DS420-COBRAS
CAPSTONE PROJECT FINAL REPORT
KDD CUP 2018 PARTICIPATION

Leo Salemann, Yash Bhandari, David Ghan, Venkata Vemulapalli

June 4, 2018

PROBLEM DESCRIPTION

The objective of the Biendata 2018 KDD Cup of Fresh Air competition is to provide 48 hour forecasts of 2.5 micron Particulate Matter (PM2.5), 10-micron Particulate Matter (PM10), and Ozone (O3) air pollutants for a collection of air quality stations in Beijing and London. Accurate forecasts of these air pollutants could contribute to significant health improvements of exposed populations, by establishing times to wear particulate masks or remain in environments with quality air filtration. Particulate matter in these size ranges have been known to penetrate deeply into human lungs and blood vessels, where they can cause multiple chronic and life-threatening conditions. The best and perhaps only way to mitigate these risks is to prevent the particles from entering the lungs in the first place.

Source: 2018 KDD Cup of Fresh Air Introduction page, background section:
https://biendata.com/competition/kdd_2018/

DATA DESCRIPTION

Our data came from four primary sources:

1. **Beindata.com** Source of weather and air quality stations, and basic weather data as described in https://biendata.com/competition/kdd_2018/data/
2. **Openstreetmap** Reference mapping imagery, used to visually observe and manually record the terrain topography (flat vs. hills vs mountains) of each weather and air quality station.
3. **DigitalGlobe** Reference mapping imagery, used to visually observe and manually record the environmental setting (urban, park, forest, farm, …) of each weather and air quality station.
4. **DarkSky.net** Secondary source found in the KDD Cup discussion board at sed to replace the original KDD Cup weather data with more parameters current *and forecasted*, matching the *actual* location of each air quality station.

Further details are provided below:

Data Sources

Item	Source	Notes
Beijing Air Quality	https://biendata.com/competition/airquality/bj	Standard input data
London Air Quality	https://biendata.com/competition/airquality/ld	Standard input data
Weather Data	DarkSky.net via python forecastio library, as described in https://biendata.com/forum/view_post_category/139	DarkSky gives us current, historical and forecast data for for specific latitude/longitude.
Beijing Air Quality Station Locations	https://www.dropbox.com/s/5lhxontpbfoyemi/Beijing_AirQuality_Stations_en.xlsx	Provides latitude & longitude

London Air Quality Station Locations	https://www.dropbox.com/s/nuy1r6psk46vsi4/London_AirQuality_Stations.csv	Provides latitude & longitude
Terrain Map, Backdrop Image	OpenStreetmap, Microsoft PowerBI, MapBox Visual for Power BI	Used for manually recording terrain type (flat, mountainous, …) in a handmade spreadsheet.
Satellite Imagery	DigitalGlobe, Microsoft PowerBI, MapBox Visual for Power BI	Used for manually recording environment type (park, suburbs, city, forest, …) in an handmade spreadsheet.

Note that there were limitations in terms of daily requests that we could make for DarkSky API. We, therefore, cached every response from DarkSky. This way we only fetched weather forecasts for the days that we haven't already fetched data already for.

DATA EXPLORATION AND ANALYZING REPORT

Data exploration took four main forms:

- **Geospatial Assessment** Understanding the location of each weather and air quality station, its environment (vegetation and topography) and nearest-neighbor relationships.
- **Null Value Inspection** Looking for patterns in null values and how best to address them.
- **Pollutants vs. Weather** Looking for relationships between changes in pollutant and weather variables.
- **Signal vs. Noise** An overall assessment of random variation in the data vs. seasonal trends or other predictable behaviors.

Geospatial Assessment

The Geospatial Assessment was performed in Microsoft PowerBI with the ArcGIS and MapBox "Visuals" or extensions. Our primary goal was to establish nearest-neighbor relationships between the various stations, and to see if any variable exhibited spatio-temporal behaviors such as drifting with the wind. In practice, we settled on inspecting each AQ and non-grid MEO point, recording the topography (flat vs. mountain) and vegetation (urban, forest, park, farm …). Examples of our results are below.

Fig 1. Topography (MSFT PowerBI + MapBox + OpenStreetMap-Outdoor)

Fig 2. Vegetation (MSFT PowerBI + MapBox + DigitalGlobe-Satellite)

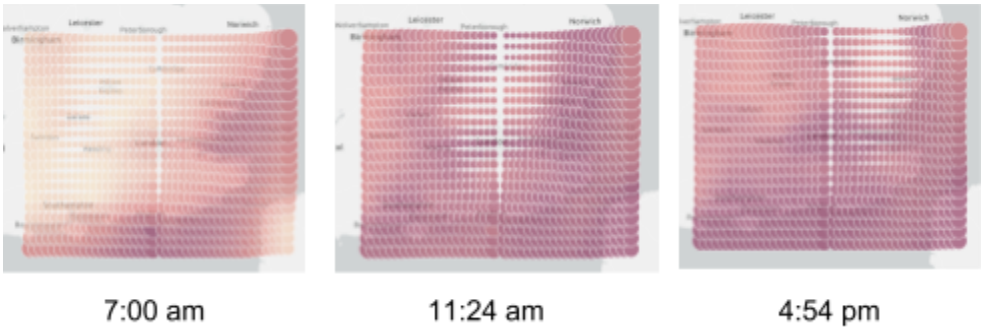


Fig 3. Spatio-Temporal Effects MSFT PowerBI + ArcGIS

Null Value Inspection.

Leo identified null values through ad-hoc inspection, and the use of line plots with sentinel values. Ad-hoc inspection refers to opening data files (csv's) in Microsoft Excel and using data filtering to see a list of unique values. It was particularly useful for categorical values. This technique revealed a few trivial cases such as a categorical **Precipitation Type** with the values *Rain*, *Snow*, and *Null*. Cross-checking some related weather variables quickly indicated *Null* should be replaced with *None* (i.e. no precipitation fell for that observation).

