Predictive Analytics: Air Pollution

Georgetown University Certificate in Data Science, Spring 2020

GEORGETOWN UNIVERSITY

Team Members

1. James Hinshaw
2. Ayesha Baig
3. Adrienne White
4. Jeremy Lykken
5. Julien Collear

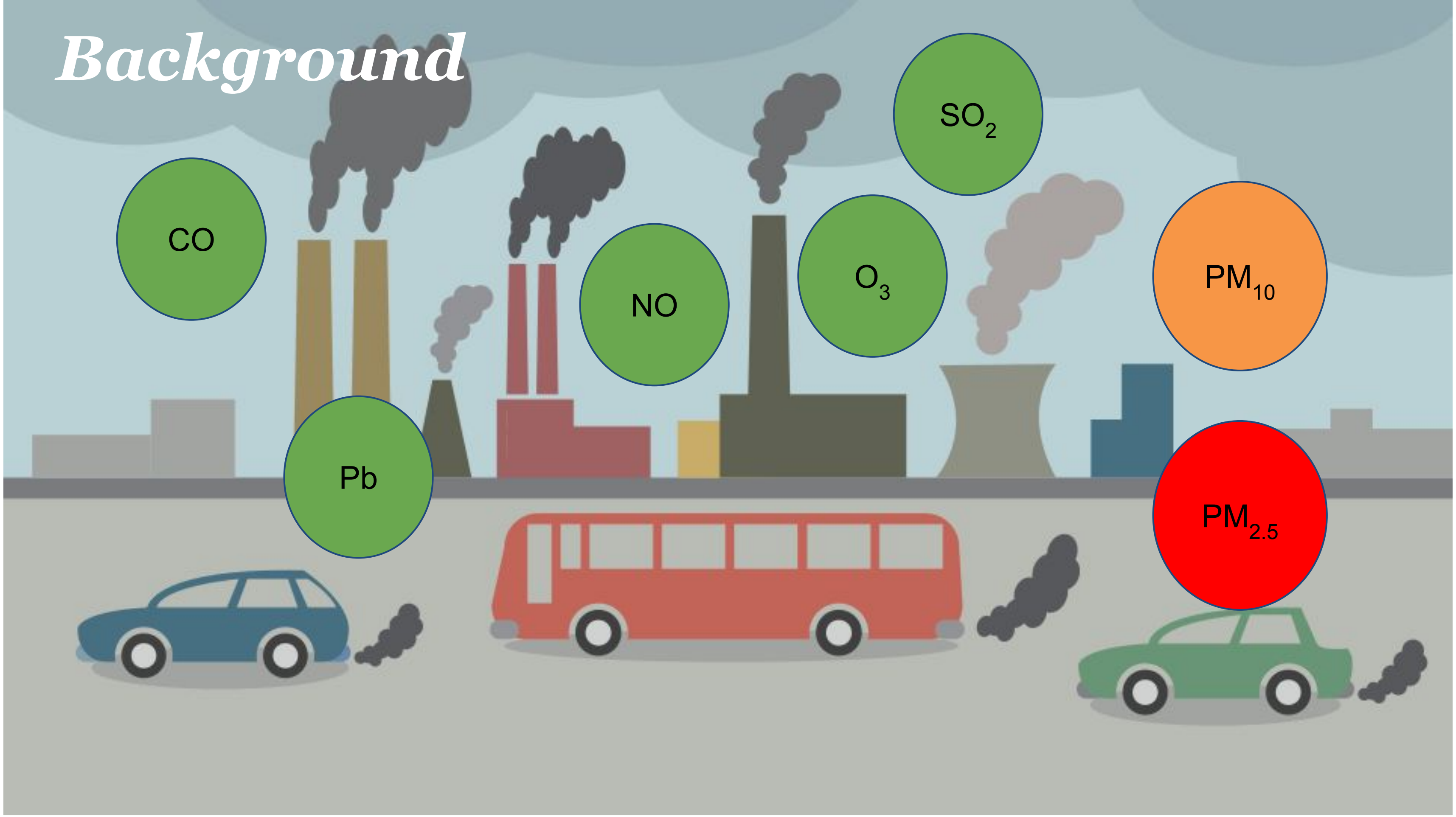
Agenda

1. Introduction
2. Methodology, Data Pipeline, Ingestion and Wrangling
3. Exploratory Data Analysis
4. Feature Engineering, Model Selection and Machine Learning
6. Results, Conclusions, Next Steps and Closing Remarks

20 Minutes

Introduction

Background



GEORGETOWN UNIVERSITY

Motivation

9

4.2 million

150 million

Motivation

- The ability to locate, measure, and predict PM2.5 levels is vital for protecting vulnerable population within local communities
- The current state of PM2.5 monitoring
- Government Sensors
  - limited access to data
  - small network size
  - expensive
  - accurate but inconsistent monitoring

Application/Hypothesis

- The goal of this study was to develop a local, high-resolution air pollution model which can both predict changes in $PM_{2.5}$ levels and provide residents in the District of Columbia a tool for making healthier decisions about where and when they spend their time outside.
- Hypothesis
    - The accuracy of air quality predictions will be influenced by
        - spatial and temporal gaps
        - location features
        - meteorological inputs
- How are we going to do this?
    - public and private sensors
    - high resolution measurements
    - consider the impact of location, location features, and weather inputs on pm25 concentrations

Methodology

The input values/columns

```
cols = ['x', 'y', 'population', 'dist-mroads',
        'dist-setl', 'dist-coast', 'dist-forest', 'slope', 'elevation',
        'dayofweek', 'sin_day', 'cos_day', 'sin_year', 'cos_year', 'TEMP',
        'DEW', 'SKY', 'VIS', 'ATM', 'Wind-Rate', 'sin_wind', 'cos_wind']
y = df['pm25']
```

Classification Model Methodology

```
In [ ]: from sklearn.tree import DecisionTreeClassifier
        parameters = {'criterion':["gini","entropy"],
                      "max_depth":[2,3,4],
                      "min_samples_leaf":[5,8,10]}
        dt = DecisionTreeClassifier(random_state=0)
        clf = GridSearchCV(dt, parameters)
        clf.fit(X_train, y_train)
```

