Hyperlocal Air Quality Prediction

Introduction

The [Environmental Defense Fund (EDF) (https://www.edf.org/airqualitymaps)](https://www.edf.org/airqualitymaps) has partnered with Google Earth Outreach to map air pollution at the **hyperlocal level**. In various cities around the world, mobile air quality sensors were used to gather air pollution data at street level. This new approach allows researchers to collect much more data at a granular level and showed how air poluution varied over very short distances.

Our project focuses on the data collected in [Houston (https://www.edf.org/airqualitymaps/houston)](https://www.edf.org/airqualitymaps/houston), where low cost mobile sensors were outfitted on city fleet vehicles and Google Street View cars, in addition to gathering data from stationary sensors. For this report, we have focused on only one type of pollution - NO2 emissions.

Based on our review of literature and exploring various datasets, we have decided the scope of our project to explore the use of machine learning models to predict air quality at any location in Houston based on meteorlogical conditions, sources of emissions from various facilities, and traffic.

Inspiration

Our project is inspired and informed by Varsha Gopalakrishnan's "Hyperlocal Air Quality Prediction using Machine Learning" project, outlined on [Medium (https://towardsdatascience.com/hyperlocal-air-quality-prediction-using-machine-learning-ed3a661b9a71)](https://towardsdatascience.com/hyperlocal-air-quality-prediction-using-machine-learning-ed3a661b9a71). Gopalakrishnan performed her analysis using Oakland data, and we intend to replicate most aspects of the approach for Houston.

Data

The level of pollution at any place depends on a number of factors like traffic on major streets; emissions from different facilities like railroads, ports, and industries; and meteorological factors like temperature and precipitation. Our target variable is Air Quality Data obtained from the aforementioned pilot done by EDF. We conducted exploratory data analysis for the air quality dataset in our [Houston air quality data Jupyter Notebook (https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/air_quality.ipynb)](https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/air_quality.ipynb).

We classified our potential features into the following three buckets:

1. Traffic Data - [Link to notebook (https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/traffic_data.ipynb)](https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/traffic_data.ipynb)

We look at the geographical range for which EDF collected the air quality data in Houston. The coordinates for the area around Houston is based on minimum and maximum of latitude and longitude from the EDF data. We chose this area as the bounding box to determine location of all traffic signals / intersections within that boundary.

The Overpass API from Open Street Maps is used to determine the location of all traffic signals (intersections) within a given bounding box. The Overpy library is used to send the request to the API and this call returns the latitude and longitude of all traffic signals. Next, the distance between each traffic intersection and each point in the monitoring data is measured. A traffic score is calculated as the 'Number of traffic intersections within 1,000 ft of each point in the monitoring data."

Going forward, in addition to intersections, we will also extend the analysis to look at the distance of bus stops and highways from the NO2 monitoring points.

2. Emissions Data - [Link to notebook (https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/facility_data.ipynb)](https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/facility_data.ipynb)

We use NEI's data for point sources - which provides facility specific pollutant information from across the US. Similar to how we bound based on a boundary box region, we consider only those facilities that fall within the (lat, long) range for which EDF data was collected.

After grouping some of the sources based on their types and range of emission values, we calculate the distance between the location of every pollutant data read (from the EDF data) to each facility. We also calculate the emissions per distance measure for each such row, with the idea being that both the quantity of NO2 emission from the facility, as well as the distance from the facility will affect NO2 concentration at a certain point

Going forward, we will invest time on three fronts:

1. Understanding sources currently marked as 'unknown', since they could add vital information (and variance) to the dataset
2. Carrying out a more robuts outlier treatment of the NO2 distribution
3. Calculating additional aggregated numeric metrics such as number of point sources (per source_group) in a given radius from the monitoring point

3. Meteorological factors - [Link to notebook (https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/met_data.ipynb)](https://github.com/vjoseph21/air-quality-prediction/blob/main/notebooks/cleaning/met_data.ipynb)

Following Gopalakrishnan's approach, we use the [Daymet (https://daymet.ornl.gov/web_services)](https://daymet.ornl.gov/web_services) API to collect our meteorological data. Daymet's "daily surface weather and climatological summaries" provide daily data at a 1km grid granularity, extrapolated from less-granular meteorological observations. Through approximately 200 calls to the Daymet API, we gathered the available daily data (maximum temperature, minimum temperature, shortwave radiation, vapor pressure, and precipitation) at latitudes and longitudes corresponding to the locations of our air quality measurements, and produced averages for each metric across the 9 months when EDF was collecting air quality data in Houston.

