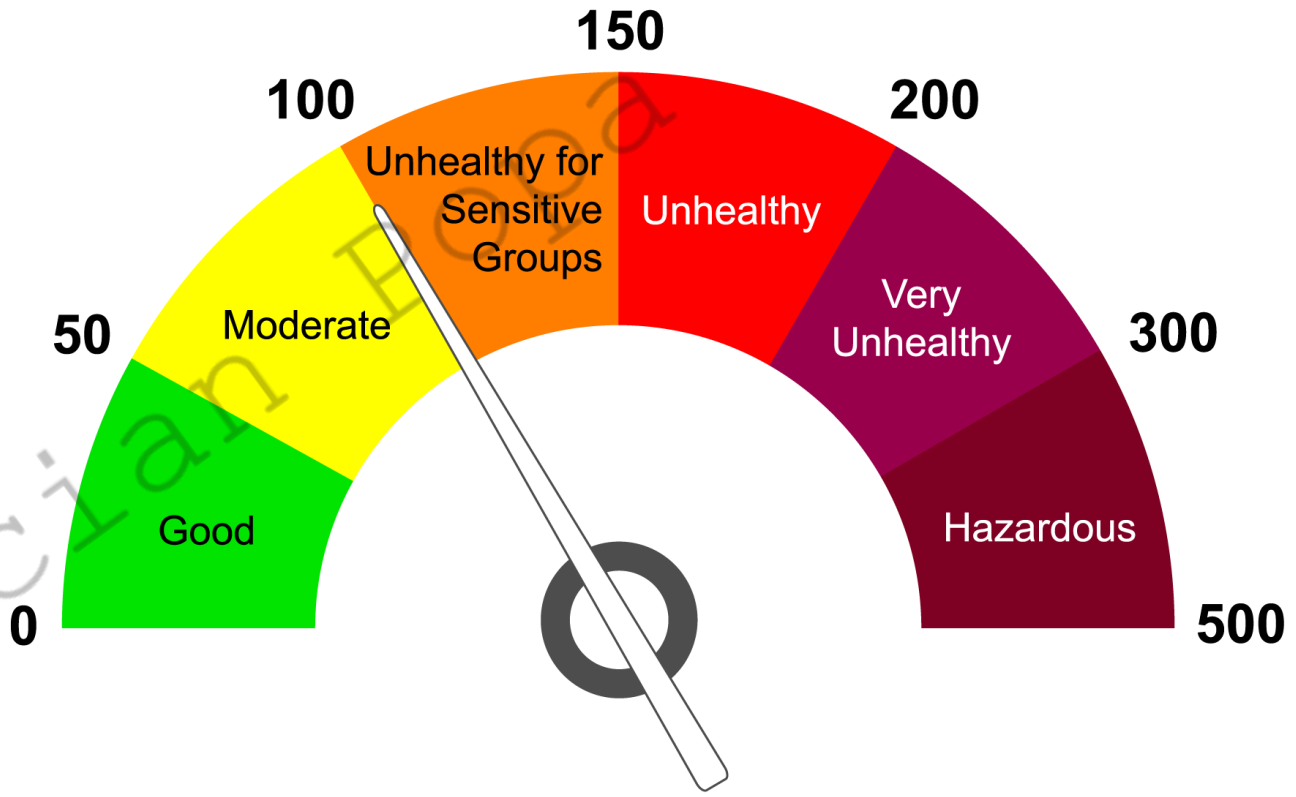
Vasile Lucian Popa

February 2022

Air Quality Analysis and Predictive Modelling

Outline

- Executive Summary;
- Introduction;
- Methodology;
- Modelling and results;
- Conclusion.



Executive Summary

- ***Summary of methodologies:***
- - Data Collection;
- - Data cleaning and feature engineering.
- - Exploratory Data Analysis with Data Visualization;
- - Machine Learning Prediction.

- ***Summary of all results:***
- - Exploratory Data Analysis result;
- - Interactive analytics in screenshots;
- - Predictive Analytics result.

Introduction

- Air pollution will endanger human health and life in big cities, especially to the elderly and children. This is not an individual problem of one person but a global problem. Therefore, many countries in the world made air pollution monitoring and control stations in many cities to observe air pollutants such as NO2, CO, SO2, PM2.5, and PM10 to alert the citizens about pollution index which exceeds the quality threshold.
- Particulate Matter PM 2.5 is a fine atmospheric pollutant that has a diameter of fewer than 2.5 micrometers. Particulate Matter PM10 is a coarse particulate that is 10 micrometers or less in diameter. Carbon Monoxide CO is a product of combustion of fuel such as coal, wood, or natural gas. Vehicular emission contributes to the majority of carbon monoxide let into our atmosphere. Nitrogen dioxide or nitrogen oxide expelled from high-temperature combustion: sulfur dioxide SO2 and Sulphur Oxides SO produced by volcanoes and in industrial processes. Petroleum and Coal often contain sulfur compounds, and their combustion generates sulfur dioxide. Air pollution is caused by the presence of poison gases and substances; therefore, it is impacted by the meteorological factors of a particular place, such as temperature, humidity, rain, and wind.
- In this project, the regression analysis technique is used to evaluate the relationship between these factors and predict nitrogen oxides NOx based on other parameters;

- ***The main objective of the analysis is to focus on prediction.***

Dataset description

The dataset contains 9358 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level, within an Italian city. Data were recorded from March 2004 to February 2005 (one year) representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses. Ground Truth hourly averaged concentrations for CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) and were provided by a co-located reference certified analyzer.

Dataset attribute information:

- - Date (DD/MM/YYYY)
- - Time (HH.MM.SS)
- - True hourly averaged concentration CO in mg/m^3 (reference analyzer)
- - PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)
- - True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)
- - True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)
- - PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)
- - True hourly averaged NOx concentration in ppb (reference analyzer)
- - PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)
- - True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)
- - PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO2 targeted)
- - PT08.S5 (indium oxide) hourly averaged sensor response (nominally O3 targeted)
- - Temperature in Â°C
- - Relative Humidity (%)
- - AH Absolute Humidity.

METHODOLOGY

Data Collection

- This dataset is from the UCI machine learning repository and contains hourly averaged responses from an air quality multi-sensor device that was located in a significantly polluted area at road level in an undisclosed Italian city. This data was collected over the course of aprox one year (from March 2004 - February 2005)

- *For reference*: https://archive.ics.uci.edu/ml/datasets/Air+Quality#

- *Note*: Evidences of cross-sensitivities as well as both concept and sensor drifts are present as described in De Vito et al., Sens. And Act. B, Vol. 129,2,2008 (citation required) eventually affecting sensors concentration estimation capabilities.

Data cleaning and feature engineering

- Dates were parsed so we can have a format that we could use:

```
In [204]: df.Date.dtype
Out[204]: dtype('O')

In [205]: df.set_index("Date", inplace=True)

In [206]: df.index = pd.to_datetime(df.index)

In [207]: type(df.index)
Out[207]: pandas.core.indexes.datetimes.DatetimeIndex

In [208]: df['Time'] = pd.to_datetime(df['Time'],format= '%H.%M.%S').dt.hour

In [209]: df.head()
Out[209]:
```

Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
2004-10-03	18.0	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578
2004-10-03	19.0	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255
2004-10-03	20.0	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502
2004-10-03	21.0	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867
2004-10-03	22.0	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888

Data cleaning and feature engineering.

- Removed columns we don’t need or we can not use:

In [196]:
```
# load and read the data
data = pd.read_csv('data/AirQualityUCI.csv', delimiter=";" ,decimal=",")

# drop unecessary columns
data = data.drop(["Unnamed: 15","Unnamed: 16"], axis=1)

df = data.copy() # Keep a copy of original data
```

In [197]:
```
# check data
df.head()
```

Out[197]:

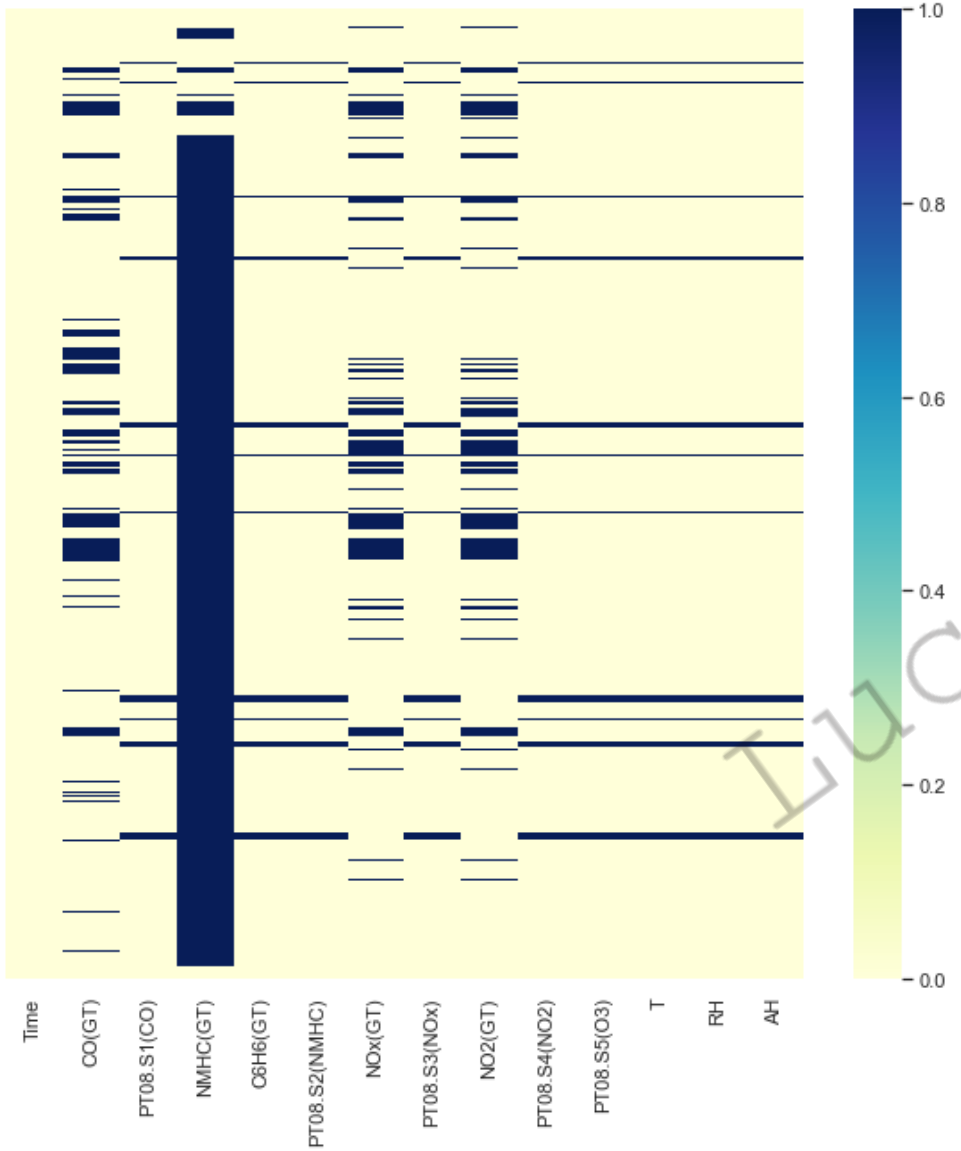
	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH
0	10/03/2004	18.00.00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9
1	10/03/2004	19.00.00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7
2	10/03/2004	20.00.00	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0
3	10/03/2004	21.00.00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0
4	10/03/2004	22.00.00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6

