Project Report: The Spectral Soil Modeler (Phase 2)

Course: Software Systems Development

Project: The Spectral Soil Modeler: An Automated ML Workflow
Team - 01

Akshat Kotadia	(2025201005)
Jewel Joseph	(2025201047)
Gaurav Patel	(2025201065)
Parv Shah	(2025201093)
Eshwar Pingili	(2025204030)

Supervisor: **Dr. Abhishek Singh**
Date of Submission: November 13, 2025

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

1. Current Understanding of the Project

1.1. Project Background and Motivation

This project, "The Spectral Soil Modeler," is addressing a significant real-world challenge in computational soil science. The current workflow for researchers at IIIT Hyderabad's Laboratory for Spatial Informatics (LSI) presents a major research bottleneck.

1.2. The Problem Statement

To build predictive models for soil properties (e.g., Clay, Sand, or Total Organic Carbon) from spectral data, researchers must manually and tediously test every combination of 3 spectral preprocessing techniques and 5 machine learning algorithms. This process is repetitive, slow, and severely limits the speed at which new models can be developed and hypotheses can be tested.

1.3. Our Solution

Our team has engineered "The Spectral Soil Modeler," a purpose-built web application designed to eliminate this inefficiency. Our solution fully automates the entire pipeline of creation, training, and validation.
The application allows a researcher to simply:

1. Upload their spectral dataset.
2. Select their target soil property.
3. Click "Run."

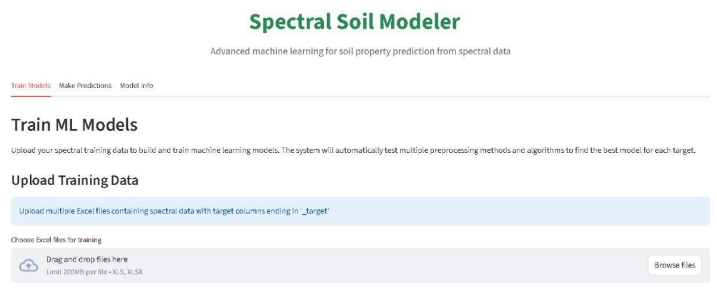
The application then executes a comprehensive, automated process in the background, which includes:

- Systematic Preprocessing: Applies all specified preprocessing techniques (Reflectance, Absorbance, Continuum Removal, plus a 'None' baseline) to the feature set.
- Automated Model Training: Trains all five specified ML models (PLSR, Cubist, GBRT, KRR, SVR) on each of the preprocessed datasets.
- Integrated Hyperparameter Tuning: Uses 5-fold cross-validation with GridSearchCV to automatically find the best hyperparameters for each model, ensuring a fair and robust comparison.
- Interactive Visualization: Presents the results in an interactive dashboard, allowing researchers to instantly compare all 15 model combinations on a ranked leaderboard, diagnose performance with prediction plots, and understand model drivers via feature importance plots.

This tool transforms a task that could take days of manual effort into a single, automated session, directly accelerating the scientific research cycle.

2. UI Screens and User Journey

The application is designed as a **single-page Streamlit dashboard**. The user's journey is linear and intuitive, guiding them from data input to results analysis.



Spectral Soil Modeler

Advanced machine learning for soil property prediction from spectral data

Train ML Models

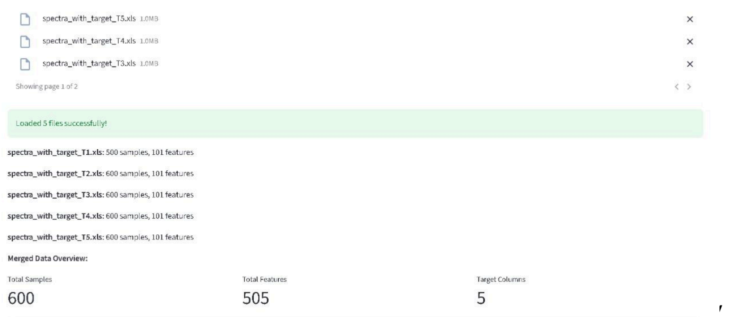
Upload your spectral training data to build and train machine learning models. The system will automatically test multiple preprocessing methods and algorithms to find the best model for each target.

Upload Training Data

Upload multiple Excel files containing spectral data with target columns ending in '_target'

Choose Excel files for training

Figure 1 – Training Screen: Uploading spectral datasets.
The user opens the application and is greeted by a file uploader in the left sidebar. The user uploads their spectral dataset (.csv or .xls). Upon a successful upload, the main panel displays a preview of the first few rows, allowing the user to validate that their file has been loaded and parsed correctly.



spectra_with_target_T5.xls 1.0MB

spectra_with_target_T4.xls 1.0MB

spectra_with_target_T3.xls 1.0MB

Showing page 1 of 2

Loaded 5 files successfully!

