GeoAI Challenge for Air Pollution Susceptibility Mapping by ITU

Competition username: xiaoiornman

Xiao Liu

October, 2023

Abstract

This document first briefly introduces the GeoAI challenge dedicated to air pollution susceptibility map in Milan region as well as the used data and features inside. Then, the models used by the author and the results obtained are presented. Finally, a brief conclusion is given that the simple Naive Bayes model developed by the author almost perfectly predicted the AQI of at the locations listed in the test data, a recommendation of using Geographically weighted regression (GWR) for future model training is also given.

Figure 1: Logo of AI for Good, organized by ITU.

1. Introduction

This report serves as an assisting material for the attached source code file (main.py) scripted for the GeoAI Challenge organized by the International Telecommunication Union (ITU), as part of the AI for Good campaign. The topic of the challenge is air pollution susceptibility in the city of Milan, Italy.

1.1. Problem Description

The objective of this challenge is to use machine learning to produce air pollution susceptibility maps at the city level (5m spatial resolution) which will support decision-making to improve the resilience of the city. The air pollution at any given location is measured as an index value to provide a general panorama of the different pollution levels. This leads to the target parameter of this challenge, the Air Quality Index (AQI).

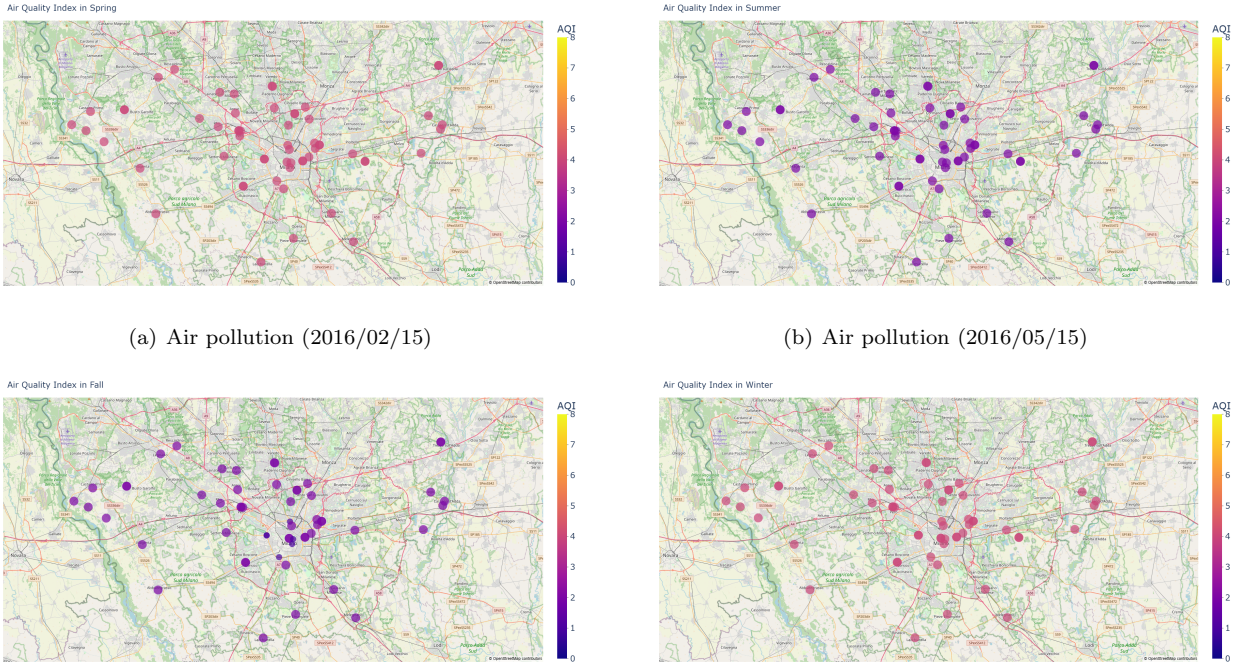
1.2. Feature Description

In order to properly estimate the AQI, feature variables including coordinates, season info and meteorological data (such as temperature, precipitation, relative humidity, solar radiation, wind speed and direction at a daily temporal resolution) are considered, even though it was found that the best solution resulted from using coordinates and season features only.

2. Data

The data used for this challenge are all downloaded from the competition platform (Zindi), no additional data are used. The collected data include the following files:

1. **Train.csv**, labelled training dataset with all Meteorological and Air pollution time series information as well as with AQI (target). The AQI for each season are plotted in Figure 2
2. **Test.csv**, includes only the location ID to be tested and their coordinates and desired season number.
3. Meteorological timeseries data (2016-2022) which includes temperature, precipitation, relative humidity, solar radiation, wind speed and direction at a daily temporal resolution.
4. Air pollution timeseries data (2016-2021) which includes NOx, SO2, CO, O3, PM2.5, PM10, and benzene at a daily temporal resolution.
5. Digital terrain model, land cover, geology, plan curvature, hill shade, slope, SPI, TRI, TWI are provided at the stations' points and as a continuous representation at 100-m resolution.
6. **SampleSubmission.csv**, used as a reference for the submission file format.



