SEQANA ORGC Soil Profile Dataset Review Analysis by Abhishek Dave

Dataset Overview

This dataset is about the ORGC measurement of the soil profile. The soil data was sourced from an Excel file containing information on soil profiles, layers, and measurement methods. The source data included fields related to geographic coordinates, soil layer properties, and organic carbon measurements.

Data Fields Description and Understanding

- **profile_id**: Unique identifier for each soil profile.
- **Profile_layer_id:** Unique identifier for soil profile layer.
- **Layer_name:** Name of the soil layer**.**
- **Upper_depth:** start point/position/distance/ of the soil layer to measure orgc_values**.**
- **Lower_depth:** endpoint/position/distance/ of the soil layer to measure orgc_values.
- **Litter:** Amount of litter in the soil layer.
- **orgc_value**: Organic carbon value measured in the soil layer for individual orgc_method.
- **Orgc_avg_value**: Average organic carbon value for a soil layer measured by different orgc_method.
- **orgc_date**: Date when the organic carbon value was measured.
- **orgc_method**: method used for measurement of orgc_value.
- **Orgc_dataset_id**: Identifier for the dataset associated with the soil profiles and orgc profile codes.
- **Orgc_profile_code**: Unique Code for each soil profile.
- **X**: Longitude coordinate of the soil profile location.
- **Y**: Latitude coordinate of the soil profile location.
- **Country_name**: Name of the country where the orgc_value was collected.

Approach to data quality issues

ORGC Method Format

- **orgc_method** came as a dictionary pattern string like *{"1:calculation = not specified, detection = not specified, reaction = not specified, sample pretreatment = sieved over 2 mm sieve, spectral = false, temperature = not specified, treatment = not specified", "2:calculation = not specified, detection = not specified, reaction = not specified, sample pretreatment = sieved over 2 mm sieve, spectral = false, temperature = not specified, treatment = not specified"}*

the 1: and 2: means the number of times an orgc_method applied on an individual profile.

- Also, specific methods contain attributes like calculation, reaction, detection, and so on that were used to measure orgc_values for individual soil profile_layer.
- **Solution**: Normalize it consistently by dismantling the attributes with their values under each method instance and storing them in the database under 3NF structure.

ORGC Values

- **orgc_value** came as a dictionary pattern string like {1:1.80,2:4.10} based on the number of times an **orgc_method** applied on an individual profile_layer.
- Also contains null values, meaning on specific soil layer ORGC was not measured or calculated.
- **Solution**: Normalize and associate it with each soil profile layer where its specific method instance is used to measure this orgc_value.

ORGC Date

- **orgc_date** came as a dictionary pattern string like {1:1960-12-19,2:1960-12-19} based on the number of times an orgc_method applied on an individual profile_layer.
- Also contains different date format {1:1992/02/17} and some invalid dates like {1:????-??-??} within pattern string
- **Solution**: Standardize date formats to a consistent format (e.g., YYYY-MM-DD) and normalize and associate it with each soil profile layer to get when its specific method instance is used on the soil profile layer.

Missing Data

- **Issue**: Presence of missing values in critical fields such as **orgc_value, orgc_value_avg, orgc_date, layer_name, litter**.
- **Approach Handling missing data**: Given the context of soil science, it makes sense not to impute missing data because making assumptions could misinterpretation and misrepresent the real-world conditions.
- **Understanding**:
    - Missing **layer_name**, even if an **orgc_value** is recorded, can indeed lead to misinformation. This is because the layer name is important for interpreting where the organic carbon values apply. If too many null values are present in key attributes, it compromises the dataset's quality and reliability.

Handle Outliers

- **Issue**: Outliers in fields such as **orgc_value and orgc_value_avg** may affect data quality and may increase bias later in data analysis.
- **Approach Handling outliers**: Based on the soil science domain expert's guidance, we can apply statistical methods to detect and handle outliers either by capping them or

removing them but without it we cannot remove the outliers on our own to loose data without knowledge.

Inconsistencies

- **Issue**: Inconsistencies in data format types, such as mismatches in fields like **orgc_profile_code and orgc_date**, where mixed or incorrect values may be present.
- **Approach**: Implement data validation rules to enforce consistent formats. Correct inconsistencies by transforming values into a standardized format (e.g., converting dates to a uniform format like **%Y-%m-%d** and ensuring the reformat values in string in orgc_profile_code).

