**Crop Yield Prediction**

This project contains a series of Python notebooks designed to facilitate data preparation, model fitting and evaluation for prediction crop yield of Maize and Soy based on Producer Data dataset. The goal is to enable effective data processing, model training, and prediction of crop yield using machine learning techniques.

**1\_Data\_Preparation.ipynb**

The script is designed for preparing crop yield data before feeding it into machine learning models with a focus on cleaning, feature engineering, and balancing class distribution. Main Tasks:

1. **Loading and Cleaning Data**: Importing data, filtering it based on NDVI and coverage thresholds to ensure high-quality samples.
2. **Feature Engineering**: Calculating vegetation index derivatives (e.g., NDVI, EVI) for time-based analysis.
3. **Class Rebalancing**: Limiting the number of samples for majority crop classes to avoid overrepresentation, ensuring diversity within classes like Maize and Soy.

**2\_Fit\_Classifier.ipynb**

The script outlines the process of predicting crop yield using an XGBoost model with feature selection techniques, such as Sequential Forward Selection (SFS), to optimize model performance. Main Tasks:

1. **Loading the Dataset**: Import and shuffle the dataset for model training.
2. **Feature Engineering**: Add cyclical features (week\_sin, week\_cos) and calculate vegetation growth rates (e.g., NDVI, EVI).
3. **Defining Predictors and Labels**: Dynamically select predictor columns and set the target variable.
4. **Training the XGBoost Model**: Fit the model with parallel processing for efficiency.
5. **Feature Selection**: Compute feature importance and use SFS to select the top features for yield prediction.
6. **Model Evaluation**: Split the data into training and testing sets, then evaluate using R² and RMSE.
7. **Saving the Model**: Save the trained model and selected features for future use.

**3\_Simulation.ipynb**

This script loads a pre-trained XGBoost model and selected features to perform crop yield prediction, and visualize results using time series plots for Maize and Soy yields. Main Tasks:

1. **Load Pre-Trained Model and Features**: Retrieve the saved XGBoost model and selected features from previous training.
2. **Feature Engineering**: Add cyclical week features and calculate vegetation growth rates.
3. **Model Evaluation**: Fit the model to the full dataset and calculate performance metrics (RMSE, R², MAE).
4. **Plotting Results**: Visualize feature importance and generate time series scatter plots for Maize and Soy.

**4\_ Simulation.py**

This script loads a pre-trained XGBoost model and selected features to perform crop yield prediction, and visualize results using time series plots for Maize and Soy yields. The main tasks are:

1. **Loading Pre-Trained Model and Selected Features**: The script loads a previously trained XGBoost model and the corresponding selected features from external files to avoid re-training.
2. **Feature Engineering**: New features are created, such as sine/cosine transformations of the 'week' column to capture seasonality, and growth rates for vegetation indices (NDVI, EVI, LAI) are added.
3. **Using Pre-Trained Model for Prediction**: The pre-trained XGBoost model is used to predict crop yield directly on the cleaned dataset.
4. **Evaluate Performance by Week**: The script calculates performance metrics (RMSE, R², MAE) for each unique week in the dataset and stores the results in a DataFrame.
5. **Save Results DataFrame**: A DataFrame is created containing observed and predicted crop types, along with additional fields (crop type, field ID, year, and week). This is saved to a CSV file.

**4\_ SimulationBoots.py**

The script is designed to evaluate a pre-trained XGBoost model to perform crop yield prediction, with a focus on bootstrapped accuracy per crop per week and smoothed visualization of the results. The main tasks are:

1. **Loading Pre-Trained Model and Selected Features**: The script loads a previously trained XGBoost model and corresponding selected features from external files to perform predictions on the dataset.
2. **Feature Engineering**: The script generates new features such as sine/cosine transformations of the 'week' column and calculates growth rates for vegetation indices (NDVI, EVI, LAI), along with a combined vegetation index.
3. **Using Pre-Trained Model for Prediction**: The pre-trained XGBoost model is used to predict crop yield directly on the cleaned dataset.
4. **Bootstrapping Accuracy Calculation per Crop per Week**: The script calculates bootstrapped accuracy yield on a weekly basis, providing both the mean accuracy and standard deviation. This method ensures more robust accuracy estimates by resampling the data multiple times.
5. **Smoothing and Plotting Accuracy Results**: A Gaussian smoothing function is applied to the weekly bootstrapped accuracy results, and the smoothed accuracy per crop is plotted with a shadowed border to represent the standard deviation.
6. **Saving Results and Plot**: The bootstrapped accuracy results are saved as a CSV file, and the accuracy plot is saved as a PNG image in the output folder.

**General Instructions:**

1. **Environment Setup**:
   * Ensure that you have the necessary Python packages installed, including pandas, scikit-learn, XGBoost, and matplotlib.
   * The notebooks are intended to be run in a Jupyter environment.
2. **Workflow**:
   * Start with **1\_Data\_Preparation.ipynb** to prepare the data.
   * Use **2\_Fit\_Classifier.ipynb** to train a classification model for predicting crop types.
   * Finally, use **3\_Simulation.ipynb** to use the generated model for the yield predictions and evaluation.
   * Additionally, use **4\_Simulation.py** to use the generated model for the yield predictions and evaluation within a simple python code.
   * Use **4\_SimulationBoots.py** to add thebootstrapped accuracy for the yield predictions on a weekly basis.
3. **Outputs**:
   * The notebooks generate various outputs such as CSV files, scatter plots, and saved models for further use. Each notebook will save its output in the specified directory.