Soil Organic Carbon Prediction

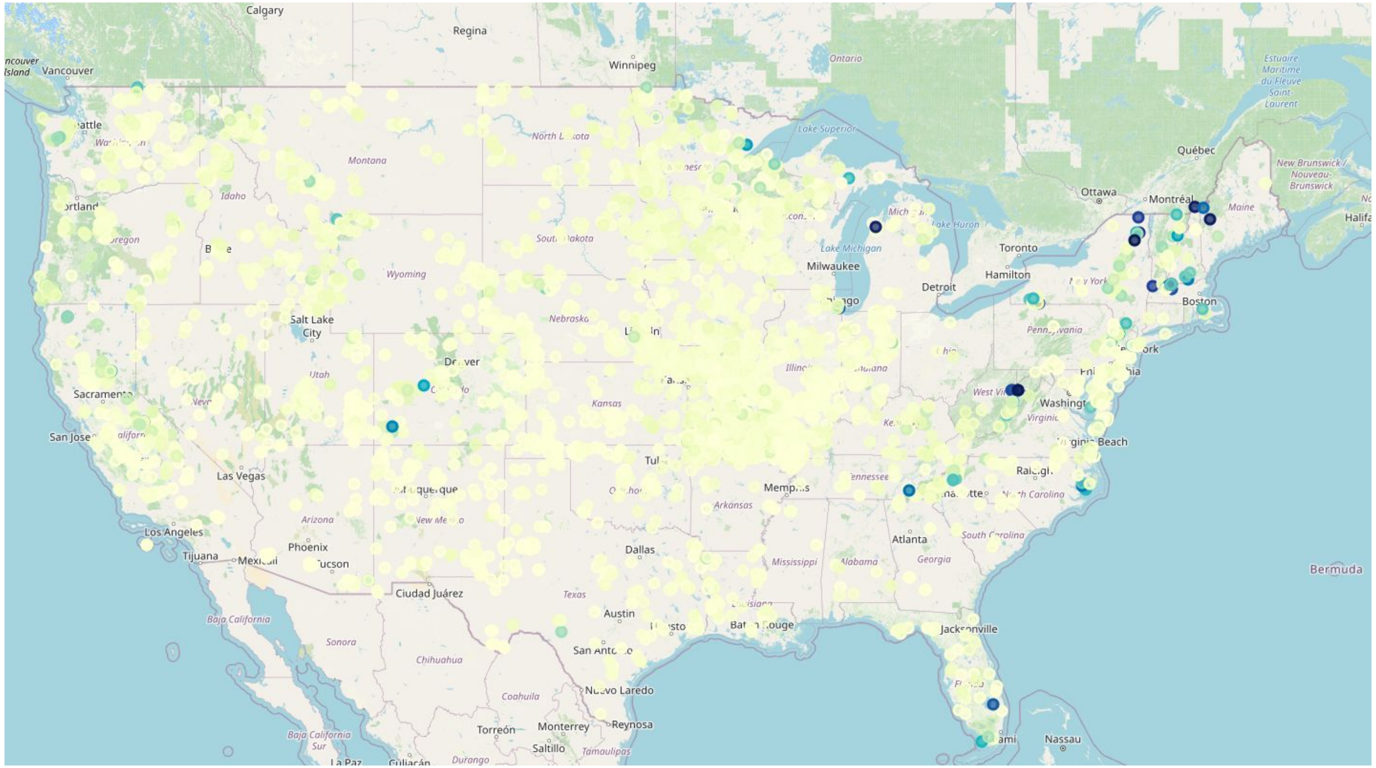
Maria Oros

Data Collection of Soil Dynamic Properties in CONUS

Data Preparation

Data about the soil organic carbon was collected from Neon across the contiguous USA (CONUS), spanning 1985 to 2021. For location or soil profile, 6 depths were considered, leading to a total of 43K (~7Kx6) raw data observations.

Later the soil covariate properties were collected from GEE, due to probabilistic covariates, terrain, land use and land change and temporal covariates such as precipitation and temperature, those last were aggregate by mean, min and max over the year of interest.

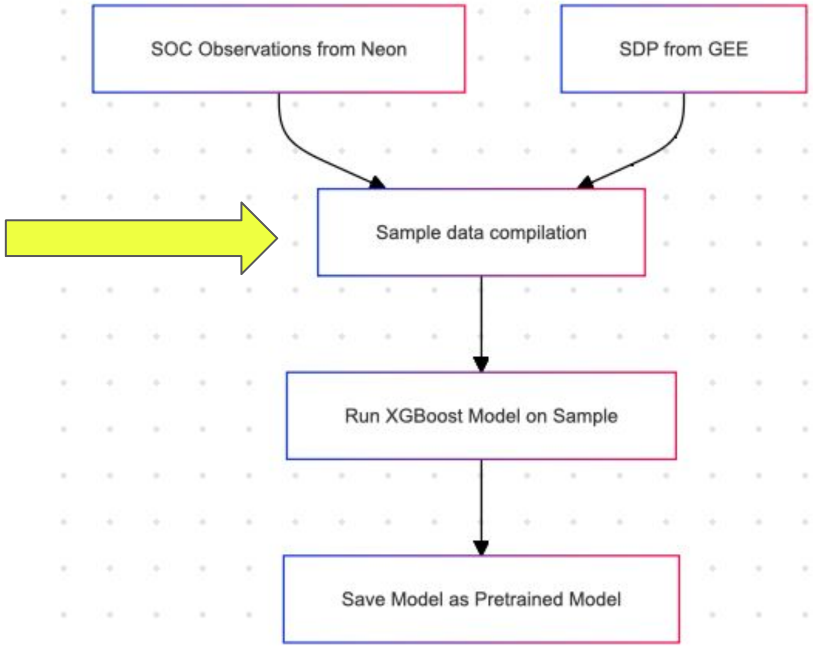


SOC Observations - Sample

Soil organic carbon observations (NEON)

Covariates: soil dynamic properties (GEE)

lat	lon	year	depth	soc	om	Min tem	Max tem	…	hillshade
x1	y1	2018	30cm	1.1	.	.	.	.	.
x1	y1	2018	100cm	1.3	.	.	.	.	.
x2	y2	2018	30cm	1.3	.	.	.	.	.
xn	yn	2019		1.1	.	.	.	.	.

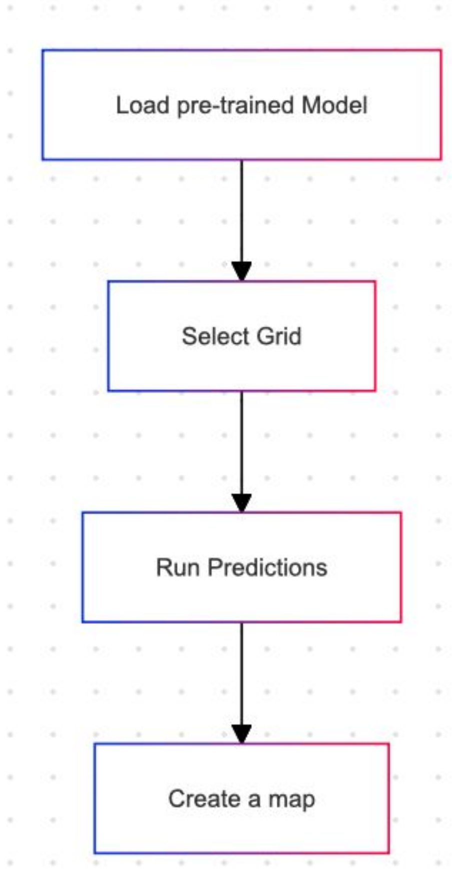

SOC Observations from Neon
SDP from GEE
Sample data compilation
Run XGBoost Model on Sample
Save Model as Pretrained Model

Grid on an AOI (Example WI 2018 on a 500mx500m resolution)

Grid points

Covariates: soil dynamic properties (GEE)

lat	lon	year	depth	[redacted]	om	Min tem	Max tem	…	hillshade
x1	y1	2018	30cm	[redacted]	.	.	.	.	.
x1	y1	2018	100cm	[redacted]	.	.	.	.	.
x2	y2	2018	30cm	[redacted]	.	.	.	.	.
				[redacted]					
				[redacted]					
xn	yn	2018		[redacted]	.	.	.	.	.



Variables to describe the soil organic carbon

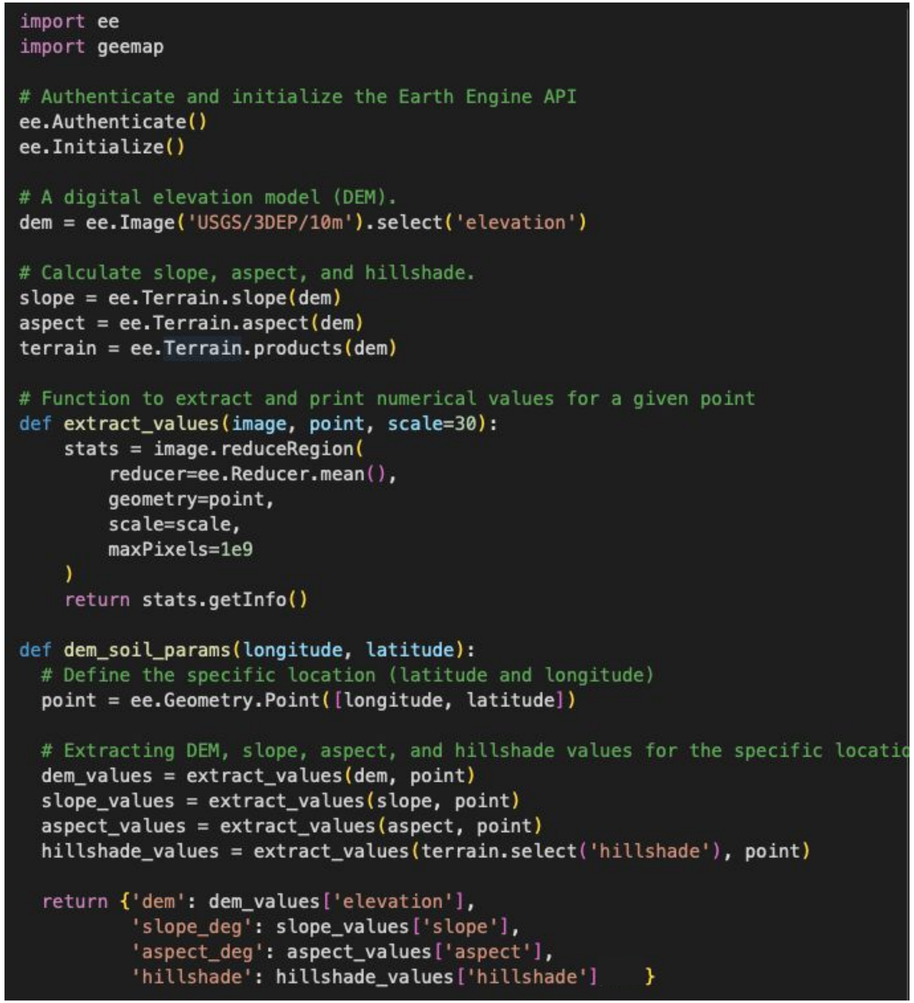
- Time varying variables: total precipitation, min max and avg temperature from Daymet
- Stable Variables
  - Polaris:
    - Probabilistic: clay, silt, sand, ph, bd, om
  - GEE:
    - Terrain: dem, hillshade, slope, Aspect, // twi*, mrvbf* skip those
    - Land use, land change from `USFS/GTAC/LCMS/v2023-9`

Dashboard to visualize the data:
https://connect.doit.wisc.edu/content/cfa49680-e57f-44b2-9c8f-41313c9e072c
Data:
https://drive.google.com/file/d/101kazyuvTPr-i4R21dmvJyB1QT9yYHvK/view?usp=sharing

Extract terrain features for specific locations

Procedure: initialized a **GEE session in python**
A Digital Elevation Model (DEM) from the USGS 3D Elevation Program (3DEP) at 10m resolution is loaded and the 'elevation' band is selected.
Input: specific location (lat, lon)
Output:

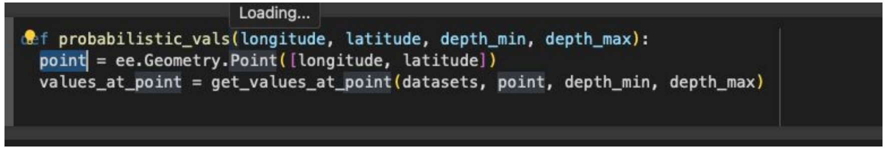
- dem
- slope
- aspect
- hillshade



```
import ee
import geemap

# Authenticate and initialize the Earth Engine API
ee.Authenticate()
ee.Initialize()

# A digital elevation model (DEM).
dem = ee.Image('USGS/3DEP/10m').select('elevation')

# Calculate slope, aspect, and hillshade.
slope = ee.Terrain.slope(dem)
aspect = ee.Terrain.aspect(dem)
terrain = ee.Terrain.products(dem)

# Function to extract and print numerical values for a given point
def extract_values(image, point, scale=30):
    stats = image.reduceRegion(
        reducer=ee.Reducer.mean(),
        geometry=point,
        scale=scale,
        maxPixels=1e9
    )
    return stats.getInfo()

def dem_soil_params(longitude, latitude):
  # Define the specific location (latitude and longitude)
  point = ee.Geometry.Point([longitude, latitude])

  # Extracting DEM, slope, aspect, and hillshade values for the specific locatio
  dem_values = extract_values(dem, point)
  slope_values = extract_values(slope, point)
  aspect_values = extract_values(aspect, point)
  hillshade_values = extract_values(terrain.select('hillshade'), point)

  return {'dem': dem_values['elevation'],
          'slope_deg': slope_values['slope'],
          'aspect_deg': aspect_values['aspect'],
          'hillshade': hillshade_values['hillshade']    }
```