Going forward, we plan to explore alternate approaches to collapsing the daily data for each air quality measurement site - for example, it's possible that minimums or maximums over the 9-month period in question would be more meaningful than averages.

Modeling

We will use this dataset to develop a model to predict air quality in neighbourhoods in Houston where the EDF project did not collect air quality data. We plan to first train the machine learning model to predict NO2 emissions on the same locations as the EDF data and then use the trained model and additional traffic, features, and emissions data to preduct concentrations elsewhere in Houston.

For the baseline model, we have merged all the features mentioned above into one dataset and used linear regression for feature selection. We decided to use a threshold of $\pm 90\%$ to indicate very high correlation between features and dropped them to avoid multicollineairty. We then use the remaining features to run a simple linear regression to guage the siginificance of each feature on NO2 emissions. Our results indicate that minimum temperature, precipitation, and number of intersections are the most important features to predict air quality.

In [1]:
```
#Import basic python packages for data analysis and plotting
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.lines as mlines
import pylab as plot
import matplotlib
import random
import seaborn as sns
from mpl_toolkits.axes_grid1 import make_axes_locatable
import math
import time

### Import Scipy stats packages
from scipy.stats import pearsonr
from scipy.stats import boxcox


# Import statsmodel packages
import statsmodels.api as sm
from statsmodels.formula.api import ols
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices

# import sklearn packages
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
from sklearn.model_selection import GridSearchCV, cross_validate, cross_val
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn import linear_model
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer
from sklearn.metrics import r2_score
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import ElasticNet

import re
import os


import warnings
warnings.filterwarnings("ignore")

sns.set(style = 'whitegrid')
sns.set_palette('bright')
%matplotlib inline
```

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/base/tsa_mode
l.py:7: FutureWarning: pandas.Int64Index is deprecated and will be remove
d from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/base/tsa_mode
l.py:7: FutureWarning: pandas.Float64Index is deprecated and will be remo
ved from pandas in a future version. Use pandas.Index with the appropriat
e dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
```

Dataset prep

In [3]:

```
# merge all datasets to create master_df

root = os.path.dirname(os.path.dirname(os.getcwd()))

df_aq = pd.read_csv(root + "/data/cleaned/air_quality_NO2.csv", index_col=0
df_met = pd.read_csv(root + "/data/cleaned/nO2_met.csv", index_col=0)
df_fac = pd.read_csv(root + "/data/cleaned/no2_fac_data.csv", index_col=0)
# df_fac.drop(df_fac.columns[df_fac.columns.str.contains('_emsdist')], axis
df_traffic = pd.read_csv(root + "/data/cleaned/intersection_final.csv", ind

df_m1 = df_aq.merge(df_met, on = ['latitude', 'longitude'], how = 'inner')
df_m2 = df_m1.merge(df_fac, on = ['latitude', 'longitude'], how = 'inner')
df_merged = df_m2.merge(df_traffic, on = ['latitude', 'longitude'], how = '
df_merged.drop(columns = ['latitude', 'longitude'], inplace=True)
```

Feature selection

In [4]:

```
# create feature list and scale
feature_df = df_merged.drop(columns = ['value'])

#Standardizing features
feature_df_scaled = pd.DataFrame(StandardScaler().fit_transform(feature_df)

df_merged_scaled = pd.concat([df_merged['value'], feature_df_scaled], axis
```

In [5]:
```python
# ols fit on individual features

r2 = []
for column in feature_df.columns[0:]:
    r2_val = sm.OLS(df_merged['value'], feature_df[column]).fit().rsquared
    r2.append(r2_val)
r2_score_df = pd.DataFrame({'Feature': feature_df.columns, 'Individual_R2':


r2_score_df.sort_values('Individual_R2', ascending = False)
```

Out[5]:

	Feature	Individual_R2
0	prcp	0.756036
4	vp	0.749088
3	tmin	0.748959
2	tmax	0.748711
1	srad	0.748691
55	4861111-Petroleum-low_dist	0.722380
76	6642811-FoodPlants-low_emsdist	0.721422
36	18343611-RailYard-high_emsdist	0.720754
72	6642011-FoodPlants-high_emsdist	0.706980
63	4942411-Electricity-Heating-high_dist	0.694085
45	4057311-Electricity-Heating-low_dist	0.694071

Lasso Regression for Feature Selection

In [6]:
```python
# identifying all the features that have a correlation higher than 0.90 or

