**There are two stages in this project.**

Stage 1 – Collecting satellite images and DEM data and building a postgreSQL database. Clustering analysis were done based on pixel brightness and DEM.

Stage 2 – Using the database and supervised learning, the model processes the soil delineation of new locations

Stage 1

**Step 0: Select Location**

1. Locations were selected by using geojson.io webpage.
   1. Only Rectangular images will work
2. Selected 10 random locations and created “##.geojson” files from the website
3. Variable site\_num will need to be manually assigned when running the jupyter notebook. (from 1 to 10)

**Step 1: Load image data**

1. I used your API to download color bands, but filter out images with too many NaNs and images with excessive clouds.
2. To detect images with clouds, I used standard deviation of ‘near infrared’ and ‘red’ color bands. I used clustering analysis to group abnormal images and excluded them
3. After the images were selected, I normalized all images into same 0~1 scale and average them.
   1. I even used the image with full grown crops because crop healthiness shown on WDRVI images could be an indirect indication of soil organic matter contents

**Step 2: Load SSURGO data**

1. Downloaded the soil information and geographical coordinates of polygons using your API.

**Step 3: Load DEM**

1. Downloaded DEM using your API.
2. Converted lat/long coordinates to actual distance in meters to visualize the soil polygons in orthogonal coordinates.
3. Since DEMs usually come in coarser pixel resolution, I change them in the same dimension as satellite images using 2D interpolation
4. Calculated slope based on the DEM, but there are too much noises and was not included in the final feature selection.
5. Finalized feature selection (seven feature matrix)

**Step 4: Feature scaling**

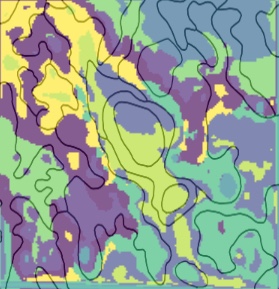
1. Using MinMaxScaler, features in the matrix are on the same numerical scale of 0 ~ 1
2. PCA was performed to reduce the number of features, but decided not to use principal components. All seven features will be used for clustering.

**Step 5: Clustering analysis**

1. Clustering analysis were done on the feature matrix to group similar pixels. The number of clusters were assigned as the number of unique areas with different om\_r values from SSURGO plus 3. For example, if the area has 10 unique organic matter values found in SSURGO database, then the clustering analysis will be done with 13 clusters pre-defined.
2. I first used Silhouette score to measure the correct number of clusters in one site but this step is merely to provide a means of labeling dataset for supervised learning in the later stage, so I fixed the number of clusters and it turned out that number of clusters in a location is not very critical for the final product.

**Step 6: Prepare training dataset for Supervised Learning (Stage 2)**

1. The clustered pixels look similar to the SSURGO polygon shapes, but they do NOT have meaningful SOM values. Therefore, a connection between these pixels and actual book values need to be made.



1. As seen in the figure above, new clustered pixels were overlaid with SSURGO polygon. For each polygon, I selected the most popular cluster labels on the polygon and assigned the SSURGO published value.
2. I repeated these process with other location – changing ‘site\_num’ variable in the very first code block – to increase the number of labeled pixels to be used in supervised learning.

**Step 7: Save training dataset to SQL table**

1. The dataset is created in postgreSQL database.
2. You need to change the username and password for your application.

Stage 2

First part of this stage is same as Stage 1: go to the website geojson.io and select a new location, then save the coordinates in json format.

**Step 1: Load image data**

1. Load the image data in ‘geojson’ format

**Step 2: same as in Stage 1**

**Step 3: same as in Stage 1**

**Step 4: : same as in Stage 1**

**Step 5: Supervised learning**

1. Read SQL database: read the database created from Stage 1
2. Split train/test dataset using sci-kit learn’s cross validation module
3. Establish performance metric: mean squared error
4. Fit a regressor model: Decision tree regressor was used
5. Check model parameters
   1. Learning curves were drawn to prevent model bias based on one of the regressor’s parameter: max depth in growing trees
   2. Complexity graph were drawn to prevent model variance and select just right max\_depth parameter value
6. Predict the soil organic matter on a test site
   1. Results are in a separate folder
7. Plotting final results: NEW SOIL MAP
   1. Plotting new soil map on jupyter notebook

**Example Soil maps**

Each pair are in same color scale