In [198]:
```
df.shape # (rows, columns)
```

Out[198]: (9471, 15)

Data cleaning and feature engineering.

- Checked the missing values by column:

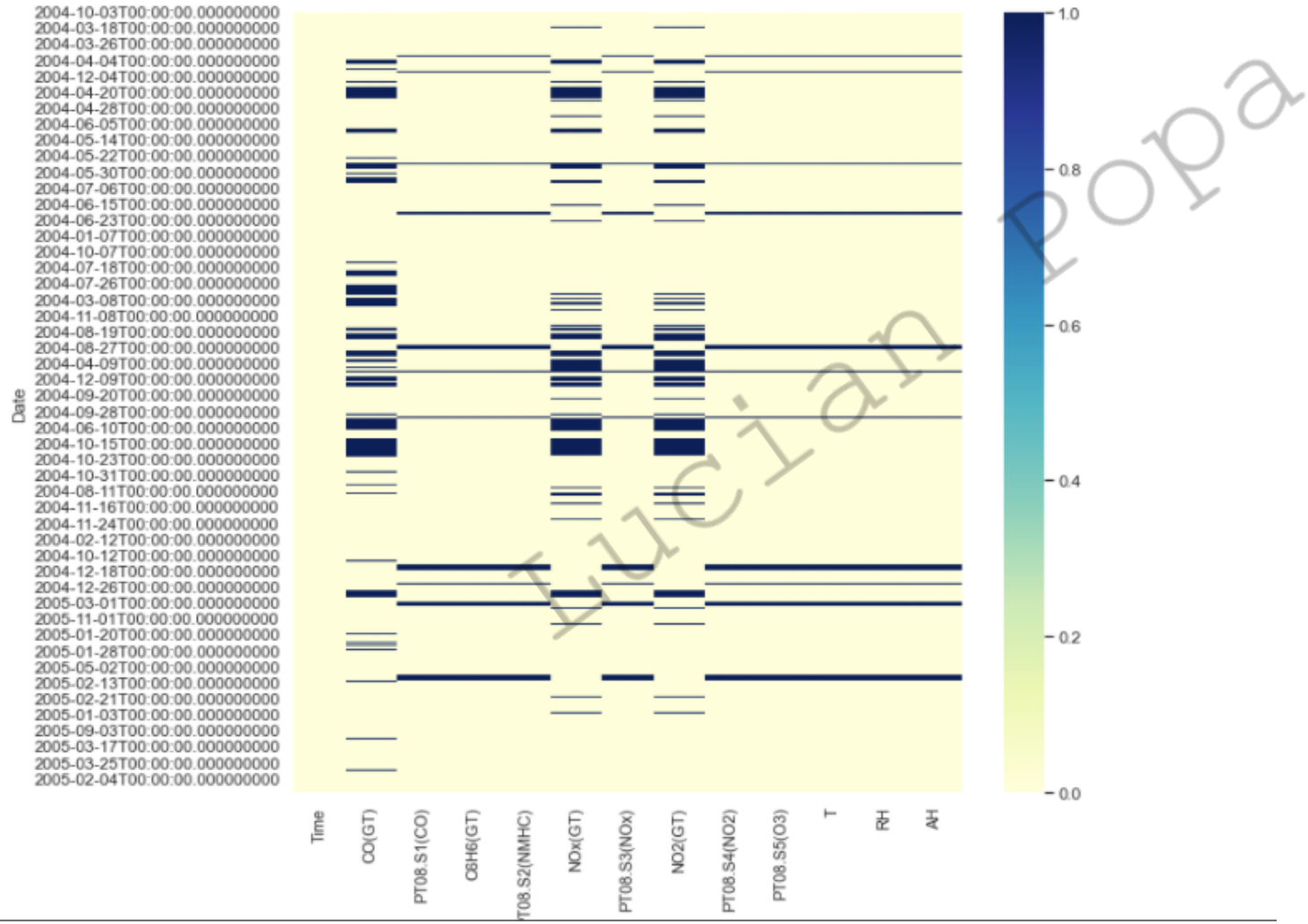


The NHMC(GT) column is missing many values, more than 85% values are NaN; 8443 out of 9357. Thus this column can be removed from the dataset as these values are very less likely to be of any significant importance in this dataset.

Data cleaning and feature engineering.

- Checked the missing values by column after we removed NHMC(GT) :

```
In [213]: # plot dataframe with NA values after dropping the column above
          fig, ax = plt.subplots(figsize= (10, 10))
          ax = sns.heatmap(df.isin([-200]),cmap="YlGnBu")
```



Data cleaning and feature engineering

- Fill NaN values with average of particular date and forward fill method for removing the leftover nan values

In [214]:
```
df.replace(to_replace= -200, value= np.NaN, inplace= True)
```

In [215]:
```
# Fill NaN values with average of particular date

def remove_outlier(col):
    df[col] = df.groupby('Date')[col].transform(lambda x: x.fillna(x.mean()))
```

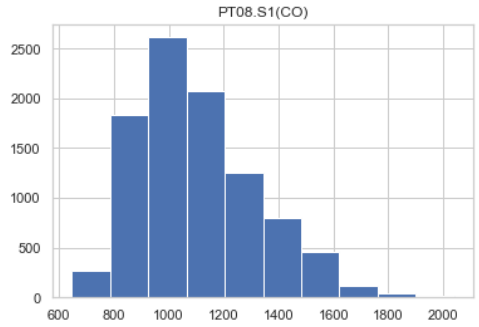
In [216]:
```
# forward fill method for removing the leftover nan values

df.fillna(method='ffill', inplace= True)
```

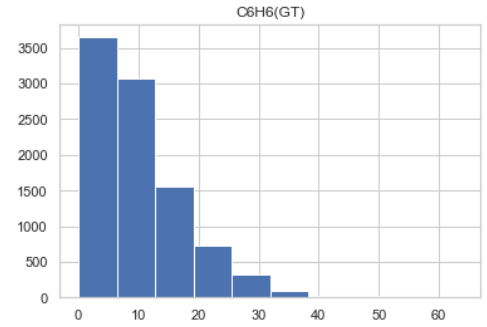
In [217]:
```
df.isnull().any()
```

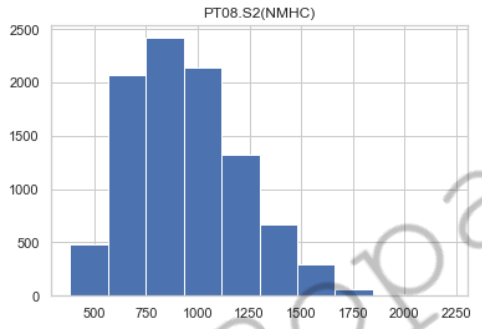
Out[217]:
```
Time            False
CO(GT)          False
PT08.S1(CO)     False
C6H6(GT)        False
PT08.S2(NMHC)   False
NOx(GT)         False
PT08.S3(NOx)    False
NO2(GT)         False
PT08.S4(NO2)    False
PT08.S5(O3)     False
T               False
RH              False
AH              False
dtype: bool
```

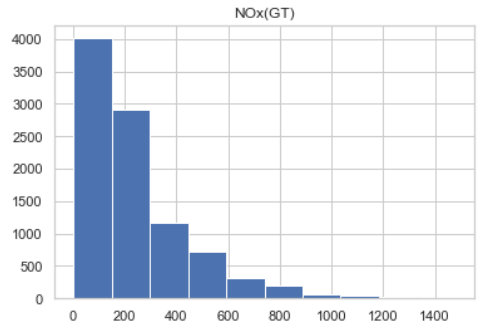
EDA with Data Visualization

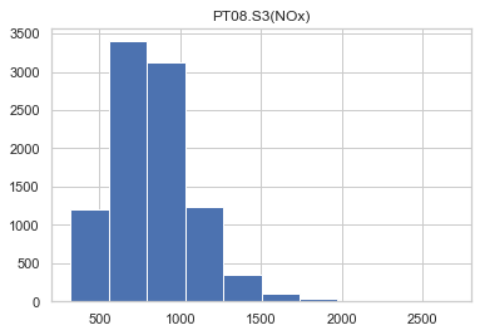
- Plot the column using Histograms so we can see the distribution of the data:

