CSCI 5980/8980: Spatial Enabled Artificial Intelligence

Assignment 2 (10 points)

Due Date: 2022/03/08 11:59:00 CST

1. Overview of the Assignment

In this assignment, you will learn to use machine learning methods to support real-world spatial artificial intelligence tasks. You will implement the approach proposed in the paper "Mining Public Datasets for Modeling Intra-City PM2.5 Concentrations at a Fine Spatial Resolution" [Lin et al. 2017]. The approach aims to generate fine-grained air quality prediction over a region using sensor observations ($PM_{2.5}$ concentrations) and geographic feature data. You will apply clustering, classification, and regression techniques to solve the problem. You can access the data on Google Drive.

2. Programming Requirements and Environment Settings

a. You must use **Python** to implement all tasks.

b. Programming Environment: Python 3.7 (you can use the same environment as Assignment 1)

c. You need to download and use QGIS to visualize results.

3. Assignment Datasets

This assignment provides the following datasets [download]:

a. Datasets of Purple Air sensor locations, named as **county_sensor_locations_[train/test].csv**

The Purple Air sensor location data contain the sensors in Los Angeles County (LA County), represented as sensor IDs, longitude, and latitude in WGS84. The sensors with missing observations are excluded from this assignment. We split the locations into train (80%) and test (20%) sets. You will use the train samples to build a model and generate prediction at test locations (a total of 27) for quantitative evaluation.

b. Datasets of Purple Air sensor observations, named as **county_sensor_observations_[train/test].csv**

The sensor observation data contain the observed $PM_{2.5}$ concentrations (column "pm2.5") at the sensors in LA County, reported hourly from January to February in 2022. We split the observations into training and testing sets based on the locations.

c. One dataset of grid locations, named as **county_grids.csv**

You will generate the prediction of $PM_{2.5}$ concentrations at grid locations in LA County (see Figure 1) for qualitative evaluation. Each row in the dataset represents a cell covering a region of $1{,}000 \times 1{,}000$ square meters, described by grid ID, longitude and latitude of the centroid of the cell, and the geometry of the cell (in the form of text).

d. Datasets of geographic features, named as **county_[sensor/grid]_geographic_features.csv**

You will use the geographic features generated from OpenStreetMap to describe the surrounding environment of a location (see assignment 1). We have generated geographic features using the buffers

from 100 meters to 3,000 meters with an interval of 100 meters. The datasets are available for the sensor locations (the geographic features for train and test locations are in the same file) and the grid locations.