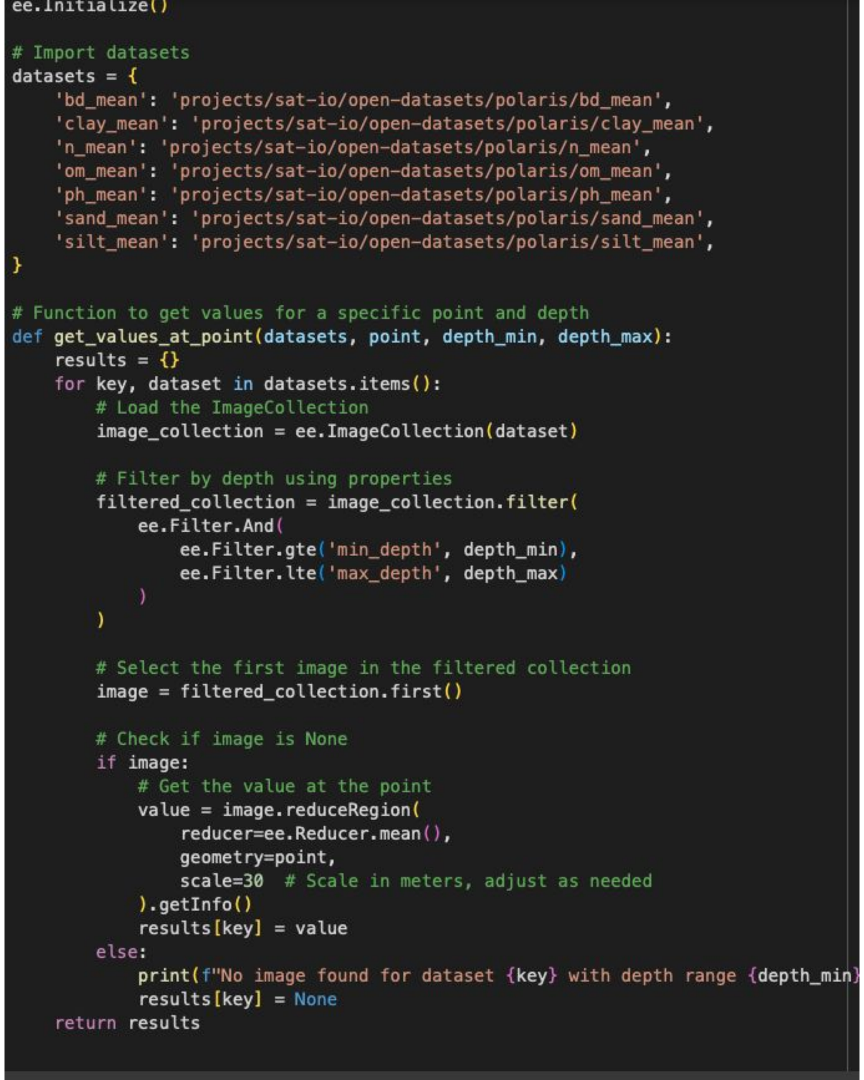
No time dependent Variables - Probabilistic variables

Procedure: initialized a GEE session in python
Input: specific location (lat, lon) and depth



```python
def probabilistic_vals(longitude, latitude, depth_min, depth_max):
  point = ee.Geometry.Point([longitude, latitude])
  values_at_point = get_values_at_point(datasets, point, depth_min, depth_max)
```

Output:
- Clay
- Silt
- Sand
- Bulk density
- Organic Matter
- ph



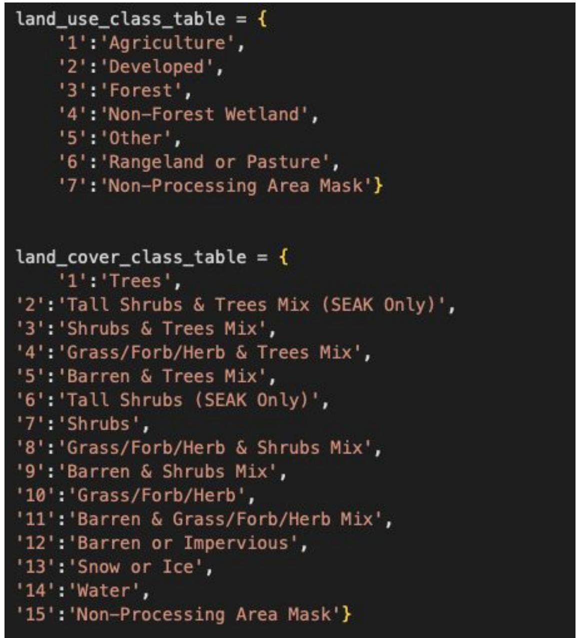
```python
ee.Initialize()

# Import datasets
datasets = {
    'bd_mean': 'projects/sat-io/open-datasets/polaris/bd_mean',
    'clay_mean': 'projects/sat-io/open-datasets/polaris/clay_mean',
    'n_mean': 'projects/sat-io/open-datasets/polaris/n_mean',
    'om_mean': 'projects/sat-io/open-datasets/polaris/om_mean',
    'ph_mean': 'projects/sat-io/open-datasets/polaris/ph_mean',
    'sand_mean': 'projects/sat-io/open-datasets/polaris/sand_mean',
    'silt_mean': 'projects/sat-io/open-datasets/polaris/silt_mean',
}

# Function to get values for a specific point and depth
def get_values_at_point(datasets, point, depth_min, depth_max):
    results = {}
    for key, dataset in datasets.items():
        # Load the ImageCollection
        image_collection = ee.ImageCollection(dataset)

        # Filter by depth using properties
        filtered_collection = image_collection.filter(
            ee.Filter.And(
                ee.Filter.gte('min_depth', depth_min),
                ee.Filter.lte('max_depth', depth_max)
            )
        )

        # Select the first image in the filtered collection
        image = filtered_collection.first()

        # Check if image is None
        if image:
            # Get the value at the point
            value = image.reduceRegion(
                reducer=ee.Reducer.mean(),
                geometry=point,
                scale=30  # Scale in meters, adjust as needed
            ).getInfo()
            results[key] = value
        else:
            print(f"No image found for dataset {key} with depth range {depth_min}
            results[key] = None
    return results
```

LULC

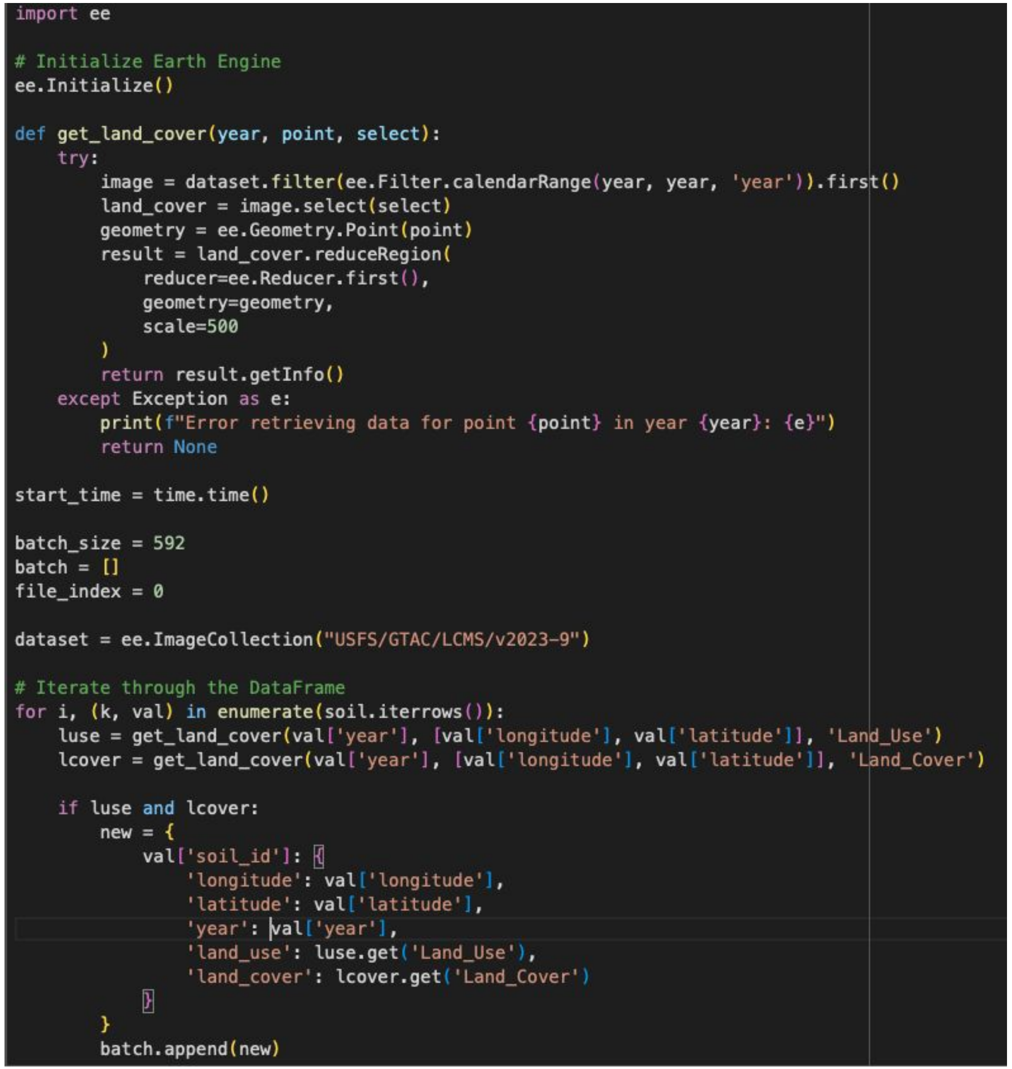
Procedure: initialized a GEE session in python
Input: specific location (lat, lon) and year

- Land Use encoded
- Land cover encoded

Then turn it into a labeling:



```
land_use_class_table = {
    '1':'Agriculture',
    '2':'Developed',
    '3':'Forest',
    '4':'Non-Forest Wetland',
    '5':'Other',
    '6':'Rangeland or Pasture',
    '7':'Non-Processing Area Mask'}


land_cover_class_table = {
    '1':'Trees',
'2':'Tall Shrubs & Trees Mix (SEAK Only)',
'3':'Shrubs & Trees Mix',
'4':'Grass/Forb/Herb & Trees Mix',
'5':'Barren & Trees Mix',
'6':'Tall Shrubs (SEAK Only)',
'7':'Shrubs',
'8':'Grass/Forb/Herb & Shrubs Mix',
'9':'Barren & Shrubs Mix',
'10':'Grass/Forb/Herb',
'11':'Barren & Grass/Forb/Herb Mix',
'12':'Barren or Impervious',
'13':'Snow or Ice',
'14':'Water',
'15':'Non-Processing Area Mask'}
```

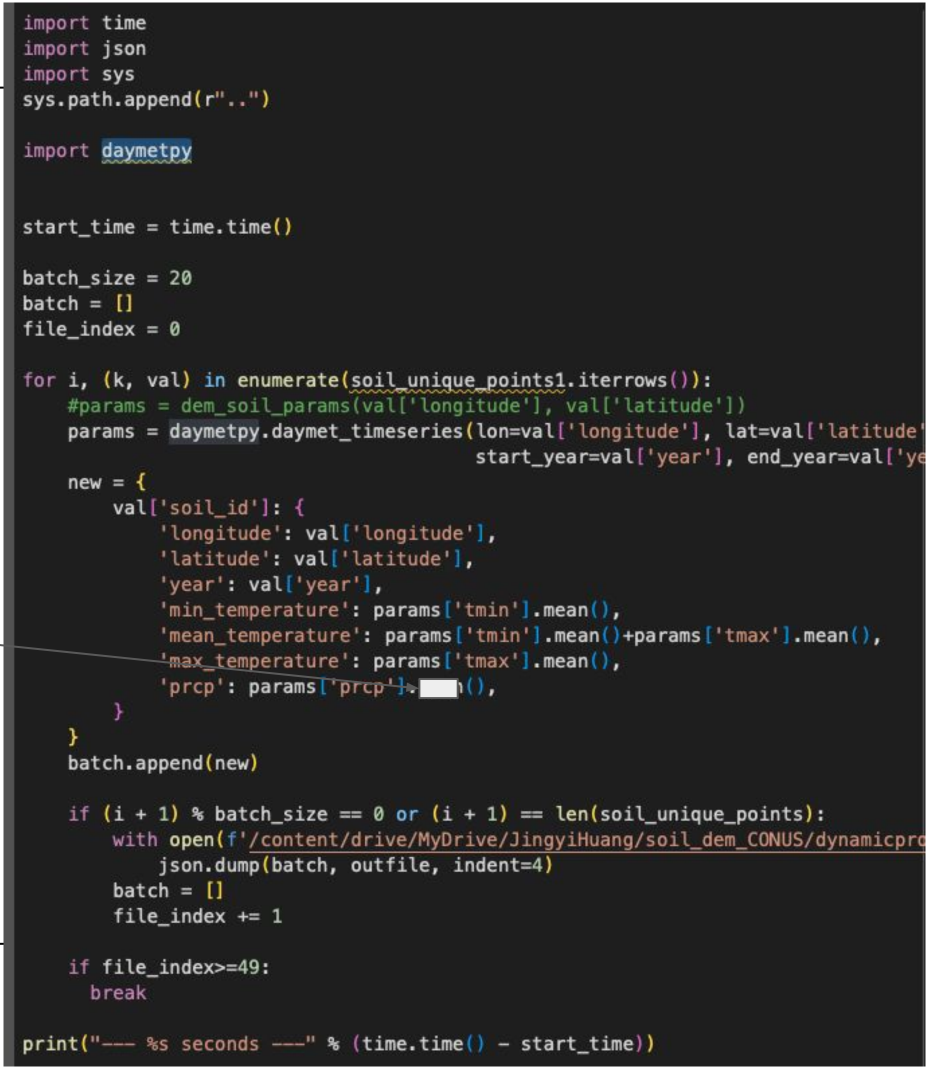


```
import ee

# Initialize Earth Engine
ee.Initialize()

def get_land_cover(year, point, select):
    try:
        image = dataset.filter(ee.Filter.calendarRange(year, year, 'year')).first()
        land_cover = image.select(select)
        geometry = ee.Geometry.Point(point)
        result = land_cover.reduceRegion(
            reducer=ee.Reducer.first(),
            geometry=geometry,
            scale=500
        )
        return result.getInfo()
    except Exception as e:
        print(f"Error retrieving data for point {point} in year {year}: {e}")
        return None

start_time = time.time()

batch_size = 592
batch = []
file_index = 0

dataset = ee.ImageCollection("USFS/GTAC/LCMS/v2023-9")

# Iterate through the DataFrame
for i, (k, val) in enumerate(soil.iterrows()):
    luse = get_land_cover(val['year'], [val['longitude'], val['latitude']], 'Land_Use')
    lcover = get_land_cover(val['year'], [val['longitude'], val['latitude']], 'Land_Cover')

    if luse and lcover:
        new = {
            val['soil_id']: {
                'longitude': val['longitude'],
                'latitude': val['latitude'],
                'year': val['year'],
                'land_use': luse.get('Land_Use'),
                'land_cover': lcover.get('Land_Cover')
            }
        }
        batch.append(new)
```

Time Varying Variables

Procedure: used **daymet** library from python
Input: specific location (lat, lon) and year
Output:

- Averaged precipitation** sum
- mean temperature
- max temperature
- mean temperature
- Total Precipitation



```python
import time
import json
import sys
sys.path.append(r"..")

import daymetpy


start_time = time.time()

batch_size = 20
batch = []
file_index = 0

for i, (k, val) in enumerate(soil_unique_points1.iterrows()):
    #params = dem_soil_params(val['longitude'], val['latitude'])
    params = daymetpy.daymet_timeseries(lon=val['longitude'], lat=val['latitude'
                                        start_year=val['year'], end_year=val['ye
    new = {
        val['soil_id']: {
            'longitude': val['longitude'],
            'latitude': val['latitude'],
            'year': val['year'],
            'min_temperature': params['tmin'].mean(),
            'mean_temperature': params['tmin'].mean()+params['tmax'].mean(),
            'max_temperature': params['tmax'].mean(),
            'prcp': params['prcp'].[redacted]n(),
        }
    }
    batch.append(new)

    if (i + 1) % batch_size == 0 or (i + 1) == len(soil_unique_points):
        with open(f'/content/drive/MyDrive/JingyiHuang/soil_dem_CONUS/dynamicpro
            json.dump(batch, outfile, indent=4)
        batch = []
        file_index += 1

    if file_index>=49:
      break

print("--- %s seconds ---" % (time.time() - start_time))
```