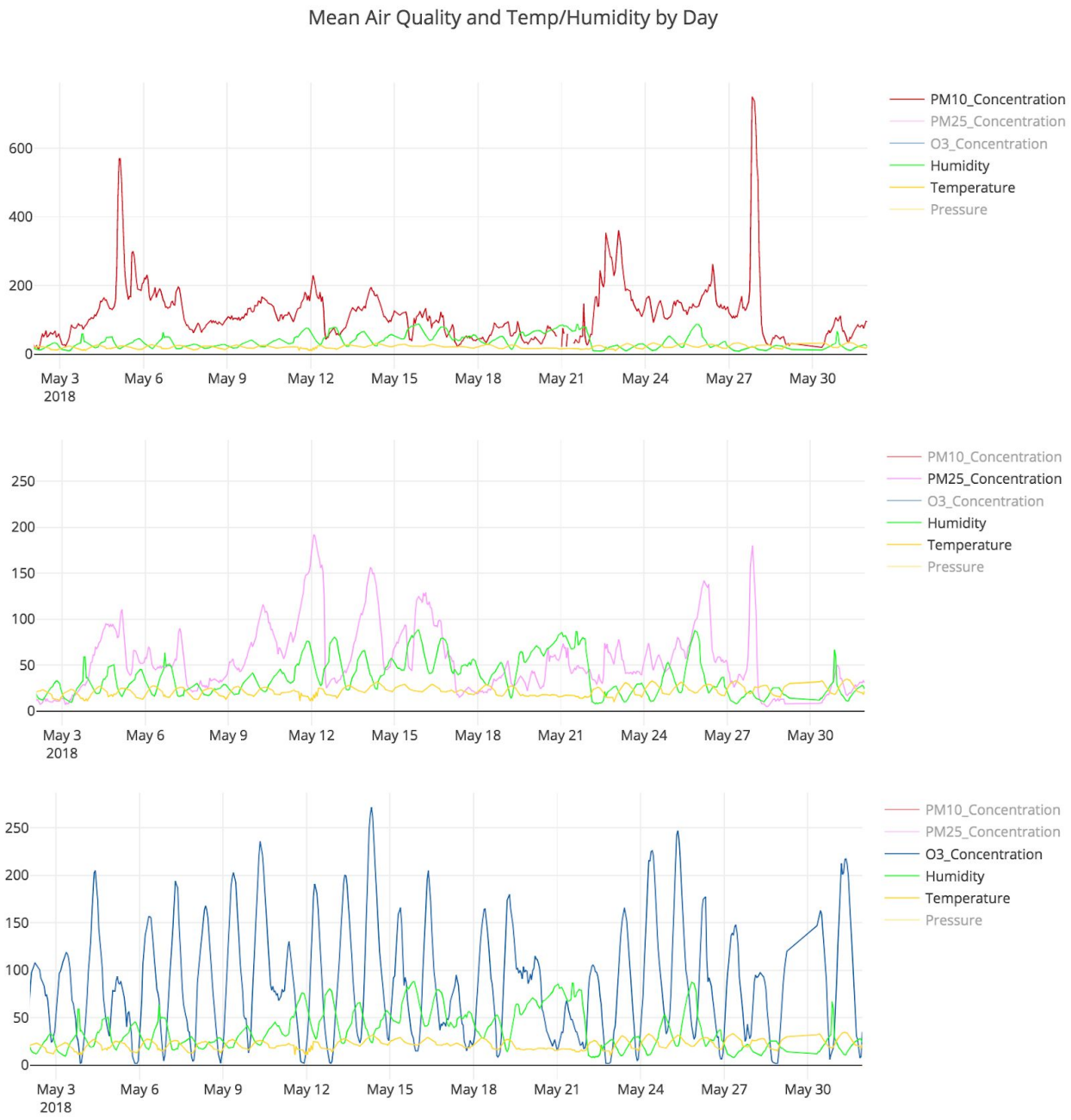
Line-plotting with sentinel values involved taking a copy of the input data, and replacing nulls with numeric numeric "sentinel values" that didn't exist in the data set beforehand. In practice, we used negative values such as -1 or -100. Plotting each variable as function of datetime revealed three main patterns, discussed in the Feature Engineering section.

A complete list of plots is in the Appendix.

Relationships Between Pollutants and Weather Variables

David explored and visualized the target variables PM2.5, PM10, and O3 alongside weather and meteorological data. It was determined that a number of weather features could be correlated with our target variables. Significant features included temperature, humidity, and air pressure. There was also some seasonality in the target variables over time, most noticeably for O3 levels. A few time periods over the course of each year demonstrated surprisingly low levels of PM2.5, PM10 and O3.

Below is a visualization of our target variables in comparison to some key weather variables over the course of the month of May 2018.



Signal vs. Noise

As part of the code development, Yash performed preliminary examination of the data. The analysis indicated that the randomness in the data was far stronger than the seasonality or trends. This indicated that any feature modeling task would be very prone to overfitting.

FEATURE ENGINEERING

Feature engineering took three major approaches

- **Leveraging Spatial and Temporal Relationships.** Early concept, abandoned.
- **Joining DarkSky.net Weather Data.** Broke from using BienData.com weather API.
- **Resolving Null Values.** Implemented in the later half of the competition.

Leveraging Spatial and Temporal Relationships

In the beginning, we were hoping to use spatio-temporal relationships such as, "the conditions at station A are driven by the conditions observed as station B two hours earlier when the wind speed is C and wind direction is D. We had envisioned a very wide table in which every observation held not only variables for a "primary" station, but also for its four "nearest neighbors." In practice, this was abandoned for a simpler approach.

Joining DarkSky.net Weather Data

The big breakthrough came with the weather data from DarkSky.net. The weather information as obtained from the competition website contained only the past weather observations and did not contain the future predictions. This implied that we needed one model for the past data and another one for the forecasts. This was infeasible by design since our goal was to make pollution forecasts. Thanks to the data from DarkSky.net, we could use the model trained on the past pollution and weather data for the future forecasts.

Moreover, with DarkSky we could have weather for the exact location of each air quality station, and we could have current *and future/forecast values*. Because of data at exact locations, we did not have to worry about interpolation from the weather stations to the air quality stations. And this was a huge time and effort saving for us.

The flip-side of this decision was that there is an additional risk introduced because of the change. The risk is that the DarkSky data may not match well against the weather information obtained from the competition website.This risk is identified as a scope of improvement for the project.

For our Beijing weather feature engineering from the DarkSky data, we determined a few key aspects to consider:

- **Zhiwuyuan air quality station (zhiwuyuan_aq) has numerous null columns.** This included PM10_Concentration, PM25_Concentration, O3_Concentration, SO2_Concentration, NO2_Concentration, CO_Concentration. It was not possible to remove the station because it was a part of the competition's required predictions.
- **Some DarkSky weather data had nulls until 4/23 10am.** These features included visibility, uvIndex, windGust, ozone, precipIntensity, precipProbability, pressure are null until after 4/23 10am.

London DarkSky weather data had a number of challenges to it as well:

- **Weather features did not have values until after 4/22 9pm.** These features included uvIndex, windGust, ozone, precipIntensity, and precipProbability.
- **Air quality station C9 had several anomalous values at 4/25 3pm.** Some features were from 10 times to 100 times larger than all other values.
- **A few air quality stations had nulls for the target variables PM10 and PM2.5.** BX9, CT2, and TD5 had mostly null values for PM10_Concentration, while BX1 had mostly null values for PM25_Concentration.

Resolving Null Values

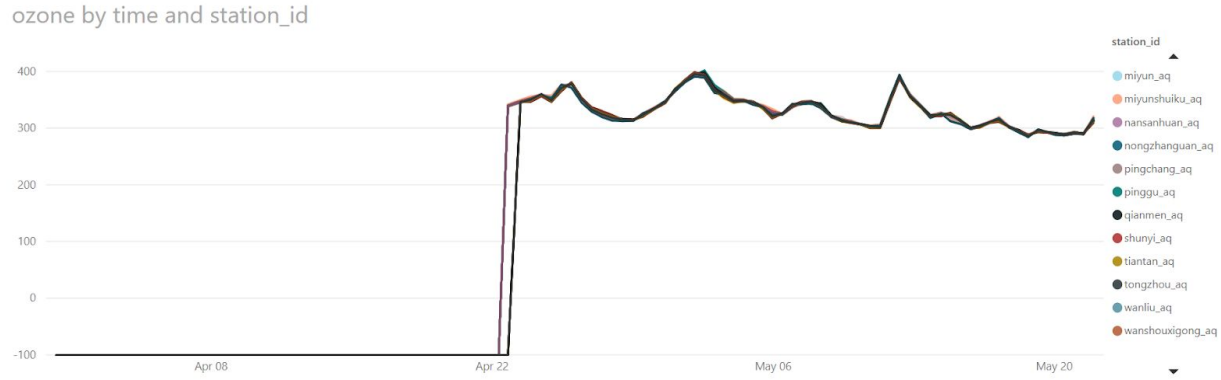
Nulls revealed through ad-hoc inspection were resolved through two main techniques.

- **Data that was complete for one city, but not another**. DarkSky windBearing and windSpeed were null only for Beijing, so we "split" our pipelines to let us treat each city uniquely, such that wind bearing and speed could be included for London and dropped for Beijing.
- **Consistently absent categorical value.** Precipitation, a categorical variable whose original values consisted of "rain," "snow", and "null". Observing related variables, precipIntensity, precipProbability, we were able to determine "null" was equivalent to "none."

