Pedretti Beatrice, 943805

Regression assignment

1. Preliminary analysis: loading and checking data

Firstly, I imported some useful tools and loaded the .csv file in the Jupiter Notebook, where I rapidly looked at some statistical properties of the dataset thanks to the functions data.head(), `data.describe()` and `data.info()`.

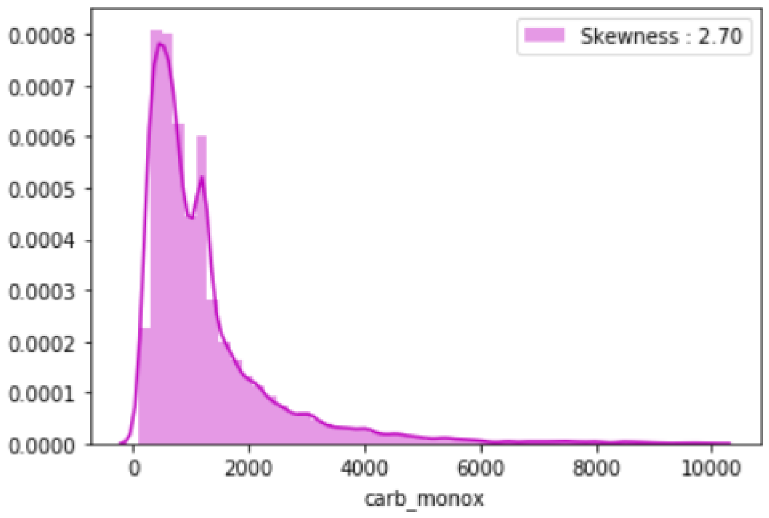
From this preliminary analysis I was able to observe that:

- The dataset has 14000 observations
- The explanatory features are mostly numerical
- There are some missing data

2. EDA

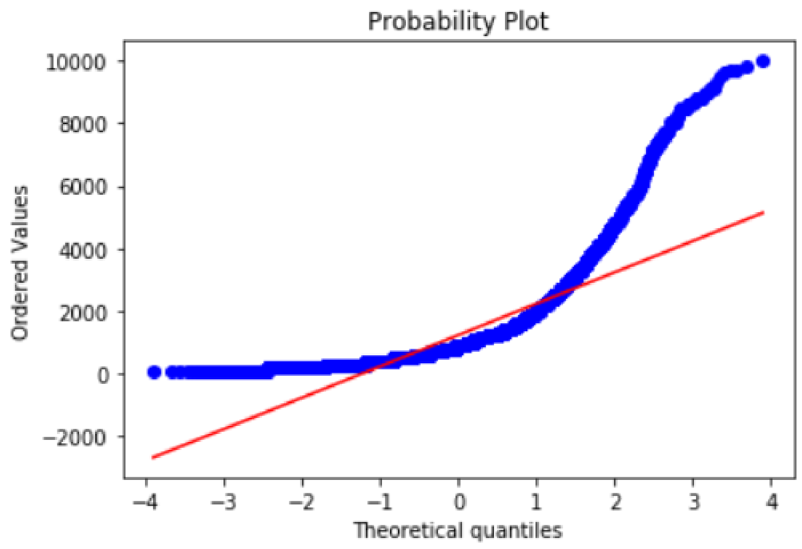
‘carb_monox’

By plotting the target variable we can infer that carb_monox deviates from the normal distribution, that it has appreciable skewness, and that it shows peakedness, with values of Skewness= 2.695517 and Kurtosis=9.883100.