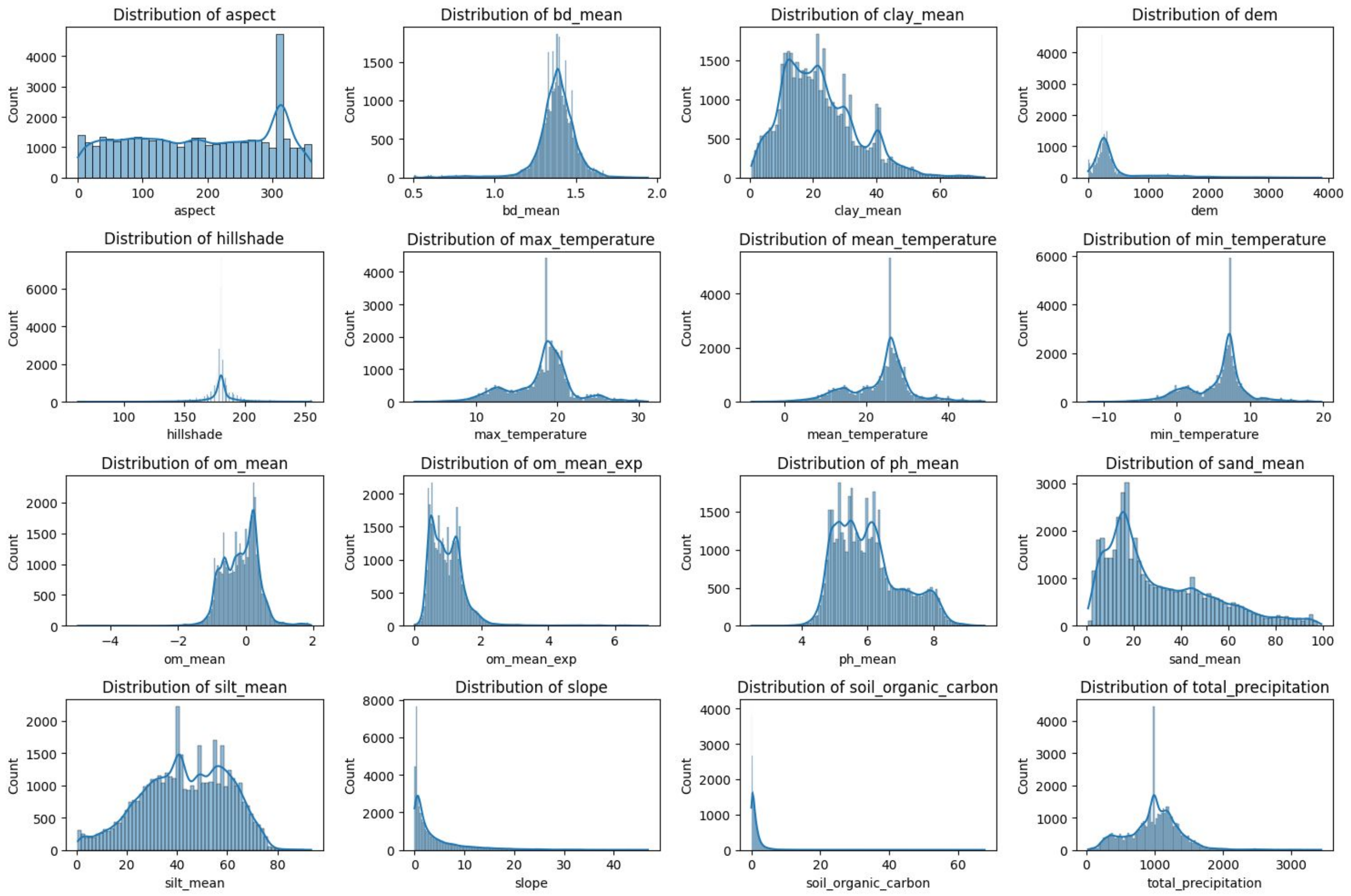
Data Description

Maria Oros

Summary of the Collected Data

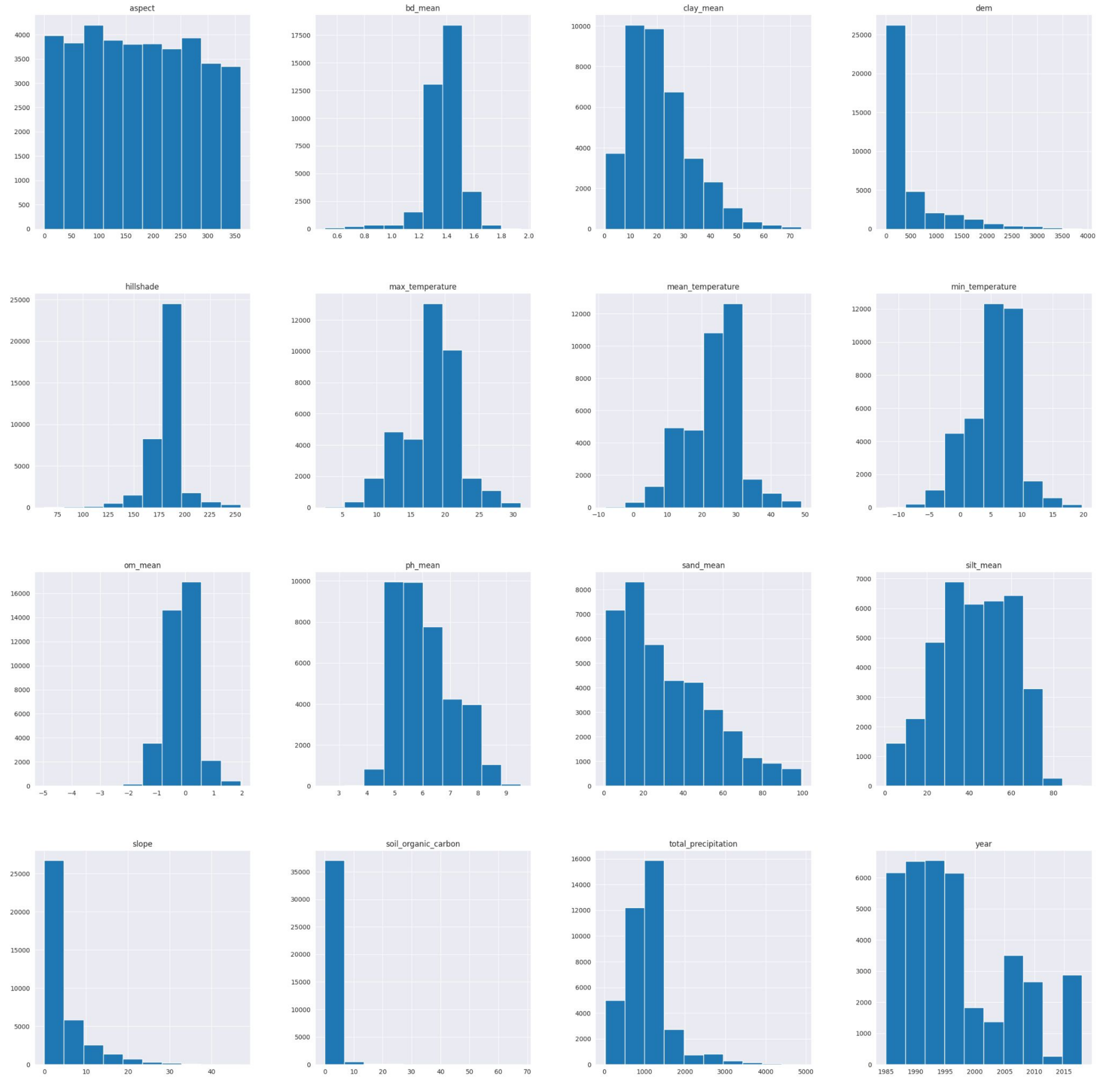
Category	Variables	Description	Units	Spatial Resolution
	silt	silt percentage	%	30m
	sand	sand percentage	%	30m
	clay	clay percentage	%	30m
	bd	bulk density	g/cm3	30m
	om	organic matter	log10(%)	30m
Soil	ph	soil pH in H20	N/A	30m
	Digital Elevation Model			30m
	Slope			30m
	Aspect		degrees	30m
Topography	Hillshade		degrees	30m
	mean temperature		C	1km x 1km
	max temperature		C	1kmx1km
	mean temperature		C	1km x 1km
Climate	Total Precipitation			1kmx1km

Distribution of the data - CONUS



A particular pick is observed in temperature and slope.

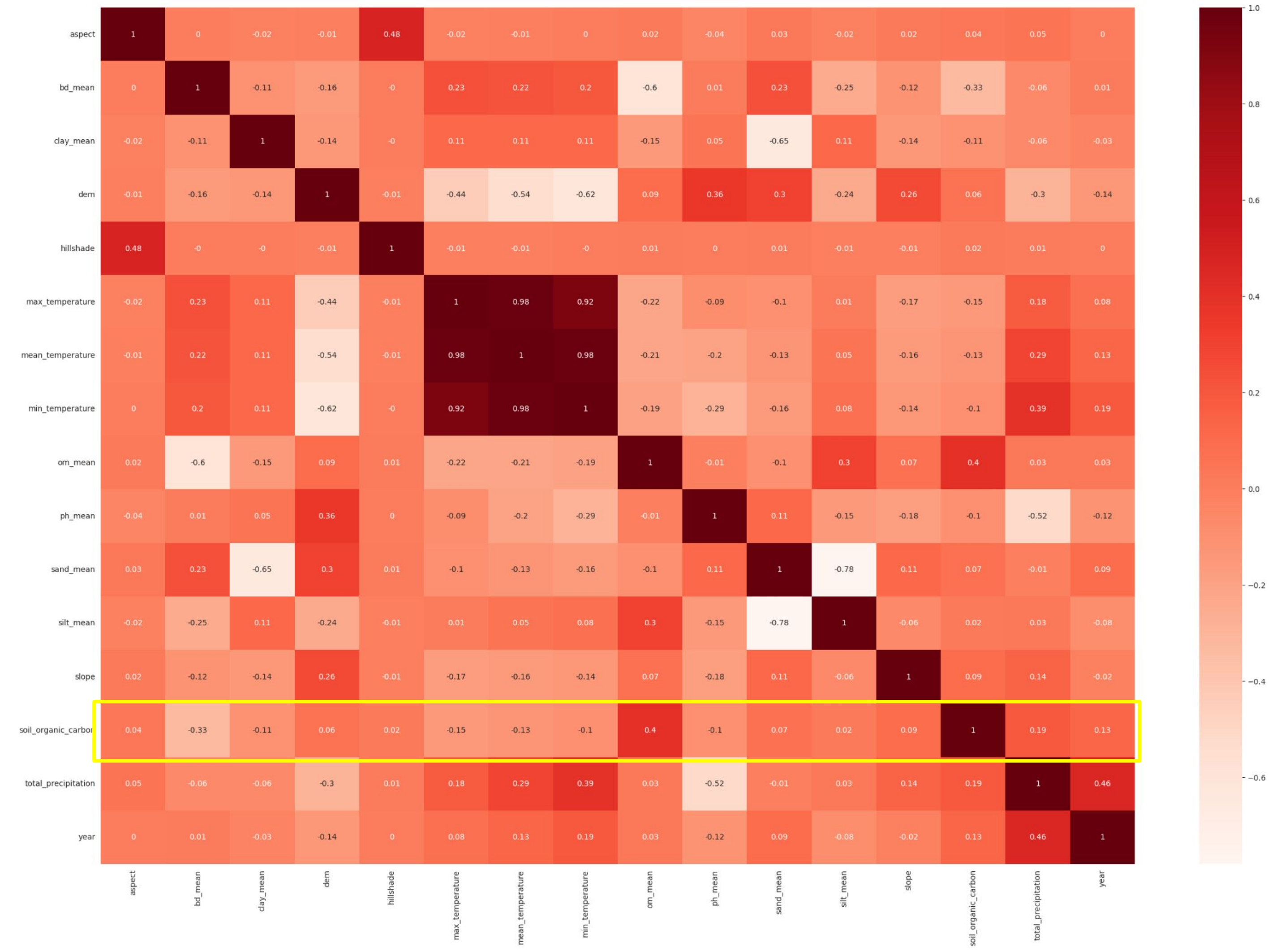
Data seems consistent.
Units:
Temperature - annual aggregations **°C**
Hillshade degrees 0-360°



Distribution of the data - CONUS

A particular pick is observed in temperature and slope.

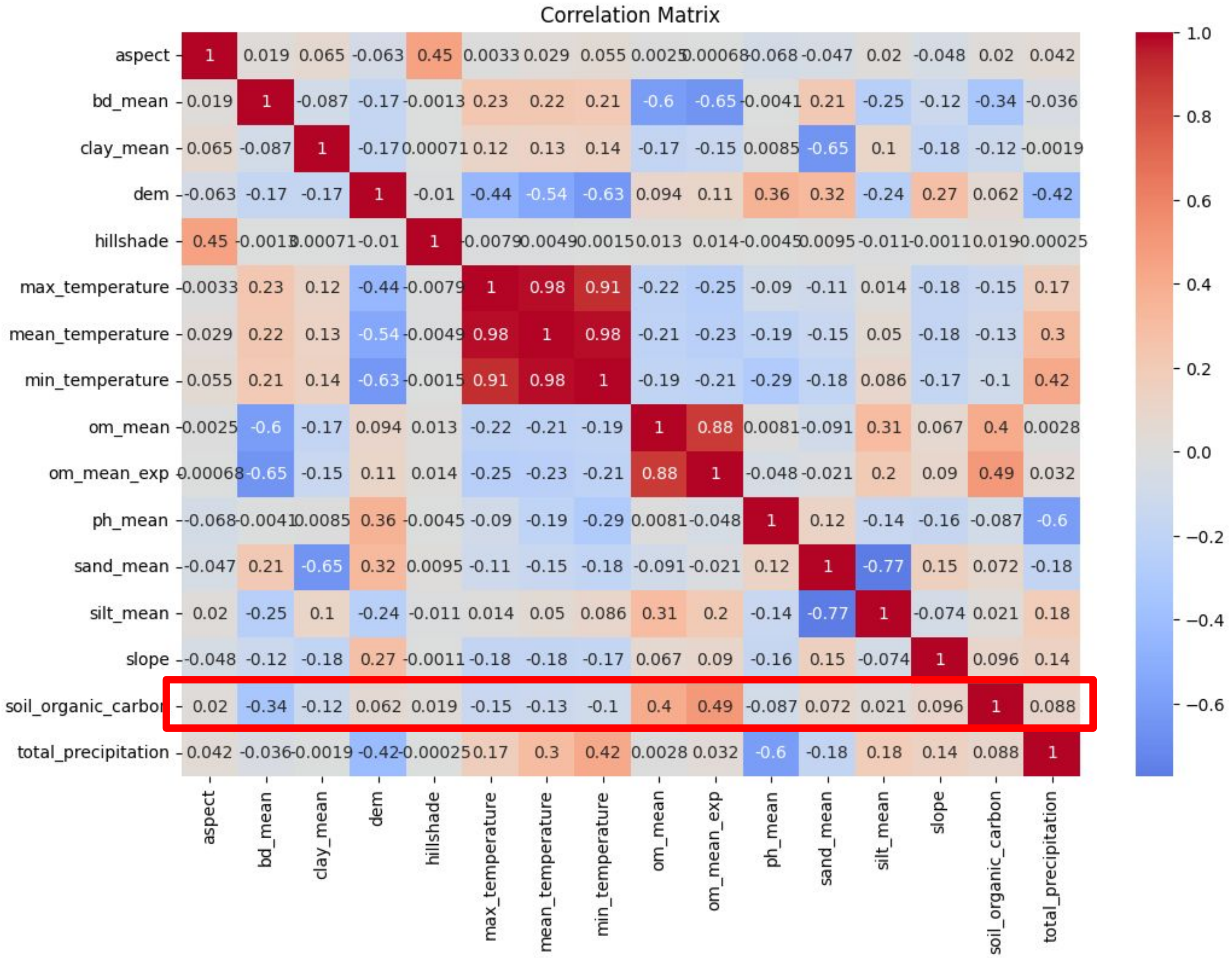
Data seems consistent.
Units:
Temperature - annual aggregations **°C**
Hillshade degrees 0-360**°**