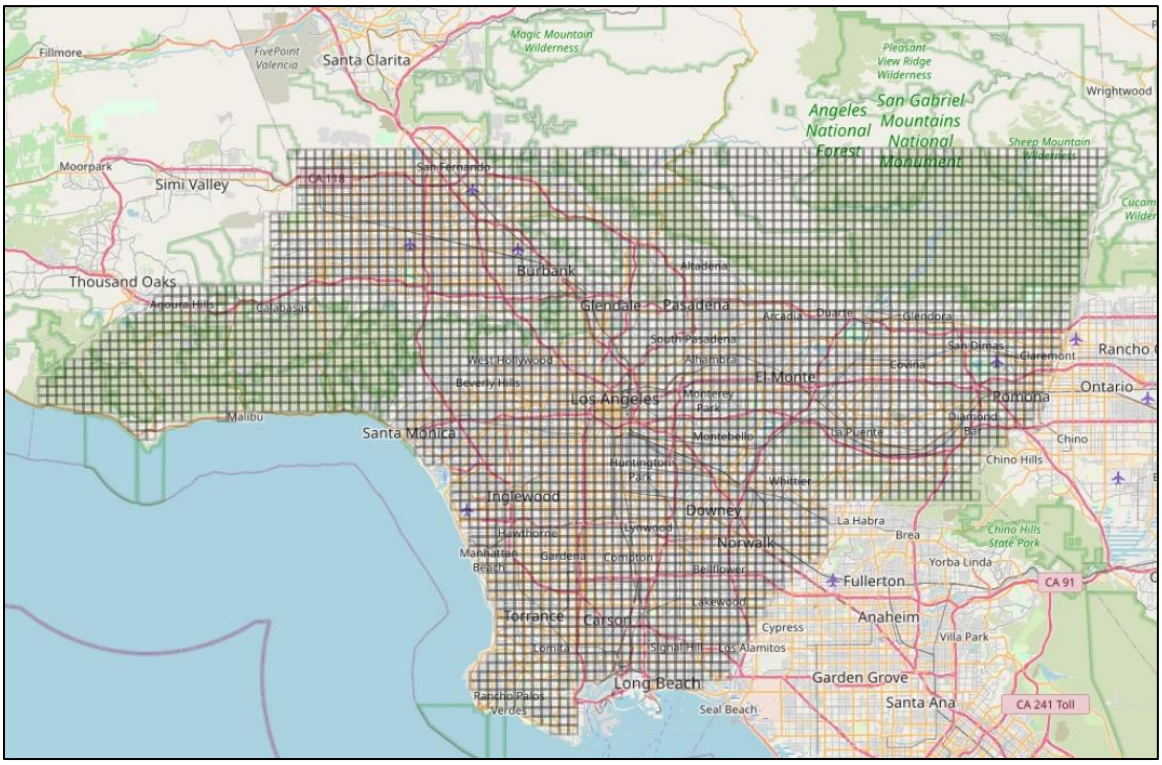


Figure 1: Grid locations (a total of 4234 cells) in LA County

4. Deliverables and Tasks

You need to turn in a zip file, named as **[your_login_id]_assignment2.zip** (all lowercase), [1] e.g., lin00786_assignment2.zip, containing the following files:

a. [REQUIRED] Code scripts: two Python script named as **task1.py** and **task2.py**

b. [REQUIRED] A model file named as **model.save**

c. [REQUIRED] Two result files named as **test_prediction.csv** and **grid_prediction.csv**

d. [REQUIRED] A readme document in Word describing the results, named as **readme.docx**

e. [OPTIONAL] You can include other Python scripts to support your programs (e.g., callable functions).

4.1. Task 1: Identifying Important Geographic Features (2pts)

In this task, you will write the code in task1.py to automatically identify important geographic features for predicting air quality from training samples. You can follow the paper or the steps below.

[1] The name of your email address, e.g., if your email address is lin00786@umn.edu, your id would be lin00786

Step 1: Clustering Sensor Locations

In this step, the goal is to group sensors that have similar temporal patterns on $PM_{2.5}$ concentrations. For example, locations near the highway would show higher $PM_{2.5}$ concentrations during the daytime than those near parks. Thus, these locations can form into two clusters.

The paper uses K-means to cluster the available sensors based on the collected time series of $PM_{2.5}$ concentrations, i.e., the observation at a time point forms one feature dimension. You can use the elbow point strategy to decide the number of clusters. Also, you can apply other clustering algorithms. Optionally, you can do dimension reduction on the time series first and then cluster the lower-dimensional representations. **You need to report the number of clusters you are choosing and the clustering results in a dictionary format as {sensor ID: cluster label}.**

Step 2: Constructing Geographic Abstraction Vectors

This step aims to convert geographic features (e.g., Table 1) to geographic abstraction vectors (e.g., Table 2). The features in the vectors are the distinct geographic features, represented as a combination of [Geo Feature, Feature Type, Buffer Size]. For example, in Table 1, there are seven distinct features (e.g., Highway_Pedestrain_500). Sensor A has values in five of the features, and sensor B has values in four of them. Table 2 shows the constructed vectors for sensor A and sensor B. If the location does not have a particular feature, the value will be 0. **You need to report the number of features in the geographic abstraction vector.**

Table 1: Geographic features

GID	Sensor ID	Geometry Type	Buffer Size	Geo Feature	Feature Type	Value
1	A	line	500	Highway	Pedestrian	200
2	A	line	1,000	Highway	Motorway	300
3	A	polygon	500	Landuse	Green_land	1,000
4	A	polygon	1,000	Landuse	Water_area	1,500
5	A	point	500	Building	House	40
6	B	line	500	Highway	Motorway	150
7	B	line	1,000	Highway	Motorway	200
8	B	line	500	Highway	Primary_road	500
9	B	polygon	1,000	Landuse	Water_area	750

Table 2: Geographic abstraction vectors

	Highway Pedestrian 500	Highway Motorway 500	Landuse Green_land 500	Landuse Water _area 500	Building House 500	Highway Motorway 1000	Highway Primary_road 500
A	200	0	1,000	1,500	40	300	0
B	0	150	0	750	0	200	500

