approach

November 11, 2019

1 Approach

I started by thinking which variables affect air quality, and split those into sources of pollution, and causes of dispersion

sources of pollution: - pollution from cars that intersection (car.count) - pollution from traffic/industrial activity in other areas of the city (congestion model, hours since sunrise) - pollution coming in from other regions by wind

causes of dispersion: - wind (velocity and direction) - temperature gradient between altitudes

1.1 variables used

I scaled all of the following variables to minimize the p-value, with out choosing any power greater than 3 or smaller than .3333; i also made use of log transforms. The reasoning for doing all of these transforms was more or less the same: I am not too familiar with the physics of how these variables interact, and because I didn't have a strong opinion one way or the other, I went with what the data says, within reason.

`car.count`
`wind.velocity`
`temperature.30`
`temp_diff` (`temperature1 - temperature30`)

- the absolute value of the difference in temperatures between 1m and 30m
- the reasoning behind adding this interaction term is that it is affected by cloud cover, precipitation, and humidity, both of which may influence so2 dispersion, and may be indicative of previous wind patterns
- additionally, it may be indicative of negative/positive pressure that may keep pollution close to the ground, or suck it up into the atmosphereto dissipate

absolute value of `temp_diff`

- similar reasoning as above

`total_congestion`

- I wrote a simple congestion model taking into account the day index and time of day to estimate the congestion/industrial activity in the city

- the model took into account differences in activity between months, days of the week, and hours in the day between weekdays and weekend
- interestingly, the congestion model seemed to have stronger predictive value than car.count; I included these two charts at the end of my code
- i used the data from this article: https://www.wbez.org/shows/curious-city/when-is-chicagoarea-traffic-the-worst/73cf66f9-8a44-4227-8619-68220de78ad3

`since_sunrise`

- the hours since sunrise
- the reasoning behind including this is that it gives us another approximation of how much time has passed since morning rush hour, and the start of most industrial activity for the day

`sin`

- sin of angle of wind direction, using north as 90 degrees
- reasoning for transforming wind direction to sin and cos component is that intuitively because the numbers loop back on themselves, and it doesn;'t make sense to treat 359 degrees as wildly different than 2 degrees

`cos_abs`

- absolute value of the the angle of wind direction, using north as 90 degrees
- i chose to use the abs value of the cos simply because it seemed to have a high p_value, which was surprising to me; if anything, i would have expected the opposite (that abs(sin) and cos would be the 2 with a high pvalue)

2 other variables

I opted to not include some given variables that I deemed redundant after calculating the interaction variables

`temperature.30`

- i used this one to calculate temp_diff, but opted to drop it in the final model , as much of that information was already captured in `temp_diff`, `temp_diff_abs`, and `temperature.30`

`day.index`

- I used this to calculate `total_congestion` and `since_sunrise` , and so dropped it, as much of the inherent information was already captured

`time.of.day`

- same reasoning as above

`wind.direction`

- this was already accounted for in sin and cos_abs, so felt it was unnecesary to include it

3 scaling features

I started by sorting all 9 features by the pvalue of their regression on `so2`, in descending order.

I fit the first feature (`total_congestion`) to `so2`, experimenting with different scaling functions (log, and powers between .33 and 3)

For each of the following features, I followed the following steps:

- re-sort the features by their pvalue (descending order) against the residual of all previously encountered features (including `total_congestion`)
- experiment with scaling factors (log, and powers between .33 and 3)
- scale each feature by scale resulting in lowest pvalue, and update the residual to include this newly encountered feature

4 handling outliers

Before scaling the features and fitting the model, I dropped all rows that had an `so2` reading above 150

5 time estimate

this assignment took me most of the weekend

6 run instructions

this project was done in a jupyter notebook (`.ipynb`), though i also saved a less aesthetically pleasing copy as a standard `.py` file