threshold_1 = 0.90
threshold_2 = -0.90

def features_high_corr(df_features_corr):
    columns = np.full((df_features_corr.shape[0],), True, dtype=bool)
    for i in range(df_features_corr.shape[0]):
        for j in range(i+1, df_features_corr.shape[0]):
            if (df_features_corr.iloc[i,j] >= threshold_1) | (df_features_c
                if columns[j]:
                    columns[j] = False
    selected_columns = df_features_corr.columns[columns]
    return selected_columns
```

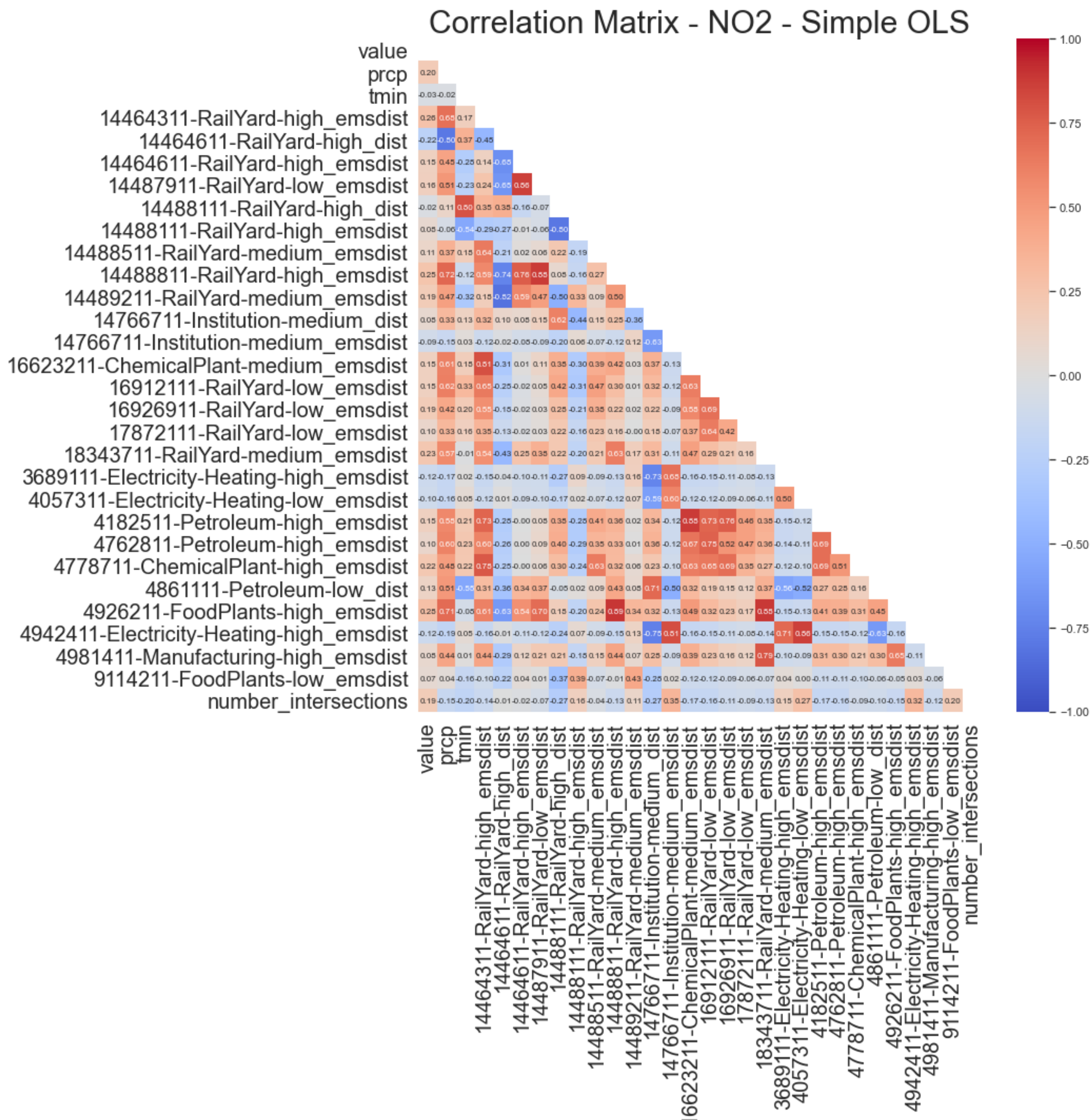
In [7]:
```
# list of features that are not highly correlated
features_corr = feature_df.corr()
features_OLS = features_high_corr(features_corr)
print("Features in BC dataset that are not highly correlated: ")
print(features_OLS)
```