Step 3: Computing Feature Importance

In this step, our goal is to automatically identify which geographic features within what buffer size have the most impact on $PM_{2.5}$ concentrations. The paper uses the grouped sensors as the label (the dependent variable) and their geographic abstractions as the predictor features to train a random forest classification model. You might need to preprocess the geographic abstraction vectors with normalization (e.g., min-max-scaler) before applying machine learning algorithms. Then you will extract feature importance from the random forest model. **You need to report the top 20 features with the highest importance in a table (see the format below).**

Geo Feature	Feature Type	Buffer Size	Importance (%)
highway	service	2900	0.005234

Step 4: Saving Model

You will save your model to a file, including scaler for normalization and selected features (e.g., picking top 10% important features). You will load this model for the prediction. There is no restriction on what and how to save, but pickle file is recommended.

Execution Commands:

```
usage: task1.py [-h] [--sensor_loc SENSOR_LOC] [--sensor_obs SENSOR_OBS]
                [--sensor_geo SENSOR_GEO] [--model_path MODEL_PATH]

optional arguments:
  -h, --help            show this help message and exit
  --sensor_loc SENSOR_LOC
                        The file to the locations (train).
  --sensor_obs SENSOR_OBS
                        The file to the sensor observations (train).
  --sensor_geo SENSOR_GEO
                        The file to the sensor geographic features.
  --model_path MODEL_PATH
                        The file to save the model.
```

4.2. Task 2: Generating Predictions (3pts)

In this task, you will write the code in task2.py to generate predictions for given locations. The given locations can be test samples or grid locations. You can follow the paper or the steps below.

Step 1: Constructing Geo-Context

In this step, you need to construct geo-context for the observed sensor and given locations, respectively. The geo-context are the vectors containing the selected geographic features from the model in Task 1. These features replace the original geographic abstraction and become a description of the geographic environment around a location for predicting $PM_{2.5}$ concentrations.

Step 2: Generating Predictions

To predict $PM_{2.5}$ concatenation **at a time point** for **a given location** that does not have sensor, the paper trains a random forest regression model with the geo-context (as the predictors) and the $PM_{2.5}$ concatenations (as the dependent variable) at that time from all available train sensors.

You need to generate predictions for the test locations (a total of 27) and grid locations (a total of 4234) at all time points (a total of 745). **You need to save the predictions in CSV files (see the format below).**

Table 3: Prediction file format for test samples

sensor_id	timestamp	prediction
1852	2022-01-01 00:00:00-08:00	29.51327

Table 4: Prediction file format for grid locations

grid_id	timestamp	prediction
1	2022-01-01 00:00:00-08:00	18.90042

Execution Commands:

```
usage: task2.py [-h] [--sensor_loc SENSOR_LOC] [--sensor_obs SENSOR_OBS]
                [--sensor_geo SENSOR_GEO] [--model_path MODEL_PATH]
                [--target_loc TARGET_LOC] [--target_geo TARGET_GEO]
                [--prediction PREDICTION]

optional arguments:
  -h, --help            show this help message and exit
  --sensor_loc SENSOR_LOC
                        The file to the sensor locations (train).
  --sensor_obs SENSOR_OBS
                        The file to the sensor observations (train).
  --sensor_geo SENSOR_GEO
                        The file to the sensor geographic features.
  --model_path MODEL_PATH
                        The file of the model.
  --target_loc TARGET_LOC
                        The file to the target locations (test or grid).
  --target_geo TARGET_GEO
                        The file to the geographic features of target
                        locations (sensor or grid).
  --prediction PREDICTION
                        The file to save the predictions.
```

4.3. Task 3: Evaluation

In this task, you will write the code to evaluate your predictions quantitatively for test samples and quantitatively for grid locations.

For test locations, you will report the MSE and R2 scores by comparing the predictions to the ground truth (i.e., sensor observations for test locations). **Specifically, you will need to (1) the overall MSE and R2 scores, and (2) the MSE and R2 scores at each hour (hour is from 0 to 23). For (2), you need to plot the trend of RMSE and R2 scores (y-axis) along the hours (x-axis).**

For grid locations, you will use QGIS to visualize the predictions at a time (see the steps in Appendix). You will join the predictions with cell geometry on grid ID and save as a new file. You need to show the spatial visualization for the average $PM_{2.5}$ predictions in the morning (6am to 9am), afternoon (4pm to 7pm), weekday (Monday to Friday), and weekend (Saturday and Sunday). **You need to report the four screen shots and discuss the spatial pattern of the predictions in the README.** Optionally, you can use the same color scale for the plots.