Regression Model Methodology

```
model_dict = {
    "DecisionTreeRegressor" : {"model":DecisionTreeRegressor,
                               "parameter":{
                                   "criterion": ["mse", "friedman_mse"],
                                   "splitter" : ["best", "random"],
                                   "min_samples_leaf" : [1,2,3]
                               },
                              'par':{
                                  "criterion": ["mse"],
                                   "splitter" : ["best", "random"],
                              }},
    "MLPRegressor" : {"model":MLPRegressor,
                               "parameter":{
                                   "hidden_layer_sizes": [100,200],
                                   "activation" : ['identity', 'logistic', 'tanh', 'relu'],
                                   "alpha" : [0.0001,0.001]
                               },
                                 'par':{
                                  "hidden_layer_sizes": [100],
                                   "activation" : ['identity'],
                              }},
    "KNeighborsRegressor" : {"model":KNeighborsRegressor,
                               "parameter":{
                                   "n_neighbors": [5,6],
                                   "algorithm" : ['auto', 'ball_tree']
                               },
                             "par":{
                                   "n_neighbors": [5],
                                   "algorithm" : ['auto']
                           }
}}
#list of models to be used
models = ['DecisionTreeRegressor', 'MLPRegressor', 'KNeighborsRegressor']
```

Data Pipeline

GEORGETOWN UNIVERSITY

Data Pipeline

Open AQ
Purple Air
Worldpop
Weather
Oceanic

Ingest

Wrangling/EDA
(Pandas)

Machine Learning: Supervised
(Scikit-Learn, Pandas)

Machine Learning: Supervised
(Scikit-Learn, Pandas,)

Storage on AWS , tmp and GitHub

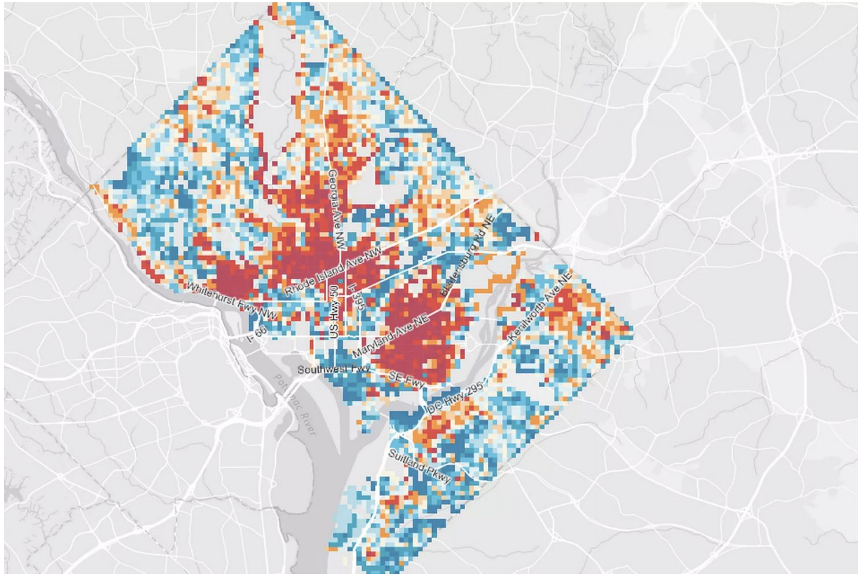
Ingestion and Wrangling

Ingestion & Wrangling

- Relied on 4 main data sources:
  - Open AQ
  - PurpleAir
  - WorldPop
  - Weather Data

- Ground sensors based on geographic locations.
  - In the DC metro area however it was expanded to beyond to get a clearer picture of the

- Switched to
  - CSV based as opposed to all the different file sources we had.



Pandas,

Scikit-learn, Numpy

Matplotlib

Wrangling

Weather / meteo data extract

weather data extraction made manually and storde locally

Data wrangling done here below

In [13]:
```
df = pd.read_csv("bigtable.csv")
df['time'] = pd.to_datetime(df["datetime"],format="%Y-%m-%d %H:%M:%S %Z")
df['date'] = pd.Series(df.time).dt.strftime("%Y-%m-%d")
df['hour'] = pd.Series(df.time).dt.strftime("%H")

print("nb ground station measurement : {}".format(len(df)))


#2019-01-01T02:40:00
df_meteo = pd.read_csv("weather/weather.csv")
df_meteo['time'] = pd.to_datetime(df_meteo["DATE"],format="%Y-%m-%dT%H:%M:%S")
df_meteo['date'] = pd.Series(df_meteo.time).dt.strftime("%Y-%m-%d")
df_meteo['hour'] = pd.Series(df_meteo.time).dt.strftime("%H")
df_meteo['datehour'] = pd.Series(df_meteo.time).dt.strftime("%Y-%m-%d-%H")

print("nb meteo : {}".format(len(df_meteo)))

df_meteo.head()
```

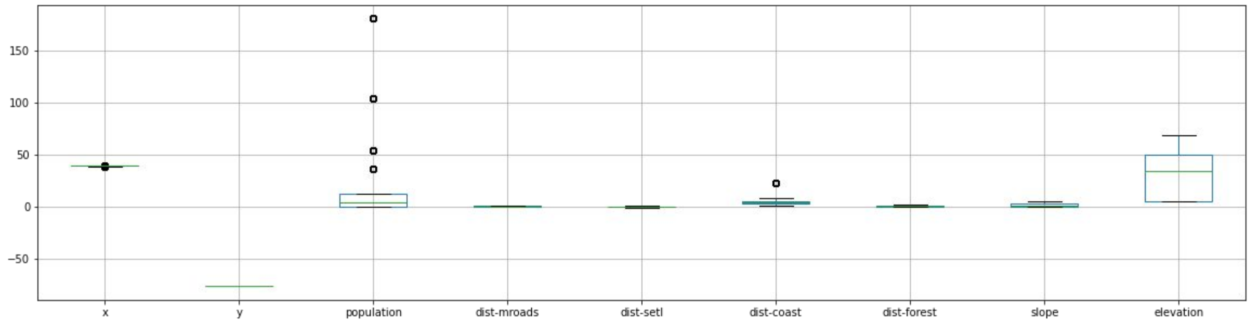
```
nb ground station measurement : 152624
nb meteo : 14449
```