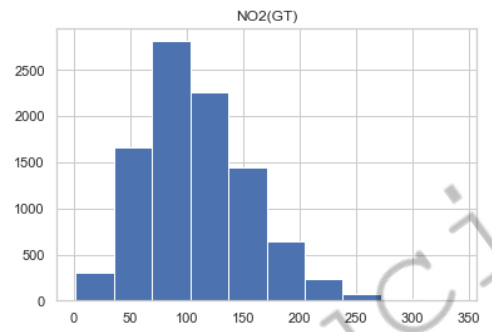


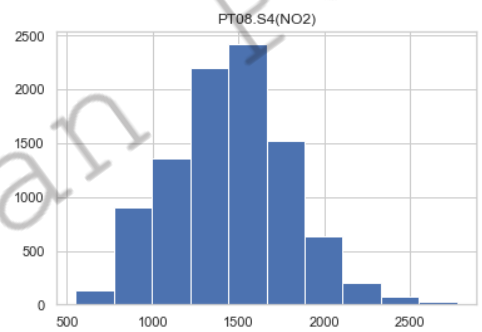


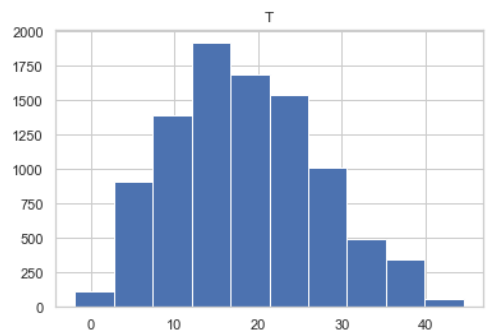


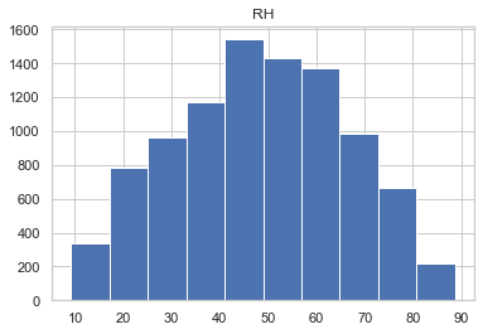


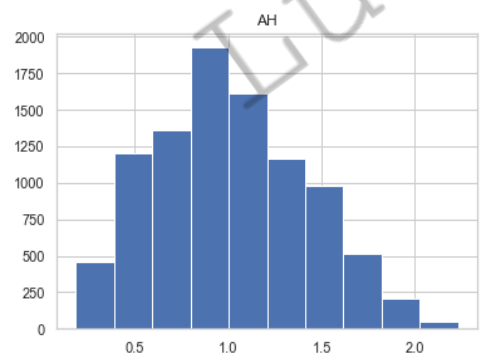


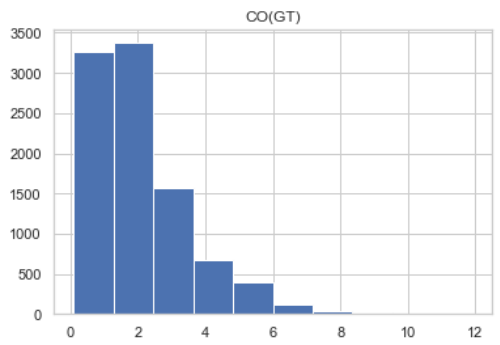








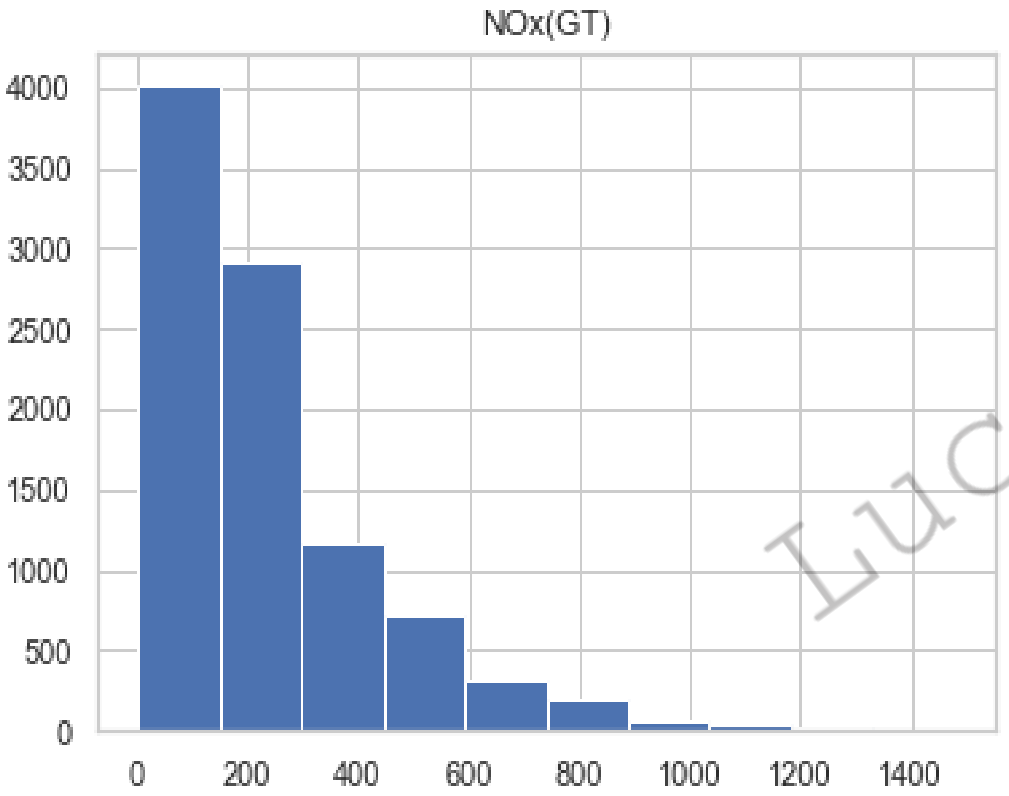




EDA with Data Visualization

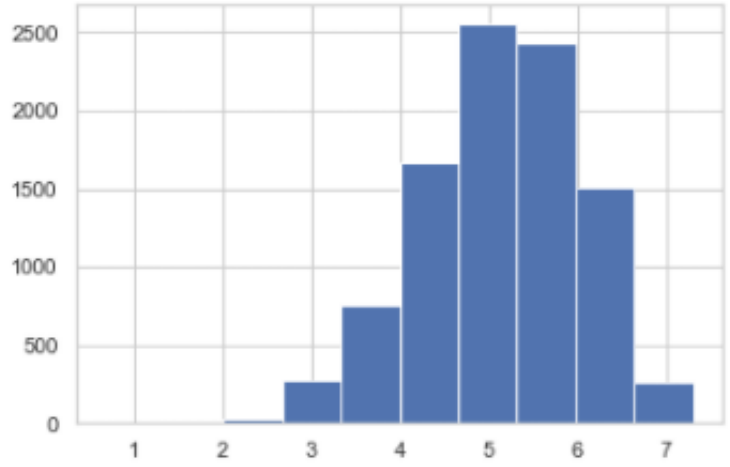
- Performed data transformation on the target variable (Nox(GT) for better predictions when modelling.

- Before:

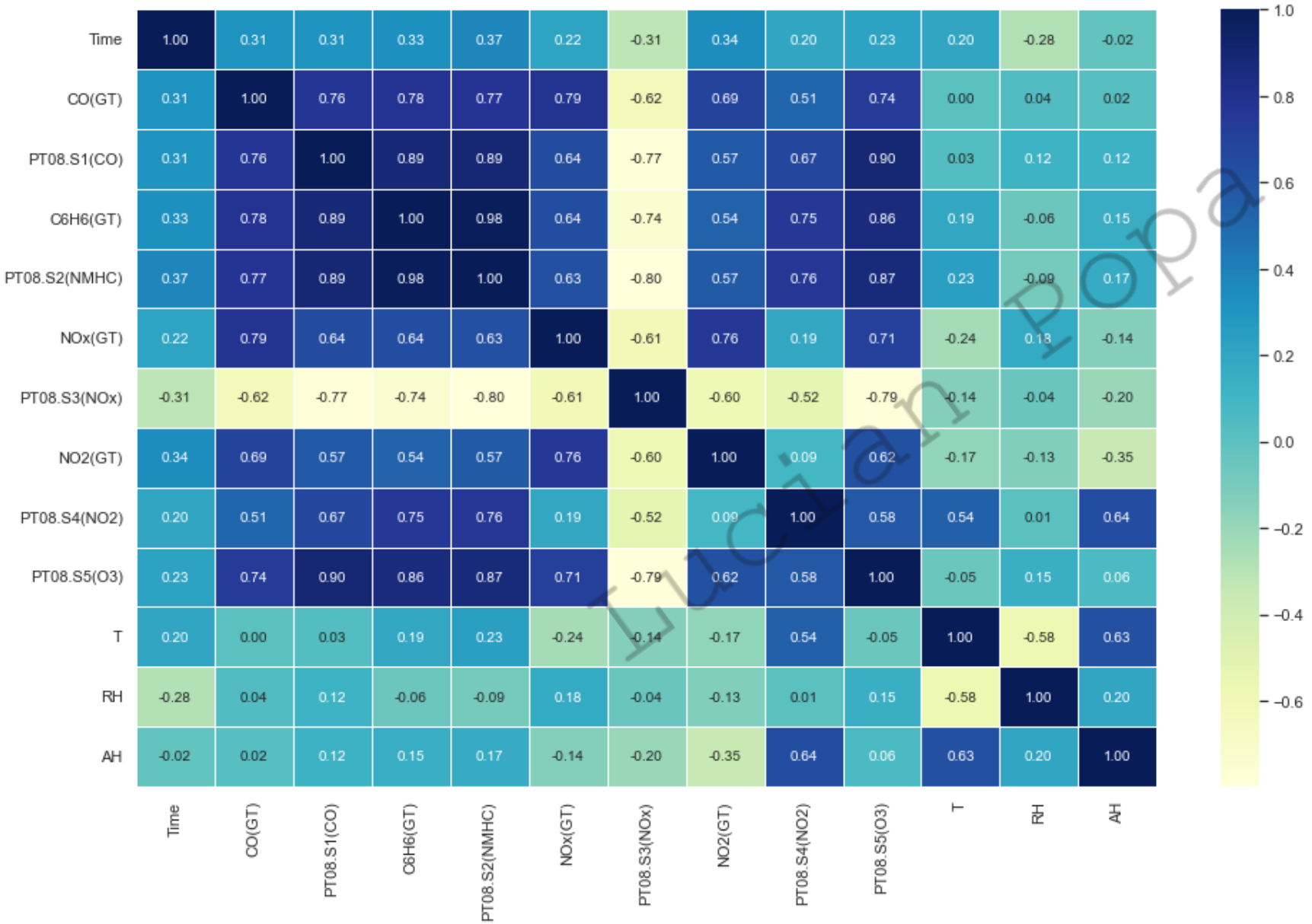


After:

```
In [220]: # Log-transform the skewed features
          plt.hist(df['NOx(GT)'].apply(lambda x: np.log(x)))

Out[220]: (array([1.000e+00, 5.000e+00, 2.700e+01, 2.720e+02, 7.500e+02, 1.671e+03,
                  2.552e+03, 2.430e+03, 1.503e+03, 2.600e+02]),
           array([0.69314718, 1.35374461, 2.01434204, 2.67493947, 3.33553689,
                  3.99613432, 4.65673175, 5.31732918, 5.97792661, 6.63852403,
                  7.29912146]),
           <BarContainer object of 10 artists>)
```



EDA with Data Visualization



Plotted correlation matrix so we can find the correlation between the variables and choose the ones we are going to work with.

Predictive Analysis (Regression)

Building the model:

- Load the dataset into NumPy and Pandas;
- Transform data;
- Split the data into training and test data sets;
- Pick up which type of ML algorithm I want to use (used LinearRegression, Ridge, Lasso, ElasticNet, Decision Tree and RandomForestRegressor);
- Set parameters and use GridSearchCV to find the best ones;
- Fit data int the GridSearch CV and train on the dataset.

Evaluating the model:

- Check accuracy of the models;
- Tune hyperparameters of each model;

Improving the model:

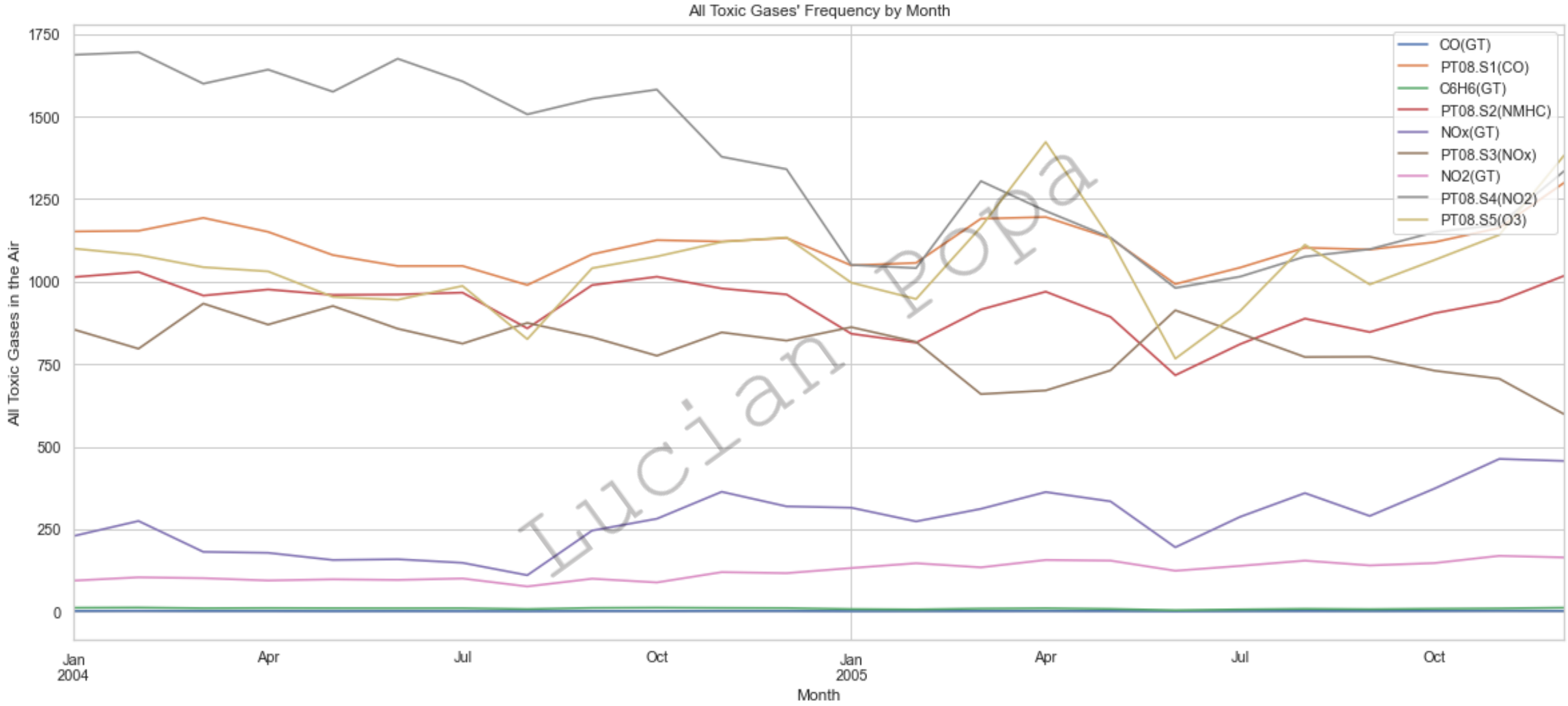
- Feature Engineering;
- Algorithm tuning;

Finding the best model:

- Evaluate the best performing model, on the accuracy.