spectra_with_target_T1.xls: 500 samples, 101 features

spectra_with_target_T2.xls: 600 samples, 101 features

spectra_with_target_T3.xls: 600 samples, 101 features

spectra_with_target_T4.xls: 600 samples, 101 features

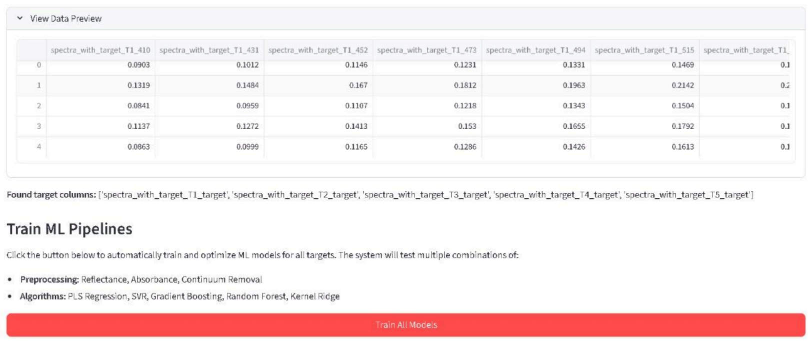
spectra_with_target_T5.xls: 600 samples, 101 features

Merged Data Overview:

Total Samples	Total Features	Target Columns
600	505	5

[illegible]

Next, the user selects their target variable (e.g., 'target') from a dropdown menu in the sidebar, which is automatically populated with the columns from their file. Below this, the application confirms the shape of the features (X) and target (y). Once the target is selected, the user clicks the "Run ML Pipeline" button.
The application enters its processing state. The main window displays a dynamic status message (e.g., Processing: Absorbance - GBRT - Fold 4/5...). This feedback assures the user that the system is working and shows them the progress of the automated training and validation.



View Data Preview

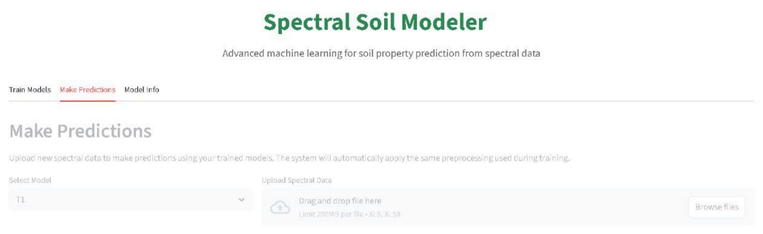
	spectra_with_target_T1_410	spectra_with_target_T1_431	spectra_with_target_T1_452	spectra_with_target_T1_473	spectra_with_target_T1_494	spectra_with_target_T1_515	spectra_with_target_T1_
0	0.0903	0.1012	0.1146	0.1231	0.1331	0.1469	0.1
1	0.1319	0.1484	0.167	0.1812	0.1963	0.2142	0.2
2	0.0841	0.0959	0.1107	0.1218	0.1343	0.1504	0.1
3	0.1137	0.1272	0.1413	0.153	0.1655	0.1792	0.1
4	0.0863	0.0999	0.1165	0.1286	0.1426	0.1613	0.1

Found target columns: ['spectra_with_target_T1_target', 'spectra_with_target_T2_target', 'spectra_with_target_T3_target', 'spectra_with_target_T4_target', 'spectra_with_target_T5_target']

Train ML Pipelines

Click the button below to automatically train and optimize ML models for all targets. The system will test multiple combinations of:

- **Preprocessing:** Reflectance, Absorbance, Continuum Removal
- **Algorithms:** PLS Regression, SVR, Gradient Boosting, Random Forest, Kernel Ridge


Spectral Soil Modeler

Advanced machine learning for soil property prediction from spectral data

Make Predictions

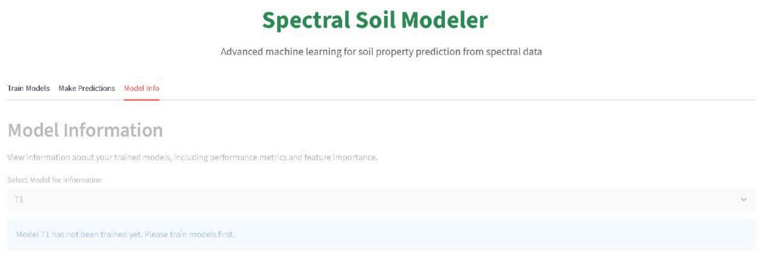
Upload new spectral data to make predictions using your trained models. The system will automatically apply the same preprocessing used during training.

Select Model: T1

Upload Spectral Data

Figure 4 – Make Predictions tab.
Finally, the user can investigate any model in detail. A new dropdown menu, "Select a model... for detailed analysis," allows the user to select a row from the leaderboard. The dashboard then updates to show two diagnostic plots for the selected model:

1. Predicted vs. Actual Values: A scatter plot (aggregating all 5 CV folds) that visualizes the model's predictive accuracy against a 1:1 reference line.
2. Permutation Feature Importance: A horizontal bar chart displaying the top 10 most predictive spectral bands (features), helping researchers understand why a model is working.