	aspect	bd_mean	clay_mean	dem	hillshade	max_temperature	mean_temperature	min_temperature	om_mean	ph_mean	sand_mean	silt_mean	slope	soil_organic_carbon	total_precipitation	year
aspect	1	0	-0.02	-0.01	0.48	-0.02	-0.01	0	0.02	-0.04	0.03	-0.02	0.02	0.04	0.05	0
bd_mean	0	1	-0.11	-0.16	-0	0.23	0.22	0.2	-0.6	0.01	0.23	-0.25	-0.12	-0.33	-0.06	0.01
clay_mean	-0.02	-0.11	1	-0.14	-0	0.11	0.11	0.11	-0.15	0.05	-0.65	0.11	-0.14	-0.11	-0.06	-0.03
dem	-0.01	-0.16	-0.14	1	-0.01	-0.44	-0.54	-0.62	0.09	0.36	0.3	-0.24	0.26	0.06	-0.3	-0.14
hillshade	0.48	-0	-0	-0.01	1	-0.01	-0.01	-0	0.01	0	0.01	-0.01	-0.01	0.02	0.01	0
max_temperature	-0.02	0.23	0.11	-0.44	-0.01	1	0.98	0.92	-0.22	-0.09	-0.1	0.01	-0.17	-0.15	0.18	0.08
mean_temperature	-0.01	0.22	0.11	-0.54	-0.01	0.98	1	0.98	-0.21	-0.2	-0.13	0.05	-0.16	-0.13	0.29	0.13
min_temperature	0	0.2	0.11	-0.62	-0	0.92	0.98	1	-0.19	-0.29	-0.16	0.08	-0.14	-0.1	0.39	0.19
om_mean	0.02	-0.6	-0.15	0.09	0.01	-0.22	-0.21	-0.19	1	-0.01	-0.1	0.3	0.07	0.4	0.03	0.03
ph_mean	-0.04	0.01	0.05	0.36	0	-0.09	-0.2	-0.29	-0.01	1	0.11	-0.15	-0.18	-0.1	-0.52	-0.12
sand_mean	0.03	0.23	-0.65	0.3	0.01	-0.1	-0.13	-0.16	-0.1	0.11	1	-0.78	0.11	0.07	-0.01	0.09
silt_mean	-0.02	-0.25	0.11	-0.24	-0.01	0.01	0.05	0.08	0.3	-0.15	-0.78	1	-0.06	0.02	0.03	-0.08
slope	0.02	-0.12	-0.14	0.26	-0.01	-0.17	-0.16	-0.14	0.07	-0.18	0.11	-0.06	1	0.09	0.14	-0.02
soil_organic_carbor	0.04	-0.33	-0.11	0.06	0.02	-0.15	-0.13	-0.1	0.4	-0.1	0.07	0.02	0.09	1	0.19	0.13
total_precipitation	0.05	-0.06	-0.06	-0.3	0.01	0.18	0.29	0.39	0.03	-0.52	-0.01	0.03	0.14	0.19	1	0.46
year	0	0.01	-0.03	-0.14	0	0.08	0.13	0.19	0.03	-0.12	0.09	-0.08	-0.02	0.13	0.46	1

More correlated (absolute values) variables with soil organic carbon are:

	soil_organic_carbon
Bd_mean	0.33
Om_mean	0.40



	aspect	bd_mean	clay_mean	dem	hillshade	max_temperature	mean_temperature	min_temperature	om_mean	om_mean_exp	ph_mean	sand_mean	silt_mean	slope	soil_organic_carbon	total_precipitation
aspect	1	0.019	0.065	-0.063	0.45	0.0033	0.029	0.055	0.0025	0.00068	-0.068	-0.047	0.02	-0.048	0.02	0.042
bd_mean	0.019	1	-0.087	-0.17	-0.0013	0.23	0.22	0.21	-0.6	-0.65	-0.0041	0.21	-0.25	-0.12	-0.34	-0.036
clay_mean	0.065	-0.087	1	-0.17	0.00071	0.12	0.13	0.14	-0.17	-0.15	0.0085	-0.65	0.1	-0.18	-0.12	-0.0019
dem	-0.063	-0.17	-0.17	1	-0.01	-0.44	-0.54	-0.63	0.094	0.11	0.36	0.32	-0.24	0.27	0.062	-0.42
hillshade	0.45	-0.0013	0.00071	-0.01	1	-0.0079	-0.0049	-0.0015	0.013	0.014	-0.0045	0.0095	-0.011	-0.0011	0.019	-0.00025
max_temperature	0.0033	0.23	0.12	-0.44	-0.0079	1	0.98	0.91	-0.22	-0.25	-0.09	-0.11	0.014	-0.18	-0.15	0.17
mean_temperature	0.029	0.22	0.13	-0.54	-0.0049	0.98	1	0.98	-0.21	-0.23	-0.19	-0.15	0.05	-0.18	-0.13	0.3
min_temperature	0.055	0.21	0.14	-0.63	-0.0015	0.91	0.98	1	-0.19	-0.21	-0.29	-0.18	0.086	-0.17	-0.1	0.42
om_mean	0.0025	-0.6	-0.17	0.094	0.013	-0.22	-0.21	-0.19	1	0.88	0.0081	-0.091	0.31	0.067	0.4	0.0028
om_mean_exp	0.00068	-0.65	-0.15	0.11	0.014	-0.25	-0.23	-0.21	0.88	1	-0.048	-0.021	0.2	0.09	0.49	0.032
ph_mean	-0.068	-0.0041	0.0085	0.36	-0.0045	-0.09	-0.19	-0.29	0.0081	-0.048	1	0.12	-0.14	-0.16	-0.087	-0.6
sand_mean	-0.047	0.21	-0.65	0.32	0.0095	-0.11	-0.15	-0.18	-0.091	-0.021	0.12	1	-0.77	0.15	0.072	-0.18
silt_mean	0.02	-0.25	0.1	-0.24	-0.011	0.014	0.05	0.086	0.31	0.2	-0.14	-0.77	1	-0.074	0.021	0.18
slope	-0.048	-0.12	-0.18	0.27	-0.0011	-0.18	-0.18	-0.17	0.067	0.09	-0.16	0.15	-0.074	1	0.096	0.14
soil_organic_carbon	0.02	-0.34	-0.12	0.062	0.019	-0.15	-0.13	-0.1	0.4	0.49	-0.087	0.072	0.021	0.096	1	0.088
total_precipitation	0.042	-0.036	-0.0019	-0.42	-0.00025	0.17	0.3	0.42	0.0028	0.032	-0.6	-0.18	0.18	0.14	0.088	1

Correlations (>=.4):

- SOC
  - OM,OM exp
- Aspect
  - hillshade
- BD
  - OM (negative)
- Sand:
  - Clay (negative)
  - Silt (negative)
- Dem
  - Temperature
  - Precipitation
- Ph
  - Precipitation

SOC Prediction

A ML approach

Model Development

Train and test set definition:
The samples were randomly split based on soil profiles with 80% of the total profiles for calibration and 20% of the soil profiles for validation.

Variables:

- `'depth_cm',`
- `'total_precipitation', 'mean_temperature',`
- `'dem', 'slope', 'aspect', 'hillshade',`
- `'Bd_mean' (bulk density), 'clay_mean', 'om_mean', 'ph_mean', 'sand_mean'`
- `'land_use', 'land_change' (as dummy variables)`

Model Development

Model Selection:

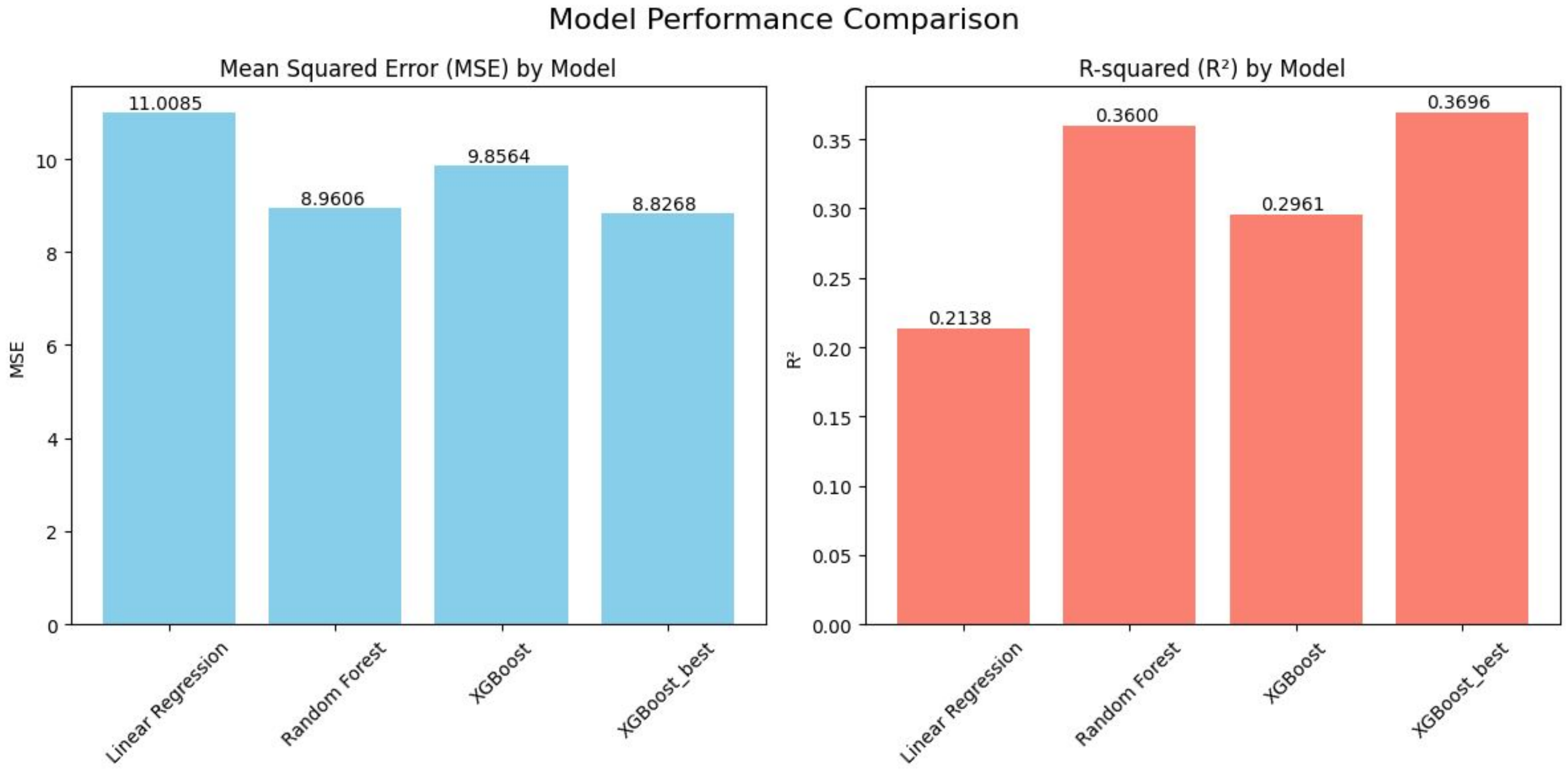
The model fitting was performed in python software. The sklearn library was used along with the train function to fit the model using the calibration dataset in 5 models:

- Linear Regression
- SV Regression was performed also but excluded given the low performance.
- Random Forest
- XGBoost
- XGBoost by parameter optimization using Grid Search.

Model Evaluation:

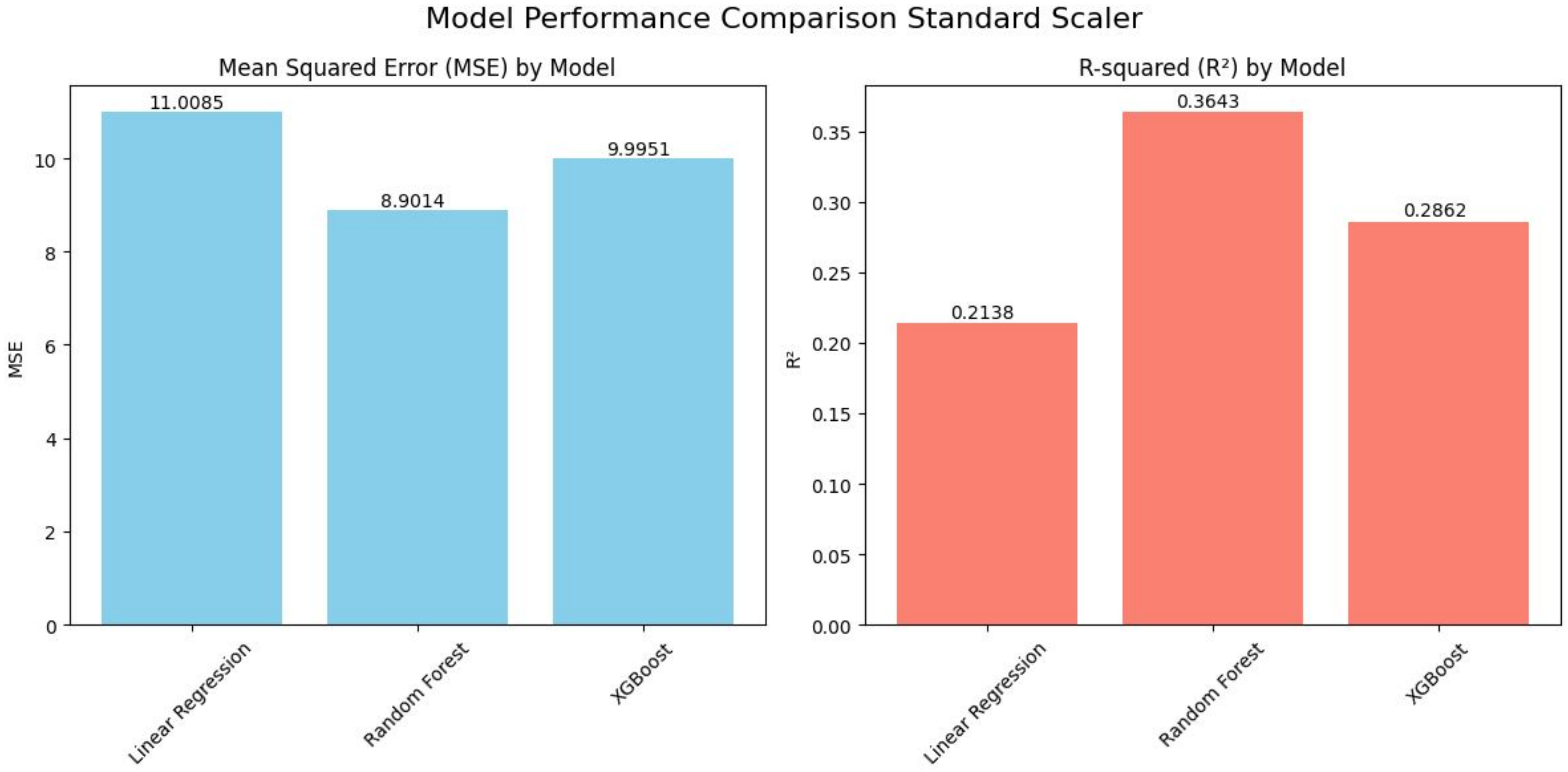
Both R2 and MSE were recorded on each case.