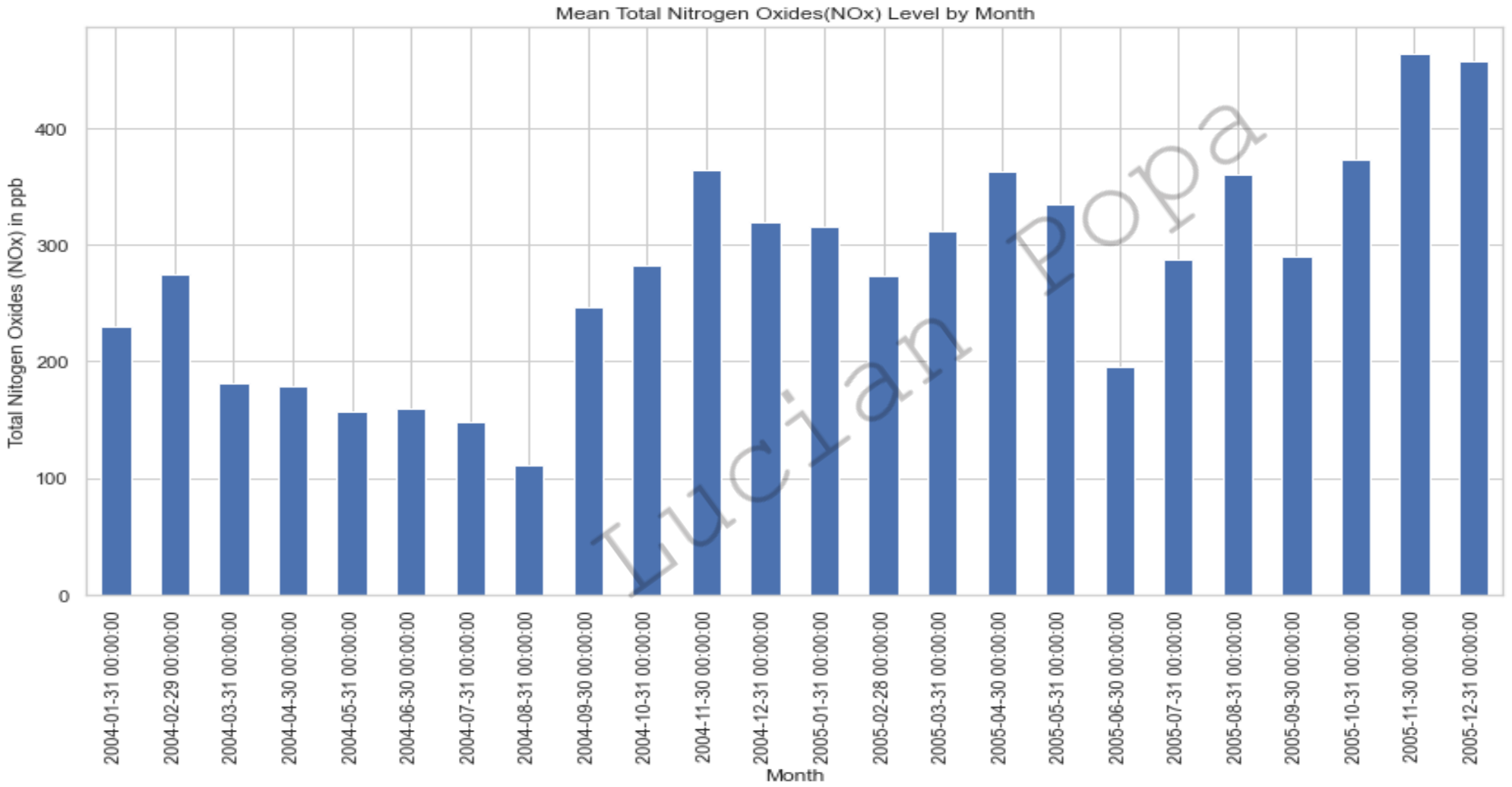
Insights drown from EDA

All Toxic Gases - Frequency by Month



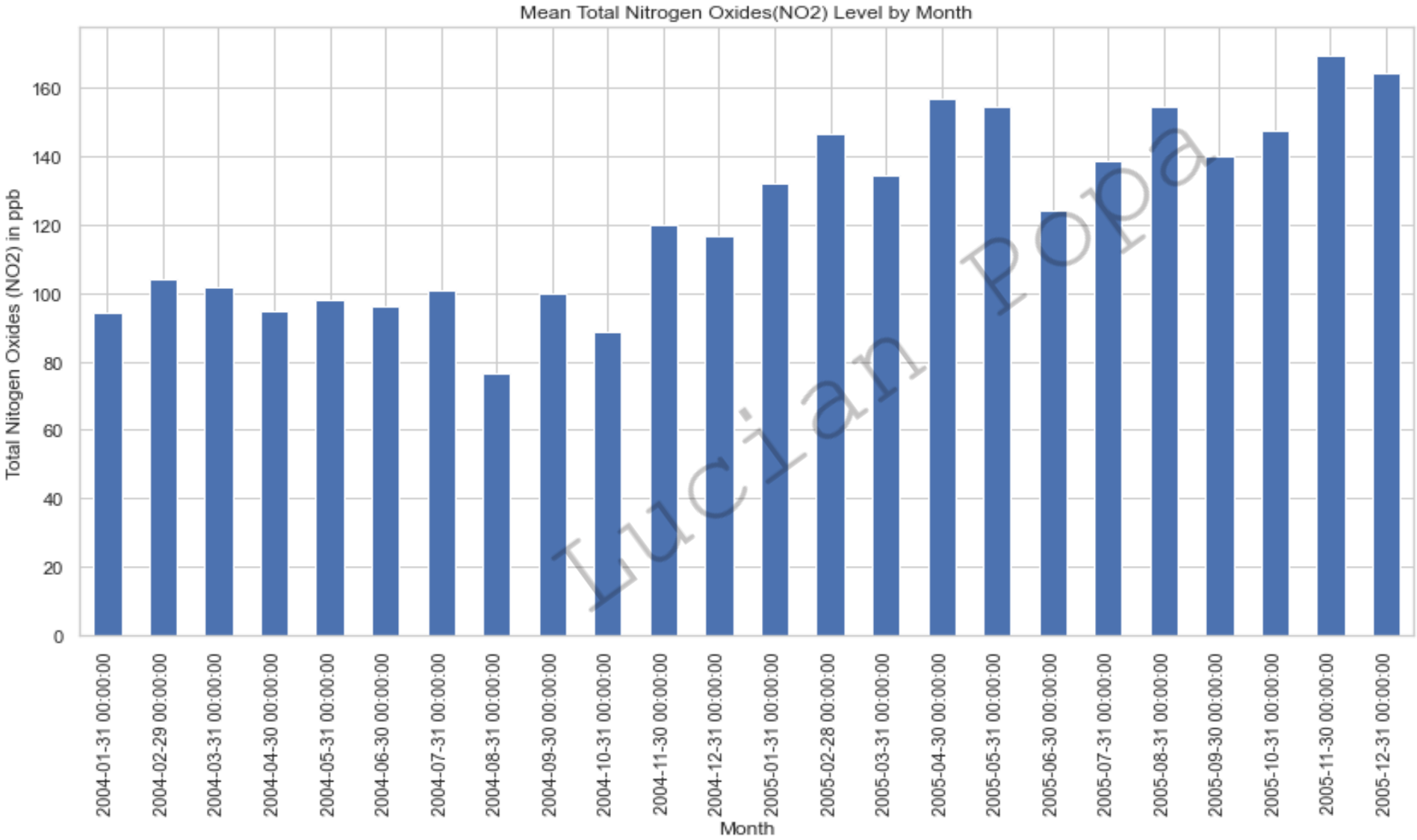
Mean Total Nitrogen Oxides(NOx) Level by Month

We can see in this plot how the Nox levels are increasing every month making this a reason to be worried and willing to take actions.



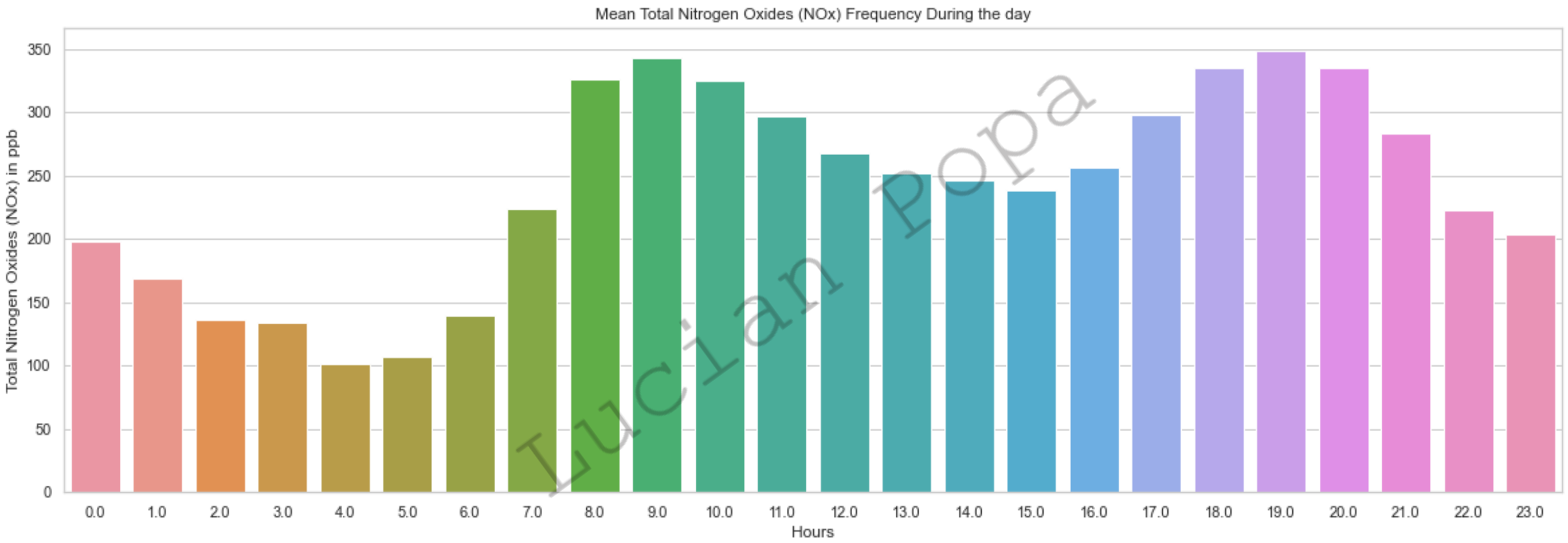
Mean Total Nitrogen Oxides(NO2) Level by Month

Also, the levels of NO2 are increasing every month.



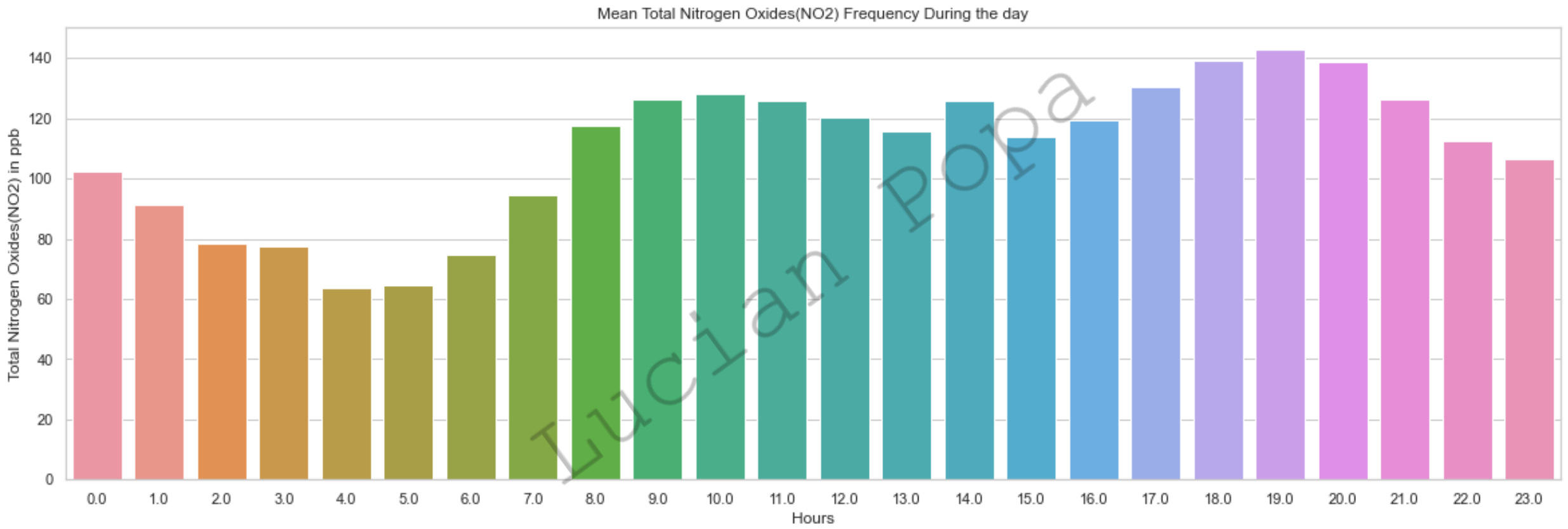
Mean Total Nitrogen Oxides (NOx) Frequency During the day

Switching to the levels of Nox during the day, we can definitely see the increase in the level of Nox at specific hours.



