Rainfall Prediction Model Documentation

1. Overview and Objectives

Accurate rainfall prediction is crucial for agriculture and disaster preparedness. In this project, we explore how indigenous knowledge — local farmers' forecasts based on natural indicators — can be modeled to predict rainfall events.

Thus, the **Rainfall Prediction Model** is designed to forecast rainfall intensity categorized into four levels: **No Rain**, **Small Rain**, **Medium Rain**, and **Heavy Rain**.

This solution was developed as part of **Zindi's Ghana Indigenous Challenge**, leveraging **Indigenous Ecological Indicators (IEIs)**.

Objectives

The main objectives of this project are:

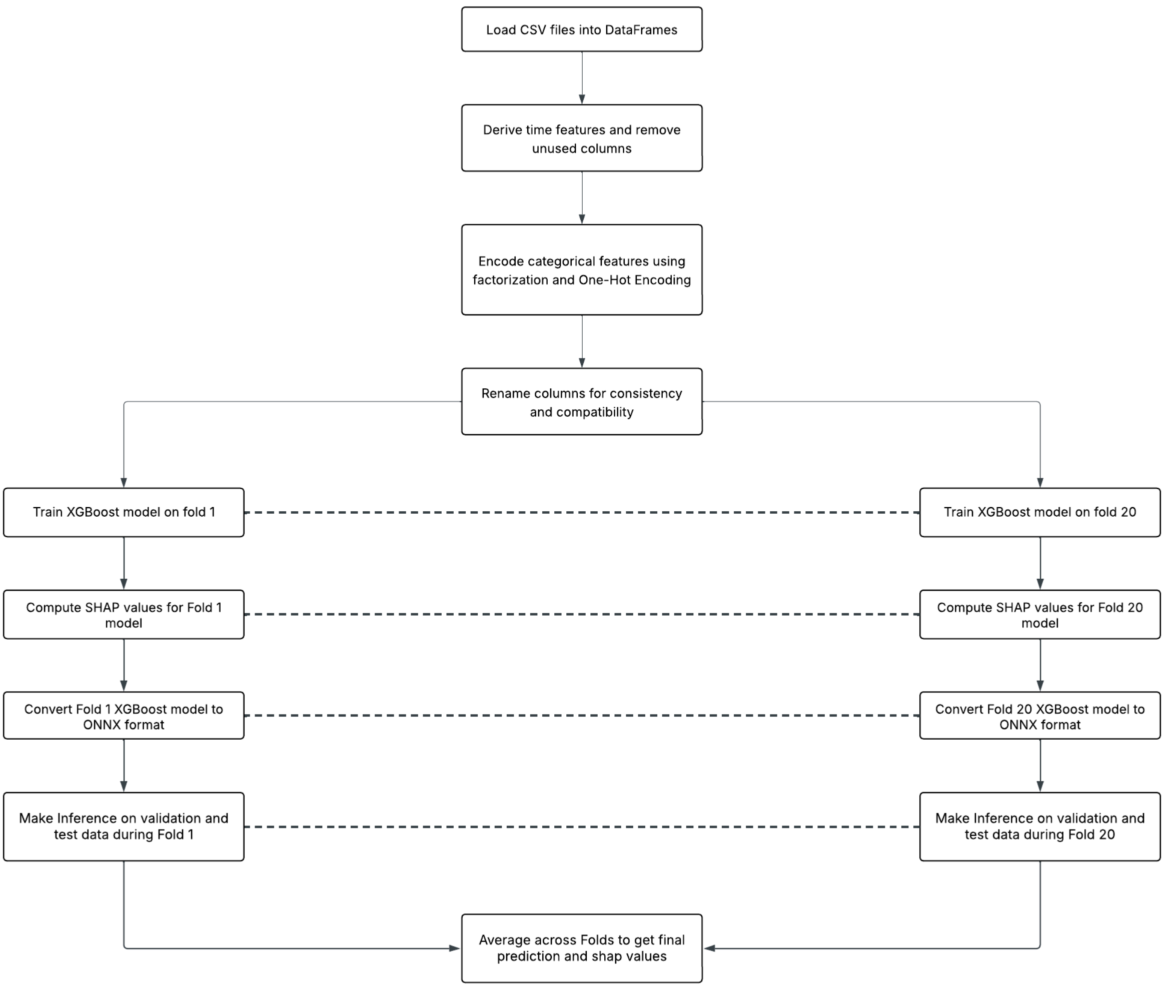
- To accurately predict rainfall categories using engineered features derived from raw IEIs.
- To address class imbalance using **stratified cross-validation** and robust metrics such as the **F1-macro score**.
- To train **XGBoost models** using **20 stratified folds** for enhanced stability and reduced variance.

2. Architecture Overview

The model follows a **modular pipeline** encompassing data extraction, transformation, modeling, evaluation, and inference.

Architecture Flow

1. Raw IEIs are extracted from the Zindi dataset.
2. Data is cleaned and transformed through **feature engineering**.
3. Categorical features are encoded using **One-Hot Encoding**.
4. The transformed data is used to train **XGBoost models** using **20 stratified folds**.
5. Cross-validation ensures stable model performance.
6. **SHAP values** are computed for interpretability.
7. The ensemble of best models is used for final inference and submission.