XGBoost_best emerged as the top-performing model based on both MSE and R2.
Random Forest also showed strong performance with a relatively low MSE and high R2.
Linear Regression and **XGBoost** had higher MSE values, indicating less accurate predictions.

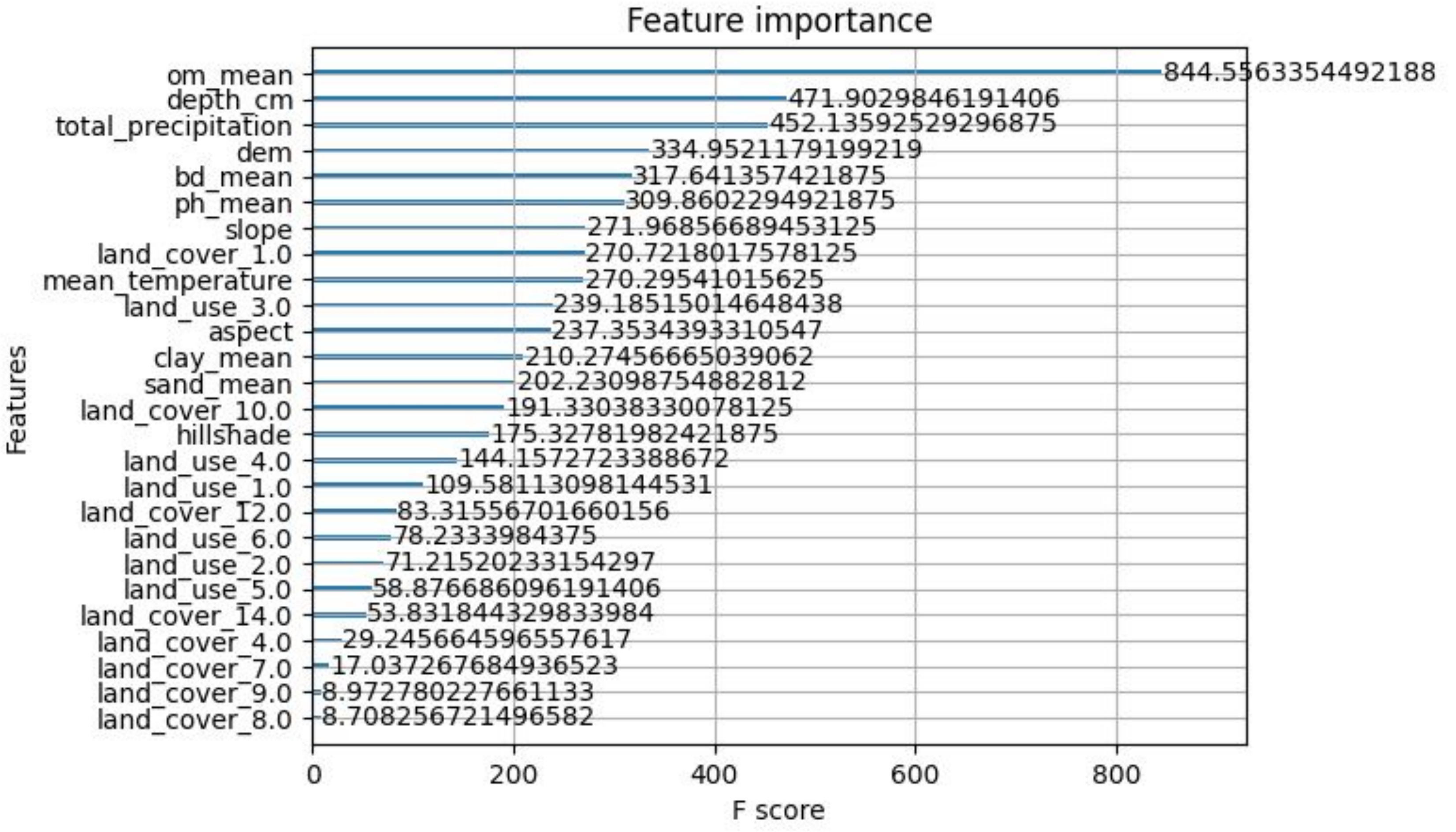


Standard Scaler

- **XGBoost_best** emerged as the top-performing model based on both MSE and R2.
- **Random Forest** also showed strong performance with a relatively low MSE and high R2.
- **Linear Regression** and **XGBoost** had higher MSE values, indicating less accurate predictions.
- dkmc.fdmclmds.cm.sdmc.dsc.l.c



Feature importance



Model Development

Takeover

The standard scaler was also evaluated but discarded as the performance was marginal compared with the -non standardized covariates- previous cases. Also by including standard scaler we would add a new complexity in the operationalization of the model as

The predictive model XGBoost performs adequately but there is still room for improvement in terms of incorporating additional data sources or mechanistic models by data assimilation.

The model was then operationalized by API to provide estimates in the contiguous United States.

Map in WI

August 28

Soil Organic Carbon Map Creation - Phase 1: the covariates

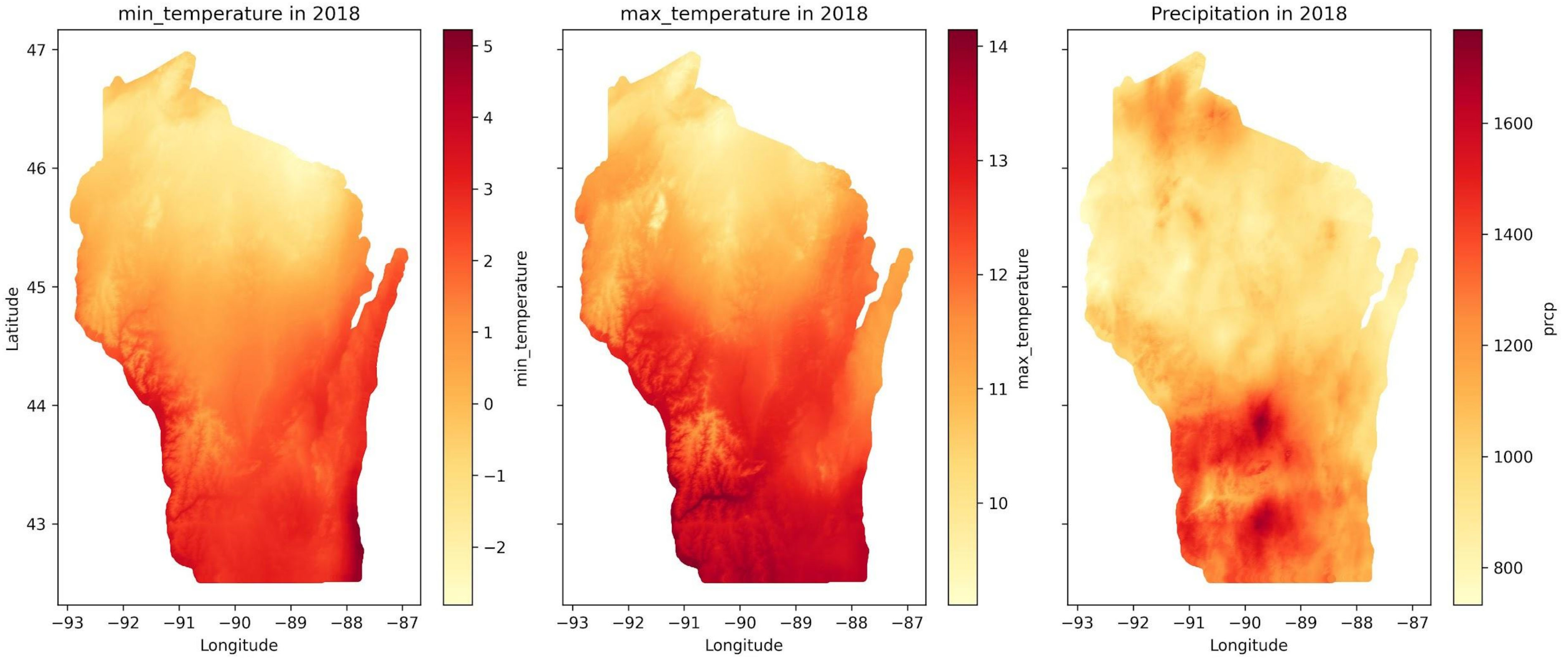
The Grid of 500mx500m - year 2018

To create a grid on soil organic carbon the next steps were performed:

- Build a grid in terms of latitude, longitude in the delimited area of WI (a total of 150K unique points)
- Each of those points were assigned to a soil id that represents the uniqueness of the latitude, longitude.
- Over the previous dataset, I have performed a collection of the covariates that play a role in the soil organic carbon model (slide 16).
    - Data set dimensions (27 variables because of the dummy variables in lu-lc, 150K rows)

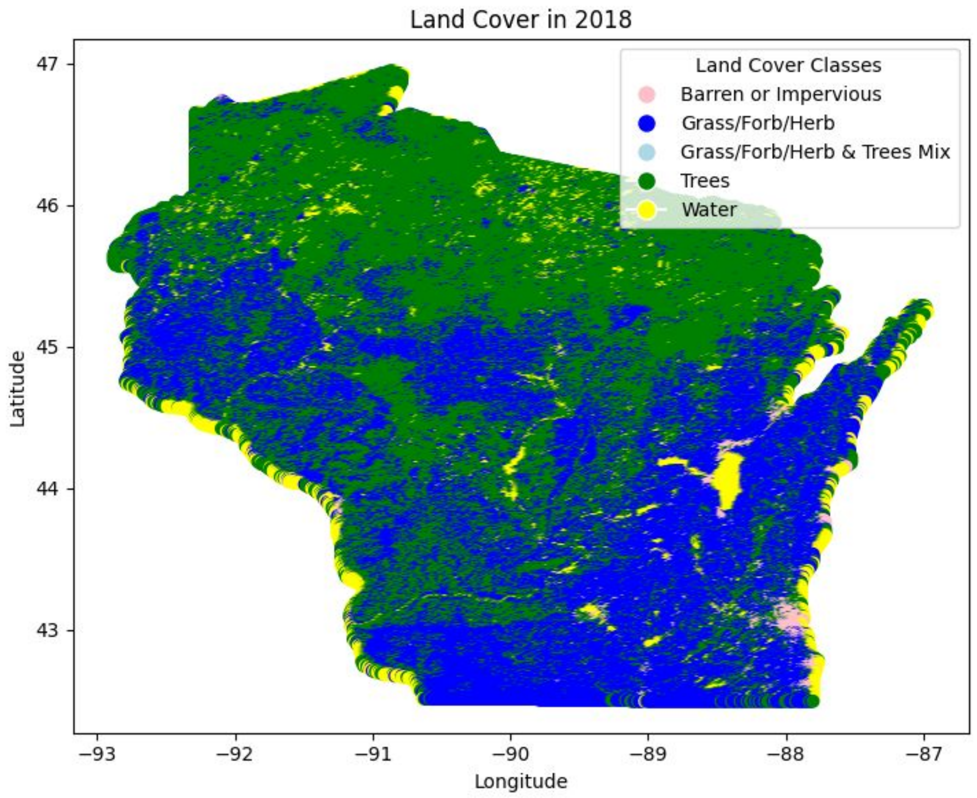
Dynamic Variables: Annual minimum and maximum temperature (C) and precipitation

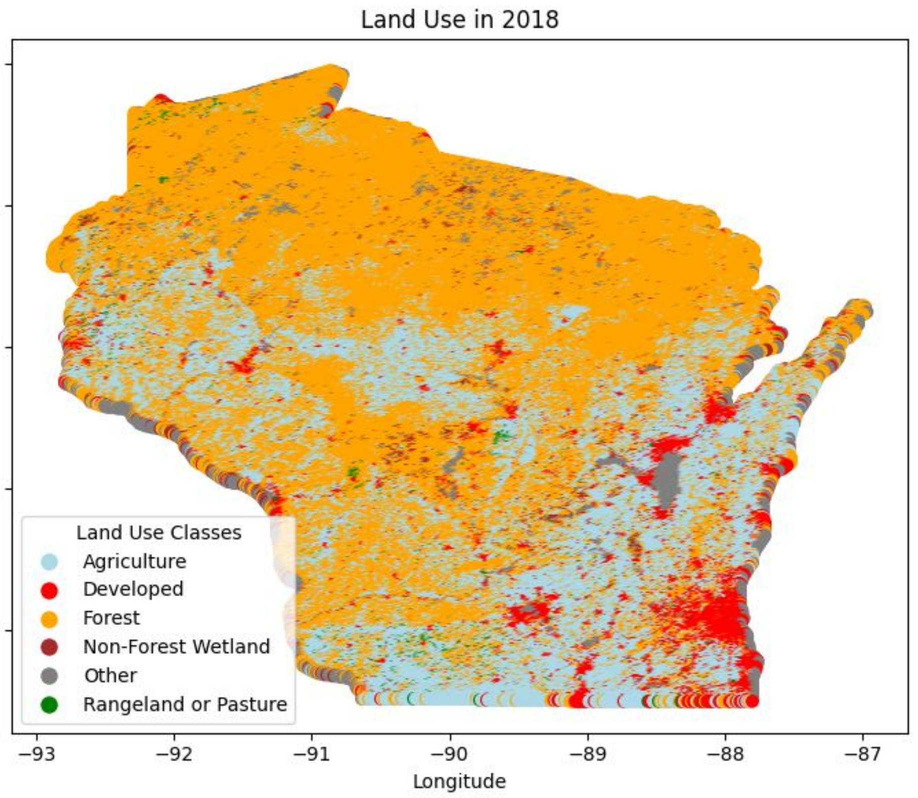
Maps in 2018 across Wisconsin, USA.