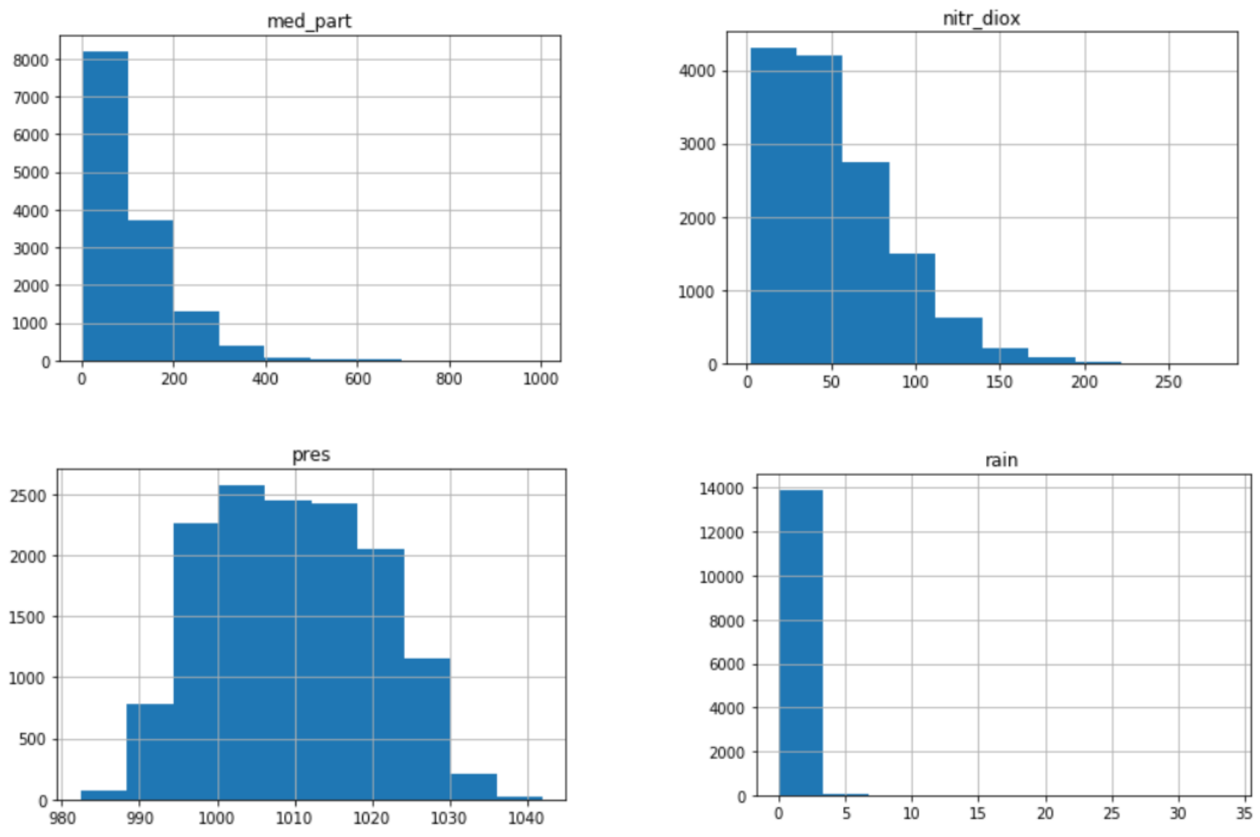
I debated whether to apply a logaritmic transformation or not, but eventually decided not to because transforming the variable did not seem to improve my prediction. Moreover, it seemed pointless to transform the variable, only to apply the inverse transformation after modeling.

It is also evident in the QQ plot how the distribution is far from being normal:



Explanatory variables

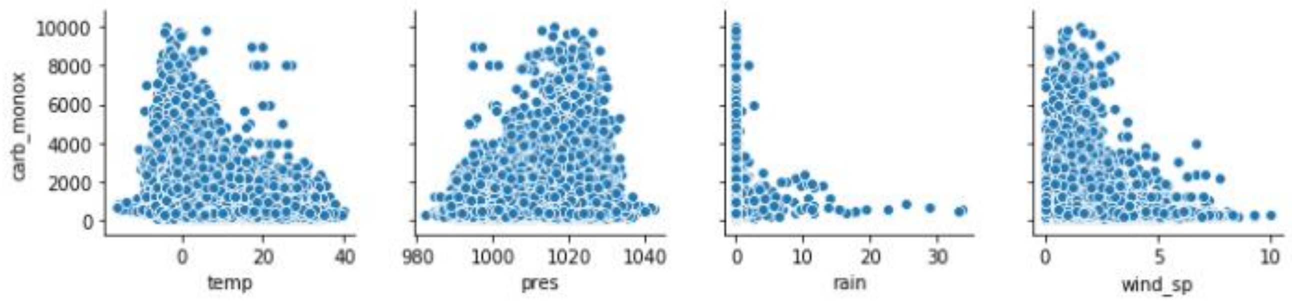
By analyzing the distribution of the explanatory variables, we notice that some features (such as 'sulf_diox', 'small_part', 'med_part', 'nitr_diox') show a decreasing curve (maybe exponential?), while 'pres' seems to have an almost Gaussian distribution and 'rain' follows a Pareto distribution.