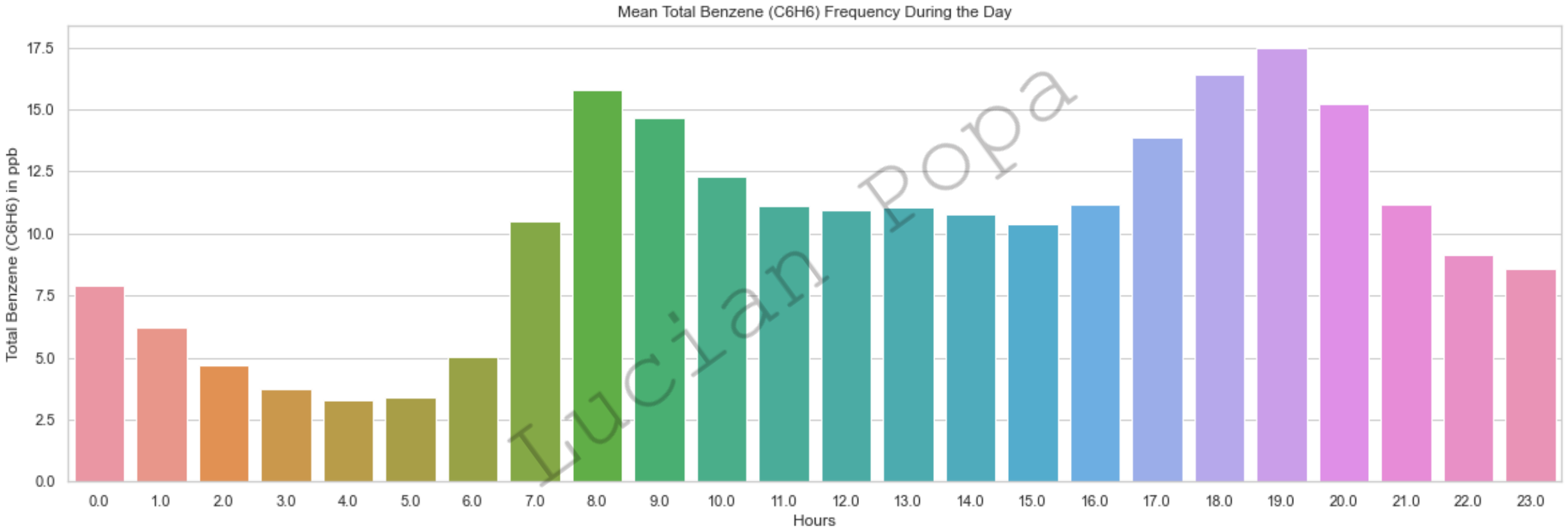
Mean Total Nitrogen Oxides(NO2) Frequency During the day

On the NO2 plot by hours we can see the same pattern, higher levels at the same hours as Nox.



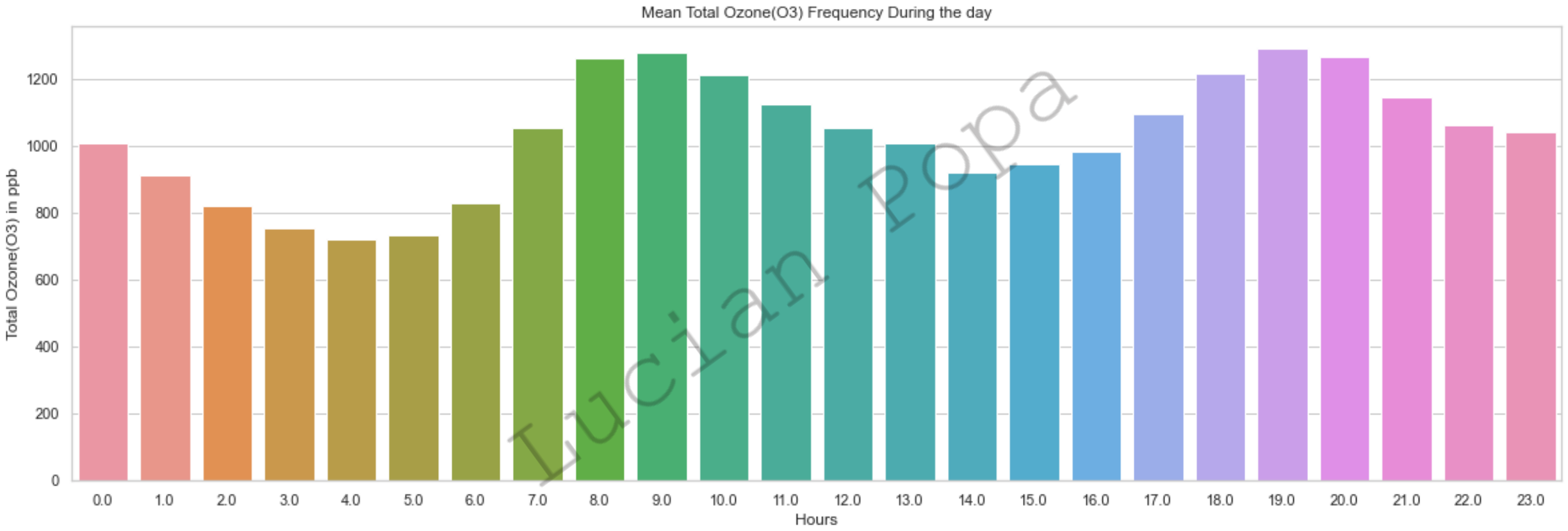
Mean Total Benzene (C6H6) Frequency During the Day

What about Benzene (C6H6) levels? Can we find a pattern? Yes, the same as NO2 and Nox.



Mean Total Ozone(O3) Frequency During the day

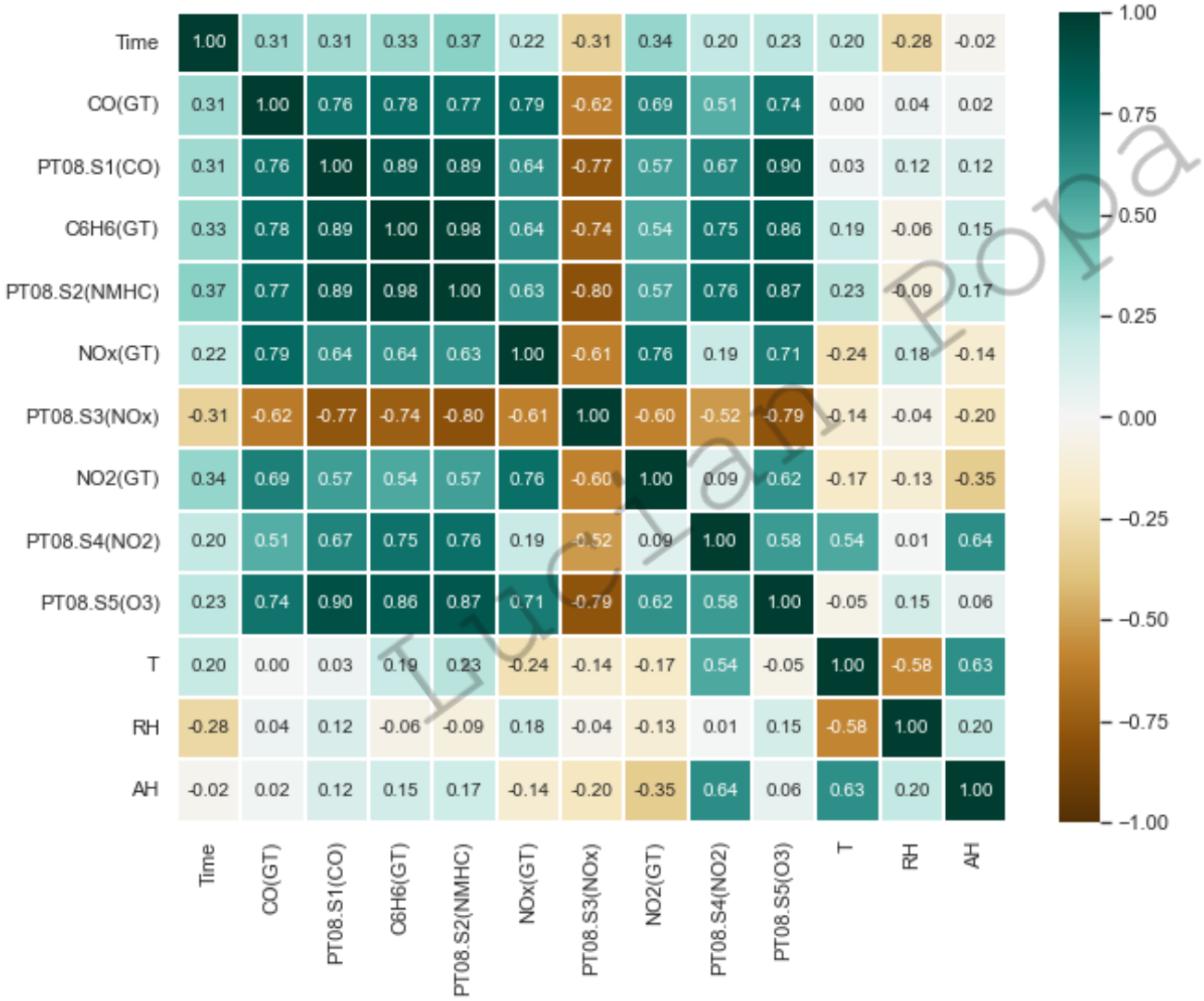
What about the Ozone levels? Will we find the same patterns? YES!