5. Grading criteria

a. Your code should be runnable and be able to generate results correctly (that is, generating the models and predictions with the submitted Python scripts successfully) to receive full points on Task 4.1 and Task 4.2 (a total of 5 points).

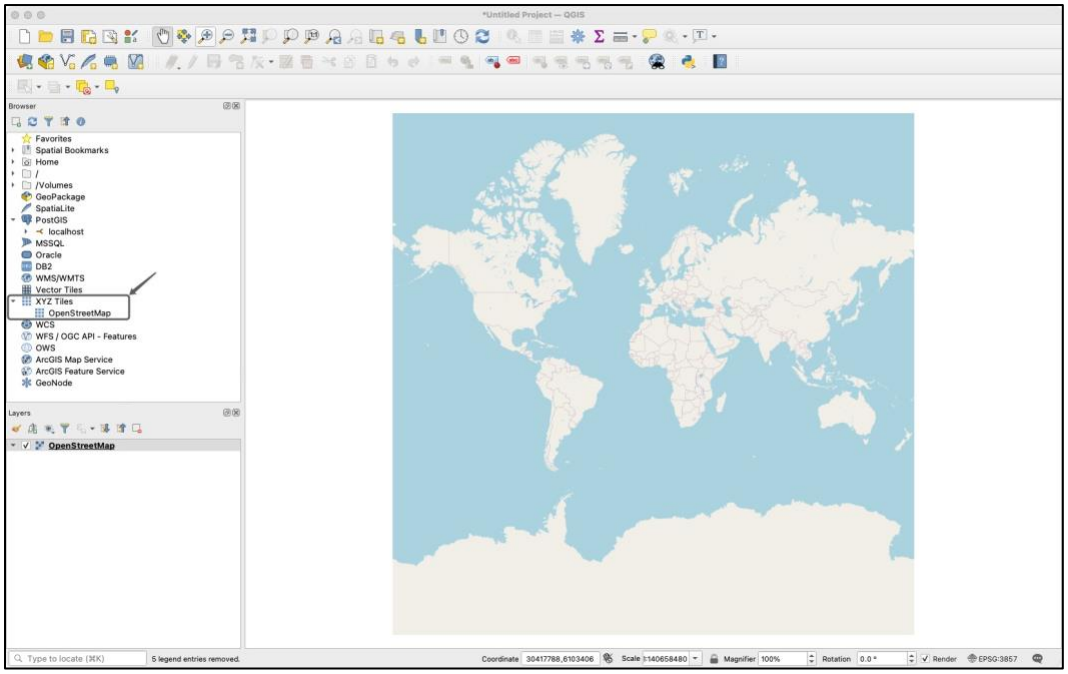
b. In the readme document, you should report the following things:

   1. The number of clusters and the cluster label for each sensor location (0.5pt)
   2. The total number of features in the geographic abstraction vector (0.5pt)
   3. The top 20 important features and their importance (1pt)
   4. Overall MSE and R2 on test samples and the plot showing hourly MSE and R2 scores (1pt)
   5. Four plots of fine-grained prediction results (screen shots) (1pt)
   6. Your findings and discussion on the selected features, MSE and R2 scores, and plots of fine-grained predictions in several sentences (**no more than 300 words**) (1pt)