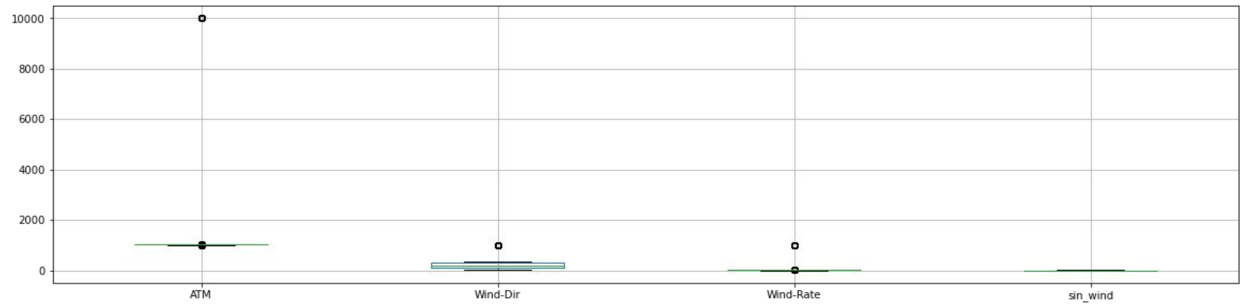
EDA
Exploratory Data Analysis

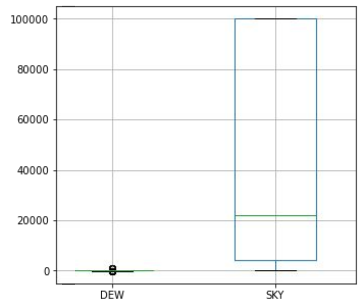
Univariate Analysis

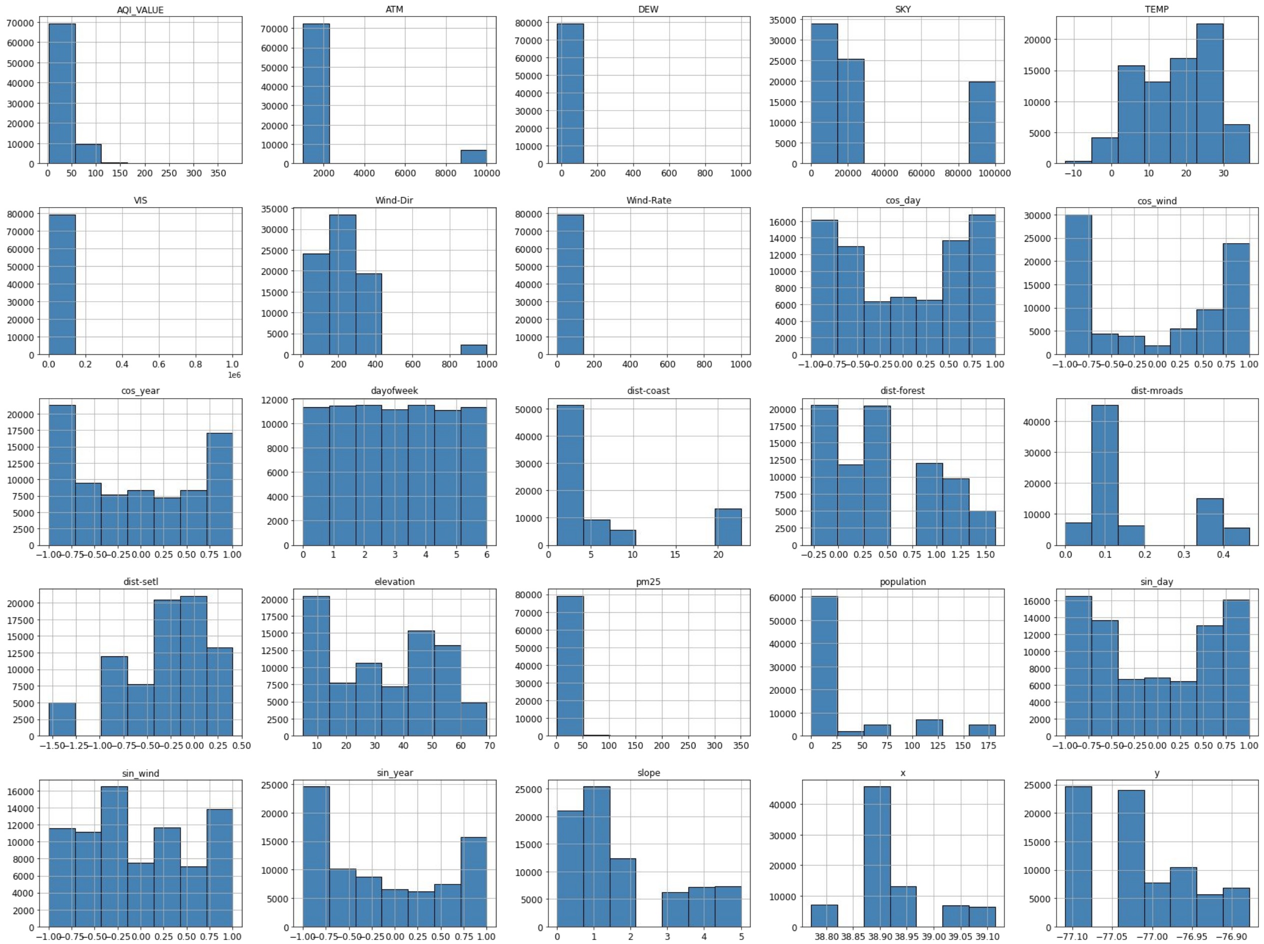
- Goal
  - understanding of individual features within themselves
  - shape, center, and spread of individual features
- Tools
  - boxcharts, histograms, descriptive statistics
- Findings
  - Original Shape: (105398, 30)
  - Missing data
  - Outliers for pm25 and most of our weather features
- Outcomes
  - removed instances with Nan (-7724)
  - IQR to remove weather outliers (+30000)

Outlier Detection








AQI_VALUE
ATM
DEW
SKY
TEMP
VIS
Wind-Dir
Wind-Rate
cos_day
cos_wind
cos_year
dayofweek
dist-coast
dist-forest
dist-mroads
dist-setl
elevation
pm25
population
sin_day
sin_wind
sin_year
slope
x
y

Bivariate Analysis

- Goal
  - understanding of the relationship between the features in our dataset
  - Pearson correlation
- Tools
  - correlation
  - correlation heatmaps
  - scatter plots
- Outcomes
  - eliminated any linear models for machine learning
  - helped eliminate many non-essential features (e.g. VIS, SKY)