```
Features in BC dataset that are not highly correlated:
Index(['prcp', 'tmin', '14464311-RailYard-high_emsdist',
       '14464611-RailYard-high_dist', '14464611-RailYard-high_emsdist',
       '14487911-RailYard-low_emsdist', '14488111-RailYard-high_dist',
       '14488111-RailYard-high_emsdist', '14488511-RailYard-medium_emsdis
t',
       '14488811-RailYard-high_emsdist', '14489211-RailYard-medium_emsdis
t',
       '14766711-Institution-medium_dist',
       '14766711-Institution-medium_emsdist',
       '16623211-ChemicalPlant-medium_emsdist',
       '16912111-RailYard-low_emsdist', '16926911-RailYard-low_emsdist',
       '17872111-RailYard-low_emsdist', '18343711-RailYard-medium_emsdis
t',
       '3689111-Electricity-Heating-high_emsdist',
       '4057311-Electricity-Heating-low_emsdist',
       '4182511-Petroleum-high_emsdist', '4762811-Petroleum-high_emsdis
t',
       '4778711-ChemicalPlant-high_emsdist', '4861111-Petroleum-low_dis
t',
       '4926211-FoodPlants-high_emsdist',
       '4942411-Electricity-Heating-high_emsdist',
       '4981411-Manufacturing-high_emsdist', '9114211-FoodPlants-low_emsd
ist',
       'number_intersections'],
      dtype='object')
```

In [8]:
```
#Create a dataframe with air quality value and selected columns from above
OLS_df = df_merged[['value']].join(df_merged[list(features_OLS)])
```

In [9]:
```
## Plot Correlation matrix
OLS_df_corr = OLS_df.corr()
features_corr_mat = OLS_df_corr.values

print(plt.get_backend())

# close any existing plots
plt.close("all")

# mask out the top triangle
features_corr_mat[np.triu_indices_from(features_corr_mat)] = np.nan

fig, ax = plt.subplots(figsize=(15, 15))

hm = sns.heatmap(features_corr_mat, cbar=True, vmin = -1, vmax = 1, center
                 fmt='.2f', annot_kws={'size': 8}, annot=True,
                 square=False, cmap = 'coolwarm')

ticks = np.arange(OLS_df_corr.shape[0]) + 0.5
ax.set_xticks(ticks)
ax.set_xticklabels(OLS_df_corr.columns, rotation=90, fontsize=20)
ax.set_yticks(ticks)
ax.set_yticklabels(OLS_df_corr.index, rotation=360, fontsize=20)

ax.set_title('Correlation Matrix - NO2 - Simple OLS', fontsize  = 30)
plt.tight_layout()
```

```
module://matplotlib_inline.backend_inline
```



Fitting an OLS Model on the Shortlisted Features