---

3. ETL Process

3.1 Extract

Data Source: Zindi competition dataset

Files Included:

- `train.csv` – Historical data with rainfall labels
- `test.csv` – Unlabeled data for prediction
- `sample_submission.csv` – Submission format template

Data Format: CSV (comma-separated)
Extraction Method: Data loaded using `pandas.read_csv()`

Considerations:
The dataset size allows for in-memory processing. Missing values and categorical inconsistencies are handled during transformation.

3.2 Transform

Feature Engineering

- Derived temporal features: `month`, `day`, `hour`, `day_of_week`, and `pred_date (YYYY-MM-DD)` from the `prediction_time` column.

Handling Missing Values

- The `indicator` column contained missing values filled with the placeholder `"missing"`.
- No statistical imputation was used as no meaningful patterns were observed.

Categorical Encoding

- `community` and `indicator` were initially factorized.
- **One-Hot Encoding** was applied to `community`, `district`, `indicator`, and `pred_date` for interpretability and to prevent target leakage.

Class Imbalance Handling

- **StratifiedKFold (20 splits)** maintained consistent class proportions across folds.

Scaling

- No scaling was applied since **XGBoost** is invariant to monotonic transformations.

Columns Dropped

`["prediction_time", "indicator_description", "time_observed", "ID"]`

Column Renaming

- Columns were renamed to the `f%d` format for compatibility with **ONNX** exports.

3.3 Load

- Transformed data was loaded into **pandas DataFrames** for model training and inference.

---

4. Data Modeling

4.1 Model Overview

The model utilizes **XGBoost (Extreme Gradient Boosting)** for **multi-class classification** with four output categories.
The chosen objective function is `multi:softprob`, which outputs probability distributions across all rainfall classes.

4.2 Feature Summary

Feature Type	Example Features	Encoding Method
Temporal	Month, Day, Hour, Day of Week	Derived numerically
Categorical	Community, District, Indicator, Prediction Date	One-Hot Encoding
Numerical	User ID, Predicted Intensity, Confidence, Forecast Length	Numeric

4.3 Model Training

Cross-Validation

- **20 stratified folds** ensure robust evaluation and preserve class balance across splits.

Evaluation Metric

- **mlogloss** was chosen to measure how well the model predicts class probabilities, not just class labels.

Training Parameters

```
params = {
    "random_state": 44,
    "max_depth": 10,
    "colsample_bytree": 0.9,
    "subsample": 0.9,
    "n_estimators": 2000,
```

```
    "learning_rate": 0.01,
    "num_class": 4,
    "early_stopping_rounds": 25,
    "objective": "multi:softprob",
    "eval_metric": "mlogloss"
}
```

Training Process

1. Dataset split into 20 folds using **StratifiedKFold**.
2. One model trained per fold.
3. **Out-of-Fold (OOF)** predictions are used for validation.
4. Final predictions averaged across all folds.

4.4 Model Interpretation

Model interpretability was achieved using **SHAP (Shapley Additive Explanations)**. Analysis revealed that **community (asamama)**, **user_id**, and **day** were the top three features that contributed most significantly to predictions.

---

5. Inference

5.1 Inference Workflow

1. Apply the same preprocessing transformations (including One-Hot Encoding) to new data.
2. Generate predictions using all 20 trained models.
3. Average probabilities across folds.
4. Assign rainfall categories based on the highest mean probability.

5.2 Infrastructure

- Inference runs locally on **CPU** with lightweight resource requirements.

5.3 Retraining and Versioning

- Retraining follows the same preprocessing and encoding pipeline.
- All models, encoders, and artifacts are versioned using timestamped directories for reproducibility.

6. Run Time Summary

Component	Description	Approx. Run Time
Data Preprocessing	Feature engineering and encoding	255 ms
Model Training (20 folds)	XGBoost training with early stopping	299.88 s
Inference	Prediction on test and validation data	7.35 s
SHAP Analysis	Model interpretation	522.88 s
ONNX Conversion	Converting model to ONNX format	215.47 s

Note: Run times may vary depending on system configuration. GPU acceleration can further reduce training time.

7. Performance Metrics

Metric	Description	Score
Public Leaderboard	F1-Macro Score (Zindi)	**0.9653**
Private Leaderboard	F1-Macro Score (Zindi)	**0.9716**
Cross-Validation (OOF)	Mean F1-Macro (20 folds)	**0.9872**

Additional evaluation tools included **class-wise F1 scores** and **SHAP importance plots** for deeper model insights.

8. Error Handling

- Ensured the presence of required columns before processing.

---

9. Maintenance and Monitoring

9.1 Monitoring

- Performance metrics are in the script.

9.2 Maintenance

- Dependencies listed in `requirements.txt`.
- Scripts are modularized and version-controlled for maintainability.

---

10. Author Information

Author: Benson Wainaina
Competition: Zindi – *Ghana’s Indigenous Intel Challenge [BEGINNERS ONLY]*
Date: October 2025
Contact: benson.kamau1738@gmail.com