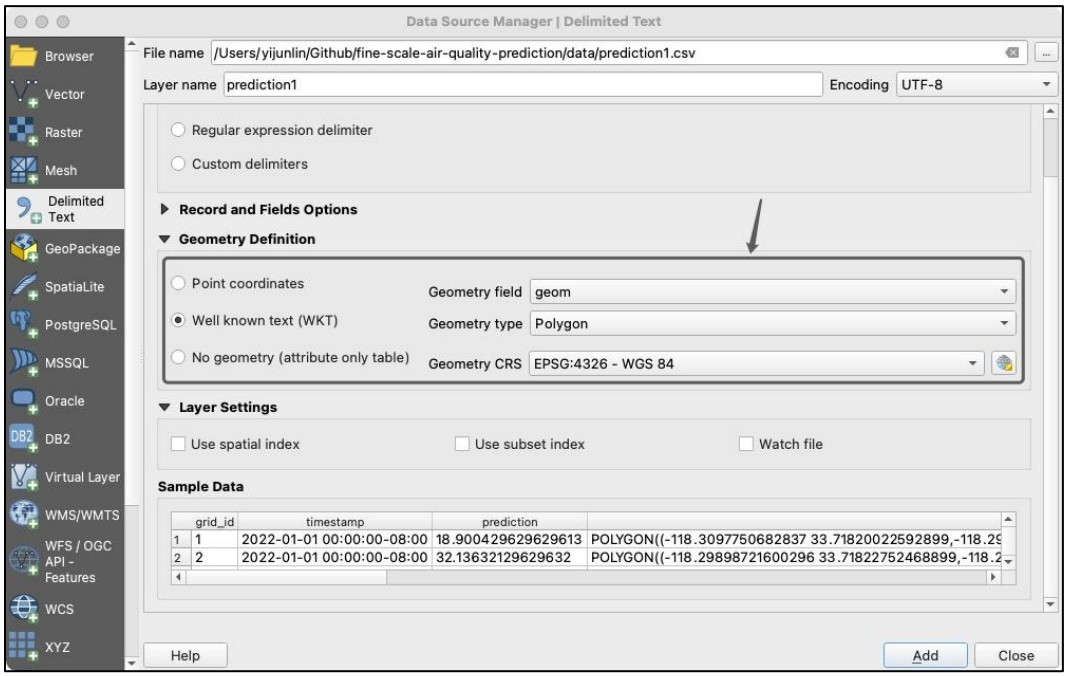
c. You can put any external libraries that are necessary to execute your code or other instructions that can help TA run your assignment.

Appendix:

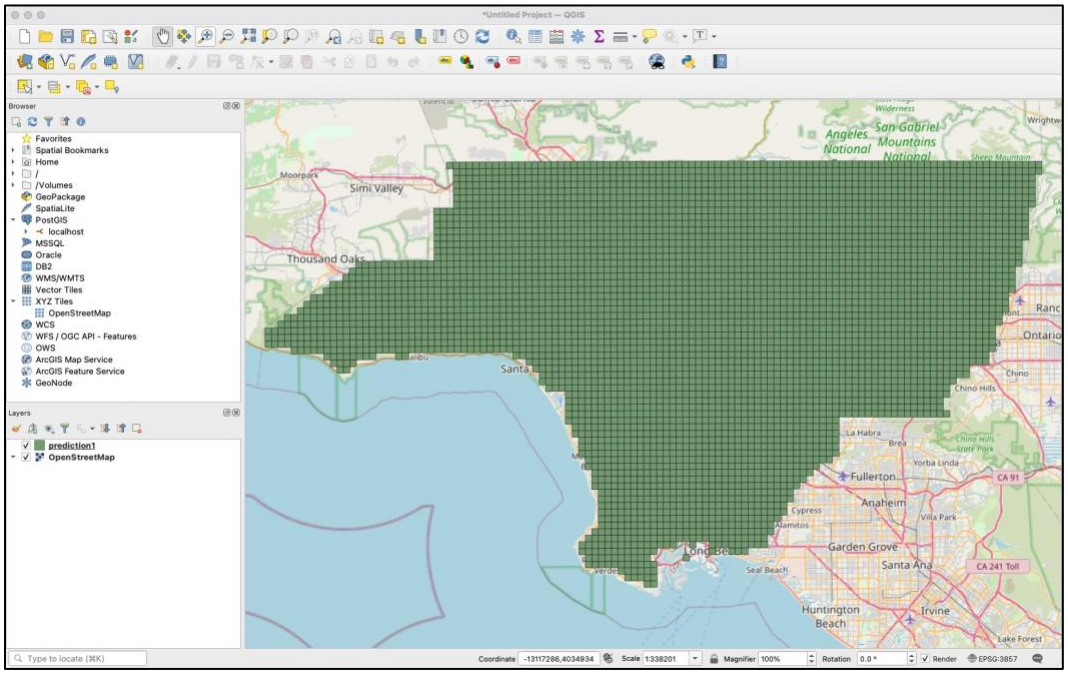
1. Load OSM from XYZ Tiles



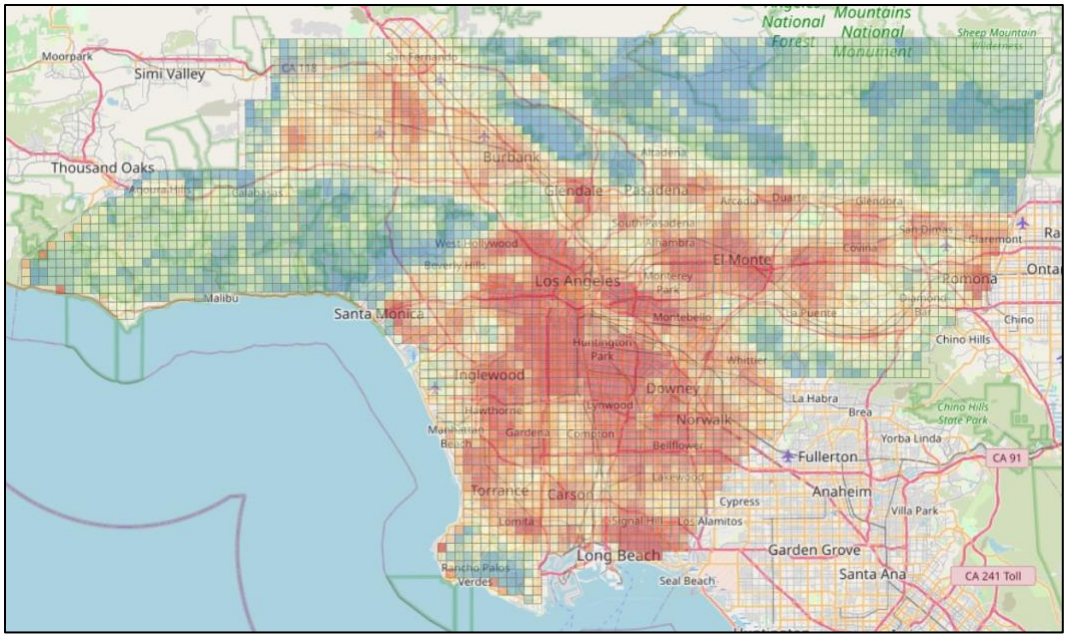
2. Load predictions at a specific time point from “Layer” -> “Add Layer” -> “Add Delimited Text Layer” -> Add the prediction file -> Fill in Geometry Definition