(a) Air pollution (2016/02/15)

(b) Air pollution (2016/05/15)

(c) Air pollution (2016/08/15)

(d) Air pollution (2016/11/15)

Figure 2: Air Quality Index at each location in training data set on a selected day (middle) for each season

Note that item 3, meteorological data, are stored in 4 files for the 4 seasons respectively. Item 4 and 5 are stored together in a single file for the whole time period. All data in these 3 items are appended to each row of the Test.csv file by fetching the rows whose center coordinates are closest to the test location. A script was developed to collect the proper row of data from these 3 data sets, see Listing 1.

3. Features

The training data contains numerous columns of features that might have influenced the final AQI, a basic interpretation from the author is listed below:

1. **Temporal Data**:
   - `date`: Date of the record.
2. **Geographical Data**:
   - `lat`: Latitude.
   - `lng`: Longitude.
   - `utm_x`: UTM coordinate x-value.
   - `utm_y`: UTM coordinate y-value.
   - `dtm_milan`: Digital Terrain Model related to Milan.
   - `aspect`: Direction the terrain faces.
   - `water_distance`: Distance to the nearest water body.

```
# function to get the proper row of
   terrain data by finding the closest
   point to the input coordinates
def get_terrain_data(lat, lng):
    # find the closest point to the
       input coordinates
    df_terrain['dist'] =
       df_terrain.apply(lambda row:
       (row['lat'] - lat) ** 2 +
       (row['lng'] - lng) ** 2, axis=1)
    df_terrain.sort_values(by=['dist'],
       inplace=True)
    # get the closest point
    df_terrain_closest =
       df_terrain.iloc[0]
    # get the terrain data of the
       closest point
    df_terrain_data =
       df_terrain_closest.iloc[5:]
    return df_terrain_data
```

Listing 1: Script to obtain data at nearest location

   - `slope`: Incline angle of terrain.
   - `plan_curvature`: Curvature of the terrain in the horizontal plane.
   - `profile_curvature`: Curvature of the

terrain in the vertical plane.

3. **Weather and Climatic Data**:
   - `temperature`: Air temperature.
   - `precipitation`: Amount of rain.
   - `humidity`: Relative humidity.
   - `global_radiation`: Solar radiation.
4. **Hydrological Data**:
   - `hydrometric_level`: Possibly the water level of rivers or reservoirs.
   - `spi`: Possibly a drought index.
   - `tri`: Topographic Ruggedness Index.
   - `twi`: Topographic Wetness Index.
5. **Wind Directions**:
   - `N`, `NE`, `E`, `SE`, `S`, `SW`, `W`, `NW`: Could refer to wind direction or wind speeds from these directions.
6. **Air Quality Data**:
   - `type`: Type of air pollutant.
   - `pm25`: PM2.5 (fine particulate matter) concentration.
   - `pm10`: PM10 (coarse particulate matter) concentration.
   - `o3`: Ozone concentration.
   - `so2`: Sulfur dioxide concentration.
   - `no2`: Nitrogen dioxide concentration.
   - `pm25_aqi`, `pm10_aqi`, `o3_aqi`, `so2_aqi`, `no2_aqi`: Air Quality Index values for respective pollutants.
   - `aqi`: General Air Quality Index.
7. **Geological and Geographical Features**:
   - `dusaf15`: Might be related to a geological survey or dataset.
   - `geologia`: Geology or geological characteristics.
   - `hillshade`: Measures of how much sunlight a given location gets during the day.
   - `ndvi_2019`: Normalized Difference Vegetation Index for the year 2019.
   - `geo_0` to `geo_6`: Perhaps categorical or numerical geological features or zones.
8. **Land Cover Data**:
   - `lc_11` to `lc_51`: These seem like land cover classifications, though without context, the specific categories are unknown.

4. Models and Results

The author experimented on 2 different algorithms using all the possible features as initial trials, respectively:

- Random Forest Regression (RFR)
- Naive Bayes (NB)

These 2 models were trained using the RandomForestRegressor and GaussianNB from the scikit-learn library while the NN model is based on the keras module of Tensorflow.

The training data and test data were all divided into 4 subsets, each subset of data correspond to one season and different models was trained for different seasons. Each model was also evaluated by computing the Mean Squared Error of the training data predictions (Table 1).