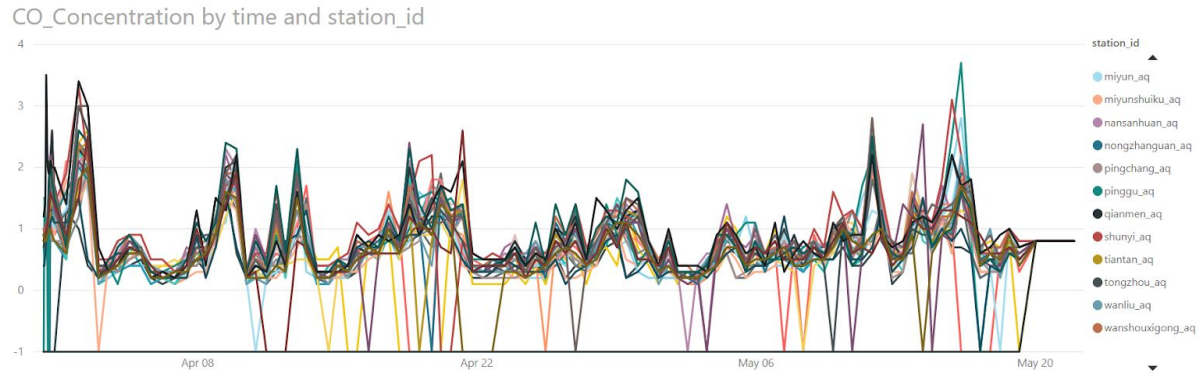
Nulls revealed though line plot exhibited three main behaviors, each with its own response.

Data with a few scattered "holes". Sometimes, a variable for a given weather station would "skip" for a given time slot and return null. In most cases, the best approach would be to take the average of the previous and subsequent values on either side of the "hole."

Data that was null until a certain date. Some variable yielded all nulls until a certain date, such as April 23. The response is, if we wish to use this variable as an input, is to base our predictions only on observations that come after this date.



Dead stations. Some stations, such as zhiwuyuan_aq, yielded no data whatsoever. In these cases, the best response is to use that station's "nearest neighbor" based on our geospatial analysis.



A complete set of anomalies and heuristics for resolving them is available in the Appendix.

In order to leverage more variables based on our feature engineering, Leo made the following modifications in the leo_v2 branch on the DS420-Cobras repository:

1. Rather than loading data from the beginning of the competition, start at 4/23/2018 10:00 UTC for Beijing and 4/25/2018 21:00 UTC for London.
2. Enable the following features for Beijing: ozone, precipIntensity, precipProbability, pressure, windGust
3. Enable the following features for London: ozone, precipIntensity, precipProbability.

Unfortunately, these changes did not yield sufficient performance in our feature-driven models (i.e. RandormForest, etc) to beat our mean or median based approaches. Moreover, mean/median performance degraded slightly when sampling values after late April rather than the full data set.

Keras/Neural Network. Towards the end of the competition window, David explore the possibility of using a Neural Network model to predict the air quality variables. This required some data transformation in order to fit within the requirements for a Neural Net using Keras.

For the weather features, the features were transformed similarly to what was done for the SMAPE model:

- Dropping features with numerous null values.
- Transforming our datetime columns into hour, day, month, and day of week.
- Dropping unneeded pollution features.
- Creating dummy columns for categorical columns.

In addition to the feature engineering mentioned above, I scaled our numeric features using MinMaxScaler from Scikit-Learn. This is required for building a Neural Net in order to keep the Gradient Descent value changing at a reasonable pace.

We performed exactly the same transformations on the submission data as we did on the training and testing data.

FEATURE AND MODEL SELECTION

Early in our analysis we noticed that ARIMA time-series analysis did not identify strong trends or strong periodic behavior. Combined with the lack of modeling flexibility in ARIMA analysis, we decided on not using ARIMA style time-series modeling. We decided to use techniques like linear regression, random forest, and neural networks. That decision influenced our feature engineering.

For each of the pollutants that we needed to make predictions, we created independent models. We employed 5 fold cross validation to measure the performance of our models. Most of our models did not have hyperparameters. So, we needed just the training and test set. We created a number of simple models for our analysis:

1. Means - Mean value of the pollutant since the time data was available
2. Smape - Constant value that would minimize the smape error for a given station. Each station had a unique value
3. Random forest - Output of the default random forest model in Python

4. Median - Median value of the pollutant since the time data was available
5. Lasso - Output of the default lasso model in Python
6. Mean-Median Ensemble - Average of the mean and median values for each of the stations
7. Station-Hour median - Median value of each station at the specific hour

Our experience with the leaderboard score indicated that the Means, smape, Median, and Mean-Median ensemble model gave the best results.

David explored XGBoost and Neural Networks (with Keras) as alternative regressors. The real challenge of taking advantage of these models was determining the necessary features to produce a tenable model. Features with excessive NaNs were removed from our model training, which reduced a useable data quite a bit. There was some opportunity to fill the NaNs with other features. For example determining "cloudCover" from the over weather "summary" for the day (clear, rainy, etc.). Out-of-the-box XGBoost's regressor did not improve our SMAPE score, but it was decided to refocus on feature engineering rather than parameter tuning. David also explore the option of using a Dense Neural Net with Keras. The SMAPE score on our test set was remarkably improved, but once submitted to the BienData.com website, our results were far worse than expected. This is likely due to overfitting on our training/test data.

MODEL PERFORMANCE

We are 57th on the leaderboard with a weighted score of 0.4578 .

54	—	613A	0.60	0.48	0.48	0.45	0.45	0.7083	0.4544	58
55	—	我们讲道理	0.80	0.52	2.00	0.49	0.60	0.6247	0.4565	79
56	—	MO	0.65	0.48	0.51	0.49	0.53	0.5275	0.4567	88
57	—	DS420-Cobras	0.68	0.60	0.62	0.57	0.49	0.5799	0.4578	81
58	—	UKROP	0.68	0.50	0.61	0.50	0.56	0.6095	0.4586	76
59	—	Jing_KDD	0.70	0.57	0.55	0.56	0.50	0.4911	0.4594	93

We used R-squared error and SMAPE error functions as explained below.

We, initially, used R-square error as the performance metric as that was the default metric for many of the modeling techniques (linear regression and random forest). As per the performance metric, lasso regression vastly outperformed simple mean prediction and the random forest outperformed lasso. We submitted the results containing the means for each of the station (which was our placeholder), lasso predictions for each station, and random forest predictions for each of the stations. To our surprise, just simply predicting the means outperformed lasso and random forest on the leaderboard!

In order to figure out the reason for this, we did the SMAPE error calculations based on the models. We saw that the mean prediction outperformed lasso and random forest!

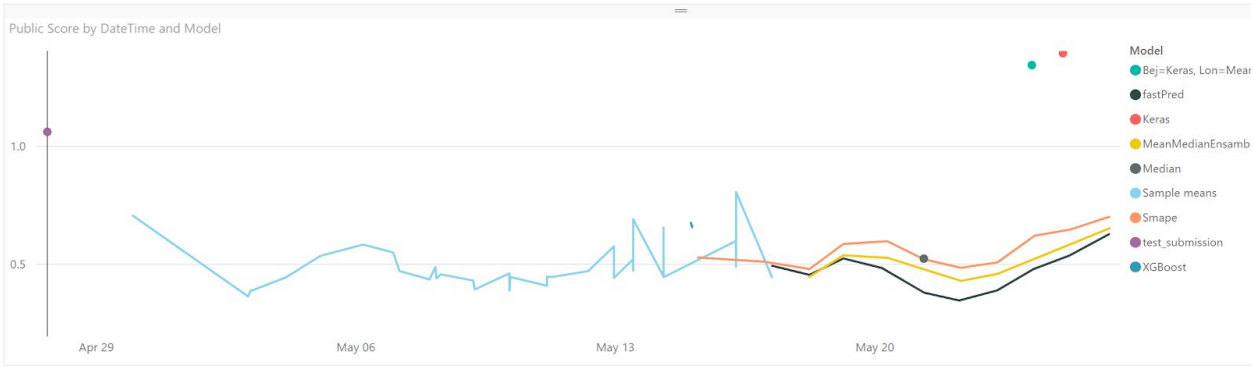
My interpretation of the anomalous behavior was that due to lack of features (or feature engineering) we did not have strong predictors for the pollutants. Also, the R-squared error is very different from SMAPE error. So, while Lasso and Random forest are designed to minimize the R-squared error, they are not particularly good with SMAPE error.