Dynamic Variables: Land Cover and Land Use

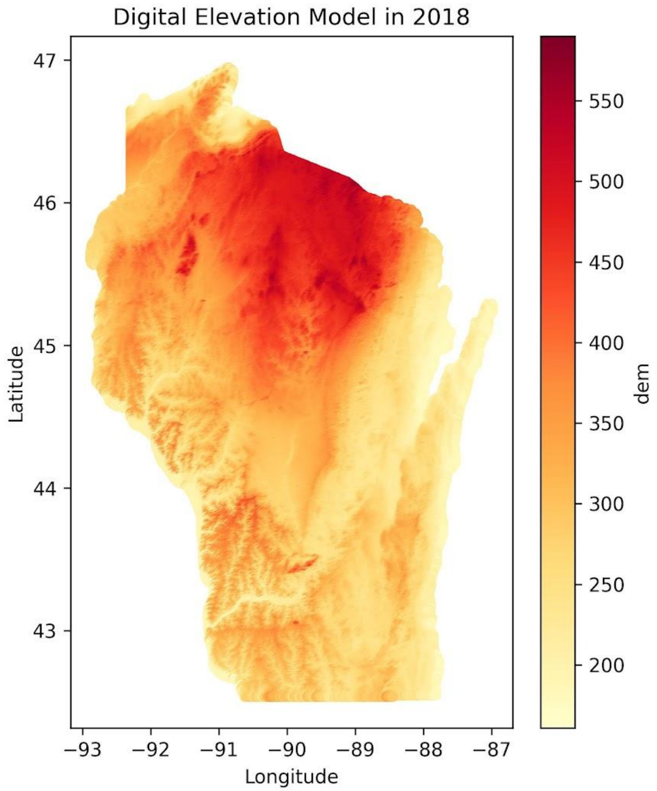
Maps in 2018 across Wisconsin, USA.

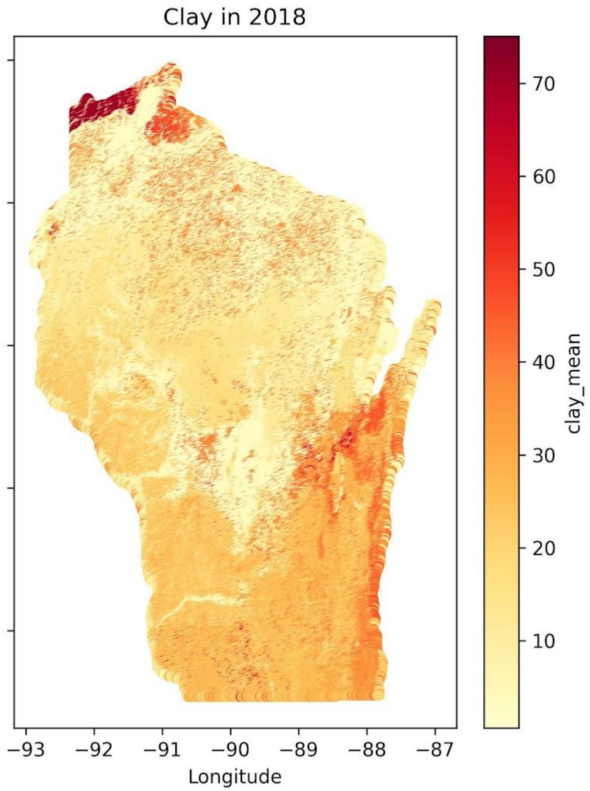


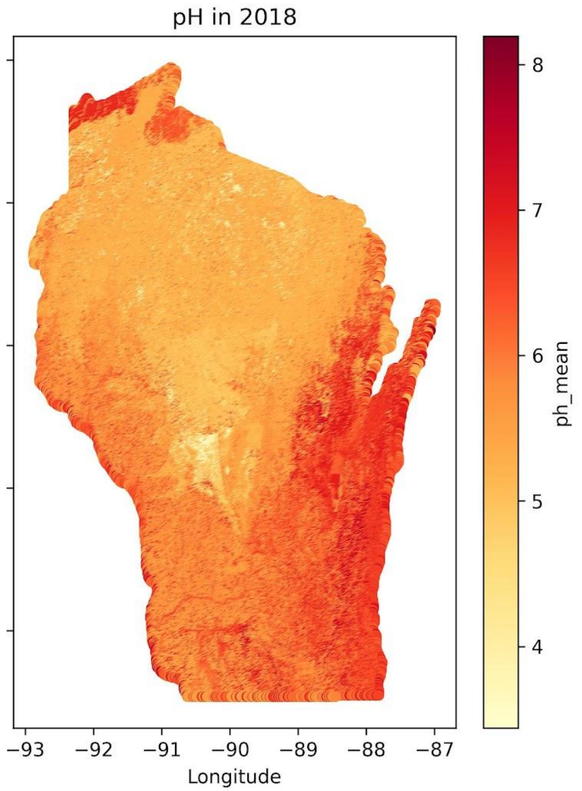


Probabilistic Variables

Maps of elevation (m), clay content (%) and pH at .30–.6 m across Wisconsin, USA.

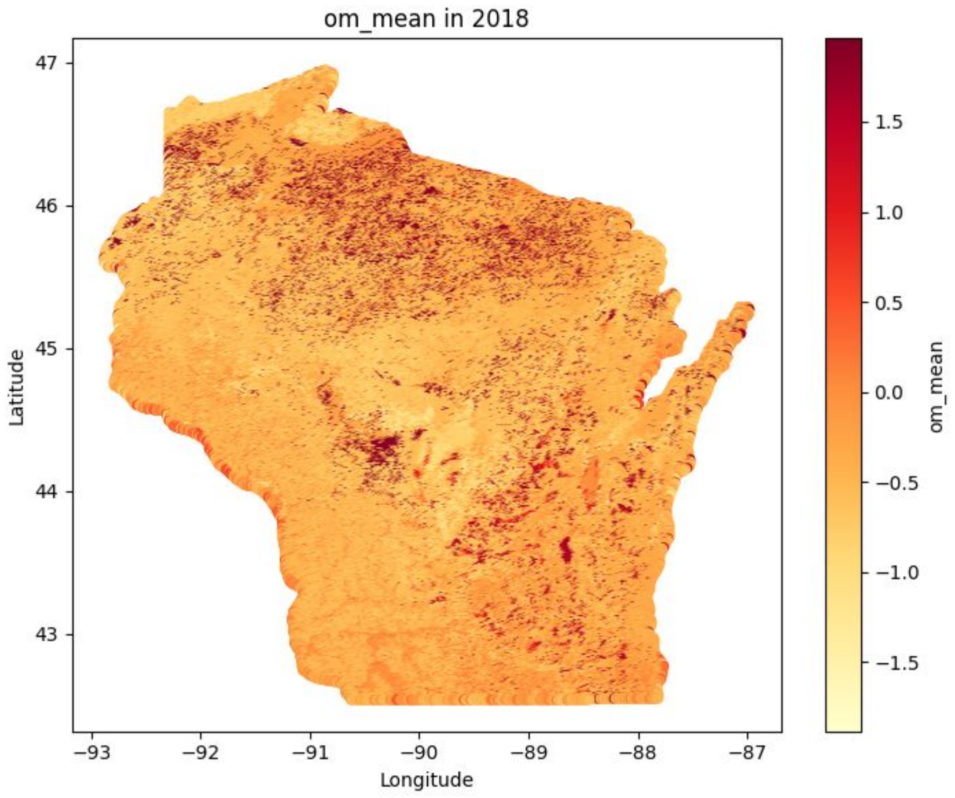


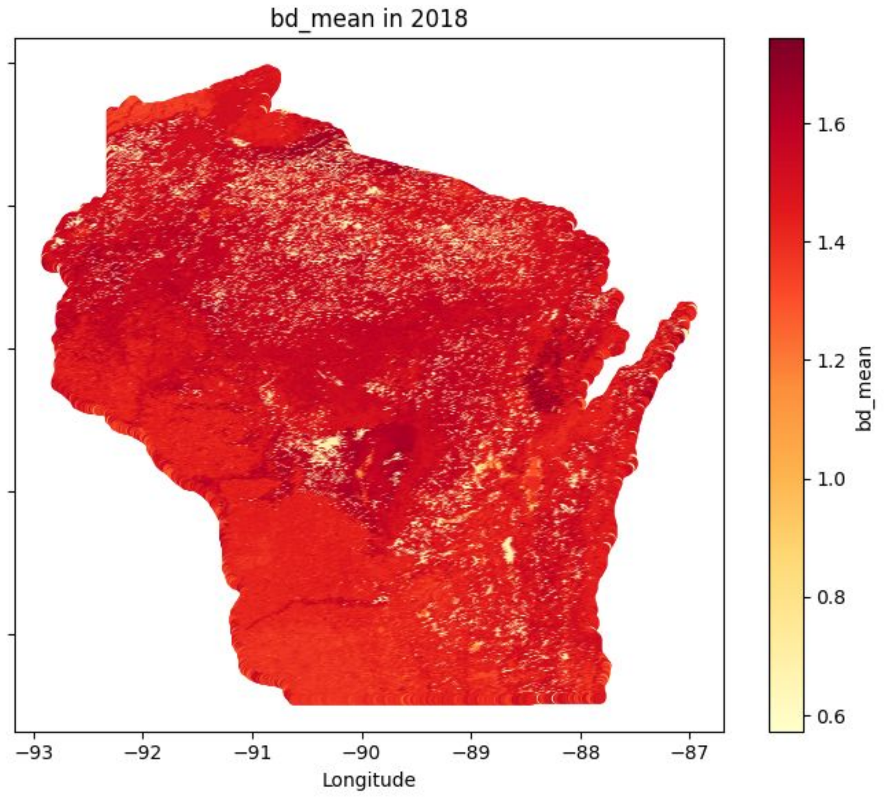




Organic Matter and bulk density

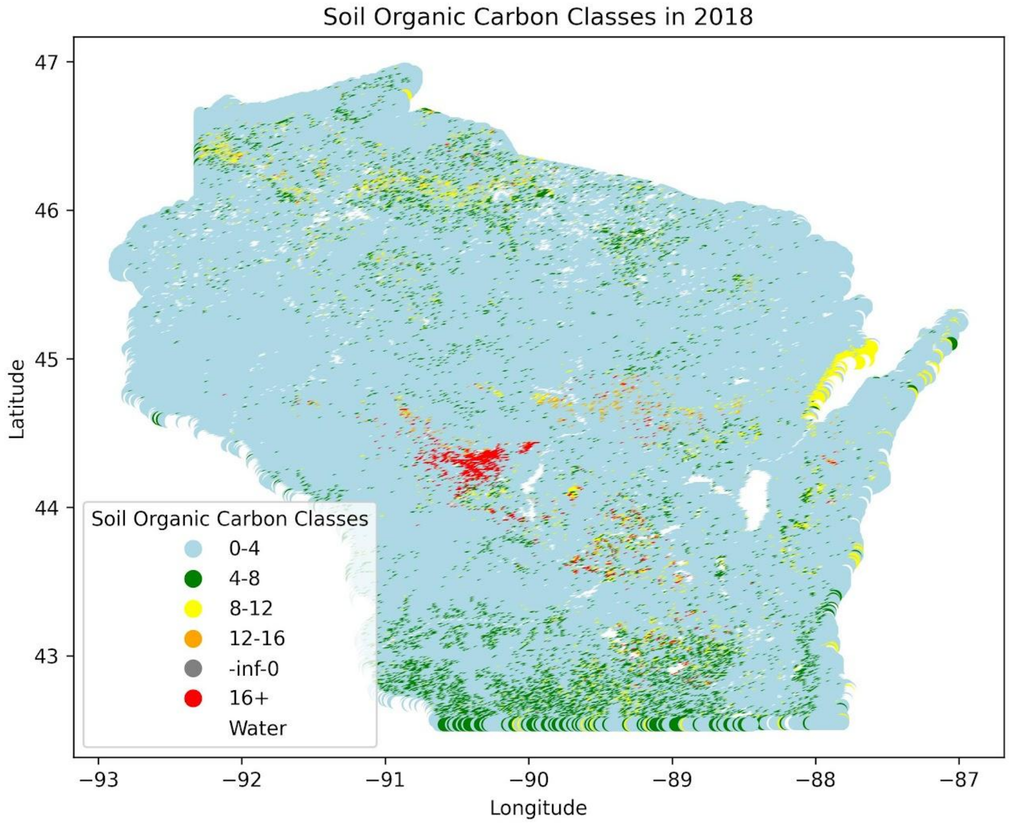
Maps to show the OM and BD as those are the variables with highest importance for soil organic carbon.





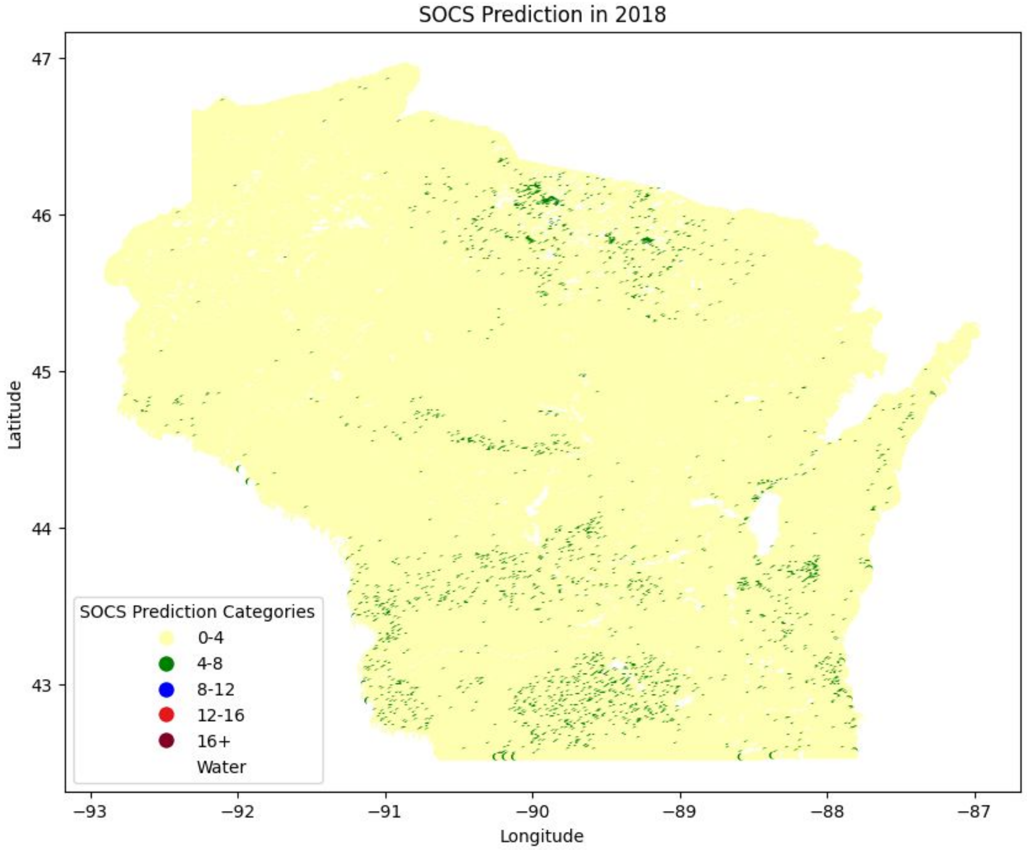
Soil organic carbon prediction in Wisconsin 2018

The SOC concentration was predicted using a XGBoost model and it had the results described in slides 18 and 19. Although the XGBoost model was accurate for the calibration data, it performed moderately based on the validation dataset.