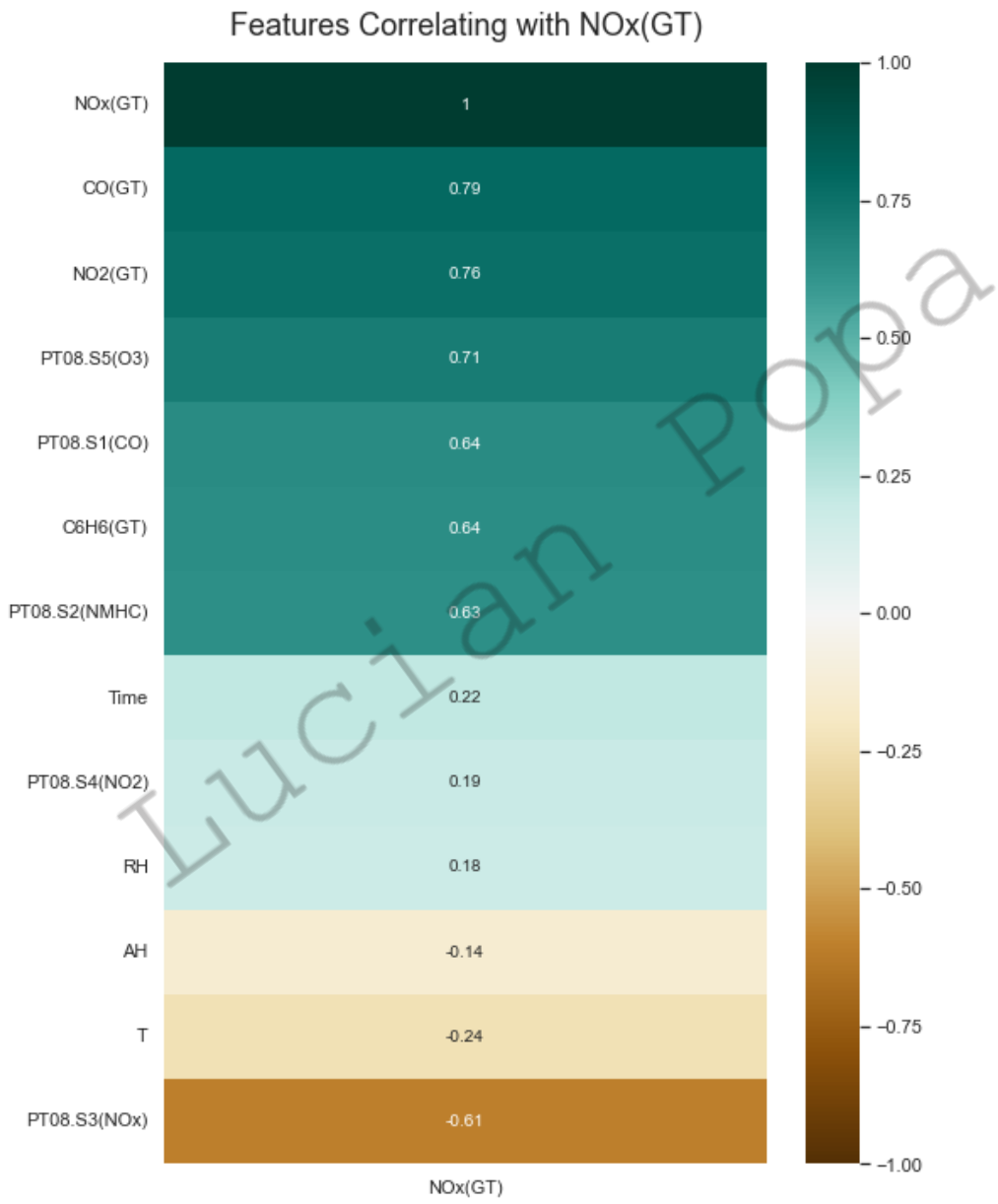
We could say that the levels of the toxic data is affected by the 'rush' hours and the moments when the traffic on the streets is really high, pushing up all the toxic gases levels.

Pearson Correlation matrix

Perform Pearson Correlation matrix to find the most correlated variables.



Correlation levels of the features to Nox(GT)



Modelling and predictions

Choose Models , fit and score the ML models

Models to load and try:

1. LinearRegression;
2. Lasso;
3. Ridge;
4. Elastic Net;
4. DecisionTree Regressor;
4. Random Forest Regressor.

Modelling

Put models in a dictionary and create function to fit and score models

In [258]:
```
# Put models in a dictionary
models = {"Linear Regression": LinearRegression(),
          "Lasso": Lasso(),
          "Ridge": Ridge(),
          "Elastic Net": ElasticNet(),
          "DecisionTreeRegressor": DecisionTreeRegressor(),
          "RandomForestRegressor": RandomForestRegressor()
         }

# Create function to fit and score models
def fit_and_score(models, X_train, X_test, y_train, y_test):
    """
    Fits and evaluates given ML models.
    models: a dict of diff SKlearn ML models;

    """
    # Set random seed
    np.random.seed(42)
    # Make a dict to keep ML scores
    model_scores = {}
    # Loop through models
    for name, model in models.items():
        # Fit the model to the data
        model.fit(X_train, y_train)
        # Evaluate the model and append its score to model_scores
        model_scores[name] = model.score(X_test, y_test)
    return model_scores
```

In [259]:
```
# score the models
model_scores = fit_and_score(models=models,
                             X_train=X_train,
                             X_test=X_test,
                             y_train=y_train,
                             y_test=y_test)
model_scores
```

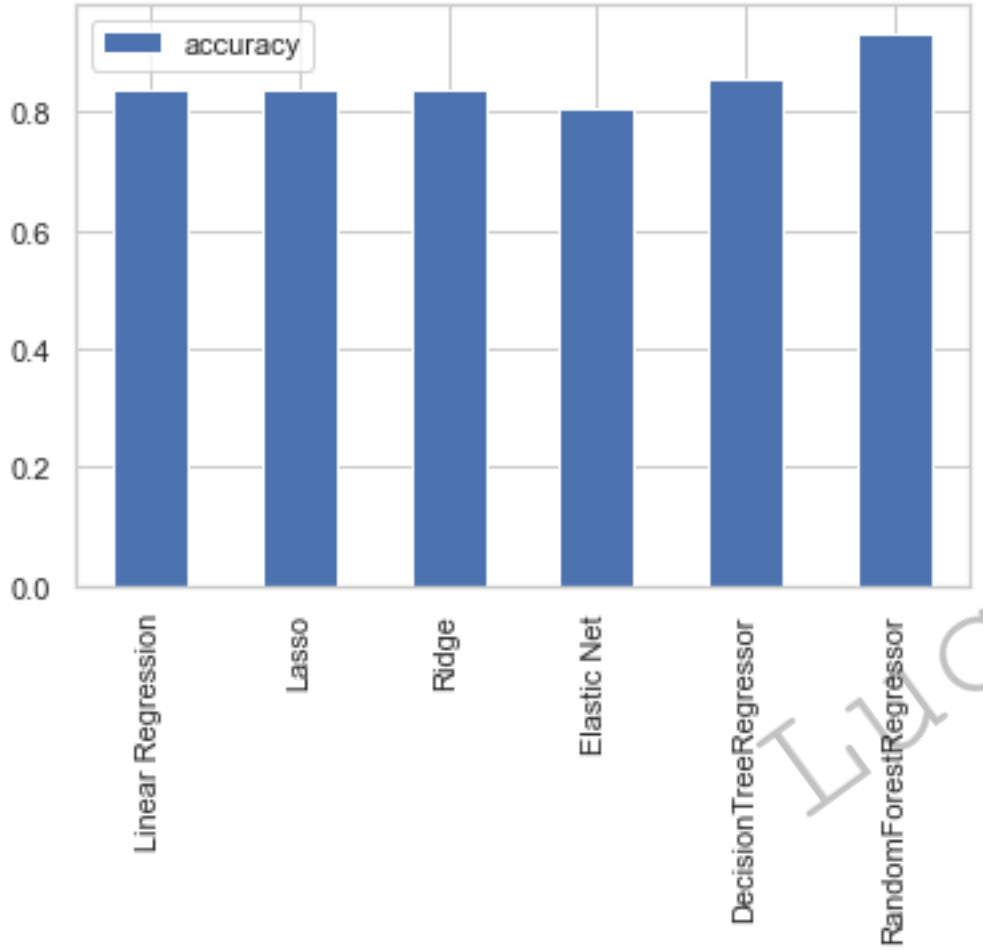
Out[259]:
```
{'Linear Regression': 0.8365953641070447,
 'Lasso': 0.8352933933739933,
 'Ridge': 0.8365791201421416,
 'Elastic Net': 0.8068324196196663,
 'DecisionTreeRegressor': 0.8538904032318794,
 'RandomForestRegressor': 0.9338367990363031}
```