We decided to invest our energies into improving the mean predictions. We created a simple SMAPE minimizer algorithm that predicts a single value for each of the station that minimizes the SMAPE error on the test set. This model sometimes outperformed the mean prediction.

We noticed that the values predicted by the SMAPE model above was very close to the median values. We, therefore, created a model that predicted the median values for each of the stations.

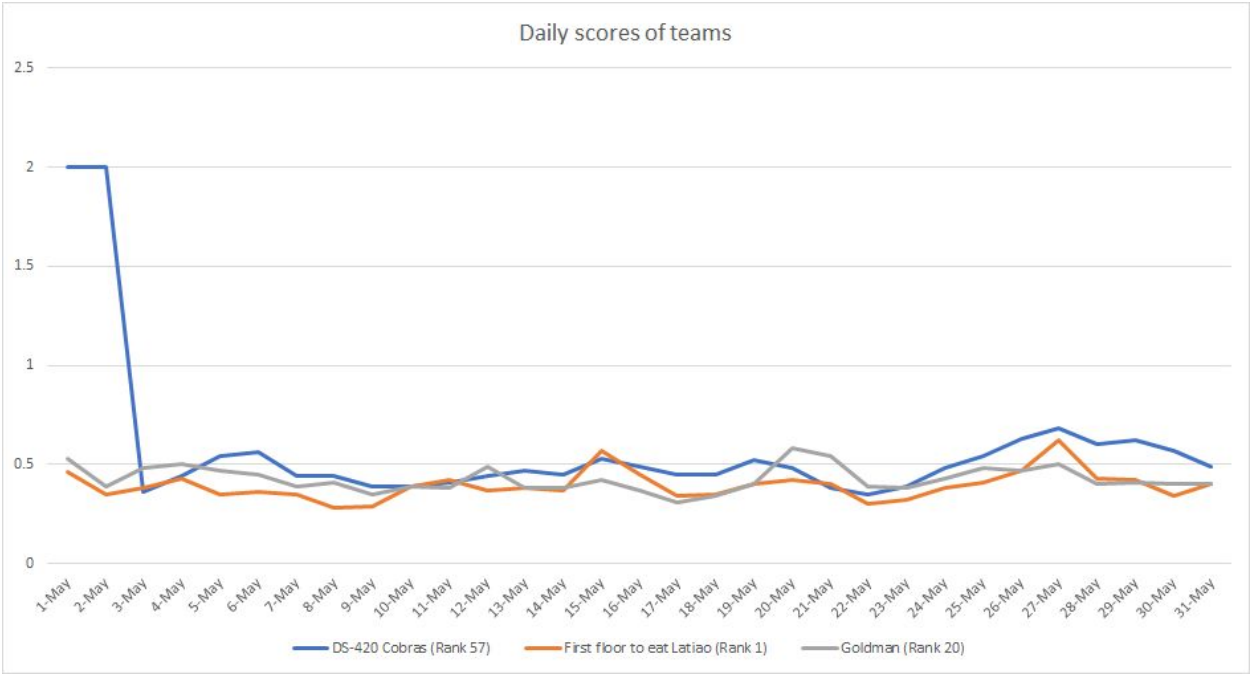
We, thought, that we could do better by computing median for each stations for every hour of the day. However, that model underperformed others significantly.

We had a lot of hopes with the XGBoost and Keras models. However, those models did not perform well on the leaderboard.



During the competition, we figured that the K-fold split should not be random (we were using random splits earlier). We used the first 4 folds to train the model and 5th fold to evaluate it. These models vastly underperformed on the leaderboard for several days in the middle of the competition. We course corrected and began using the random shuffle before performing K-fold splits. Among the 5 models that we got, we always chose the model with the smallest smape error.

During the last 10 days of the competition, we decided on using the moving average instead of average of the entire dataset. In retrospect, we feel that this was a wrong decision. The leaderboard score dipped after we made the change. This is apparent in the chart below where we are plotting the leaderboad scores of our team vs team Rank 1 vs. team rank 20. Note that the mean predictions performed close to the leading teams for the first ⅔ of the competition. Also note that we outperformed the top teams for three submissions and were close to them for many of them (barring the non-randomization experiment in the middle and the moving average mean experiment at the end).



As mentioned earlier, the models predicting the mean, median, and average of mean and median outperformed all other models in our daily submissions. There was no clear winner among the three models.

WHO DID WHAT?

This entire code and github history of the code is available here: https://github.com/DS420-Cobras/DS420-Cobras .

Leo Salemann served as the Project Leader. He established the team on the KDD Cup website, along with a a GitHub environment at https://github.com/DS420-Cobras. On GitHub, he established the basic code repository, README.md file, a GitHub team with chat feature, a GitHub project board with issues, and a Team space with a slack-like Chat functionality. Leo also performed the geospatial and null-value portions of the data exploration phase, and executed nightly runs to keep our submissions in good standing.

Yash Bhandari served as the software developer for the project. He wrote the baseline code that took inputs from the load weather and load pollution data code and called the submit script. Each of the team members contributed their data exploration, feature engineering, and feature modeling insights into this code.

Venkata Vemulapalli served as the Data Engineer for the project. I wrote the code for load_weather function which takes startDate, endDate, stations Needed, cityName, False as input arguments and return weather data in df. I also experimented with feature selection and IDEAR.ipynb.

David Ghan served as the Model Engineer. He wrote Python function submit_preds for finishing our data pipeline by streamlining our submissions according to the KDD Cup requirements. He explore different models as well as using various loss functions to attempt to achieve competitive SMAPE scores on the test data.

RETROSPECTIVE

[from Hang's ppt]:Anything that you think if you have done differently, you might have achieved better results?

The code repo was used constantly, and the chat functionality was used more than email. The project board was effective in defining initial tasks, but things became a bit more ad-hoc in the later stage of execution.

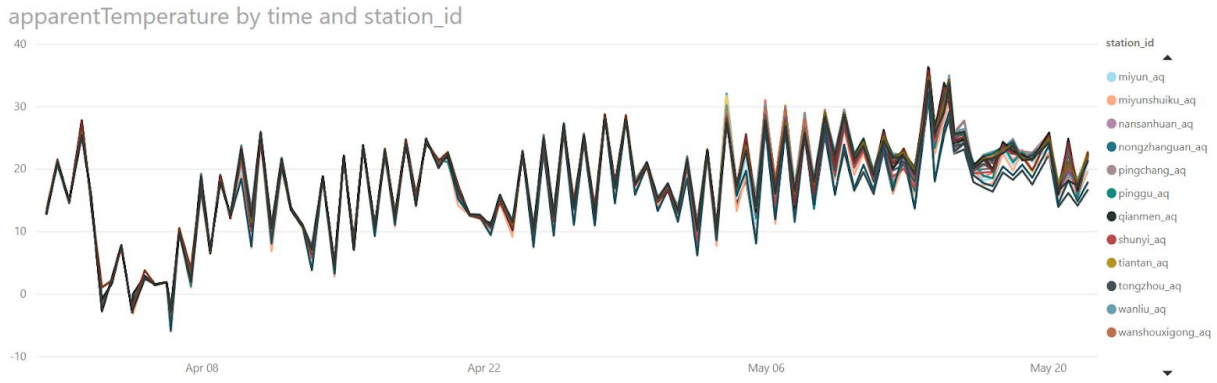
One of the things that we could have done better was accurately reproducing the leaderboard error through our submissions. The error prediction that our error function made did not match that of the leaderboard. Had we performed this experimentation before the competition started, we would have a much better command over feature engineering and modeling.