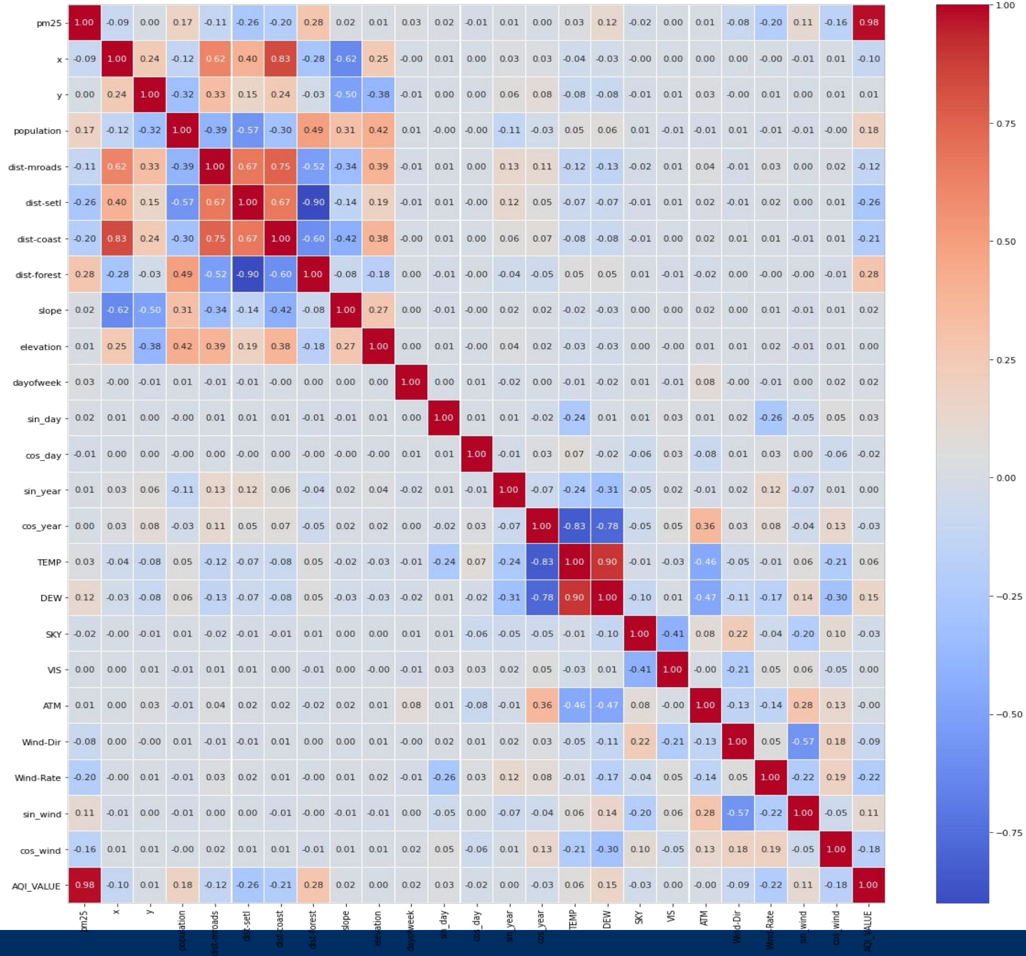
Strongly correlated variables:

AQI and pm 2.5

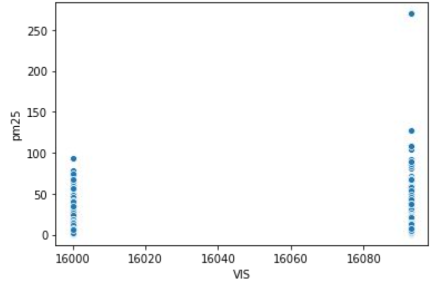
sin(year), temperature, and dewpoint

geographic variables

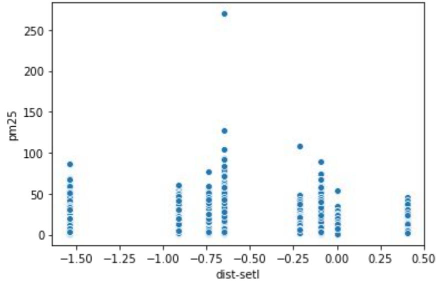
Nothing is strongly correlated with our target variables (AQI and pm 2.5)



