**Workflow for AWC Prediction Model Development – Sophia Oku**

**Project Goal:** Develop interpretable predictive models for Available Water Capacity (AWC) from measurable soil metrics, with a strong emphasis on generating explicit mathematical expressions.

**Data:** 2021 – 2023 SQM data

**Phase 1: Research and Jurisdictional Scan**

1. **Research and literature Scan**
   * Understand the limitations of previous models and research
   * Note and understand variables used in previous research as well as the reasoning behind choosing those variables

**Phase 2: Data Preparation & Understanding (Initial Setup)**

**Objective:** Clean, explore, and prepare the raw data for model training.

**Software =** VsCode **Programming language =** Python

1. **Data Loading:**
   * Load the provided .csv or .xlsx data file into a Pandas DataFrame.
   * **Input Variables (Features):** PER\_SAND, PER\_SILT, PER\_CLAY, PER\_OM, ACTIVE\_CARBON (µg/g), RESPIRATION (mg/g), PER\_AG\_STABILITY, BNA (mg/kg), pH, P\_INDEX, C\_NRATIO, PER\_C, PER\_N.
   * **Response Variable (Target):** AWC.
   * **Auxiliary Variables (for context/derivation):** SH\_ACC\_NO, SAMPLE\_NO, SOIL\_ACC\_NO, DATE\_RECD, DATE\_SMPL, FIELD\_ID, Cropping system, TEXTURE\_CLASS, FC, PWP.
     + - **Note:** While FC and PWP are provided, the direct target for the models will be AWC. AWC = FC - PWP is a derived relationship; ensure this calculation is consistent in the data or performed as needed.
   * **Drop the following variables**: SH\_ACC\_NO, SAMPLE\_NO, DATE\_RECD, DATE\_SMPL, FIELD\_ID, CROP\_SYS, TEXTURE\_CLASS. *(2022 raw data has NFI\_PLUS\_L, please drop)*
   * **Drop C\_NRATIO**: This will deal with most of the missing values
2. **Initial Data Inspection:**
   * Check data types (df.info()).
   * Inspect descriptive statistics (df.describe()).
   * Identify missing values (df.isnull().sum()).
   * Assess the distribution of numerical features (histograms, box plots).
3. **Data Cleaning & Preprocessing:**
   * **Missing Values:** Missing values will be handled by **deletion method.** This is because most of the core variables do not have missing value. Core variables are variables that most research papers have proven a significant correlation between the variables and FC or PWP.
   * **Outliers:** Identify and decide how to handle outliers (e.g., capping, transformation, or removal) if they are clearly erroneous or significantly skew the data.
   * **Feature Engineering**
     + **Soil Organic Carbon (OC):** Calculate OC = PER\_OM / 1.72. This new feature should be added to the input set.
     + Initial modeling to check the near to add **interaction terms** to the predictor variable.
4. **Feature Selection Strategy (Initial Pass - Correlation Based) !!:**
   * Calculate the **correlation matrix** between all numerical input features and AWC.
   * Visualize the correlation matrix (e.g., heatmap using Seaborn).
   * Identify features with very low correlation to AWC. While not definitively removed yet, these are candidates for later consideration in the "Best Combination of Input Vectors" step.
   * Identify highly correlated input features (multicollinearity). For highly correlated input features (e.g., PER\_C and PER\_OM given OC derivation), consider dropping one if they are redundant, to prevent issues in linear models. *Document threshold for collinearity. You can use a correlation coefficient of r = 0.6.*
5. **Data Splitting:**
   * Split the dataset into **training** and **testing** sets (e.g., 80% training, 20% testing). Use sklearn.model\_selection.train\_test\_split with random\_state for reproducibility. *Don’t forget to set the random state.*
   * The test set should be held out and *only* used for final model evaluation to ensure unbiased performance assessment.

**Phase 3: Model Architecture & Evaluation**

**Objective:** Train, evaluate, and compare models that produce explicit mathematical expressions. The reason for selecting these models is due to their ability to give an explicit mathematical expression.

**3.1: Model Training & Initial Evaluation for Each Architecture**

For each model type, follow these steps:

1. **Conventional Multiple Linear Regression (OLS):**

* **Model Type:** Parametric, produces explicit linear equation.
* **Library:** statsmodels.api (for detailed statistical output and p-values) or sklearn.linear\_model.LinearRegression (for simplicity).
* **Explicit Expression:** The model coefficients will directly give you the linear equation:

AWC = C0 + C1\*Feature1 + C2\*Feature2 + ...