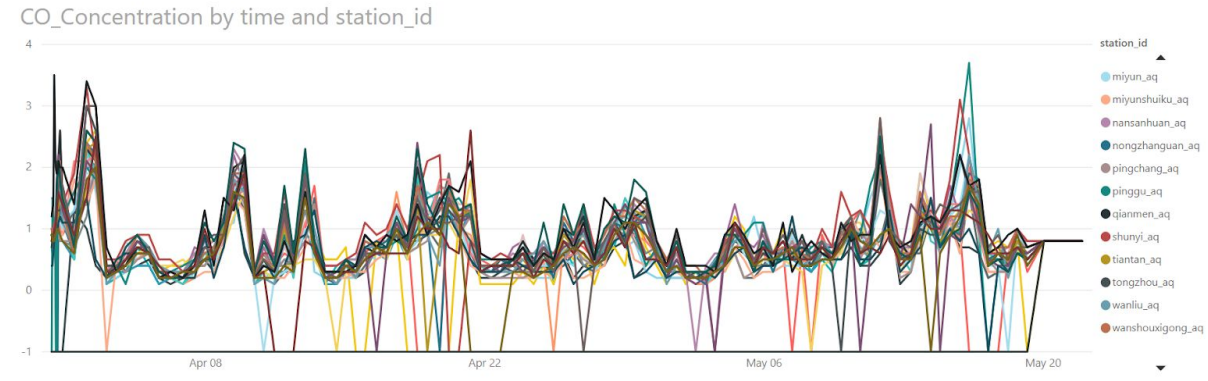
The modeling techniques that we used, like lasso, random forest, and neural networks are designed to minimize the R-squared error. Therefore, they performed poorly on the leaderboard. We could have derived SMAPE error minimization Lasso and Random Forest to get better results.

We never verified the weather measurements from DarkSky.net against the values we got from the competition website. If the DarkSky weather information is wrong/inaccurate, then that would explain the lack of good performance of the various machine learning models (like lasso, random forest, and XGBoost).

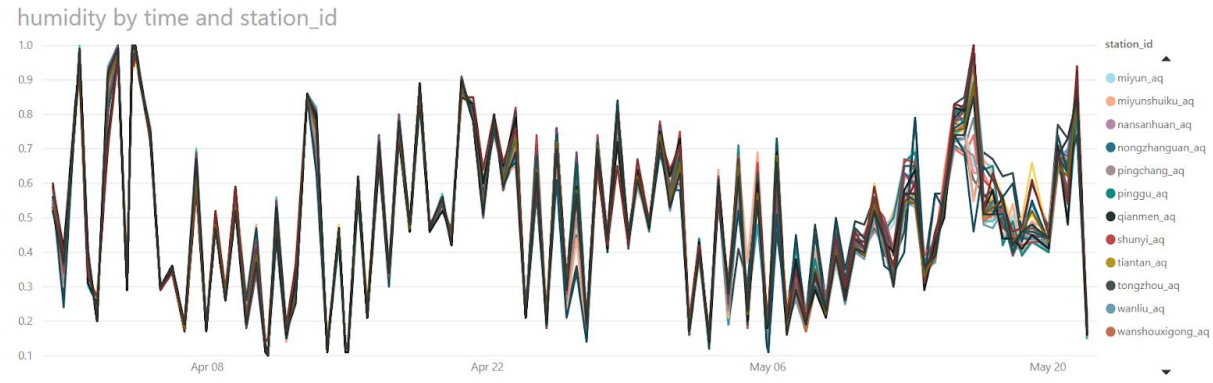
APPENDIX

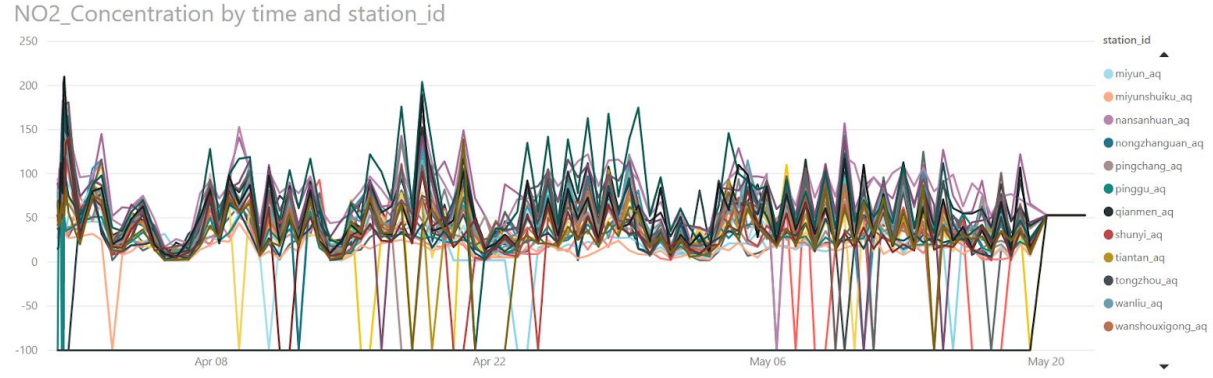
Line Plots with Sentinel Values, Beijing