```
In [10]: # OLS
OLS_corr_model = sm.OLS(df_merged['value'], feature_df[features_OLS])
OLS_corr_results = OLS_corr_model.fit()
OLS_corr_results.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	value	**R-squared (uncentered):**	0.821
Model:	OLS	**Adj. R-squared (uncentered):**	0.821
Method:	Least Squares	**F-statistic:**	1785.
Date:	Sun, 15 May 2022	**Prob (F-statistic):**	0.00
Time:	21:06:23	**Log-Likelihood:**	69548.
No. Observations:	11314	**AIC:**	-1.390e+05
Df Residuals:	11285	**BIC:**	-1.388e+05
Df Model:	29		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
prcp	-0.0003	7.61e-05	-4.345	0.000	-0.000	-0.000
tmin	0.0002	3.12e-05	5.040	0.000	9.6e-05	0.000
14464311-RailYard-high_emsdist	2.262e-05	1.68e-06	13.442	0.000	1.93e-05	2.59e-05
14464611-RailYard-high_dist	2.623e-05	7.74e-06	3.390	0.001	1.11e-05	4.14e-05
14464611-RailYard-high_emsdist	3.232e-06	4.97e-07	6.506	0.000	2.26e-06	4.21e-06
14487911-RailYard-low_emsdist	-7.192e-05	1.38e-05	-5.207	0.000	-9.9e-05	-4.48e-05
14488111-RailYard-high_dist	-0.0001	1.04e-05	-9.796	0.000	-0.000	-8.11e-05
14488111-RailYard-high_emsdist	3.182e-05	3.78e-06	8.417	0.000	2.44e-05	3.92e-05
14488511-RailYard-medium_emsdist	-1.839e-05	1.46e-06	-12.622	0.000	-2.12e-05	-1.55e-05
14488811-RailYard-high_emsdist	-4.605e-05	6.83e-06	-6.740	0.000	-5.94e-05	-3.27e-05
14489211-RailYard-medium_emsdist	0.0001	1.3e-05	11.004	0.000	0.000	0.000
14766711-Institution-medium_dist	0.0002	1.9e-05	12.172	0.000	0.000	0.000
14766711-Institution-medium_emsdist	-4.875e-06	1.52e-06	-3.217	0.001	-7.84e-06	-1.9e-06
16623211-ChemicalPlant-medium_emsdist	-1.267e-05	4.3e-06	-2.946	0.003	-2.11e-05	-4.24e-06
16912111-RailYard-low_emsdist	0.0046	0.001	8.904	0.000	0.004	0.006
16926911-RailYard-low_emsdist	0.0012	0.000	8.946	0.000	0.001	0.001
17872111-RailYard-low_emsdist	-3.81e-07	1.87e-06	-0.204	0.838	-4.04e-06	3.28e-06
18343711-RailYard-medium_emsdist	-4.099e-05	8.45e-06	-4.853	0.000	-5.75e-05	-2.44e-05
3689111-Electricity-Heating-high_emsdist	4.869e-06	1.09e-06	4.459	0.000	2.73e-06	7.01e-06
4057311-Electricity-Heating-low_emsdist	-2.843e-05	3.13e-05	-0.909	0.363	-8.97e-05	3.29e-05
4182511-Petroleum-high_emsdist	-1.051e-06	2.59e-07	-4.053	0.000	-1.56e-06	-5.43e-07

4762811-Petroleum-high_emsdist	-3.291e-08	4.91e-08	-0.670	0.503	-1.29e-07	6.34e-08
4778711-ChemicalPlant-high_emsdist	1.763e-06	3.1e-07	5.691	0.000	1.16e-06	2.37e-06
4861111-Petroleum-low_dist	-0.0002	1.57e-05	-12.104	0.000	-0.000	-0.000
4926211-FoodPlants-high_emsdist	0.0002	1.4e-05	17.672	0.000	0.000	0.000
4942411-Electricity-Heating-high_emsdist	-6.8e-06	4.09e-06	-1.664	0.096	-1.48e-05	1.21e-06
4981411-Manufacturing-high_emsdist	-3.934e-06	3.49e-07	-11.262	0.000	-4.62e-06	-3.25e-06
9114211-FoodPlants-low_emsdist	-3.455e-05	5.71e-06	-6.048	0.000	-4.57e-05	-2.34e-05
number_intersections	6.861e-05	2.16e-06	31.727	0.000	6.44e-05	7.28e-05

Omnibus:	5434.944	**Durbin-Watson:**	2.009
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	36517.513
Skew:	2.226	**Prob(JB):**	0.00
Kurtosis:	10.593	**Cond. No.**	2.45e+04

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 2.45e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [11]:
```
# R2, Coefficient and Intercept
print("The R2 is {}".format(OLS_corr_results.rsquared), "The R2 tells us th
```

```
The R2 is 0.821008850209677 The R2 tells us that there is a correlation b
etween the features identified and NO2 concentration.
```

Takeaways

From our baseline model, we observe statistically significant (if small) associations between NO2 levels and a location's average precipitation, average minimum temperature, number of nearby intersections, and numerous emission points. Going forward we'll be exploring other ways to wrangle our emissions data to deal with issues like "unknown" point sources. Additionally, we'll be adding further traffic data (highways, bus stops) and exploring alternative ways to manipulate the meterological data (minimums instead of 9-month averages, etc.). And we'll be moving beyond OLS to explore machine learning approaches and, finally, predict air quality levels in other Houston neighborhoods.