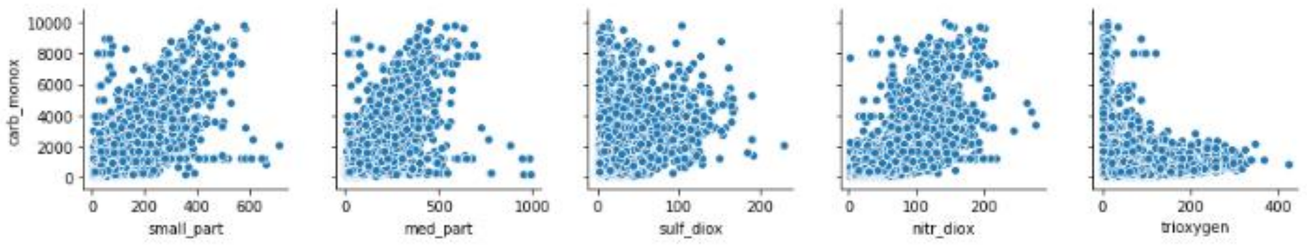


Multivariate analysis and correlation

With the help of scatterplots I observed that the only variables to have something similar to a linear correlation with our target are maybe ‘small_part’, ‘med_part’, and ‘nitr_diox’, which are the pollutants most commonly used to compute the Air Quality Index (AIQ), so it isn’t surprising that these features are the most correlated with ‘carb_monox’. However, I expected the “environmental” features, such as ‘temp’, ‘pres’ and ‘rain’ to show a higher correlation to our target.

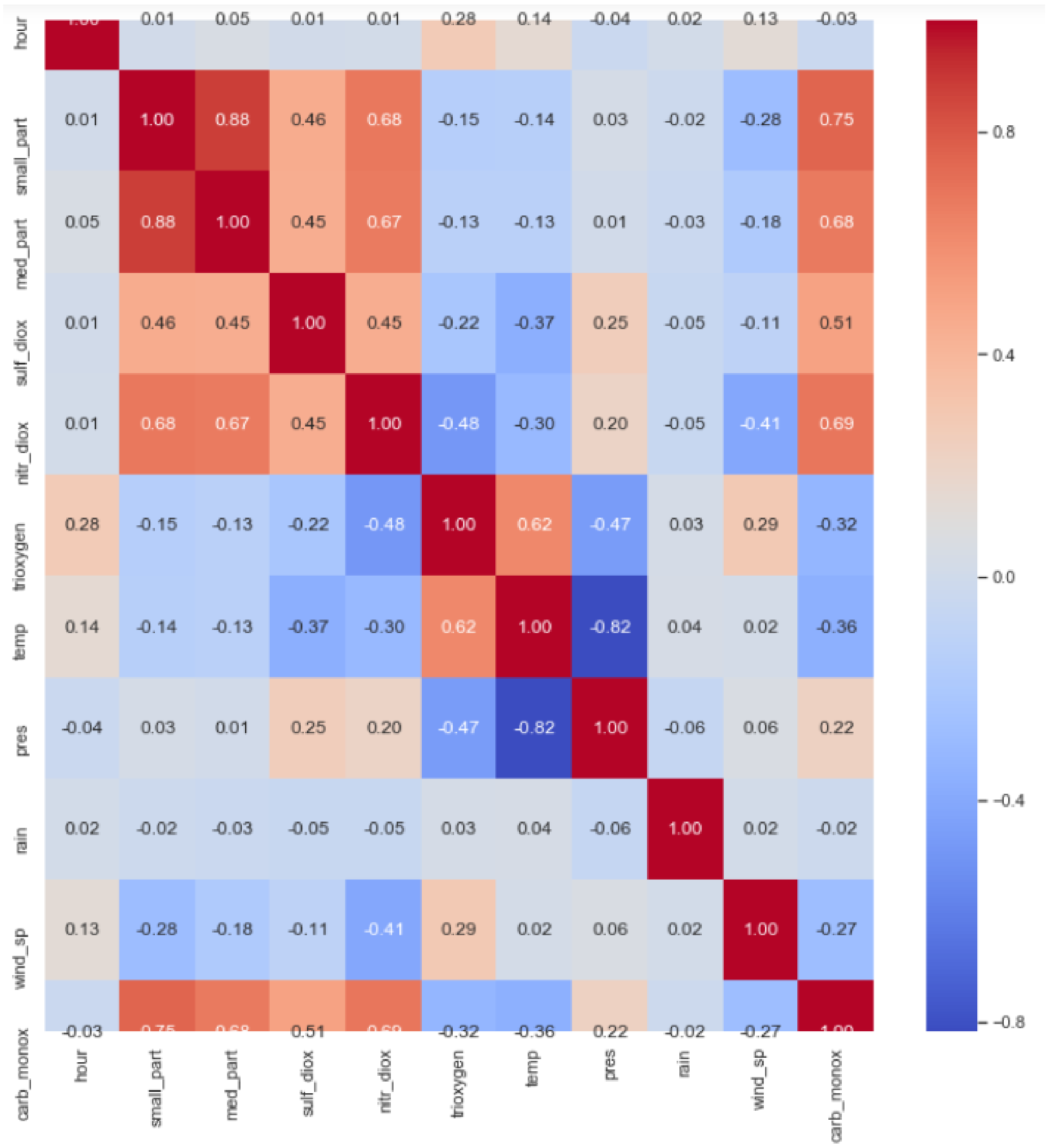
Here we can see the scatterplots of the majority of the explanatory variables, compared to ‘carb_monox’:





From the heatmap (next page) it is apparent that:

- As one could easily expect, ‘small_part’ and ‘med_part’ show multicollinearity. This is why during the feature engineering phase I have dropped one of these features entirely
- All the “pollutants” variables are correlated between each other – but are they correlated enough to fall into multicollinearity?
- ‘temp’ and ‘pres’ are highly negatively correlated; I will drop one of them too.



Outliers

Since outliers can have a dramatic effect on the prediction, at first I decided to implement functions for outlier detection: I tried with Tukey's method but I was not sure about its effectiveness, since the data is not normal. So I tried applying a logaritmic transformation on some explanatory variables, but eventually decided to look for outliers in a different way, since with log transformation and Tukey's method I witnessed a loss of generality in my regression model.

Then, I tried dropping outliers manually just by observation of the scatterplots, but decided not to proceed this way because I considered this method too empirical and not robust enough.

Eventually, I judged it would not be wise to drop outliers in this particular dataset, after I performed this further evaluation of my model:

1- After feature engineering, I set aside a small portion of the "train.csv" dataset (1000 observations to be hidden were chosen randomly)

2- I splitted the remaining 13000 data into two sets with the function `train_test_split`

3- I trained and fitted the model using these two sets

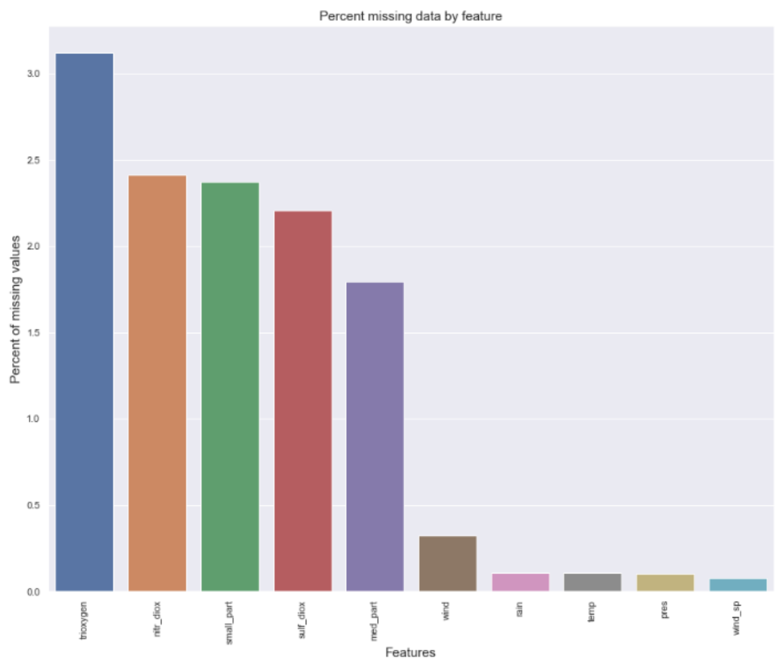
4- At last, I used my model to predict the target variable of the 1000 observations I had hidden before, and computed the MAE score on these values too