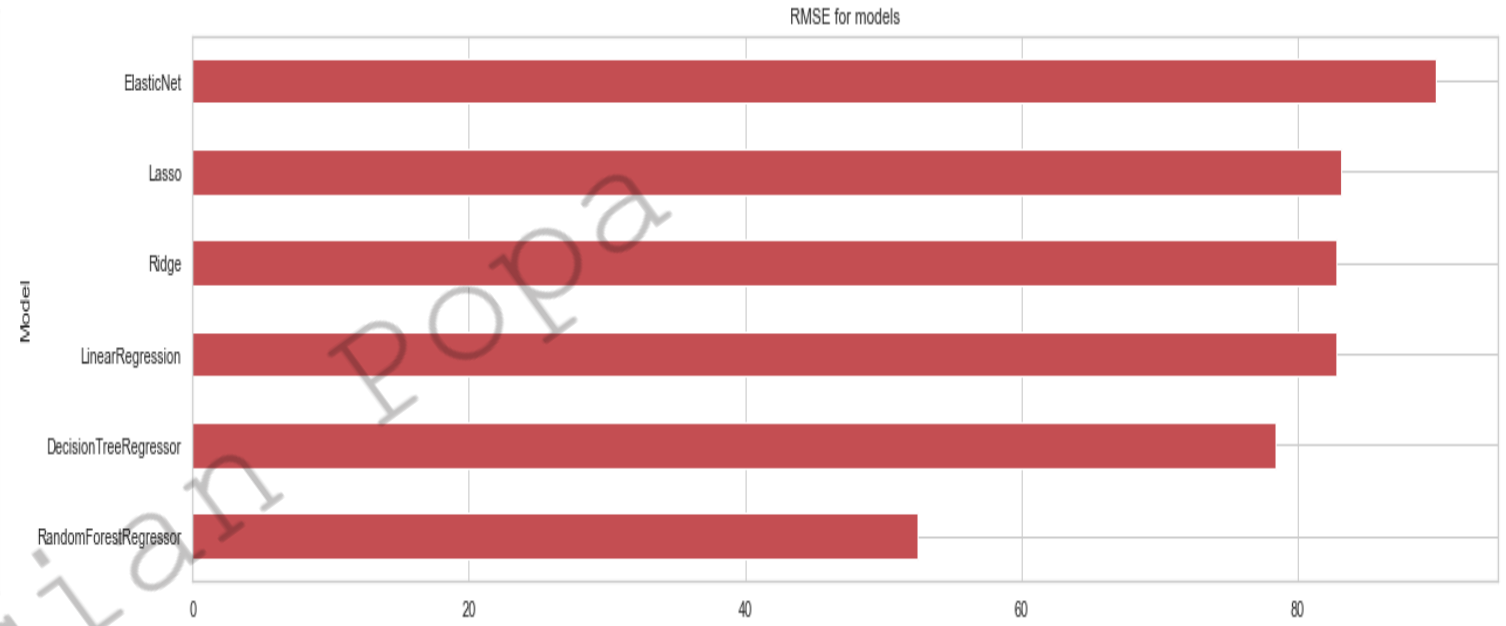

After the first split test, with the “stock” parameters on all the models, best scoring algorithm is RandomForrestRegressor with a score of: 93.38% accuracy.

Can we improve the scores on this models?

Modelling

Model Comparison and RMSE by model.





	Model	r2	RMSE
0	LinearRegression	0.836595	82.808501
1	Ridge	0.836579	82.812616
2	Lasso	0.835293	83.137746
3	ElasticNet	0.806832	90.034686
4	DecisionTreeRegressor	0.853525	78.401416
5	RandomForestRegressor	0.934268	52.520686

When comes to RMSE score, RandomForrestRegressor takes again the first place, with a score of 52.52. This is pretty good for a stock model.

Modelling

First predictions aren't always the best.

Next, after performing a few other steps, we'll check the scores again. To implement:

- Hyperparameter tunning;
- Cross validation;
- Data standardization;
- PolynomialFeature.

Modelling

After applying standardization, polynomial feature on the data and hyperparameter tunning, we tested again the models on the test data and the results were better in terms of accuracy :

In [276]:
```
# comparing accuracy scores of the models after tunning
pd.DataFrame([[linear_score, lasso_score, ridge_score]],columns=['linear', 'lasso', 'ridge'], index=['score'])
```

Out[276]:

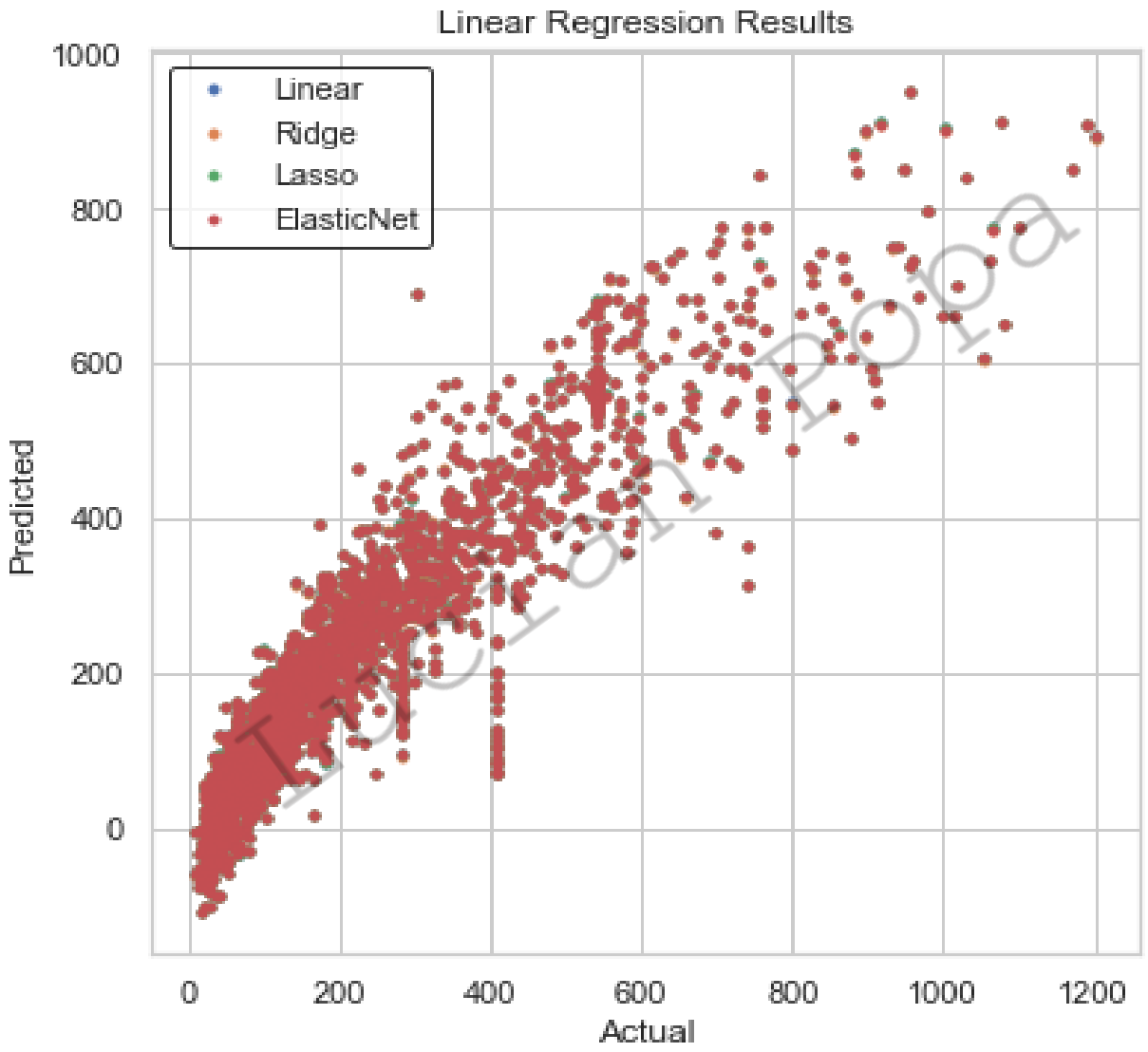
	linear	lasso	ridge
score	0.838789	0.91213	0.912645

But not much of an improvement when it comes to RMSE scores:

Out[277]:

	Linear	Lasso	Ridge	ElasticNet
rmse	82.808501	82.870802	82.808776	82.82913

Results: prediction vs actual values



Results

While we managed to improve the accuracy scores, we still couldn't beat the score of the Random Forest Regressor, even the "stock" one.

Random Forrest Regression score: 93.42% R2 score and 52.52 RMSE.

And we even improved this score, with a simple Grid Search CV setup to 93.82%

Other Linear Regression models results:

- Ridge: 91.26% accuracy, 82.80 RMSE
- Lasso: 91.21% accuracy, 82.87 RMSE

Conclusions

- In this project, the AirQuality dataset from UCI was used, for exploratory data analysis and the prediction of NOx. linear regression models (Linear baseline, ridge, lasso elasticnet, decision trees and randomforestregressor) were created and trained, using the same training and test splits, and then compared to find the best model among them.
- Based on the models findings, the simple ridge and lasso models gives the smallest Root-mean-square error and the highest score when predicting on the test data. However, the difference in scores and errors are not significant and almost identical. Therefore it is recommended as a final model as it best fits the data in terms of accuracy.
- The above models could give even better results if we used GridSearchCV or RandomizedSearchCV to optimize the models' hyperparameters. Alternatively, different techniques were used, like Random Forest Regression or Decision Tree, and the Random Forest Regression gets a higher accuracy score with a lower mean square error.

- GitHub link of the Notebook : Here

Note on conclusions

- Two of the most toxicologically significant compounds are nitric oxide (NO) and nitrogen dioxide (NO2). Nitrogen Oxides (NOx) are among the most dangerous forms of air pollution. They are produced from the reaction of nitrogen and oxygen gases in the air during combustion, especially at high temperatures. In areas of high motor vehicle traffic, such as in large cities, the amount of nitrogen oxides emitted into the atmosphere as air pollution can be significant. It is mainly due to fossil fuel combustion from both stationary sources, i.e. power generation (21%), and mobile sources, i.e. transport (44%). Other atmospheric contributions come from non-combustion processes, for example nitric acid manufacture, welding processes and the use of explosives.
- In addition, these create serious health issues. These mainly impact on respiratory conditions causing inflammation of the airways at high levels. Long term exposure can decrease lung function, increase the risk of respiratory conditions and increases the response to allergens. NOx also contributes to the formation of fine particles (PM) and ground level ozone, both of which are associated with adverse health effects.

To do next:

The above models could give even better results if we used GridSearchCV or RandomizedSearchCV to optimize the models' hyperparameters. Alternatively, different techniques were used, like Random Forest Regression or Decision Tree, and the Random Forest Regression gets a higher accuracy score with a lower mean square error.

More in depth analysis and feature engineering of the data is need it in order to get better results and predict the levels of the Nox even better.


Thank you!