Data Model Implementation

Tables and Fields

orgc_method

- **Description:** Stores detailed information about the methods used for measuring organic carbon.
- **Fields:**
    - **id (Primary Key):** Unique identifier for collective method attribute with instance.
    - **method_instance:** Identifier for the instance of the method.
    - **calculation:** Details about the calculation method used.
    - **detection:** Detection method used in the measurement.
    - **reaction:** Chemical reactions involved in the measurement.
    - **sample_pretreatment:** Description of sample pretreatment procedures.
    - **spectral:** Spectral analysis details.
    - **temperature:** Temperature conditions during measurement.
    - **treatment:** Any additional treatment details.

orgc_profile

- **Description:** Contains information about the soil profiles from which data is collected.
- **Fields:**
    - **id (Primary Key):** Unique identifier for each soil profile record.
    - **profile_id:** Identifier for the soil profile.
    - **orgc_profile_code:** ORGC Code for the soil profile.
    - **orgc_dataset_id:** Identifier for the dataset associated with the ORGC profiles.
    - **latitude:** Latitude coordinate of the soil profile location.
    - **longitude:** Longitude coordinate of the soil profile location.

- **country_name:** Name of the country where the profile was collected.

orgc_profile_layer

- **Description:** Details about different soil layers within a profile, including measurement data.
- **Fields:**
  - **id (Primary Key):** Unique identifier for each soil layer record.
  - **profile_layer_id:** Identifier for the soil layer within the profile.
  - **orgc_profile_id (Foreign Key):** Reference to the id in orgc_profile, linking the layer to the corresponding soil profile.
  - **upper_depth:** Upper depth of the soil layer.
  - **lower_depth:** Lower depth of the soil layer.
  - **layer_name:** Name of the soil layer.
  - **litter:** Amount of litter in the soil layer.
  - **orgc_method_id (Foreign Key):** Reference to the id in orgc_method, indicating the method used for measurement.
  - **orgc_value:** Organic carbon value measured in the soil layer.
  - **orgc_value_avg: A**verage organic carbon value for the soil layer.
  - **orgc_date:** Date when the measurement was taken.

Relationships Between Tables

1. **orgc_profile to orgc_profile_layer**
   - **Type: One-to-Many**
   - **Relationship:** Each orgc_profile can have multiple orgc_profile_layer records. The orgc_profile_id in orgc_profile_layer is a foreign key referencing the id in orgc_profile.
2. **orgc_method to orgc_profile_layer**
   - **Type: One-to-Many** (based on method_instance: Each orgc_method can be applied to multiple orgc_profile_layer records, but each orgc_profile_layer entry references only one orgc_method. Even though a method can be used multiple times (via different instances per profile layer) for different soil layers, each soil layer uses a single method per measurement.
   - **Relationship:** Each orgc_method can be used in single or multiple orgc_profile_layer to record orgc_value. The orgc_method_id in orgc_profile_layer is a foreign key referencing the id in orgc_method.

Step-by-Step Code Implementation

Step 1: Raw Data Extraction

- **Function Call:** extract.read_raw_data(file_name)
- **Explanation:** This reads the raw Excel file into a DataFrame (raw_df) from the specified file (seqana-data-engineering-challenge-data-wosis-belgium.xlsx).

Additional Review:

- **Function Call:** extract.generate_review_dataframes(raw_df)
- **Console output:** The metadata is displayed by breaking it into multiple smaller sections, allowing the user to examine and verify the content.

Step 2: Data Quality Checks on Metadata Level

1. Check Data Types

- **Function Call:** dataqualitychecker.check_column_data_types(raw_df, desired_column_types)
- **Explanation:** Check if columns like longitude (X), latitude (Y), and others have the correct data types (int, float, object).
- **Output:** A report indicating if there are data type mismatches in the raw data.

2. Check Patterns in Specific Columns

- **Function Call:** dataqualitychecker.check_column_patterns(raw_df, column_patterns)
- **Explanation:** Check if columns like orgc_value, orgc_date, and orgc_method adhere to expected patterns.
- **Output:** A report showing pattern mismatches (if any) in these columns.

3. Check for Duplicates

- **Function Call:** preprocessor.drop_duplicates(raw_df, df_name='raw_extracted_dataframe')
- **Explanation:** Remove any duplicate records from the raw dataset.

4. Outliers, Missing Values, and Other Checks

- **Outlier Check:**
    - **Function Call:** dataqualitychecker.check_outliers(raw_df, col_to_check_for_outliers)

- **Explanation:** Check outliers in row indices for column orgc_value_avg.

- **Missing Values Check:**
  - `col_to_check_for_missing_values` = ['X', 'Y', 'profile_id', 'profile_layer_id', 'country_name', 'layer_name', 'orgc_dataset_id', 'orgc_profile_code', 'orgc_value', 'orgc_value_avg', 'orgc_date', 'upper_depth', 'lower_depth', 'orgc_method']
  - **Function Call:** dataqualitychecker.check_missing_values(raw_df, col_to_check_for_missing_values)
  - **Explanation:** Identify missing values in all columns.
- **Advanced check on Latitude and Longitude Check:**
  - **Function Call:** dataqualitychecker.check_lat_long(raw_df, lat_column, long_column)
  - **Explanation:** Ensure latitude and longitude values are within valid ranges.
- **Advanced check on Depth Columns Check:**
  - **Function Call:** dataqualitychecker.check_depth_columns(raw_df, upper_depth_col, lower_depth_col)
  - **Explanation:** Ensure upper_depth and lower_depth columns contain valid values individually and relatable.

