Report for 5002 final project

Group number 42

Ismailov Mukhit. ID: 20565331

Abstract

This is a report of a final project of 5002 class, Fall 2018. The aim of project is to build a good model for predicting different air measurements, in particular: the concentration of atmospheric particulate matter (PM) that have a diameter of less than 2.5 micrometers (PM2.5), PM 10 micrometers or less in diameter (PM10) and ozone level (O3) in Beijing, China. All measurements are in micrograms (one-millionth of a gram) per cubic meter (ug/m3) [1]. The whole process is divided into 4 parts – data preprocessing, feature engineering, selecting right data and training models, experimentation with different features. During cross validation LGBM models show a good score.

1. Introduction

Over the past decades quality of air is getting worse due to heavy industrialization. This is especially true for big cities such as Beijing. There are couple of reasons for this: proximity of large factories to a city that produce heavy pollutants, and large population with many of them using cars which emit pollutants into the atmosphere. Importance of checking on pollutants cannot be underestimated. Particles of size less than 10 micrometers can get into the lungs causing various forms lung diseases [1].

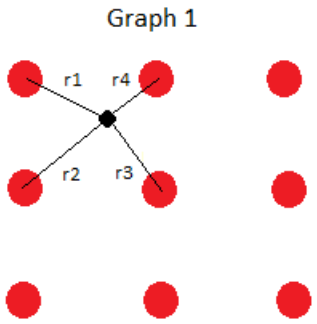
This project tries to target the problem of predicting amount of air pollutants in the air on hourly basis in different parts of Beijing. In particular, the task is to predict amount of PM2.5, PM10 and O3 in the atmosphere for 48 hours (May 1 and 2). To do this I am given air quality data for the period of January 2017 to April 2018 as well as weather data for that period.

This paper is organized in the following way: Section 2 describes data preprocessing, Section 3 discusses feature engineering on data, Section 4 shows the models I used, Section 5 talks about experiments that I have tried, Section 6 shows results, Section 7 is Conclusion.

2. Data preprocessing

There are four files related to air quality measurements – 3 for information on air quality from January 2017 to April 2018, and 1 for locations of stations that measure air quality in Beijing. In addition, there are a number of other files that have information on weather during that period. After extracting information that I believe are useful I constructed two Pandas dataframes. First one hold information on “station_id”, “time”, “PM2.5”, “PM10”, “O3”, “longitude” and “latitude”. Second, I used gridded weather information. So, weather dataframe has information on “longitude”, “latitude”, “time”, “temperature”, “pressure”, “humidity”. I dropped information about wind because I found it unreliable.

Now, I join two dataframes in a specific way. Since I have a grid, I choose 4 closest grid points to the station to get a good approximation.



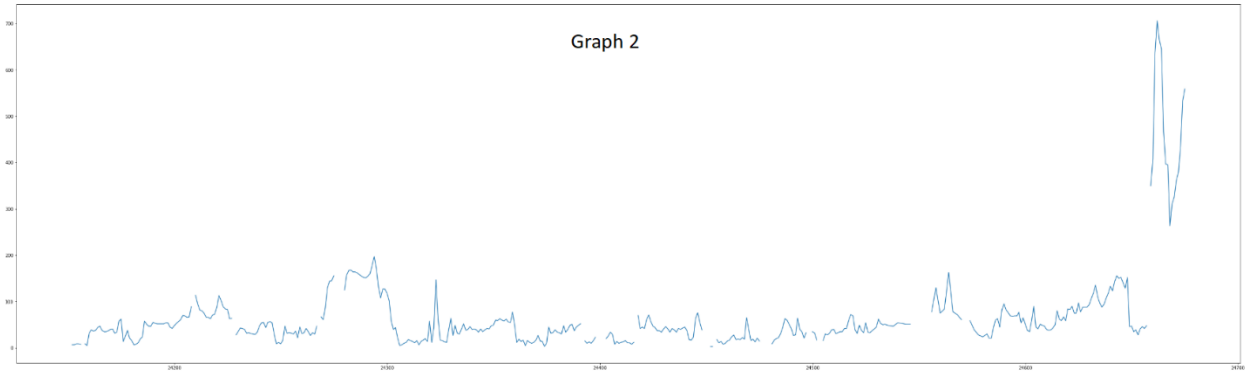
From graph 1 the idea is visible. For each grid point I assign a weight which is inversely proportional to the square of distance to the air quality station. The sum of the weights is equal to 1. Then I sum up those 4 measurements each with its own weight and get a value for the station.

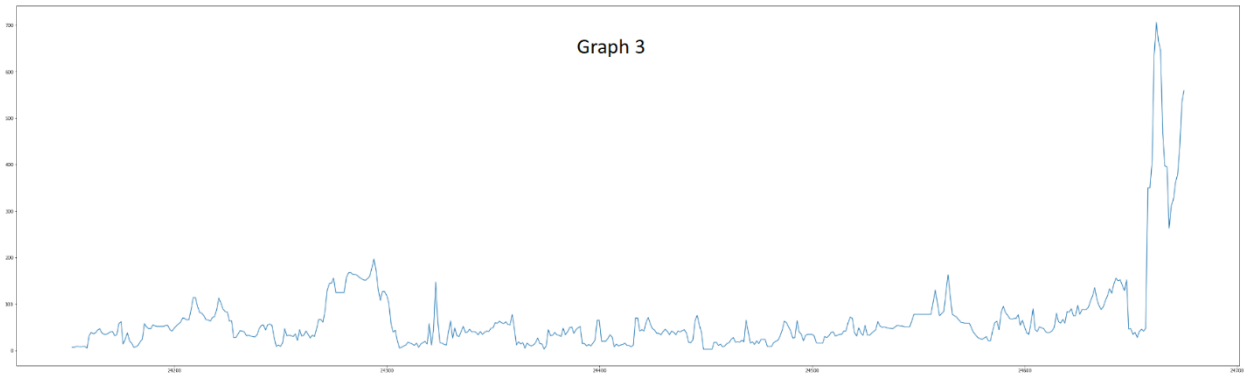
Now after dropping duplicated data points I end up having 377,265 data points. However, data has a lot of missing values. In particular, there are about 100,000 missing data points for “PM10” measurement, and about 25,000 for “PM2.5”, and “O3”. Simply removing that data is very bad because we would lose about third of our data.