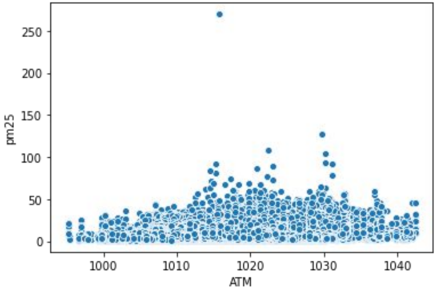
pm2.5 vs



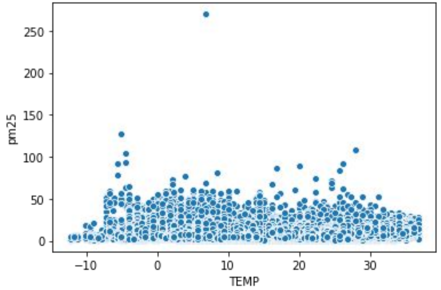
Visibility



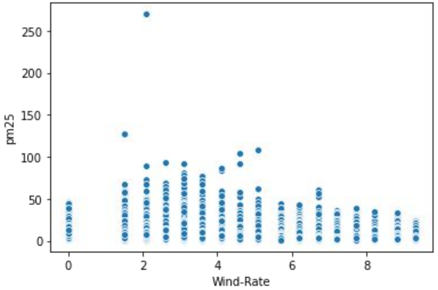
Distance from settlement



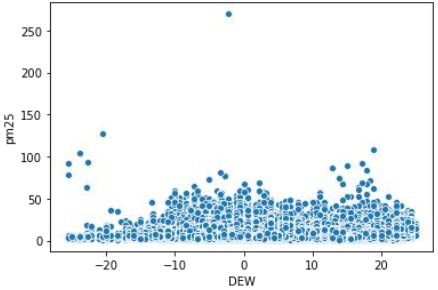
Pressure



Temperature



Windspeed

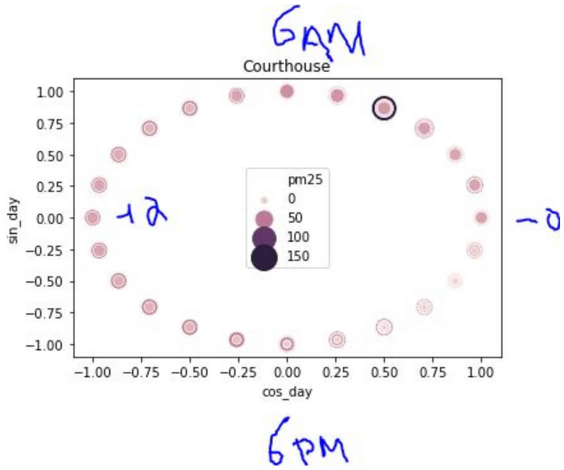


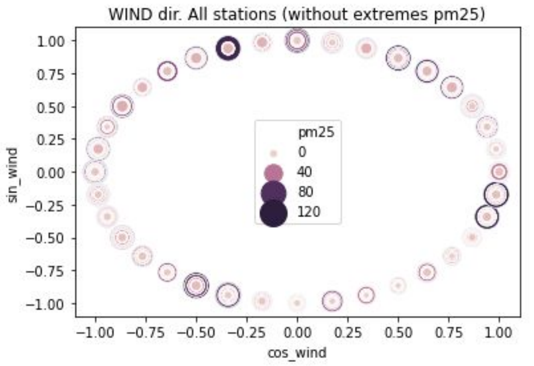
Humidity

Feature Engineering

Circular features

- datetime was splitted into 3 features :
  - sin/cos cyclic position in the year
  - sin/cos cyclic position in the day
  - category 1 to 7 for day number in the week
- Wind direction was also transformed into cos/sin cyclic



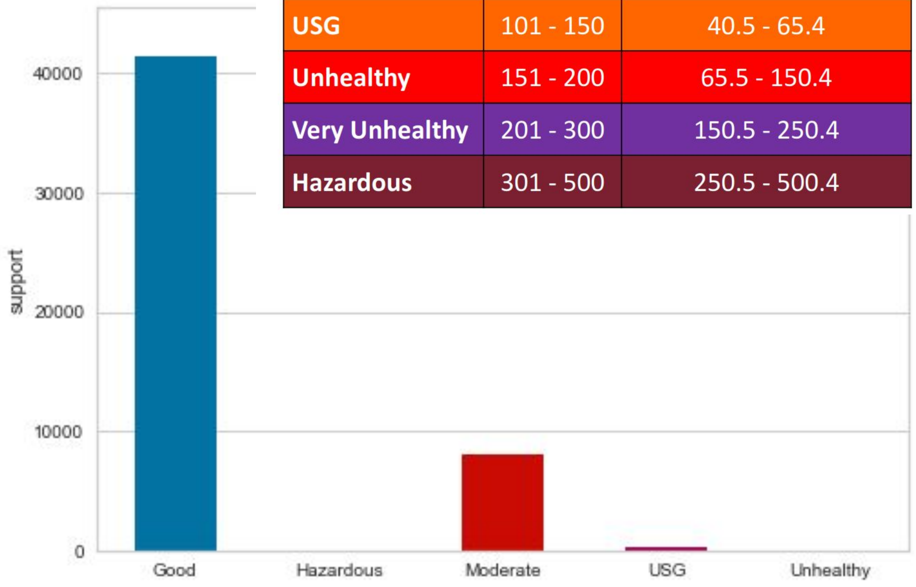


Feature Engineering

- Transformation of PM25 to EPA AQI
- Binary classification was chosen : Will the air quality be good or not.

U.S. EPA $PM_{2.5}$ AQI

AQI Category	AQI Value	24-hr Average $PM_{2.5}$ Concentration ($\mu g/m^3$)
Good	0 - 50	0 - 15.4
Moderate	51 - 100	15.5 - 40.4
USG	101 - 150	40.5 - 65.4
Unhealthy	151 - 200	65.5 - 150.4
Very Unhealthy	201 - 300	150.5 - 250.4
Hazardous	301 - 500	250.5 - 500.4