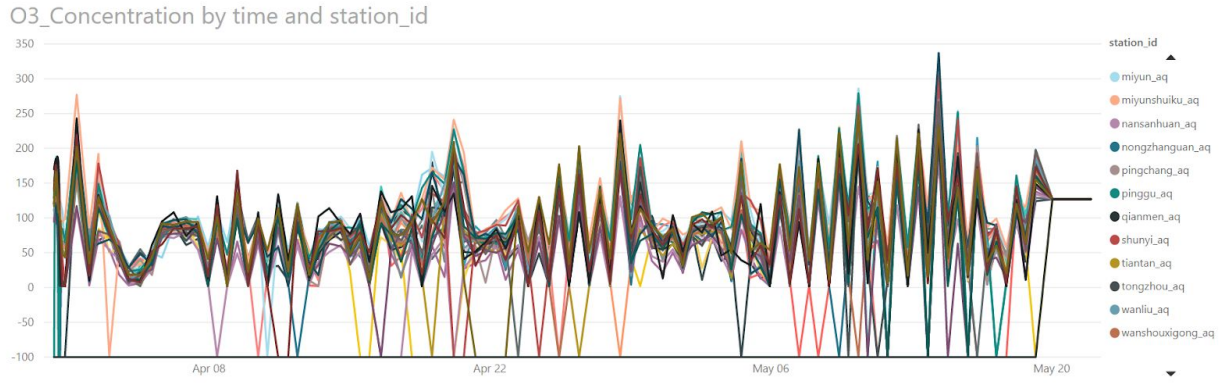


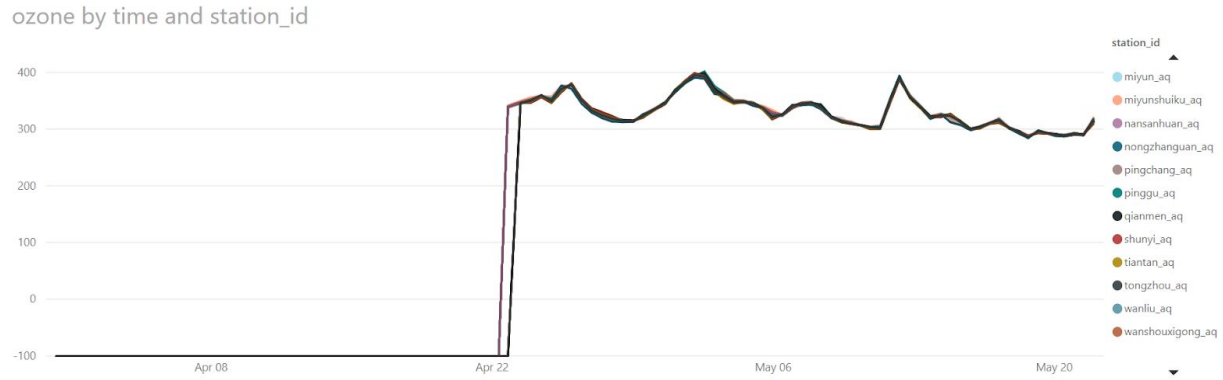


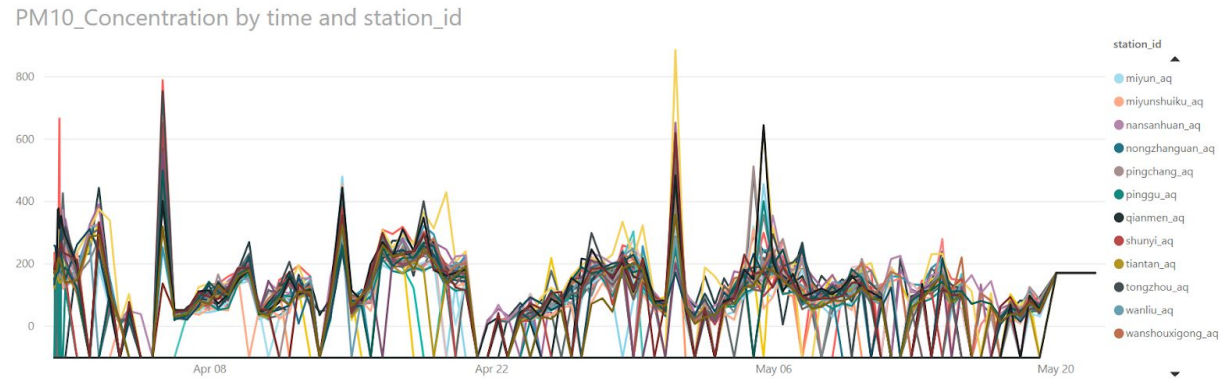

dewPoint by time and station_id
25
20
15
10
5
0
-5
-10
-15
-20
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

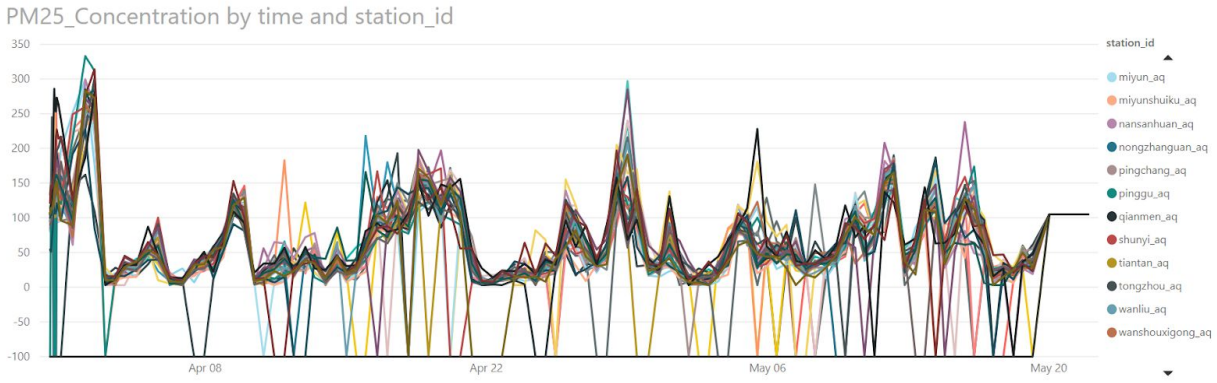

humidity by time and station_id
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

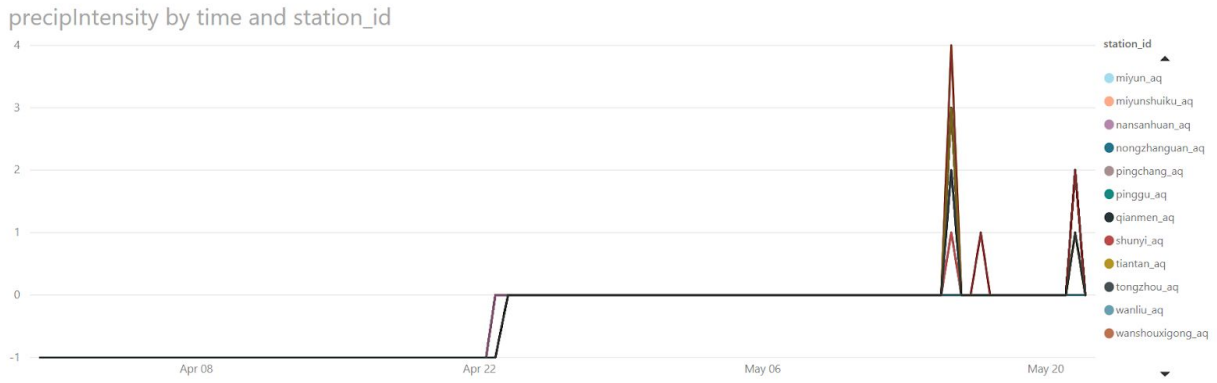

NO2_Concentration by time and station_id
250
200
150
100
50
0
-50
-100
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq


O3_Concentration by time and station_id
350
300
250
200
150
100
50
0
-50
-100
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

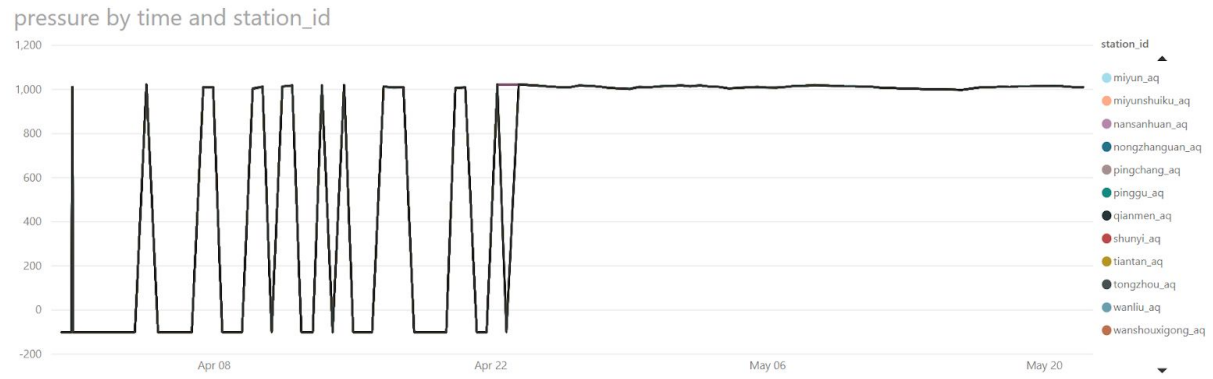

ozone by time and station_id
400
300
200
100
0
-100
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