Another method that I tried was to fill missing values with mean, median of values from other stations at that hour. So, for example, if PM2.5 measurement for station “aotizhongxin_aq” is missing at 12 a.m. on July 4, I took 34 measurements of other stations from that time and took mean, and median. Of course, when data is missing in more than one station is another problem which would be discussed later. So, I tried both mean and median values, but I found that these imputations are not good in this case. The main reason for this is that they negate different “behavior” of different stations. For example, if a station is located on a mountain it is very different from that in the valley.

What I used and find quite good is imputation with Light Gradient Boosting Machine (LGBM). I used this model instead of other Boosting algorithms because of its speed. And speed is important when you are training a total of 105 models: one model per missing air quality measurement per station. To find best hyper-parameters I used grid search package from “sklearn” library. To avoid overfitting I used cross validation on 4 folds. From the grid search I choose a model with best hyper-parameters and retrained with those parameters.

Similar to the method of filling values with mean and median, I took measurement from other stations from that time. I used a threshold of half the number of other stations – 17. This threshold says that if there are more than half the values missing at any time then other method is used to impute data. More specifically, I fill those values by looking backward and forward. This method is impressively good. To illustrate I took random interval from data. That interval contains missing values (graph 2). They show up as blanks and the line is discontinued. Graph 3 shows the same line but with filled with using a method of looking backward.





After a two-step imputation process there are no missing values left in the data.

3. Feature engineering

Now that we do not have missing values in the data, we need to predict future air quality measurements. I predict values on hourly basis, meaning that I predict one measurement at a time. To do this I use same measurements from past 72 hours together with weather measurements - temperature, pressure, and humidity. Total of 75 features to predict one value. I tried to add other features, and the results of those will be discussed in the experiments section. For example, if we need to predict a value of "O3" level in the atmosphere at 12 a.m. on July 4 for "aotizhongxin_aq" station, I take historical measurements of "O3" from July 1 12 a.m. (included) till July 4 12 a.m. (excluded). Additionally, I know temperature, pressure, humidity for 12 a.m. on July 4.

This process is used for training. Generating data for testing has a small trick to it. Basically, I need to predict measurements for two days, and thus I will be using my predicted values as input features to make predictions for further hours.

4. Model

For models to predict future values I once again used LGBM. Other models I have tried would be discussed in Section 5. Most of the hyperparameters were set to default, however on the key hyper-parameters I used grid search technique. To be more specific, I tuned "learning_rate", "num_leaves", "bagging_fraction", "feature_fraction".

Learning rate is by far most important parameter to get right in neural networks, especially in deep networks. However, in decision tree based learning algorithms you might get away with choosing this parameter wrong. But it still needs to be fine-tuned. Other parameters are responsible for making model not to overfit, or how well can a model generalize what it has learned.

To summarize I used 105 LGBM models and for each model I fine-tuned its hyper-parameters. To achieve this, I once again used cross-validation on 4 folds.

5. Experiments

When doing feature engineering I tried to extract information about date. Pandas has support for date types. I tried to add features like hour, week of month, day of week, day of month, is it an end or beginning of month, quarter, year, and several others. This is done to extract information about seasonality and/or trend. However, I faced certain difficulties which led me to abstain from adding those to my feature list. Those difficulties are how to treat those features. If treated as categorical the feature space gets large (because of probable need to one hot encoding) and with my models training for hours on simpler datasets, it could mean that I would be spending days on training. If some of them are treated as continuous then information about seasonality might get lost.

This problem can be solved by using neural networks or RNN models. This is another thing that I have tried. However, I tried Multi-Layer Perceptron, and the results were not so good. Probably, it is because I could not create a decent architecture. Unfortunately, I do not have enough experience to finetune RNN models, but I believe that that model would be more successful because you do no need to feed explicit information about seasonality.

6. Results

The score of submitted results would be calculated using Symmetric Mean Absolute Percent Error (smape):

$$smape = \frac{1}{n}\sum_{t=1}^{n}\frac{|\mathrm{F}_t - \mathrm{A}_t|}{(\mathrm{A}_t + \mathrm{F}_t)/2}$$

However, optimizing this score during training is not a good idea because it has unstable behavior near 0 values (when the value $A_t + F_t$ is close to 0).

To choose the best model during grid search and cross validation I used Mean Absolute Error (MAE). This is good enough scoring for our training.

For comparing results of different models, I used part of the data. This test data was not used during training models for comparing purposes. The results are in the table below:

	Average smape score on test data
Replacing missing values with mean	0.92
Replacing missing values with median	0.86
Imputing missing values with LGBM	0.37
Imputing missing values with LGBM + weather information	0.27
Imputing missing values with LGBM + weather information + date information	0.45
Imputing missing values with NN (MLP) + weather information + date information	0.54

As can be seen Imputing missing values with LGBM + weather information has the best smape score. So, I used that model for training on full dataset.

7. Conclusion

Currently, there is a large demand for predicting future values of some measurements based on historical data. Examples of these are some Kaggle competitions like “Rossmann Store sales” prediction, or similar to this data “Web Traffic Time Series Forecasting”. Through this project I learned how to impute missing values when there are a lot of them. Moreover, I explored several techniques to make predictions.