*(save equation expression for final paper)*

* **Performance Metrics (Training Set):**
  + - **R-squared (R2):** Proportion of variance in the dependent variable predictable from the independent variables. *Higher is better*.
    - **Mean Absolute Error (MAE):** Average absolute difference between predicted and actual values. *Lower is better*.
    - **Root Mean Squared Error (RMSE):** Square root of the average of squared differences. Penalizes larger errors more. *Lower is better*.
    - **Mean Absolute Percentage Error (MAPE):** Average percentage error. Useful for interpretability. *Lower is better*.
* **Second Initial Feature Selection Insight !!:** Examine p-values for each coefficient in statsmodels output. High p-values suggest features that might not be statistically significant in the linear relationship. At this point you can make the call to remove features based on the knowledge from the correlation matrix and linear regression model.

1. **Multivariate Adaptive Regression Splines (MARS):**
   * **Model Type:** Piecewise Explicit Mathematical Expressions (captures non-linearities by splitting the data into segments and fitting linear regressions within segments).
   * **Library:** py-earth.
   * **Performance Metrics (Training Set):** Use the same metrics as OLS (R2, MAE, RMSE, MAPE).
2. **Gene Expression Programming (GEP):**

* **Model Type:** Explicit Mathematical Expressions (or Piecewise Explicit Expressions, depending on complexity).
* **Library:** geppy.
* **Key Parameters (from (Shiri et al., 2017) for initial setup):**
  + POPULATION\_SIZE: 30
  + HEAD\_LENGTH: 8
  + NUM\_GENES: 3
  + **Genetic Operator Rates:** Mutation (0.044), Inversion (0.1), One-point Recombination (0.3), Two-point Recombination (0.3), Gene Recombination (0.1), Gene Transposition (0.1), Insertion Sequence Transposition (0.1), Root Insertion Sequence Transposition (0.1).
  + **NGEN (Number of Generations):** Start with 100-200, adjust based on convergence.
  + **PrimitiveSet:**
    - **Terminals:** All selected input features (e.g., PER\_SAND, PER\_SILT, PER\_CLAY, PER\_OM, OC, pH, etc.).
    - **Functions:** Start with common ones (+, -, \*, /) and consider adding non-linear functions based on potential relationships (e.g., math.sin, math.cos, math.log, math.exp, power, sqrt). ***The paper used F3 function set and addition linking function (implying summing gene outputs)***.
* **Fitness Function (Training Set):** You have to determine the fitness function parameters. As per the (Shiri et al., 2017), you need to evaluate "numerous absolute- and relative-error based fitness functions."

|  |  |
| --- | --- |
| **Absolute Error** | **Relative Error** |
| Mean Absolute Error (MAE) | Root Relative Squared Error (RRSE) |
| Root Mean Squared Error (RMSE) | Mean Relative Error (MRE) |
| Sum of Squared Error (SSE) | Relative Absolute Error (RAE) |
|  | Coefficient of Determination (R-squared) |

**Implementation Note:** geppy's base. Fitness takes weights. For minimization, use negative weights (e.g., weights=(-1.0,)). For maximization (like R2), use positive weights (e.g., weights=(1.0,)).

**Explicit Expression:** After GEP runs, geppy allows you to retrieve the best individual's chromosome, which can be compiled into a Python function. The expression can also be printed, but you might need parsing for easy comprehension on a human level.

**3.2: Comprehensive Input Vector (Feature Set) Selection**

**Objective:** Identify the optimal subset of input features for each model. This is essential to remove variables that don’t have any direct correlation with PWP or FC. It will also help the models run faster.

1. **Method:** Employ a systematic feature selection technique for each model type (OLS, MARS, GEP).
   * **Recursive Feature Elimination (RFE) with Cross-Validation:** (For OLS and MARS)
     + Use sklearn.feature\_selection.RFECV.
     + This method recursively removes features and builds a model on the remaining features, using cross-validation to evaluate model performance. It selects the best subset of features.
   * **GEP-specific Feature Selection:** GEP can inherently perform some feature selection if unused terminals (input variables) are pruned from the expression trees. However, to explicitly test combinations:
     + **Wrapper Methods (Forward/Backward Selection):** Start with an empty set and add features (forward) or start with all features and remove them (backward), evaluating GEP performance with each subset. This can be computationally intensive for GEP.
     + **Random Subset Selection:** Randomly try different subsets of features and run GEP.
     + *Recommendation for GEP:* Start with a full feature set as terminals. If the evolved expressions are overly complex or use few features, consider manual trimming of less correlated features initially, then run GEP. For a systematic approach, explore geppy's capabilities or implement a wrapper method.
2. **Evaluation during selection:** For each tested subset of features, evaluate the model's performance using the chosen metrics (especially R2, RMSE, RRSE for GEP) on a validation set (e.g., using k-fold cross-validation on the training data).
3. **Output:** For each model (OLS, MARS, GEP), identify the "best combination of input vectors" (feature subset) that yields the best performance.