Soil organic carbon stocks in Wisconsin 2018 (.3 - .6m)

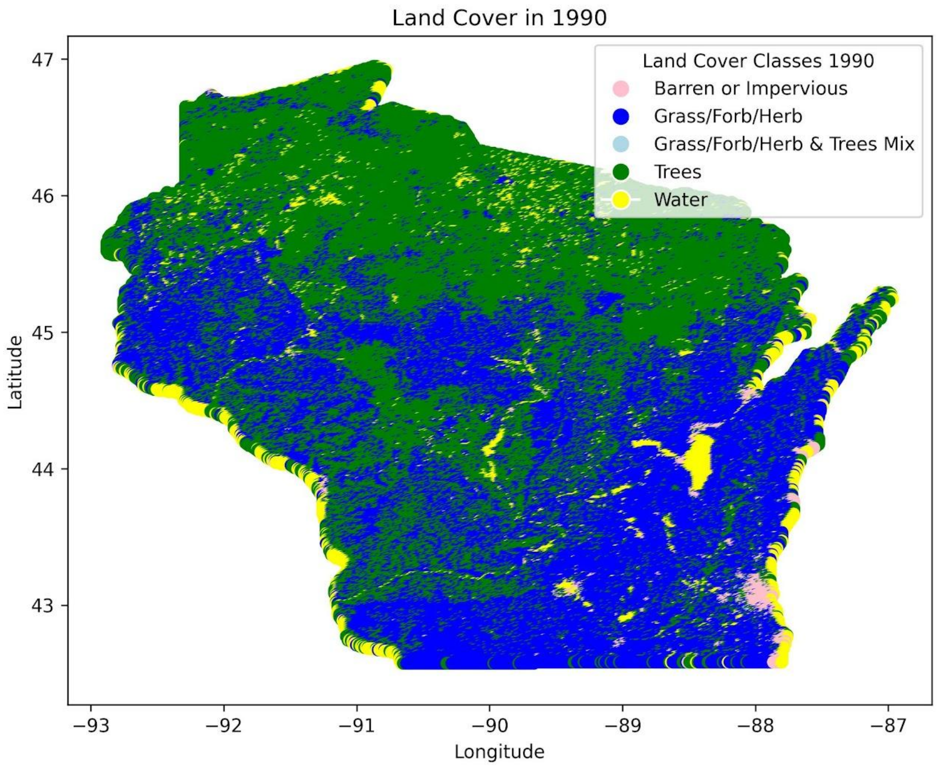
The SOC stocks are computed in terms of the predicted: soil organic carbon * 30 cm * bulk density.

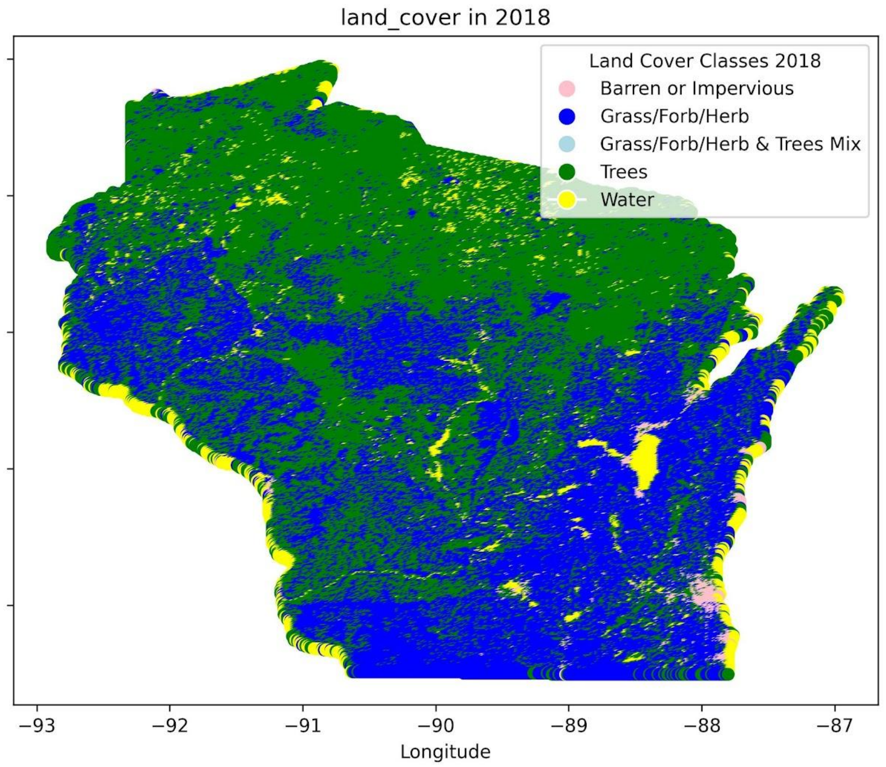


Comparing land use and land change in 1990 vs 2018

Wisconsin

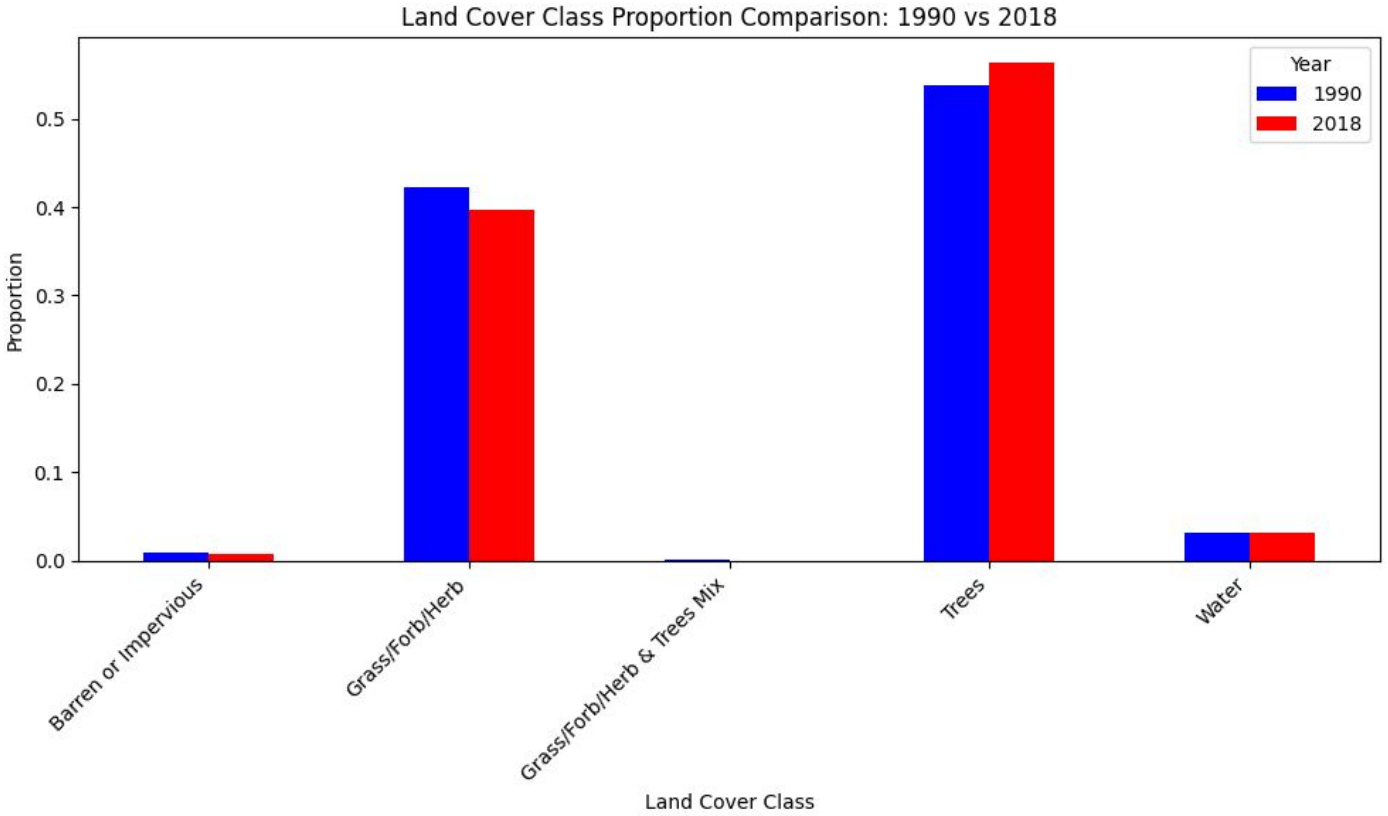
Dynamic Variables: Land Cover 1990 vs 2018





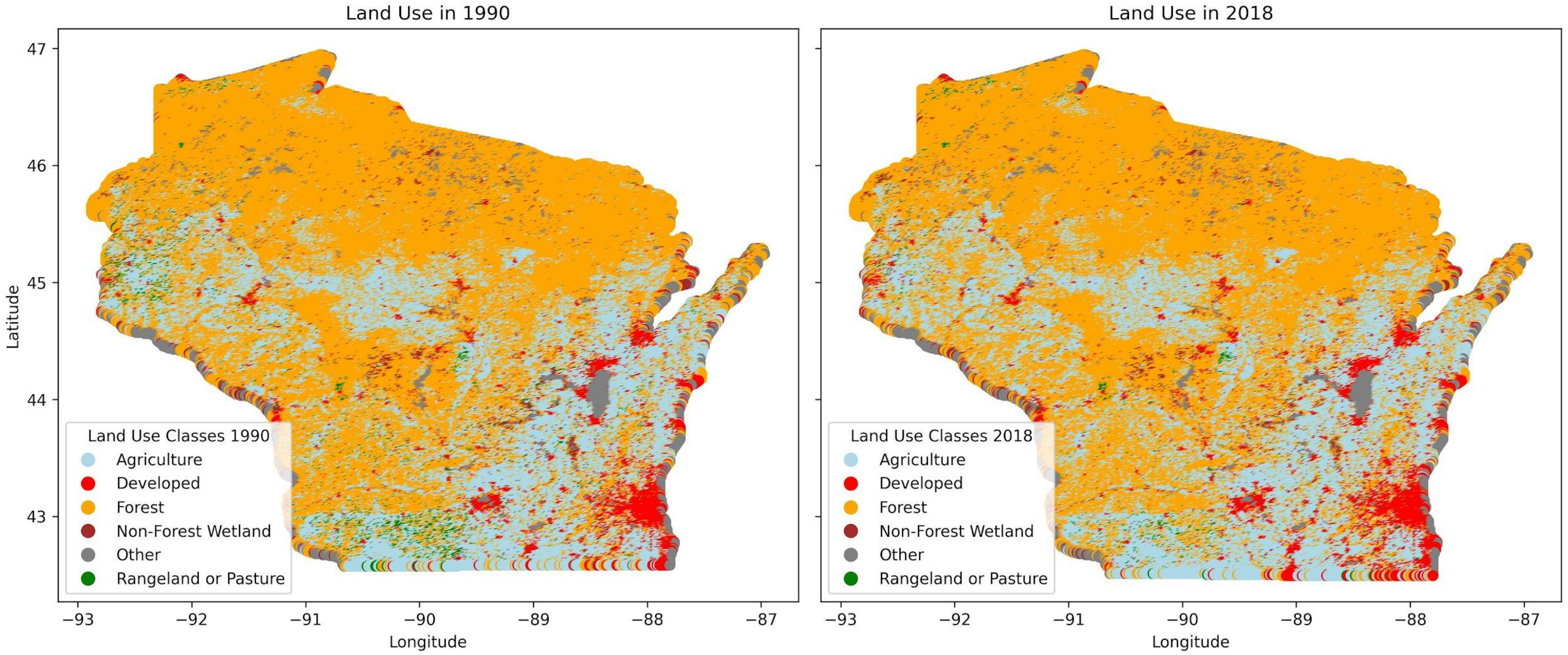
Land Cover Proportion, comparing 1990 vs 2018

Proportions are made relative to the number of observations within the year



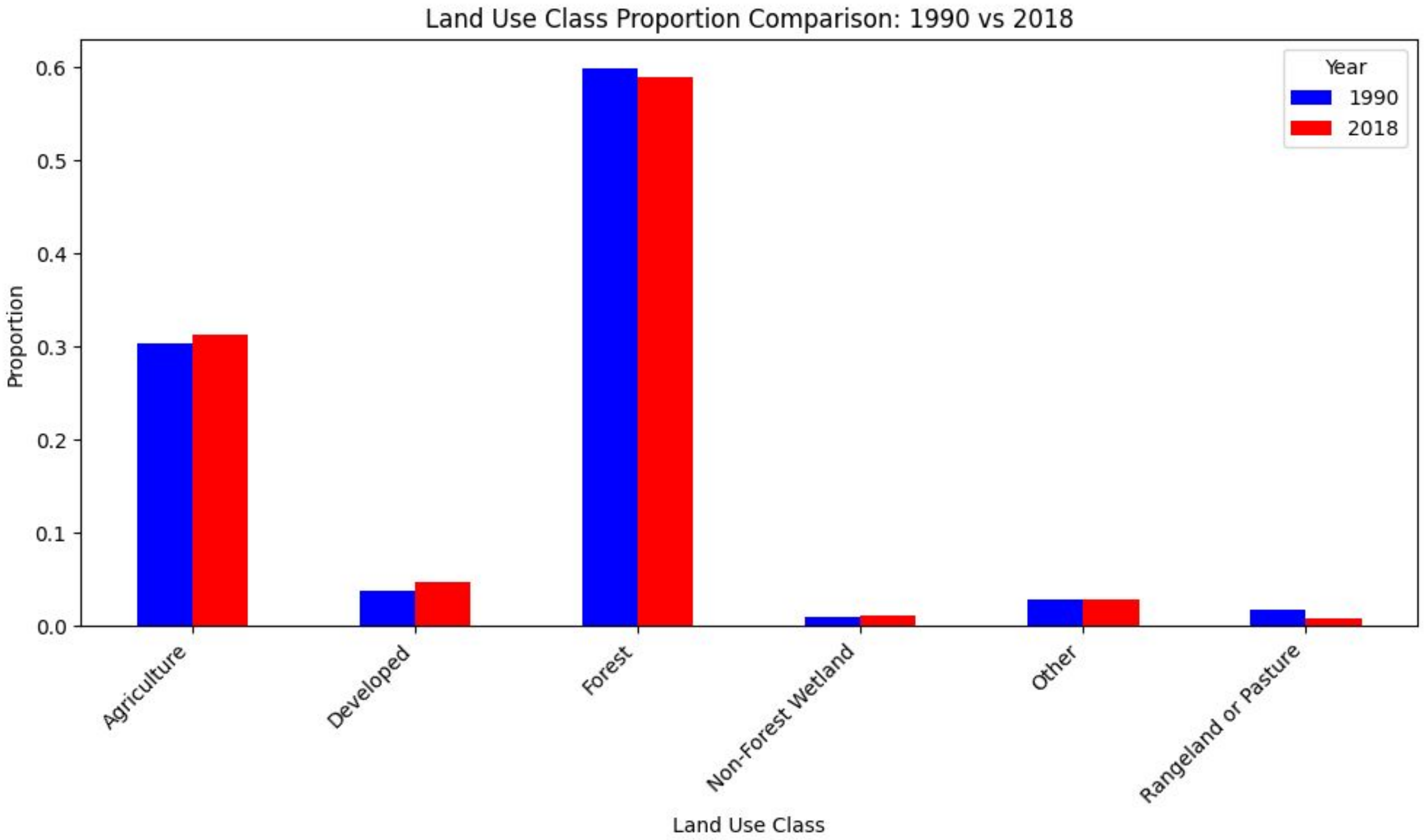
- More trees are observed from 1990 to 2018 (as proportional to LC observations within the year)
-