5- I kept the model with the most consistend MAE score between all three sets; this most consistent model was obtained by not dropping any outliers in the training set

Missing data

As a rule of thumb, variables are usually dropped where at least 10%-15% of the observations are missing, so I decided to start by computing the percentage of missing data in the whole dataset:

	Missing Ratio
trioxygen	3.121429
nitr_diox	2.414286
small_part	2.371429
sulf_diox	2.207143
med_part	1.792857
wind	0.321429
rain	0.107143
temp	0.107143
pres	0.100000
wind_sp	0.078571



‘trioxygen’ has the highest ratio of missing data, but it is only 3% of the observations, so I decided to keep all the features for the time being.
I filled all the NaN during the feature engineering phase.

3. Feature engineering

Categorical variables

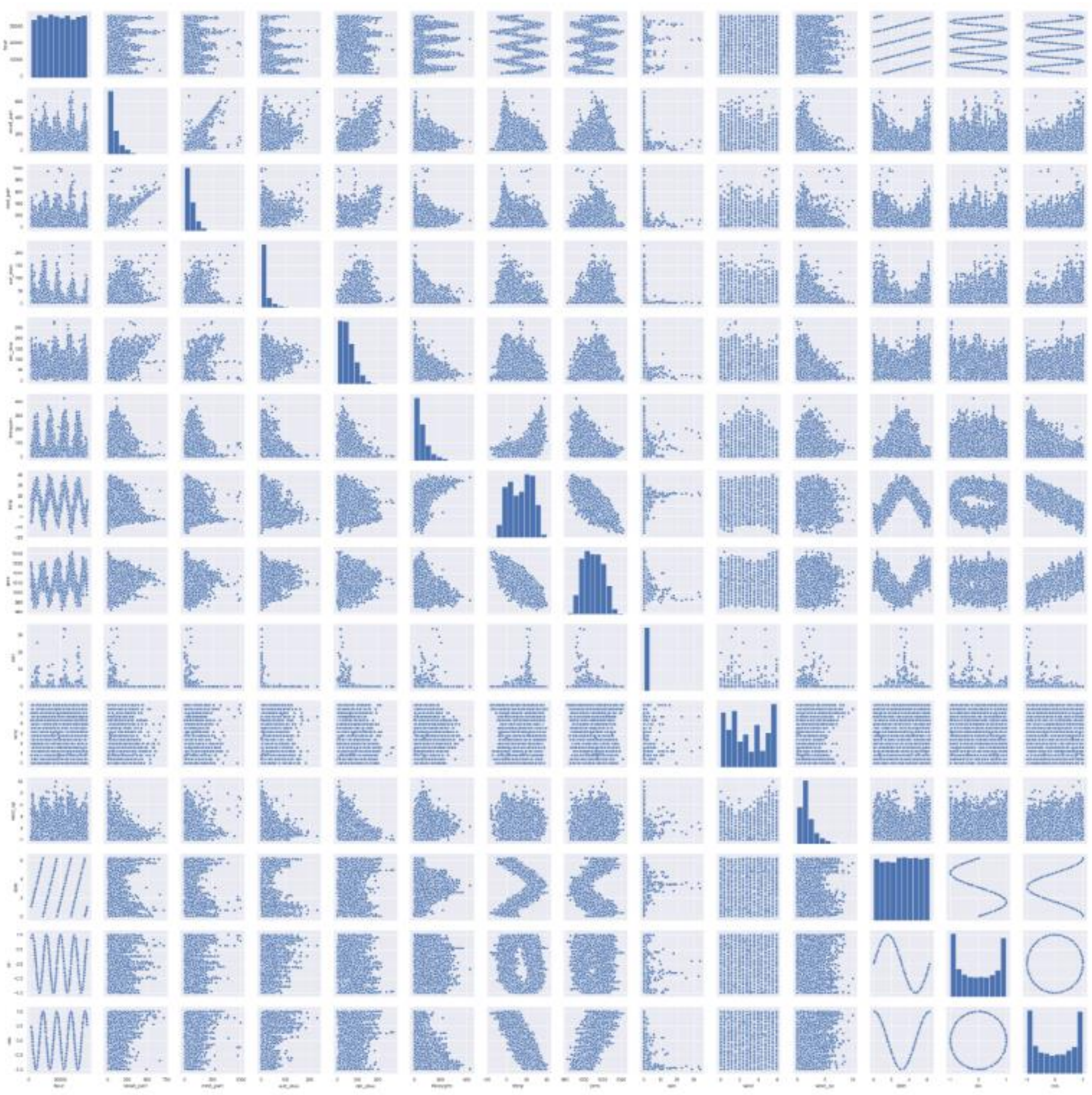
- ‘wind’: this was the first variable I tackled. I decided to encode it by mapping the wind directions into degrees first and then radians (since radians have a much smaller scale than degrees). This was eventually useless since ‘wind’ was one of the features I dropped for preforming regression.
- ‘date’ was tougher: firstly I transformed it into something that Python could easily understand, so I changed all the months into their respective number. Then, I used the time series function `datetime` to change the date ’29-2-2017’, since 2017 was not a leap year: if the hour corresponding to a ’29-2-2017’ observation was <12 a.m., the date became ’28-2-2017’, and if it was >12 a.m., it became ‘1-3-2017’. Then I used the function `total_seconds` to map the days into numbers (from 1 to 365), that I rescaled into radians to get a cyclic trend where the smallest and highest values coincide with colder months. I was still not satisfied with this ambiguity, so I computed the sine and cosine of the ‘date’ feature, creating two new features ‘sin’ and ‘cos’.