Model Selection and Machine Learning

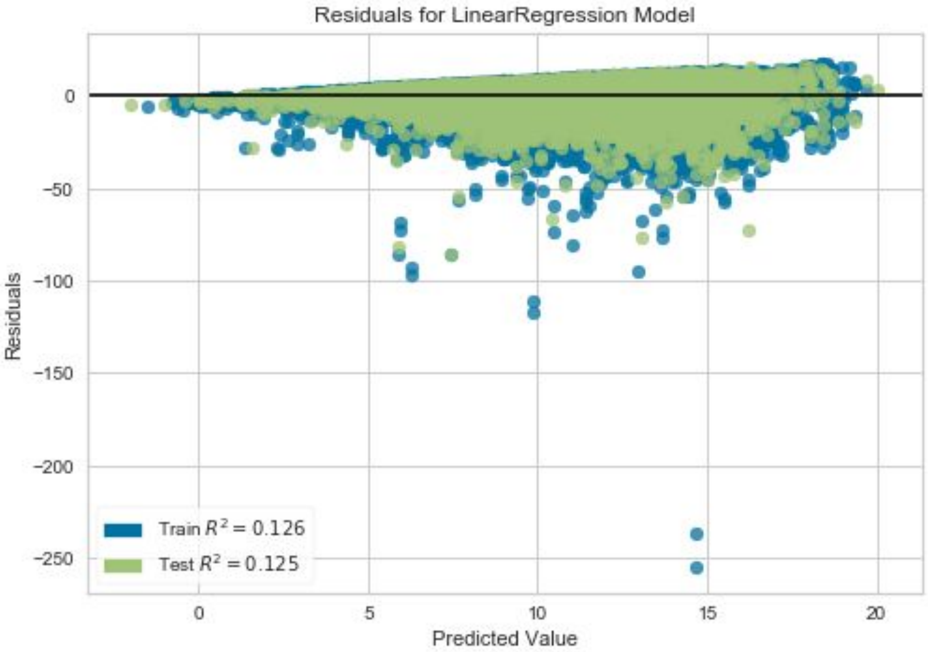
Regression models

Machine Learning

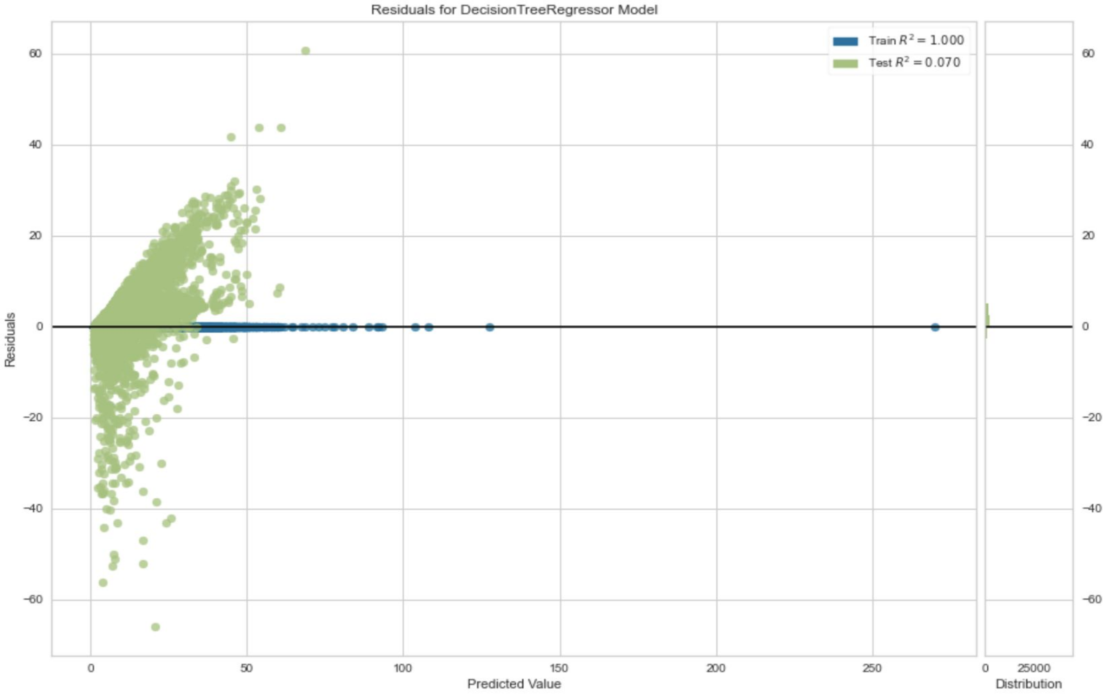
Training and Testing models

```
[54]: # dictionary mapping the model classes
model_dict = {
    "DecisionTreeRegressor" : DecisionTreeRegressor,
    "MLPRegressor" : MLPRegressor,
    "KNeighborsRegressor" : KNeighborsRegressor,
    "BayesianRidge" : BayesianRidge,
    "LinearRegression" : LinearRegression
}
#list of models to be used
models = ['DecisionTreeRegressor', 'MLPRegressor', 'KNeighborsRegressor','BayesianRidge','LinearRegression']

# storing the columns to be used for training the models
# 1 -> no filter
# 2 -> using colums with little corelation
# 3 -> using columns with more corelation
process_cols = [
    [ 'type', 'sensor',  'x', 'y',
      'population', 'dist-mroads', 'dist-setl', 'dist-coast', 'dist-forest',
      'slope', 'elevation', 'dayofweek', 'sin_day', 'cos_day', 'sin_year',
      'cos_year', 'TEMP', 'WIND', 'DEW', 'SKY', 'VIS', 'ATM']
]
```