Dynamic Variables: Land Use 1990 vs 2018



Land Use Proportion, comparing 1990 vs 2018

Proportions are made relative to the number of observations within the year



Cross table LU & LC - 2018

land_cover_class	Agriculture	Developed	Forest	Non-Forest Wetland	Other	Rangeland or Pasture
Barren or Impervious	15	1152	5	0	3	11
Grass/Forb/Herb	45065	3601	6887	1193	44	1214
Grass/Forb/Herb & Trees Mix	0	21	0	0	0	0
Trees	710	2171	79031	226	139	15
Water	13	82	166	242	3985	1

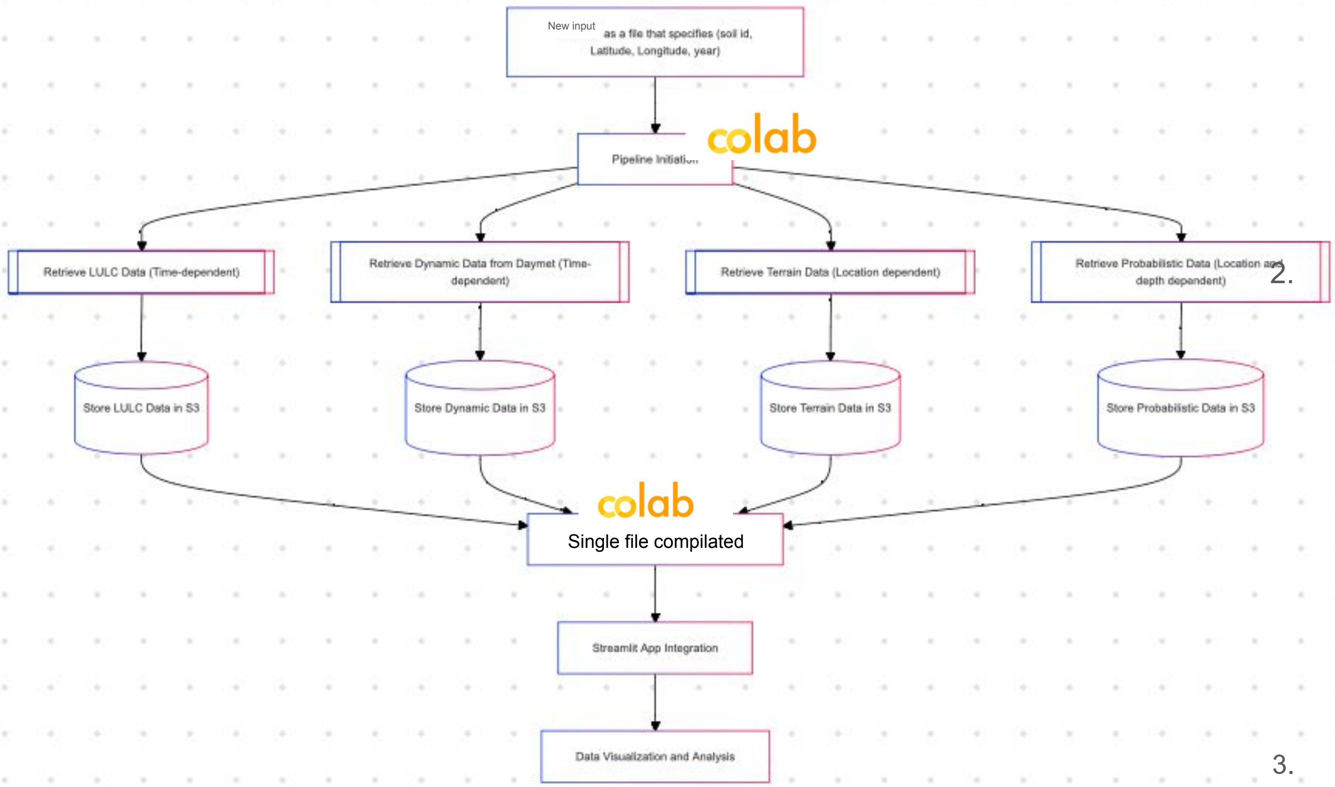
Cross table LU & LC - 1990

land_cover_class	Agriculture	Developed	Forest	Non-Forest Wetland	Other	Rangeland or Pasture
Barren or Impervious	2	1274	2	0	4	6
Grass/Forb/Herb	43784	3884	10899	1064	68	1137
Grass/Forb/Herb & Trees Mix	0	39	0	0	0	0
Trees	662	1434	74709	311	166	73
Water	8	92	215	280	3887	0

The Workflow

Cyber-Infraestructure

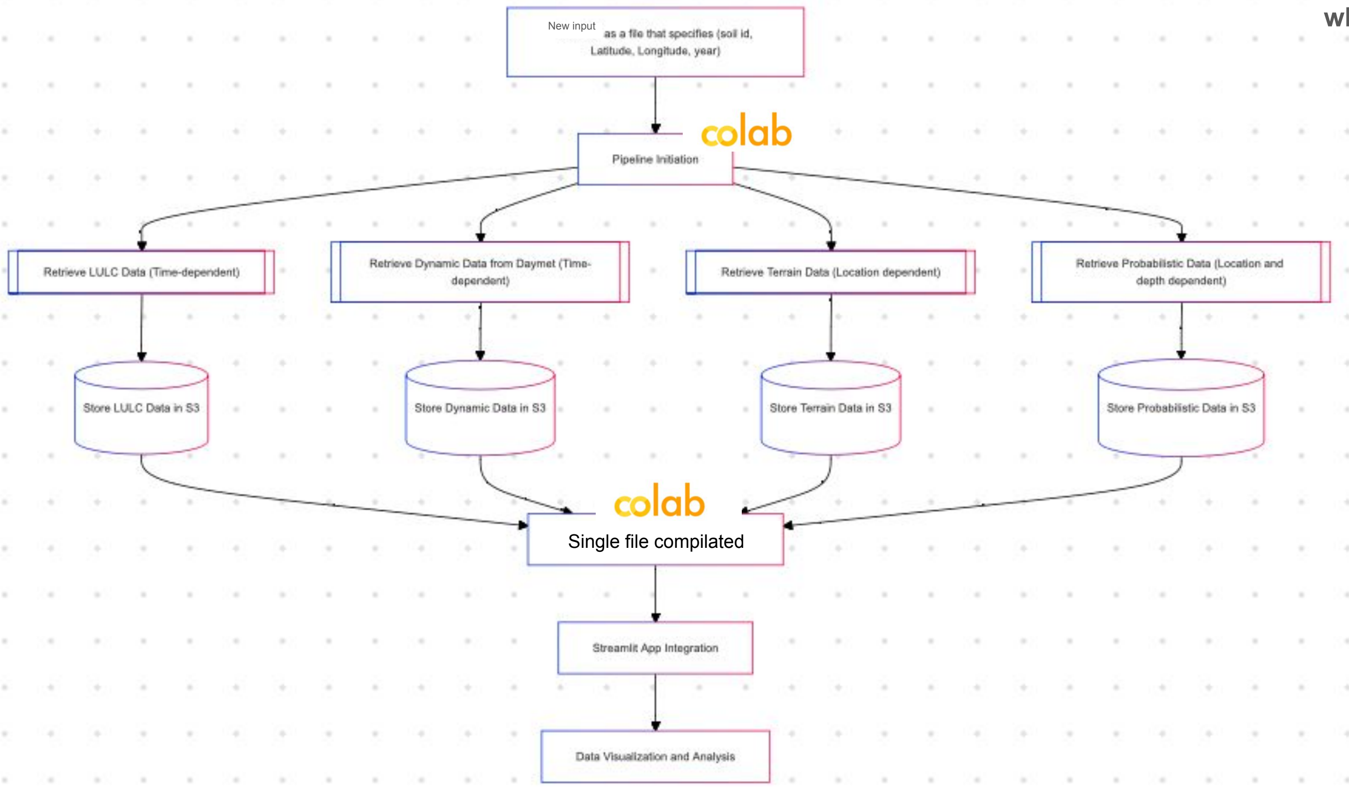
Workflow of data integration and App Visualization


New input as a file that specifies (soil id, Latitude, Longitude, year)
colab
Pipeline Initiati...
Retrieve LULC Data (Time-dependent)
Retrieve Dynamic Data from Daymet (Time-dependent)
Retrieve Terrain Data (Location dependent)
Retrieve Probabilistic Data (Location and depth dependent)
Store LULC Data in S3
Store Dynamic Data in S3
Store Terrain Data in S3
Store Probabilistic Data in S3
colab
Single file compilated
Streamlit App Integration
Data Visualization and Analysis

New input refers to one of the next options:

1. New soc observations eg a file (csv, excel, parquet) that specifies latitude, longitude, year and depth of soil, soil_organic_carbon
   a. Soil_organic_carbon belongs to the range [0,100]
   b. Latitude, longitude belongs to the CONUS
   c. Year > 1986
   d. Depth of soil is one of the values [5, 15, 30, 60, 100, 200]
2. New grid of an AOI eg a file (csv, excel, parquet) that specifies latitude, longitude, year
   a. If the AOI is not precomputed eg is not WI, proceed.
   b. Otherwise, if the AOI is WI or subarea of WI, and the year is one of the pre-computed, it won't proceed.
   c. If the AOI is WI or subarea of WI, and the year is NOT one of the pre-computed, then proceed to compute LULC and Dynamic for such year on an existing grid of WI.
3. Can be extended to the compilation of new sources of information or new input types later

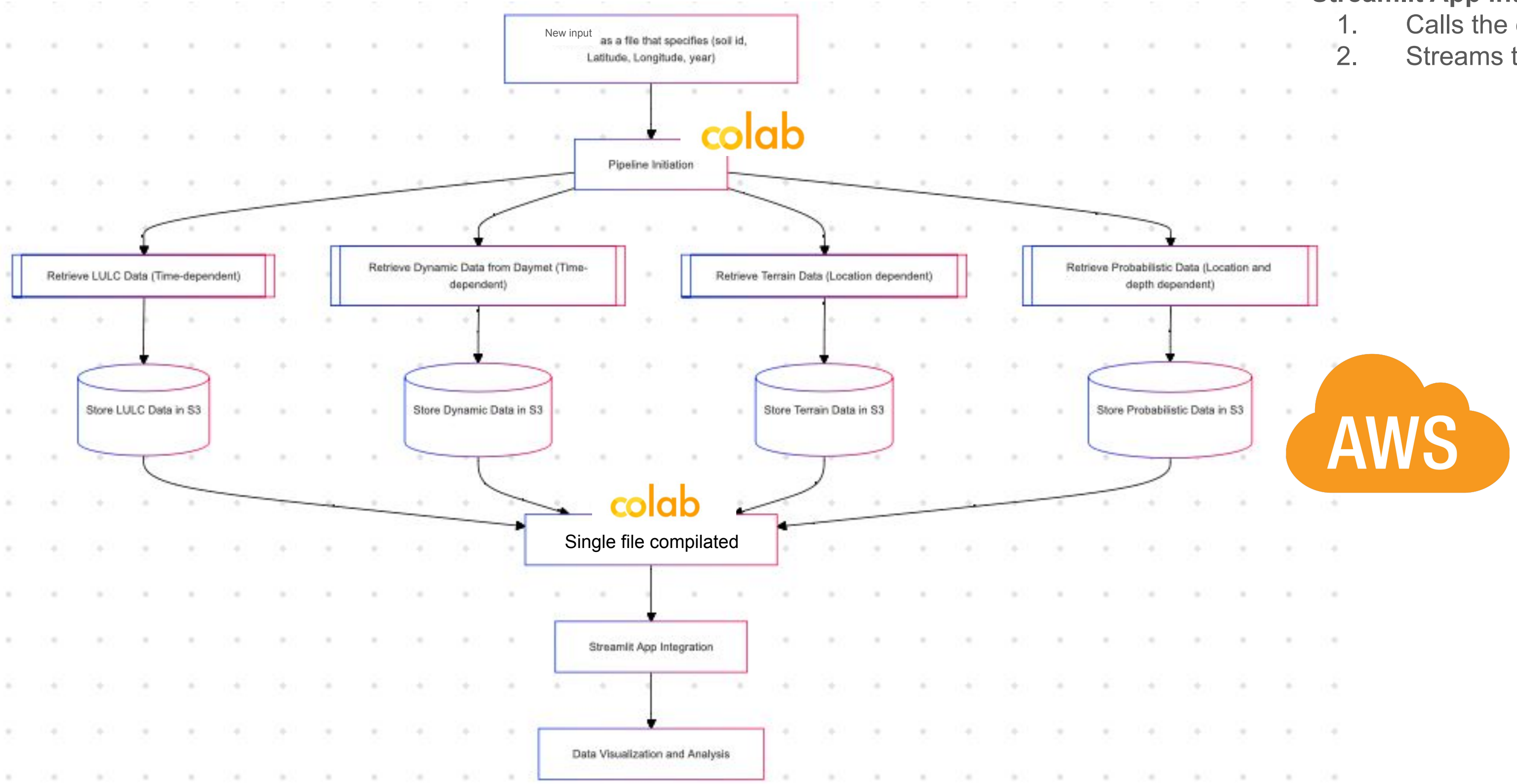
Workflow of data integration and App Visualization


New input as a file that specifies (soil id, Latitude, Longitude, year)
colab
Pipeline Initiation
Retrieve LULC Data (Time-dependent)
Retrieve Dynamic Data from Daymet (Time-dependent)
Retrieve Terrain Data (Location dependent)
Retrieve Probabilistic Data (Location and depth dependent)
Store LULC Data in S3
Store Dynamic Data in S3
Store Terrain Data in S3
Store Probabilistic Data in S3
colab
Single file compilated
Streamlit App Integration
Data Visualization and Analysis

Pipeline initiation is a jupyter notebook for which a new input needs to be specified.

1. If the input is soc_observations the new dataset in s3 will be defined in terms of the AOI following the convention: **[state]_[year]_[process]_soc**
2. If the input is a new grid it will be specified in terms of the AOI following the convention **[state]_[year]_[process]_[meters]grid**
3. The pipeline computes the data by assuming a soil_id. The soil id for a grid is computed in terms of the row order at the input, so then the file remains on such soil id before going into any of the process. The soil_id is fundamental to keep the relation one to one, or one to many when computing new fields. For example the Probabilistic data is computed in one or many soil depths, which means that the soil id will appear one or many times and needs to keep a one to many relationship.
4. Once the datasets are saved in S3, the process to merged them all and predict soil organic carbon on the grids.
5. For new inputs on soil_organic_carbon, it does not retrain the model now.

Workflow of data integration and App Visualization



New input as a file that specifies (soil id, Latitude, Longitude, year)
colab
Pipeline Initiation
Retrieve LULC Data (Time-dependent)
Retrieve Dynamic Data from Daymet (Time-dependent)
Retrieve Terrain Data (Location dependent)
Retrieve Probabilistic Data (Location and depth dependent)
Store LULC Data in S3
Store Dynamic Data in S3
Store Terrain Data in S3
Store Probabilistic Data in S3
AWS
colab
Single file compilated
Streamlit App Integration
Data Visualization and Analysis

Streamlit App integration

1. Calls the compilated datasets
2. Streams the geographical functionalities