrices for each threshold.
- After all folds, prints:
    - Features used for training.
    - Mean and standard deviation of metrics across folds.

Results Interpretation

The results show the performance of the Random Forest Classifier across 5 folds for predicting `premature_birth_risk`.

Class Distribution:

- **Training Set**: ~760,444 negative (0) and ~519,556 positive (1) samples.
- **Validation Set**: ~190,111 negative (0) and ~129,889 positive (1) samples.
- The dataset is imbalanced (~60% negative, 40% positive), but the `class_weight='balanced'` parameter helps address this.

Cross-Validation Mean Metrics:

- **AUC**: 0.9592 ± 0.0002
    - Indicates excellent discriminative ability (close to 1.0).
    - Low standard deviation suggests consistent performance across folds.

Threshold-Specific Metrics:

The model was evaluated at thresholds from 0.1 to 0.7. Below are the mean metrics across folds:

- **Threshold 0.1**:
    - F1: 0.8203 ± 0.0098
    - Accuracy: 0.8221 ± 0.0120
    - Precision: 0.6955 ± 0.0140
    - Recall: 1.0000 ± 0.0000
    - **Observation**: High recall (1.0) indicates all positive cases are correctly identified, but low precision (~0.70) suggests many false positives. This threshold is too lenient, leading to over-prediction of positive cases.

- **Threshold 0.2**:
    - F1: 0.9146 ± 0.0081
    - Accuracy: 0.9241 ± 0.0079
    - Precision: 0.8427 ± 0.0137
    - Recall: 1.0000 ± 0.0000
    - **Observation**: Improved F1 and precision compared to 0.1, with near-perfect recall. Still some false positives, but better balance.
- **Threshold 0.3**:
    - F1: 0.9228 ± 0.0003
    - Accuracy: 0.9321 ± 0.0003
    - Precision: 0.8568 ± 0.0005
    - Recall: 1.0000 ± 0.0000
    - **Observation**: Further improvement in F1 and precision, with recall still at 1.0. Very stable performance (low standard deviation).
- **Threshold 0.4**:
    - F1: 0.9232 ± 0.0003
    - Accuracy: 0.9325 ± 0.0003
    - Precision: 0.8574 ± 0.0005
    - Recall: 0.9999 ± 0.0000
    - **Observation**: Near-peak F1 score, with slight drop in recall but improved precision. False positives are further reduced.
- **Threshold 0.5** (default):
    - F1: 0.9235 ± 0.0003
    - Accuracy: 0.9328 ± 0.0003
    - Precision: 0.8581 ± 0.0006
    - Recall: 0.9997 ± 0.0001
    - **Observation**: Best balance of F1, accuracy, and precision. Recall remains near-perfect, with minimal false negatives.
- **Threshold 0.6**:
    - F1: 0.9239 ± 0.0003
    - Accuracy: 0.9333 ± 0.0003
    - Precision: 0.8608 ± 0.0007
    - Recall: 0.9969 ± 0.0005
    - **Observation**: Highest F1 and accuracy, with slightly reduced recall due to stricter threshold. Fewer false positives but slightly more false negatives.
- **Threshold 0.7**:
    - F1: 0.9210 ± 0.0006
    - Accuracy: 0.9314 ± 0.0004

    - Precision: 0.8645 ± 0.0009
    - Recall: 0.9855 ± 0.0023
    - **Observation**: Highest precision but significantly lower recall, indicating missed positive cases (false negatives increase). Too strict for this use case.

Confusion Matrices (Example from Fold 1):

- **Threshold 0.1**:
    - TN: 125,656, FP: 64,455, FN: 0, TP: 129,889
    - High FP indicates many negative cases misclassified as positive.
- **Threshold 0.5**:
    - TN: 168,803, FP: 21,308, FN: 25, TP: 129,864
    - Balanced performance with low FN and reduced FP.
- **Threshold 0.7**:
    - TN: 170,372, FP: 19,739, FN: 2,413, TP: 127,476
    - Lowest FP but increased FN, missing some positive cases.

Training Time:

- Average training time per fold: ~71 seconds (ranging from 65.63 to 81.74 seconds).
- Reasonable for a Random Forest with 100 trees and a large dataset.

Key Observations:

- **Optimal Threshold**: Threshold 0.5 or 0.6 provides the best balance of F1 score (0.9235–0.9239), accuracy (0.9328–0.9333), and recall (~0.9969–0.9997). These thresholds minimize false negatives (critical for identifying premature birth risk) while maintaining high precision.
- **Model Stability**: Low standard deviations across folds (e.g., AUC ±0.0002, F1 ±0.0003 at threshold 0.5) indicate robust and consistent performance.
- **Class Imbalance Handling**: The `class_weight='balanced'` parameter effectively handles the imbalance, as evidenced by high recall across thresholds.
- **Feature Importance**: The model leverages a mix of demographic (e.g., `AGE`, `GRAVIDA`), clinical (e.g., `HEMOGLOBIN_mean`, `BMI`), and derived features (e.g., `anemia_severe`, `hypertension`), suggesting a comprehensive risk assessment.