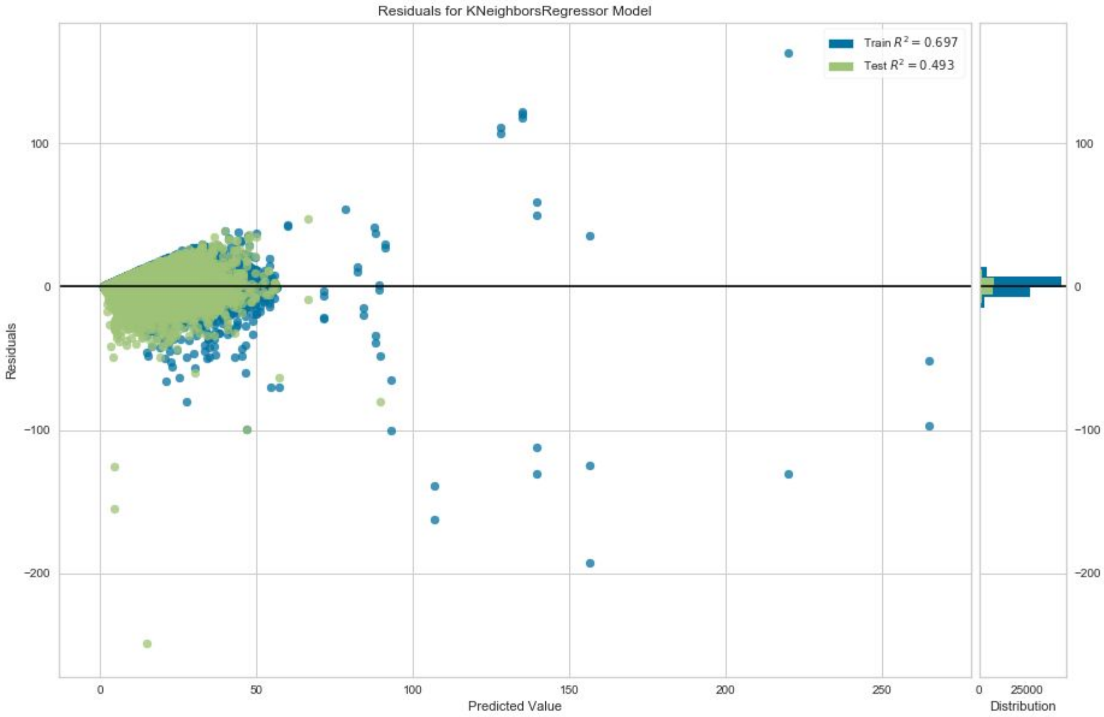
Regression Model Analysis



Decision Tree Regressor was overfit to the test sample



Residuals for KNeighbors Regressors Model



	model	group1	group2	group3	group4	group5	rsquare Var	rsquare mean
0	<class 'sklearn.linear_model._base.LinearRegre...	-0.702871	0.094141	-0.066153	-0.166127	0.043475	0.081909	-0.159507
1	<class 'sklearn.neighbors._regression.KNeighbo...	-0.227923	0.547875	0.301412	0.238347	0.152194	0.063629	0.202381
2	<class 'sklearn.linear_model._bayes.BayesianRi...	-0.699237	0.094671	-0.065771	-0.165153	0.043324	0.081174	-0.158433
3	<class 'sklearn.ensemble._gb.GradientBoostingR...	-0.026633	0.315593	0.218278	0.370938	0.084876	0.021452	0.192611
4	<class 'sklearn.neural_network._multilayer_per...	-18.503900	0.312136	0.141989	-1.167200	-0.068001	53.899367	-3.856995
5	<class 'sklearn.tree._classes.DecisionTreeRegr...	-0.996818	-0.169596	-0.679429	-0.223873	-0.099959	0.120529	-0.433935

Conclusion

- Best regression model among seems to be KNeighborsRegressor
- Mean of RSquare 0.2 with a variance of 0.06

We concluded that regressions were maybe too complex and because our data product can support a more simple binary classification, we decided to invest more time into classifiers

Classification models

Machine Learning

- Suspiciously too good results
- Overfitting or “ground station memory”

```python
models = [
    SVC(gamma='auto'),
    # NuSVC(gamma='auto'),
    LinearSVC(),
    #SGDClassifier(max_iter=100, tol=1e-3),
    KNeighborsClassifier(),
    LogisticRegression(solver='lbfgs'), #LogisticRegressionCV(cv=3),
    BaggingClassifier(),
    ExtraTreesClassifier(n_estimators=100),
    RandomForestClassifier(n_estimators=100),
    DecisionTreeClassifier()
]


for model in models:
    score_model(X, y, model)
```

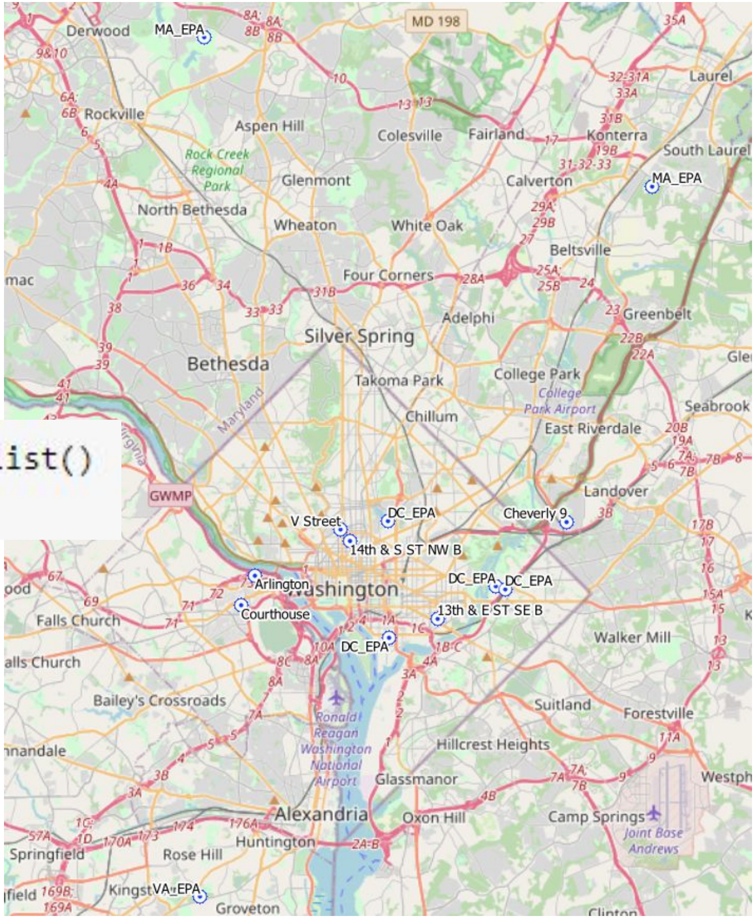
```
SVC: 0.275444345149439
LinearSVC: 0.0007155635062611806
KNeighborsClassifier: 0.6596370143149284
LogisticRegression: 0.01696885169688517
BaggingClassifier: 0.9802424242424244
ExtraTreesClassifier: 1.0
RandomForestClassifier: 1.0
DecisionTreeClassifier: 1.0
```

Machine Learning

.

- The test train split strategy was adjusted to group by station

```
groups = gs["station_id"].astype('category').cat.codes.tolist()
gkf = GroupKFold(n_splits=13)
```



Machine Learning

- We focused on the recall of predicting bad weather.

	model	group1	group2	group3	group4	group5	F1 NotG var	F1 NotG mean	Recall NotG var	Recall NotG mean
0	BaggingClassifier	0.280231	0.302059	0.459842	0.327219	0.665599	0.020623	0.406990	0.012572	0.477286
1	ExtraTreesClassifier	0.279841	0.340892	0.404481	0.390972	0.339450	0.001951	0.351127	0.035298	0.448685
2	RandomForestClassifier	0.297099	0.418886	0.448581	0.356493	0.356469	0.002819	0.375506	0.013719	0.395908
3	DecisionTreeClassifier	0.256872	0.259951	0.468585	0.344564	0.665886	0.023714	0.399171	0.012747	0.504688

Decision tree was actually the best model so far.