Spectral Soil Modeler

Advanced machine learning for soil property prediction from spectral data

Model Information

View information about your trained models, including performance metrics and feature importance.

Select Model for Information: T1

Model T1 has not been trained yet. Please train models first.

Figure 5 – Model Information tab.

After the pipeline completes, the main panel displays the "Model Leaderboard." This table ranks all 15 model combinations (5 models * 3 preprocessing techniques) based on key metrics (RPD, R-squared, and RMSE). The table also displays the specific optimal hyperparameters found by GridSearchCV for each combination

3. Revised Solution Architecture

Architecture Flow Explained:

1. Presentation Layer (Streamlit UI): The user interacts with the UI components (file uploader, select box, button) rendered by app.py.
2. Data Ingestion (load_data): When a file is uploaded, this cached function is called. It reads the file into a pandas DataFrame and stores it.
3. Configuration (select_target_variable): The user's selection from the dropdown menu is used to separate the DataFrame into the feature matrix (X) and the target vector (y).

4. Core Logic - Pipeline Orchestrator (run_ml_pipeline):
    - When the "Run" button is pressed, this main function is called.
    - Outer Loop (Preprocessing): It iterates through a list of preprocessing functions (apply_reflectance, apply_absorbance, etc.).
    - Inner Loop (Modeling): For each preprocessed dataset, it iterates through the list of model names ('PLSR', 'Cubist', etc.).
    - Model Factory (get_ml_model): This function is called to get (1) an unfitted model instance and (2) its param_grid for tuning.
    - Tuning & Validation: It uses GridSearchCV (with cv=5) to train the model and find the best_estimator_ and best_params_.
    - Results Aggregation: It stores the y_true, y_pred, and best_params_ for every fold of every combination in a DataFrame.
5. Evaluation & Visualization (Backend Logic):
    - The calculate_metrics and create_leaderboard functions are called to process the results DataFrame.
    - The plot_predictions_vs_actual and plot_permutation_feature_importance functions are called to generate the Matplotlib figures.
6. Presentation Layer (Streamlit UI): The resulting DataFrames (st.dataframe) and Matplotlib figures (st.pyplot) are rendered directly on the main page for the user to see.

4. Technology Stack Used

Core Logic - Python 3

App Framework - Streamlit - Chosen for its ability to rapidly create data-centric web applications. In our revised architecture, it serves as both the frontend GUI and the backend server running all Python logic, eliminating the need for a separate API.

Data Handling - Pandas, NumPy - The industry standard for data manipulation, numerical computation, and managing the spectral data matrices.

Machine learning - Scikit-learn - The primary ML library, providing: Models: PLSRegression, GradientBoostingRegressor, KernelRidge, SVR.Pipeline & Validation: KFold, GridSearchCV.Model Diagnosis: permutation_importance.Metrics: r2_score, mean_squared_error.

Specialized ML - Cubist Library - The Cubist model is a powerful rule-based regression algorithm not included in scikit-learn. We integrated this third-party library to meet the project's specific model requirements.

Data Visualization - Matplotlib, Seaborn - Used to generate the static plots for the detailed model analysis (Predicted vs. Actual and Feature Importance). Streamlit seamlessly renders these Matplotlib figures.

Version control - Git & Github - Essential for managing code development

5. Revisions Section (Changes Post-Phase 1)

5.1. Major Architectural Revision:
FastAPI to Monolithic Streamlit
We made a major architectural revision in Phase 2 by moving from the decoupled Streamlit–FastAPI design proposed in Phase 1 to a simplified monolithic Streamlit application. Instead of separating the frontend and backend, all components—data loading, preprocessing, ML pipeline execution, evaluation, logging, and visualization—are now executed directly within the Streamlit app.py file. This shift was driven by three factors: (1) development speed and reduced overhead compared to building and maintaining multiple API endpoints, (2) easier state management for long-running ML tasks through Streamlit's native session handling, and (3) alignment with the project's use case as a high-compute, single-user research tool rather than a multi-user distributed service. This revised architecture allowed for faster iteration, simpler debugging, and a more cohesive system overall.

5.2. Integration of Hyperparameter Tuning
Our Phase 1 plan was high-level. In Phase 2, we determined that a fair comparison between models was not possible unless each model was tuned.

- Revision: We integrated GridSearchCV inside the 5-fold cross-validation loop. This correctly tunes hyperparameters only on the training set of each fold, preventing data leakage and providing a much more realistic and trustworthy performance estimate.

5.3. Specialized Handling for Cubist

- Discovery: During testing, we found that the third-party cubist library is incompatible with GridSearchCV's parallel processing feature (n_jobs=-1), which caused the application to crash.
- Revision: We implemented a special if model_name == 'Cubist' block inside run_ml_pipeline. This block runs a manual, sequential grid search for Cubist. This solution maintains the vital hyperparameter tuning capability for Cubist without sacrificing application stability.

Thank You