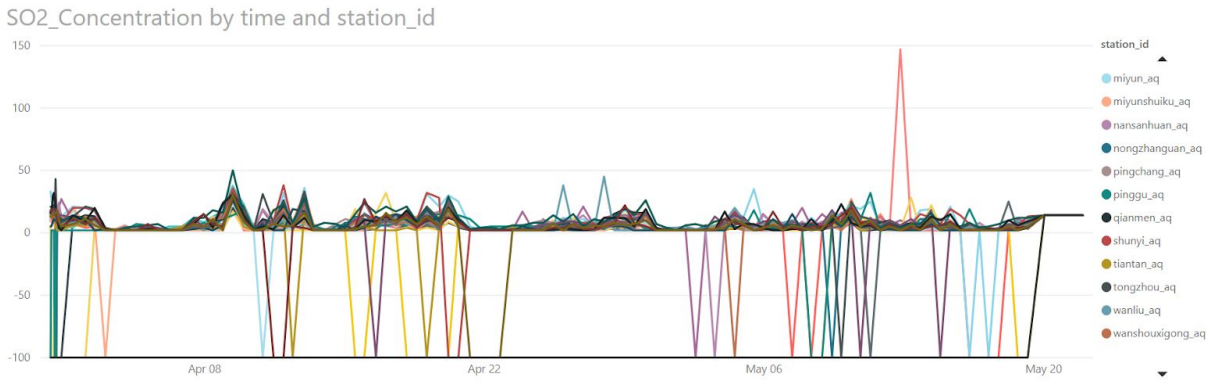

PM10_Concentration by time and station_id
800
600
400
200
0
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

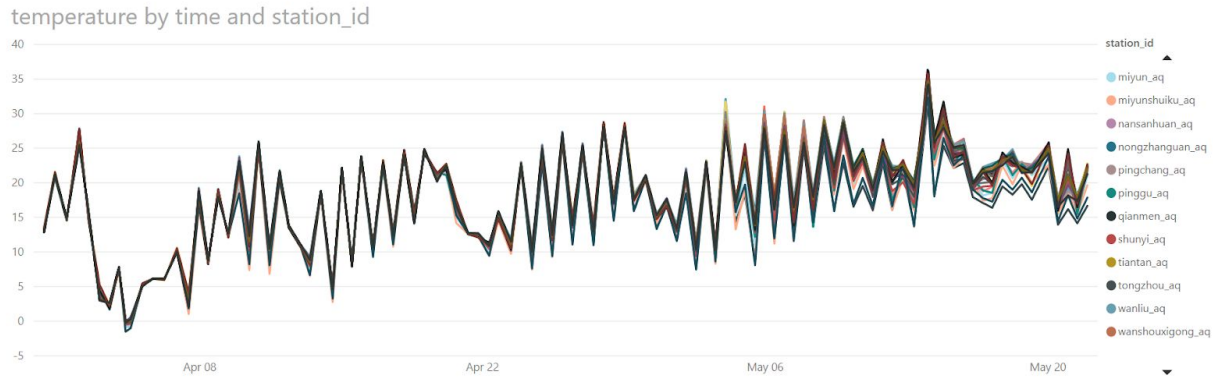

PM25_Concentration by time and station_id
350
300
250
200
150
100
50
0
-50
-100
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

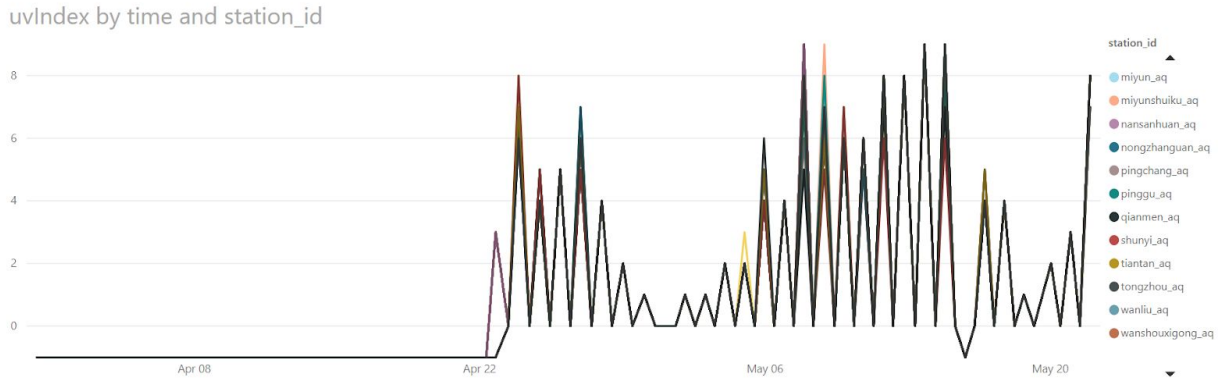

precipIntensity by time and station_id
4
3
2
1
0
-1
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

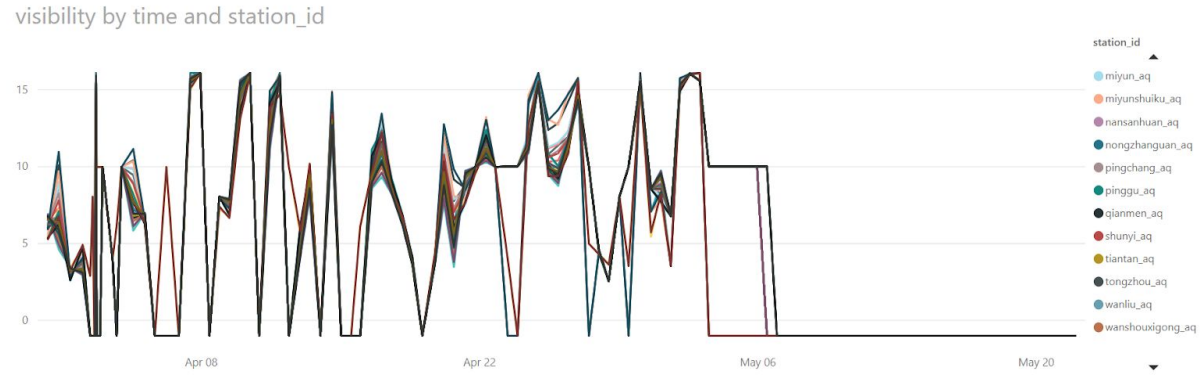

precipProbability by time and station_id
1
0
-1
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

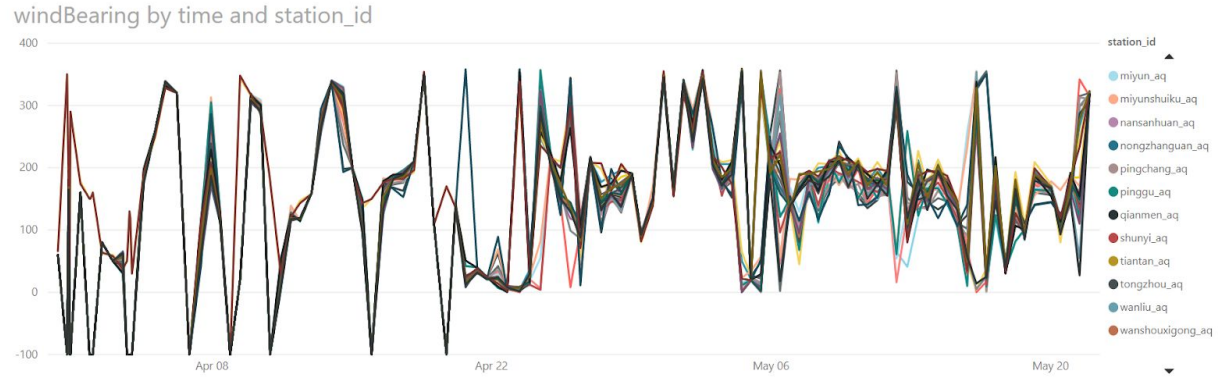

pressure by time and station_id
1,200
1,000
800
600
400
200
0
-200
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