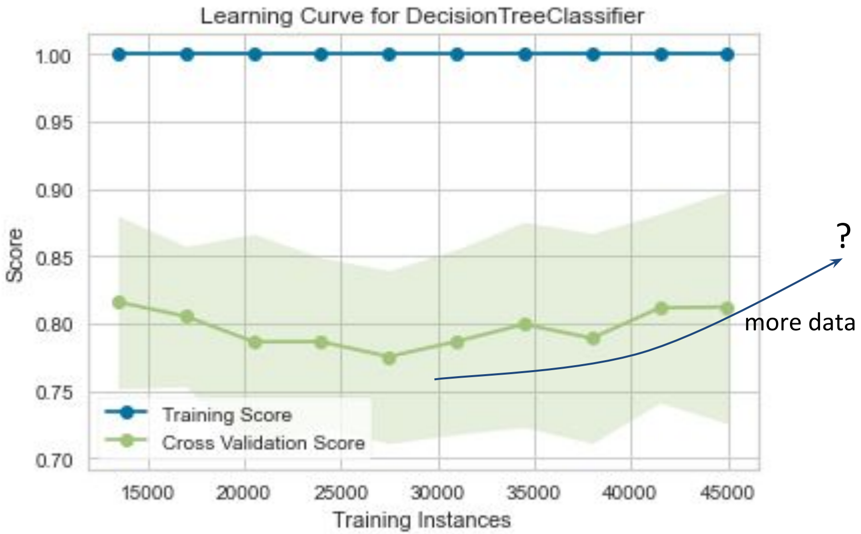
Results
Reflections,
Next Steps
Conclusion

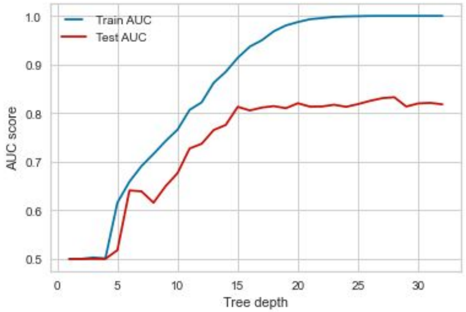
Results

We reached 0.50 recall of predicting bad air quality so far using DecisionTreeClassifier.

We didn't find substantial better results after hyper parameter tuning.

Our first conclusions are there is potential to achieve a better modelization and we need more data.





First data product preview

One minute app

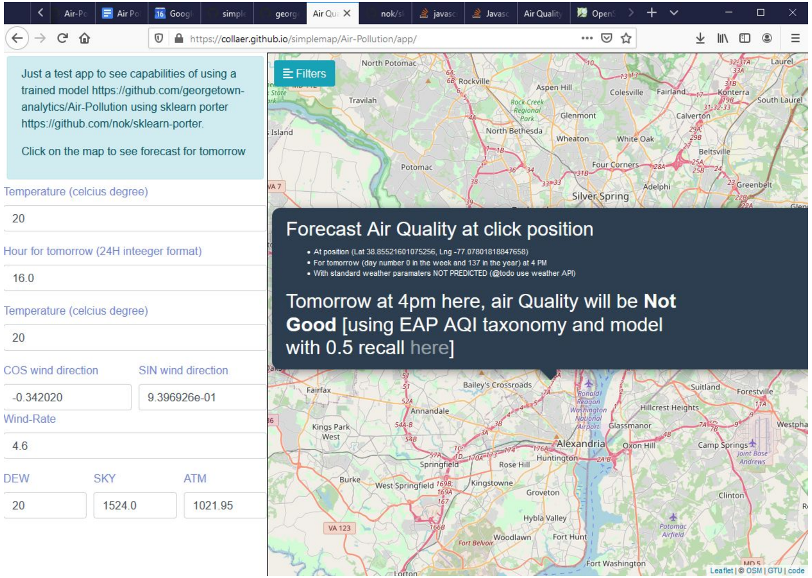
- Using https://github.com/nok/sklearn-porter to export model
    - Javascript file of 21000 lines
- Connecting the model.js with simple html/js app, using open source license libraries:
        - leaflet and leaflet plugins
        - bootstrap
        - jquery
    - And CC open data www.openstreetmap.org
- Directly published inside Github using gh-pages branch
- gh-pages option is not activated for our repo
    - URL here: https://georgetown-analytics.github.io/Air-Pollution/app/test
    - Image here: https://collaer.github.io/simplemap/Air-Pollution/app/

```
from sklearn_porter import Porter
# Export:
porter = Porter(myTree, language='js')
output = porter.export(embed_data=True)
print(output)
```

First data product preview

model.js map.js

```
var DecisionTreeClassifier = function() {

    var findMax = function(nums) {
        var index = 0;
        for (var i = 0; i < nums.length; i++) {
            index = nums[i] > nums[index] ? i : index;
        }
        return index;
    };

    this.predict = function(features) {⇔};

};

if (typeof process !== 'undefined' && typeof process.argv !== 'undefined') {
    if (process.argv.length - 2 === 14) {

        // Features:
        var features = process.argv.slice(2);

        // Prediction:
        var clf = new DecisionTreeClassifier();
        var prediction = clf.predict(features);
        console.log(prediction);

    }
}
```



Try, Try Again

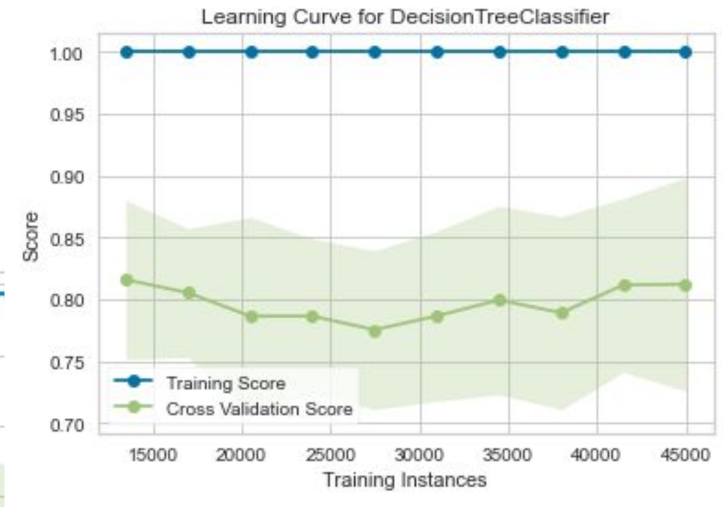
A tale of pivoting…

- Test train split data strategy, sometimes randomness is not your ally
- Data science pipeline is a loop, test, retry, removes, ...
- And of course, python stack scikit learn, yellowbrick, pandas, seaborn, etc. is awesome !

Next Steps

We believe we can reveal more information about the Air Pollution by the following improvements:
More data!

Sometimes you find something you were not looking for… Need to dig deeper but our study suggest something about ground stations efficient management: After a period (maybe lower than a year) you can move your station to another place, because we may build a model to predict that station future values...



Thank you for your time!

17 89

GEORGETOWN UNIVERSITY

Feedback, reactions, questions