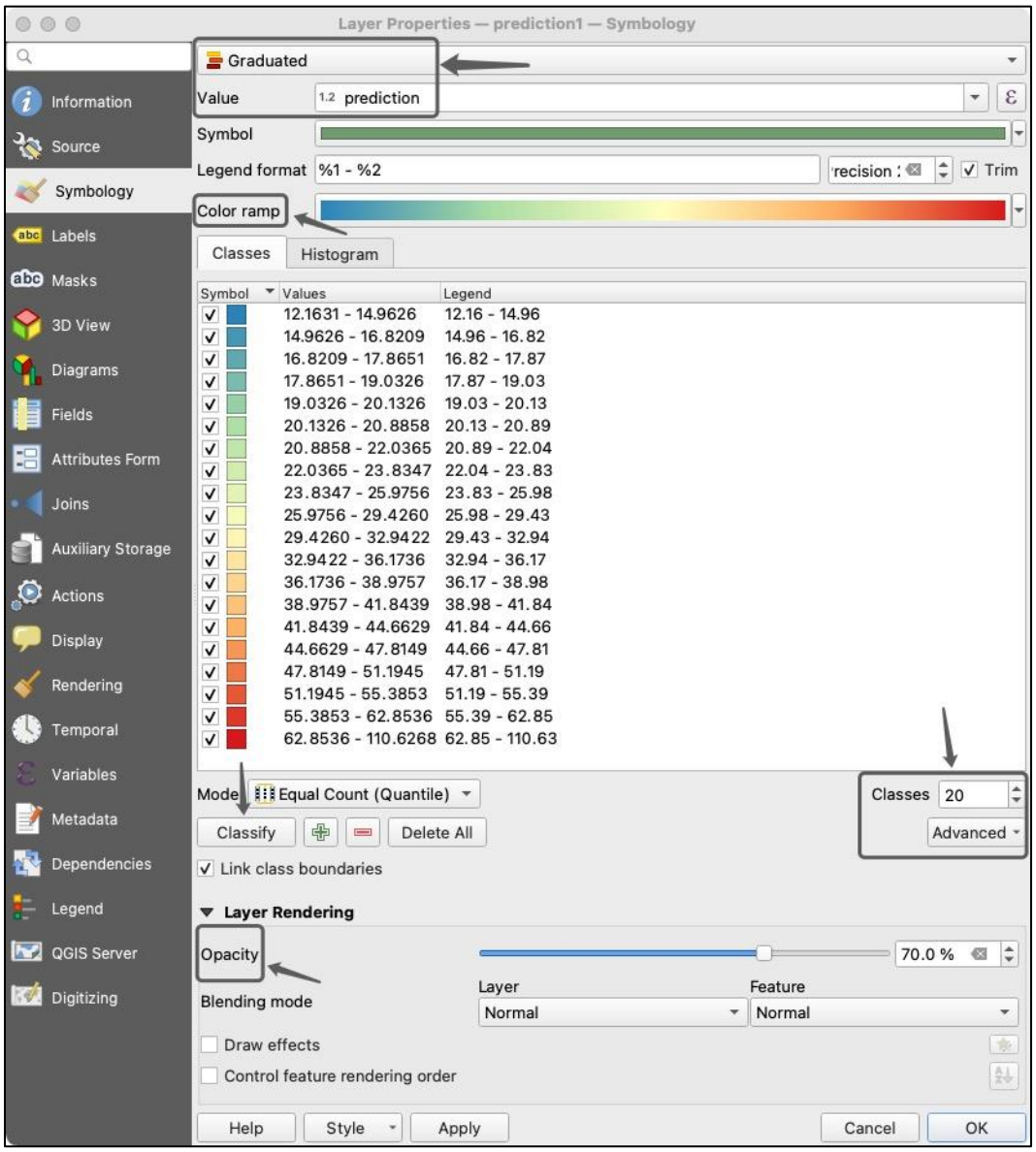


3. Zoom to the prediction layer



4. Set the color for grid -> Layer Properties -> Choose style as “Graduated” -> Set Value from the table -> Set Color ramp as “Spectral” and “Invert Color Ramp” -> Set number of classes (the more classes the more detailed the visualization, you can add fake numbers to keep the same color scale for all plots) -> Set opacity to a proper number (50% - 70%)




Layer Properties — prediction1 — Symbology
Information
Source
Symbology
Labels
Masks
3D View
Diagrams
Fields
Attributes Form
Joins
Auxiliary Storage
Actions
Display
Rendering
Temporal
Variables
Metadata
Dependencies
Legend
QGIS Server
Digitizing
Graduated
Value
1.2 prediction
Symbol
Legend format
%1 - %2
recision :
Trim
Color ramp
Classes
Histogram
Symbol
Values
Legend
12.1631 - 14.9626 12.16 - 14.96
14.9626 - 16.8209 14.96 - 16.82
16.8209 - 17.8651 16.82 - 17.87
17.8651 - 19.0326 17.87 - 19.03
19.0326 - 20.1326 19.03 - 20.13
20.1326 - 20.8858 20.13 - 20.89
20.8858 - 22.0365 20.89 - 22.04
22.0365 - 23.8347 22.04 - 23.83
23.8347 - 25.9756 23.83 - 25.98
25.9756 - 29.4260 25.98 - 29.43
29.4260 - 32.9422 29.43 - 32.94
32.9422 - 36.1736 32.94 - 36.17
36.1736 - 38.9757 36.17 - 38.98
38.9757 - 41.8439 38.98 - 41.84
41.8439 - 44.6629 41.84 - 44.66
44.6629 - 47.8149 44.66 - 47.81
47.8149 - 51.1945 47.81 - 51.19
51.1945 - 55.3853 51.19 - 55.39
55.3853 - 62.8536 55.39 - 62.85
62.8536 - 110.6268 62.85 - 110.63
Mode
Equal Count (Quantile)
Classes
20
Classify
Delete All
Advanced
Link class boundaries
Layer Rendering
Opacity
70.0 %
Blending mode
Layer
Feature
Normal
Normal
Draw effects
Control feature rendering order
Help
Style
Apply
Cancel
OK