Step 3: Data Preprocessing

1. Extract Raw Data Based on Method Instances

- **Function Call:** extract.extract_raw_data_based_on_method_instance(row)
- **Explanation:** Normalize complex data stored as dictionaries in columns like orgc_method, orgc_value, and orgc_date and extract instance-specific data.
- **Appending Preprocessed Rows:**
  - **Function Call:** preprocessor.append_preprocessed_rows(new_rows)
  - **Explanation:** Append the new preprocessed rows extracted from the original dataset.

2. Remove Duplicates Post-Preprocessing

- **Function Call:** preprocessor.drop_duplicates(df_to_preprocessed, df_name='preprocessed_dataframe')
- **Explanation:** Remove any duplicates that may have resulted from the preprocessing steps.

3. Reformat Dates

- **Function Call:** preprocessor.reformat_dates(df_preprocessed, date_column, desired_date_format)
- **Explanation:** Ensure the orgc_date field follows the standard format ('%Y-%m-%d').

4. Check Date Format

- **Function Call:** dataqualitychecker.check_date_format(df_preprocessed, ['reformat_orgc_date_for_instance'])
- **Explanation:** Verify if all reformatted dates adhere to the expected format.

Step 4: Data Transformation and Normalization

- **Function Call:** transformer.normalize_dataframes(df_preprocessed)
- **Explanation:** The data is split into multiple normalized tables (like orgc_method_df, orgc_profile_df, and orgc_profile_layer_df), corresponding to the third normal form (3NF).

Step 5: Review Normalized Data

- **Function Call:** extract.generate_review_dataframes(df_normalized_dict)
- **Explanation:** The transformed data is reviewed similarly to the raw data, allowing you to check the final structure and content of the 3NF data tables before data loading.

Step 6: Data Loading

- **Function Call:** dataloader.save_to_sqlite(df_normalized_dict, sql_script_file_name, file_name='seqana_soil_data.db')
- **Explanation:** The normalized DataFrames are written into a file ‘**'seqana_soil_data.db**’ using the schema provided in the SQL script (initialize_db.sql) that can be importable to the SQLite database for further analysis.
- **Error Handling:** If an error occurs during the loading process, the program will stop and raise the error.

Normalization (3NF) Decision for three tables

Approach:

- The data is split into three tables:
    1. **orgc_method**: Stores method-related details for each carbon measurement method.
    2. **orgc_profile**: Contains the profile-level data like latitude, longitude, and country.
    3. **orgc_profile_layer**: Stores layer-specific data for each profile, including depths, organic carbon values, and associated methods.
- **Why this structure of 3 tables:** By separating orgc_method, orgc_profile, and orgc_profile_layer data, each table focuses on one type of data. This structure avoids redundancy (e.g., repeating method details across multiple records and multiple profiles for profile_layers as a soil profile can have multiple layers for which orgc_value is calculated) and ensures referential integrity via foreign keys.

Relationships:

- orgc_profile_layer references orgc_profile to connect layers to profiles.
- orgc_profile_layer references orgc_method to link layers to their associated orgc measurement methods.

BONUS-1: Advanced Data Quality Checks Propose and implement

Propose and if possible, implement advanced data quality checks on the ingested data.

Advanced-Data Quality Checks

1. **Spatial Consistency Check**: In the code, implemented latitude and longitude validation to ensure these values fall within the **appropriate geographical range** (**latitude between -90 and 90**, and **longitude between -180 and 180**).
2. **Temporal Consistency**: In the code, implemented to dates are consistently formatted as '%Y-%m-%d'. For a more advanced check, we should also verify that dates are within a **realistic range** and **check for future or historical dates**.
3. **Depth Consistency**: In the code, implemented to check and ensure that the upper depth can be zero and non-negative, while the lower depth must be non-negative and greater than zero. To further refine this, we could implement checks to ensure that the upper depth is always less than the lower depth and align depth values with known soil horizon data.
    a) **For More depth consistency,** we can define the range between upper depth and lower depth, so the data is reliable with soil layers not further deep**.**

4. **Cross-Validation of orgc_value**: In the code, detection of outliers in orgc_value and orgc_value_avg. To advance this, we might need to consult domain expert and then take any further action.
5. **Method Consistency Check**: We validate special fields like orgc_value and orgc_date. and cross-checked that the methods used to measure organic carbon (orgc_method) align with the recorded values by method instance, by ensuring consistency and correctness in our data.