Missing values

Since the observations follow a time series, I wanted to fill the missing data with a forward and backward fill because I thought that values could not change that much in a handful of hours, but first I needed to sort the dataset with respect to an increasing time scale. I had already adjusted the ‘hour’ variable to follow the time series, so that it consisted of the hour count for the respective observation, starting from 01.01.2014.

So, I took advantage of this and used the hour count to sort and fill the dataset with the mean value of the subsequent and previous value.

After sorting by index again, I dropped the variables I deemed useless (the ones with the lowest correlation to the target variable or the ones that fell into multicollinearity), and I observed the scatterplot to see if the variables 'sin' and 'cos' showed hidden patterns:



Consequently, I dropped 'wind', 'pres', 'wind_sp', 'date' and 'med_part'.

After splitting into train and test set, I applied a StandardScaler to the data. I tried with both MinMaxScaler and RobustScaler, too, but the StandardScaler gave the best results in all model applications.

I decided to try as regressors XGBoost, Random Forest, KNN and SVR. XGBoost did not appear to be useful in this type of application, and Random Forest had a tendency to overfit my training set, so I discarded them.

I debated long and hard about whether to implement a K-Nearest Neighbor Regression or a Support Vector Regression, but eventually resorted to the latter as it seemed more robust with respect to the small experiment described in the "Outliers" section of this paper.

Eventually a SVR model with parameters {'C': 1000, 'degree': 1, 'epsilon': 10, 'gamma': 'auto', 'kernel': 'rbf'} scored a MAE= 266.4328062598784 on train set and MAE= 275.0005537337701 on test set. The results are consistend and the model does not seem to overfit, so I decided to keep this model for prediction.
The explained variance score is also ~78%, and given the variability of this dataset, I was eventually satisfied with my model.