**Crop Type Prediction**

This project contains a series of Python notebooks and scripts designed to facilitate data preparation, model fitting, and evaluation for predicting crop types such as **Maize, Soy, Sunflower, Wheat, Lucern, Pasture, Tree, Fallow, Groundnuts**, and **Sorghum**. The workflow enables effective data processing, feature engineering, machine learning model training, and simulation using a pre-trained classifier.

**1\_SB25rAll\_Fit\_Classifier.ipynb**

This script is designed to train and evaluate an **XGBoost-based crop type classifier** with a focus on feature selection and model performance assessment. The main tasks are:

* **Model Fitting and Feature Selection**: Applies XGBoost for classification and uses **Sequential Forward Selection (SFS)** to identify the most relevant features.
* **Model Evaluation**: Evaluates model performance using **accuracy**, **F1 score**, and **confusion matrix** (both raw counts and percentages).
* **Saving the Model and Features**: Exports the trained model as a .joblib file and the selected features as a .json file for later use.

**2\_SB25rAll\_Simulation.ipynb**

This notebook loads a pre-trained XGBoost model and the selected features to simulate crop type predictions and visualize results. The main tasks are:

* **Load Pre-Trained Model and Features**: Loads the .joblib model and .json feature list from the training phase.
* **Feature Engineering**: Adds cyclical encoding for week and computes vegetation growth rates.
* **Model Evaluation**: Performs prediction and computes performance metrics and confusion matrix.
* **Plotting Results**: Generates visualizations such as **feature importance** and **time series plots** for different crops.

**3\_SB25rAll\_Simulation.py**

This script evaluates a pre-trained XGBoost crop type classifier in a streamlined Python workflow. Main tasks include:

* **Loading Pre-Trained Model and Selected Features**: Imports the saved model and feature list for prediction.
* **Feature Engineering**: Adds new features (e.g., sine/cosine transformations of week, vegetation growth rates).
* **Using Pre-Trained Model for Prediction**: Applies the model directly to the cleaned dataset to predict crop types.
* **Evaluate Performance by Week**: Calculates **MAE**, **MSE**, **R²**, and **accuracy** for each unique week.
* **Save Results**: Outputs a CSV file with observed and predicted crop types, along with field ID, year, and week.

**3\_SB25rAll\_SimulationBoots.py**

This script performs a more robust evaluation using **bootstrapped accuracy** per crop type and week. The key tasks are:

* **Loading Pre-Trained Model and Selected Features**: Loads the model and features as in the previous script.
* **Feature Engineering**: Same as above, including combined vegetation indices.
* **Prediction**: Predicts crop types using the trained model.
* **Bootstrapped Accuracy Calculation per Crop per Week**: Computes mean and standard deviation of accuracy via bootstrapping.
* **Smoothing and Plotting Results**: Applies **Gaussian smoothing** to weekly results and plots accuracy per crop with standard deviation as shaded areas.
* **Save Outputs**: Stores results in CSV and PNG formats.

**4\_SB25rAll\_generate\_out\_for\_simulation.py**

This script prepares a **cleaned and feature-engineered dataset** ready for crop type prediction simulations. It ensures consistency in preprocessing between training and simulation phases. The main tasks are:

* **Load Dataset**: Reads the input dataset used for predictions.
* **Feature Engineering**:
  + Applies sine and cosine transformations of the week variable.
  + Calculates growth rates for vegetation indices (**NDVI, EVI, LAI**).
  + Generates interaction features between vegetation and soil moisture.
* **Select Required Features**: Filters the dataset to retain only those features used by the trained model.
* **Save Output**: Exports the processed dataset to a .csv file to be used in simulation scripts.

**General Instructions**

**🔧 Environment Setup:**

* Required Python packages: pandas, numpy, scikit-learn, xgboost, matplotlib, seaborn.
* Use **Jupyter Notebook** for .ipynb files; run .py scripts directly in a Python environment.

**📋 Workflow:**

1. **Train the model**  
   Use 1\_SB25rAll\_Fit\_Classifier.ipynb to train the crop type classifier and save the model and feature list.
2. **Generate simulation-ready dataset**  
   Run 4\_SB25rAll\_generate\_out\_for\_simulation.py to apply consistent preprocessing and generate a dataset ready for prediction.
3. **Run simulation and evaluate**  
   Choose either:
   * 2\_SB25rAll\_Simulation.ipynb (notebook): to simulate predictions and visualize results.
   * 3\_SB25rAll\_Simulation.py (script): to perform weekly prediction analysis and export results.
4. **Add bootstrapped accuracy metrics**  
   Optionally, use 3\_SB25rAll\_SimulationBoots.py to compute weekly **bootstrapped accuracy** per crop and create smoothed plots.

**📤 Outputs:**

* .joblib: Trained XGBoost model
* .json: Selected features
* .csv: Cleaned input data and prediction results
* .png: Accuracy plots per crop
* Plots and metrics saved in the output/ folder for review or downstream processing.