**3.3: Hyperparameter Tuning (Top Two Performing Models)**

**Objective:** Optimize the performance of the best-performing models by adjusting their internal parameters.

1. **Selection:** Based on the results from Phase 2.2 (considering explicit expression interpretability), select the two models that show the most promising performance.
2. **Method:**
   * **Grid Search Cross-Validation (sklearn.model\_selection.GridSearchCV):** Define a grid of hyperparameter values to test. The algorithm will exhaustively search through all combinations.
   * **Randomized Search Cross-Validation (sklearn.model\_selection.RandomizedSearchCV):** Define a distribution for each hyperparameter and sample a fixed number of combinations. More efficient for large search spaces.
   * **Bayesian Optimization:** (More advanced, but highly efficient) Uses a probabilistic model to select the next best hyperparameters to try.
3. **Hyperparameters to Tune (Examples):**
   * **OLS:** Not many hyperparameters beyond feature selection.
   * **MARS (py-earth):** max\_terms, max\_degree, min\_points\_per\_subset.
   * **GEP (geppy):** POPULATION\_SIZE, HEAD\_LENGTH, NUM\_GENES, Genetic Operator Rates (MUTATION\_RATE, RECOMBINATION\_RATE, TRANSPOSITION\_RATE - possibly as ranges). The exact combination of functions in the PrimitiveSet can also be considered a hyperparameter.
4. **Evaluation Metric for Tuning:** Use the primary performance metric identified (e.g., RRSE for GEP, RMSE or R2 for OLS/MARS) during the cross-validation within the tuning process.
5. **Output:** The optimal set of hyperparameters for each of the top two models.

**Phase 4: Final Model Evaluation & Delivery**

**Objective:** Evaluate the final, optimized models on unseen data and prepare for handoff.

1. **Final Model Training:** Train the top two models using their optimized hyperparameters and the best feature subsets on the *entire training set*.
2. **Test Set Evaluation:**
   * Apply the final trained models to the previously held-out **test set**.
   * Calculate all relevant performance metrics (R2, MAE, RMSE, MAPE, RRSE for GEP) on the test set. This provides an unbiased estimate of how the models will perform on new, unseen data.
3. **Explicit Expression Extraction & Presentation:**
   * For OLS: Present the equation with coefficients.
   * For MARS: Extract and present the piecewise functions (basis functions and coefficients).
   * For GEP: Extract the evolved symbolic expression. This might be a complex string that needs to be simplified or formatted for readability.
4. **Documentation & Handoff:**
   * Prepare a comprehensive report including:
     + Summary of data preprocessing steps.
     + Justification for feature selection for each model.
     + Optimal hyperparameters for the chosen models.
     + Explicit mathematical expressions for the best-performing models. *(very important!!)*
     + Performance metrics on both training and test sets for all considered models.
     + Discussion of findings, strengths, and limitations of each model.
     + Recommendations for deployment.
   * Provide well-commented Python code for each step, including:
     + Data loading and preprocessing.
     + Model definitions and training.
     + Feature selection and hyperparameter tuning scripts.
     + Evaluation and expression extraction.

**Missing Pipeline Events**

Comparison with existing models (Shiri, liu and Saxton)

**Reference papers**

Amsili, J. P., Van Es, H. M., & Schindelbeck, R. R. (2024). Pedotransfer functions for field capacity, permanent wilting point, and available water capacity based on random forest models for routine soil health analysis. *Communications in Soil Science and Plant Analysis*, *55*(13), 1967–1984. https://doi.org/10.1080/00103624.2024.2336573

Bagnall, D. K., Morgan, C. L. S., Cope, M., Bean, G. M., Cappellazzi, S., Greub, K., Liptzin, D., Norris, C. L., Rieke, E., Tracy, P., Aberle, E., Ashworth, A., Tavarez, O. B., Bary, A., Baumhardt, R. L., Gracia, A. B., Brainard, D., Brennan, J., Reyes, D. B., . . . Honeycutt, C. W. (2022). Carbon‐sensitive pedotransfer functions for plant available water. *Soil Science Society of America Journal*, *86*(3), 612–629. https://doi.org/10.1002/saj2.20395

Liu, L., & Ma, X. (2024). Prediction of soil field capacity and permanent wilting point using accessible parameters by machine learning. *AgriEngineering*, *6*(3), 2592–2611. https://doi.org/10.3390/agriengineering6030151

Saxton, K. E., & Rawls, W. J. (2006). Soil water characteristic estimates by texture and organic matter for hydrologic solutions. *Soil Science Society of America Journal*, *70*(5), 1569–1578. https://doi.org/10.2136/sssaj2005.0117

Shiri, J., Keshavarzi, A., Kisi, O., & Karimi, S. (2017). Using soil easily measured parameters for estimating soil water capacity: Soft computing approaches. *Computers and Electronics in Agriculture*, *141*, 327–339. https://doi.org/10.1016/j.compag.2017.08.012