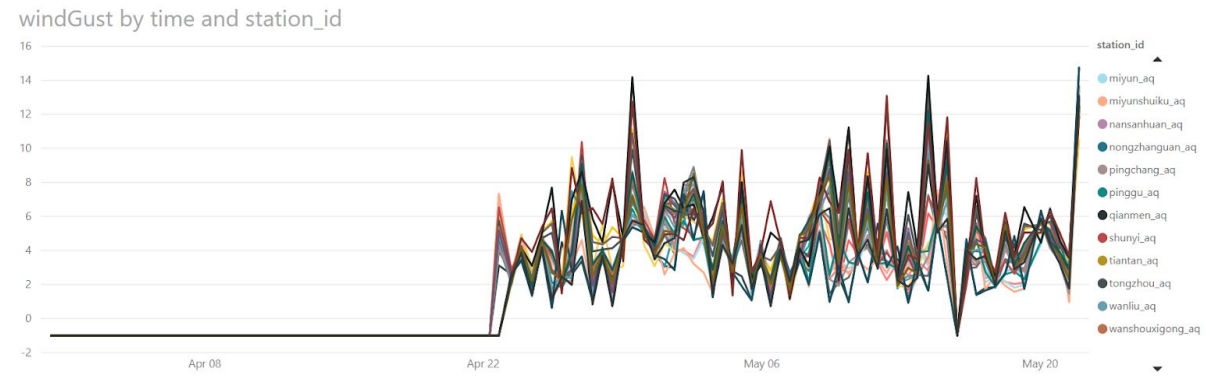

SO2_Concentration by time and station_id
150
100
50
0
-50
-100
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq

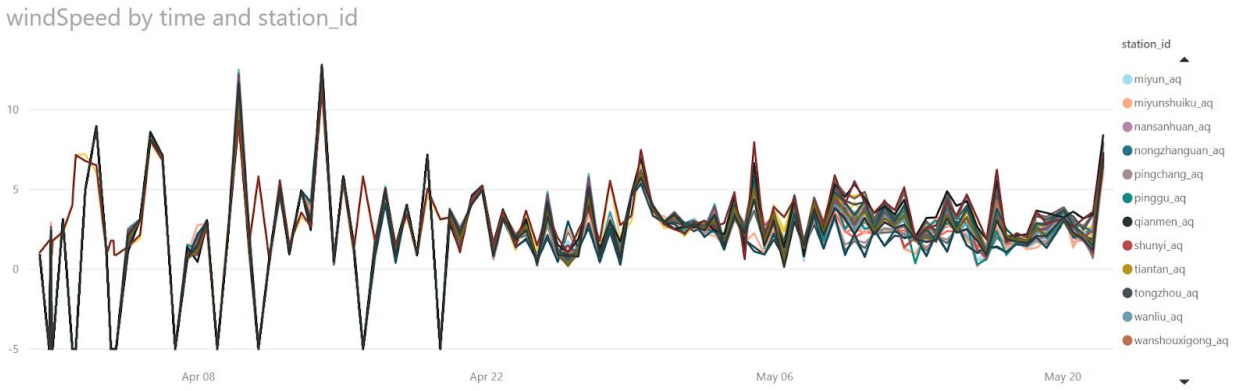

temperature by time and station_id
40
35
30
25
20
15
10
5
0
-5
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq


uvIndex by time and station_id
8
6
4
2
0
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq


visibility by time and station_id
15
10
5
0
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq


windBearing by time and station_id
400
300
200
100
0
-100
Apr 08
Apr 22
May 06
May 20
station_id
miyun_aq
miyunshuiku_aq
nansanhuan_aq
nongzhanguan_aq
pingchang_aq
pinggu_aq
qianmen_aq
shunyi_aq
tiantan_aq
tongzhou_aq
wanliu_aq
wanshouxigong_aq





Null-Value Heuristics

City	Feature	% null	Behavior	Heuristic
Beijing	precipType	46%	n/a	replace null values with the word 'none' (no quotes)
Beijing	visibility	45%	No good data after May 6 11pm	for each station: replace null with average of prior and subsequent non-null value for same station
Beijing	uvIndex	43%	No good data until Apr 22 8pm	for each station: replace null with average of prior and subsequent non-null value for same station
Beijing	windGust	43%	No good data until Apr 22 8pm	(after Apr 22 8pm) for each station: replace null with average of prior and subsequent non-null value for same station
Beijing	ozone	42%	No good data until	Everything's fine from Apr 23 10am

			Apr 23 10am	onward
Beijing	precipIntensity	42%	No good data until Apr 23 10am	Everything's fine from Apr 23 10am onward
Beijing	precipProbability	42%	No good data until Apr 23 10am	Everything's fine from Apr 23 10am onward
Beijing	pressure	27%	No good data until Apr 23 10am	Everything's fine from Apr 23 10am onward
Beijing	cloudCover	20%		If summary field says Foggy then set cloudCover to 1.0; otherwise set to 0.0
Beijing	PM10_Concentration	20%	zhiwuyuan_aq is all null; others have some scattered holes	Predict zhiwuyuan_aq's values based on those of its nearest neighbor. For others, replace null with average of prior and subsequent non-null value for same station
Beijing	PM25_Concentration	5%	zhiwuyuan_aq is all null; others have some scattered holes	Predict zhiwuyuan_aq's values based on those of its nearest neighbor. For others, replace null with average of prior and subsequent non-null value for same station
Beijing	O3_Concentration	5%	zhiwuyuan_aq is all null; others have some scattered holes	Predict zhiwuyuan_aq's values based on those of its nearest neighbor. For others, replace null with average of prior and subsequent non-null value for same station
Beijing	SO2_Concentration	4%	zhiwuyuan_aq is all null; others have some scattered holes	Predict zhiwuyuan_aq's values based on those of its nearest neighbor. For others, replace null with average of prior and subsequent non-null value for same station
Beijing	NO2_Concentration	4%	zhiwuyuan_aq is all null; others have some scattered holes	Predict zhiwuyuan_aq's values based on those of its nearest neighbor. For others, replace null with average of prior and subsequent non-null value for same station

Beijing	CO_Concentration	4%	zhiwuyuan_aq is all null; others have some scattered holes	Predict zhiwuyuan_aq's values based on those of its nearest neighbor. For others, replace null with average of prior and subsequent non-null value for same station
Beijing	windBearing	4%	Mostly-good data for whole timespan; just some scattered holes	for each station: replace null with average of prior and subsequent non-null value for same station
Beijing	windSpeed	4%	Mostly-good data for whole timespan; just some scattered holes	for each station: replace null with average of prior and subsequent non-null value for same station
London	uvIndex	44%	no good values until 4/22 9:00pm	for each station: replace null with average of prior and subsequent non-null value for same station
London	windGust	44%	no good values until 4/22 9:00pm	Station CD9 has an anomalous value at 4/25 3pm. Replace with average of predecessor and successor
London	ozone	44%	no good values until 4/22 9:00pm	Station CD9 has an anomalous value at 4/25 3pm. Replace with average of predecessor and successor
London	precipIntensity	44%	no good values until 4/22 9:00pm	all values on 4/22 9pm and after ate fine.
London	precipProbability	44%	no good values until 4/22 9:00pm	all values on 4/22 9pm and after ate fine.
London	precipType	38%	n/a	replace null values with the word 'none' (no quotes)
London	NO2_Concentration	32%	BX9 CT2 KF1 MYZ TD5 should be dropped. Maybe drop CT3	For the stations mentioned, use values from nearest neighbor. For others, replace with average of predecessor and successor
London	PM10_Concentration	26%	BX9 CT2 TD5 should be dropped	Use values from nearest neighbor.
London	cloudCover	7%		If summary field says Foggy then set cloudCover to 1.0; otherwise set to 0.0

London	visibility	0%	Station CD9 is anomalous	Station CD9 has an anomalous value at 4/25 3pm. Replace with average of predecessor and successor