Season	MSE(RFR)	MSE(NB)
Winter (1)	0.31	2.39
Spring (2)	0.24	1.83
Summer (3)	0.16	1.4
Winter (4)	0.45	3.21

Table 1: Mean Squared Error (MSE) values for each season when using all features

Besides, the accuracy info is also collected based on the public score on the leader board. The results for these two methods are shown in Table 2

As can be observed, both model seem to work badly. Thus, some more processing were done: While fitting the RFR model, the feature importance was computed and the top 15 features are plotted in Figure 4

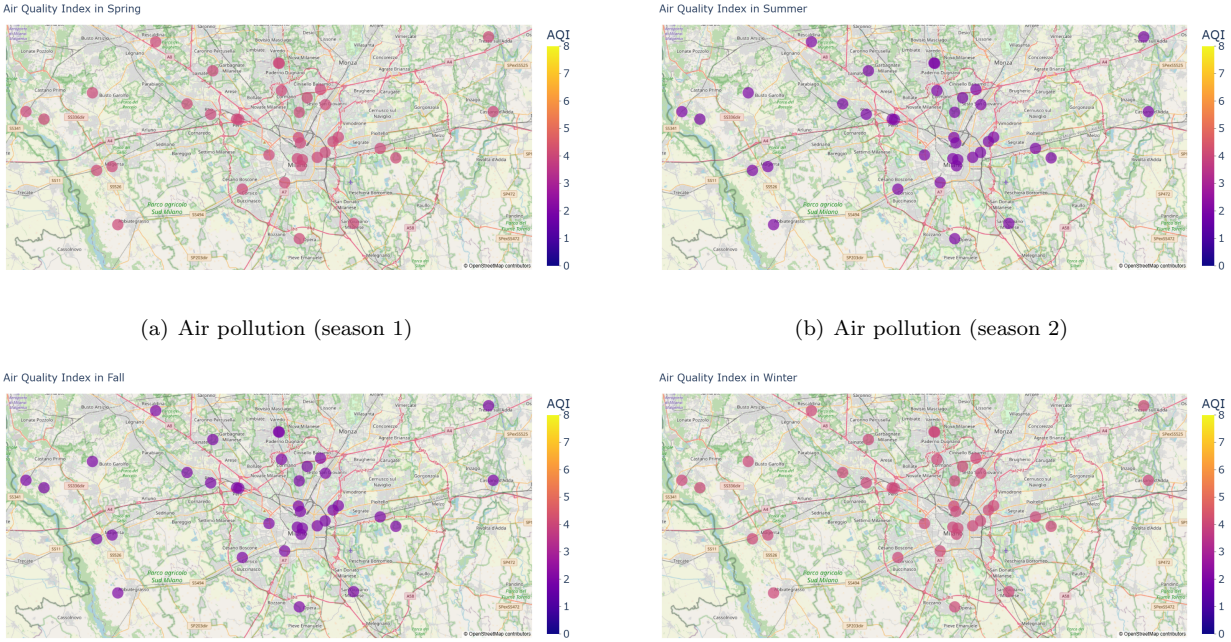
The author then tested using the minimal number of features, as provided in the original Test.csv file: latitude, longitude and season info. It turns out that by using only these information, the performance in terms of accuracy has increased dramatically, as can be seen in Table 3

Note that during the modeling process, a third model was used for the simple feature sets, this is a Neural Network (NN) model based on the Keras module of the Tensorflow library.

As can be seen, the results obtained using Naive Bayes resulted in a perfect prediction and this is thus used as the final submission file for this challenge. The source code can also be found in the

Model	Accuracy
RFR	0.25
NB	0.075

Table 2: Model accuracy when using all features



(a) Air pollution (season 1)

(b) Air pollution (season 2)

(c) Air pollution (season 3)

(d) Air pollution (season 4)

Figure 3: Air Quality Index predicted at each location in test data set based on NB model

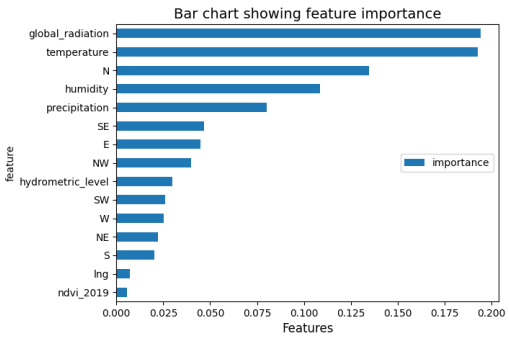


Figure 4: Feature importance that may influence Air Quality Index

attached files. The resulting air pollution susceptibility map of the Milan region is also plotted in Figure 3

5. Conclusion

It seems that the simplest model with the minimal features already produced a near perfect prediction for the test data. It is thus recommend to develop some Geographically weighted regression (GWR) models for further improving the model accuracy for any larger datasets. As for data of meteorology, Digital terrain model, land cover, geology, etc.. the abundance of such data seem to contribute in a negative way for these data set, even though their usage need to be explored more deeply.

Model	NB	RFR	NN
Accuracy (public score)	**1**	0.431	0.444
Accuracy (private score)	**0.991**	0.446	0.268

Table 3: Final accuracy of 3 different models while using minimal features