These advanced checks will improve the robustness of our data quality assurance processes and help maintain the integrity of our soil data analysis.

BONUS-2: Data Integrity, Consistency, Versioning over time?

How would you ensure data integrity and consistency over time? Suggest a solution

for tracking changes, updates and deletions without losing access to prior versions of the data.

Answer:

To ensure data integrity and consistency over time, while also tracking changes, updates, and deletions without losing access to prior versions, we can implement a **versioning system**.

I have implemented different versioning systems in my current and previous projects whose database I designed based on various schema designs such as **Star Schema design**, and **Snowflake Schema design.**

First Approach: Audit Version Schema implementation

By adapting **Audit Schema with an individual table to track granularity based on the Column Change Tracking** approach, is ideal and requires a separate **audit table for each main table** in the main database where we can track changes. Each of these audit tables would record the changes at the column level for its corresponding main table.

Example:

Main Database: Table_ABC

PK	Column_1	Column_2	Version_id	Is_active
101	A	E	1	0
102	B	F	1	0
103	C	G	2	1
104	D	H	2	1

Audit_Database: Table_ABC_audit

PK_ABC	Version_id	Changed_at	Old_value	Updated_value	Modified_at	Update_by
101	1	Column_1	A	RR	23-09-2020 08:00:00	Abhishek
103	2	Column_2	G	KJ	22-01-2021 01:00:00	Sam
102	1	Column_2	F	AA	02-10-2023 04:00:00	Jena
102	1	Column_1	B	CC	12-05-2024 15:00:00	Jena

Second Approach: Centralizing Version System to maintain data consistency

By accommodating a **version_management** Table and this **version_management_id** acting as a foreign key for the 3NF structure we cannot lose previous data, and the data loading strategy can be incremental and fast in this case. Implementing a versioning table involves creating a dedicated table to track data changes.

And **adding a column** called **is_active or is_current** in every table so that we know which latest version data we need to use while maintaining history.

version_management (

id PRIMARY KEY INTEGER,

time_stamp DATETIME,

date_of_last_revision DATETIME,

internal_dataset_version TEXT,

copyright TEXT,

license_type TEXT,

access_restrictions TEXT)

Table_A

PK	Column_2	Column_3	Column_4	**Created_at**	**version_ management _id**	**Is_current**
1	B	1	5	**23-09-2020 08:00:00**	1	0
2	C	D	5	**23-09-2020 08:00:00**	1	0
3	E	5	9	**12-10-2024 08:00:00**	2	1
4	H	7	9	**12-10-2024 08:00:00**	2	1
5	E	5	9	**12-10-2024 08:00:00**	2	1

By looking we can say, data belongs to version=2 is latest as is_current check set to 1.

References

Disclosure of Chat-GPT use:

For basic soil science, soil profile domain knowledge

To acquire basic domain knowledge about various data fields and their representations in the context of soil science due to limited time constraints to get the domain idea and limitations before starting work and thinking about the approach:

- **ORGC:** Organic Carbon Content in soil.
- **Soil Profile:** A vertical section of soil that shows its layers or horizons.
- **Profile Layer:** Specific layers within the soil profile, each with distinct properties.
- **Depths:** Measurements indicating the upper and lower boundaries of soil layers.
- **Litter:** Organic matter on the soil surface, including decomposing plant material.

Took assistance in developing improved logic the following method implementations

- **extract_raw_data_based_on_method_instance**: This method in DataExtraction class orgc_value and orgc_date for specific orgc_method instances. ChatGPT assisted in developing the approach for parsing method strings, extracting relevant values, and handling instance-specific data, ensuring accurate extraction of values and dates associated with each method instance.
- **generate_review_dataframes**: This method in DataExtraction class generates review DataFrames for a single DataFrame or a dictionary of DataFrames, providing summaries and dataset information. ChatGPT helped in formulating the approach to processing dataframes, handling missing values, and organizing summary statistics.
- ChatGPT helped in improving the logic for the method **validate_and_correct_dataframe** in DataLoading class based on defined SQL schema requirements, ensuring data integrity before data insertion into the database.

cosmetic purposes, including:

- **Commenting** and **clarifying code** for better readability.
- **Refining the ReadMe** documentation for better readability and flow.
- **Refining this analysis documentation** for better readability and organization.

A Thank-You Note:

Thank you for providing me with the opportunity to tackle this technical challenge. I found it both engaging and rewarding to delve into the complexities of soil carbon measurement and data management. The task has significantly enhanced my understanding of soil data, and the methodologies involved.

I look forward to any feedback from my analysis and am excited about the possibility of moving forward to the next round. I am eager to work together on meaningful projects that contribute to impactful solutions. Thank you once again for this opportunity!