A Machine Learning Approach to Evaluate Beijing Air Quality

By

Mingye Yang

SENIOR THESIS

Submitted in partial satisfaction of the requirements for High Honors for the degree of

BACHELOR OF SCIENCE

in

MATHEMATICS

in the

COLLEGE OF LETTERS AND SCIENCE

of the

UNIVERSITY OF CALIFORNIA,

DAVIS

Approved:

Jesús A. De Loera

June 2018

Contents

CHAPTER 1

Preliminaries

1.1. Abstract & Introduction

Air Pollution has been considered as one of the most serious environmental issues in China. In 2015, only 73 out of 338 major cities met the national standard for air quality [**1**]. Major pollutants, especially Fine particles (particulate matter with diameter less than 2.5 μm), are robustly associated with adverse health effects, including cardiac and respiratory morbidity and mortality [**2**]. Research from an independent group even showed that air pollution cause 1.6 million deaths in 2016 [**3**].

Many highly developed cities and northern "edge cities" in China usually incur air pollution problems. In this paper, I will be examining Air Quality in Beijing since it is the most representative city that suffers from air pollution.

The methods of analysis and the models we will use are inspired by Paulo Cortez and Aníbal Morais's paper ***A Data Mining Approach to Predict Forest Fires using Meteorological Data***.[**13**]

In that paper, five Data mining / Machine Learning Techniques (multiple regression, Decision Tree, Random Forest, Neural Network and Support Vector Machine) are implemented to predict the burned size of forest fires in a northeast park area in Portugal. The data-collection region is divided into a 9*9 grid and assigned spatial coordinates accordingly. Fire time (month and day), meteorological variables (temperature, relative humidity, wind and rain) , Forest Fire Weather Index Components and spatial locations are contained in the feature spaces. Feature selections and data preprocessing are conducted to ensure that the input variables are appropriate. Hyperparameters tuning are also done to boost up the accuracy.

In this paper, Beijing City is divided into 6 circles. Each air quality monitor stations location (longitude and latitude), as well as the date information (month, day, hour) are included in the feature spaces. Major air pollutant measurements are also recorded. Data-preprocessing is done to avoid NAs, divide the dataset into training set and test set, scale the features, and convert some features type if necessary. In terms of machine learning teniques, I have applied Random Forest (since its just a collection of Decision trees method and its way more accurate.) to regress the Air Quality Index and Support Vector Machine to classify the Air Quality Level. However, instead of merely applying the method and showing the results, detailed interpretations of the these two Machine Learning models are provided. I have also applied features selections in order to reduce the hyperparameter tuning time.

1.2. Data

The data used in the experiment is collected from China National Urban Air Quality Real-time Publishing Platform. Air Quality Index (AQI) is a number that measures the pollution level of the ambient air and it is measured based on 6 air pollutants: Sulfur dioxide (SO_2), Nitrogen dioxide (NO_2), Carbon monoxide (CO), Ozone (O_3), $PM_{2.5}$ (particulate matter with diameter less than 2.5 μm) and PM_{10} (particulate matter with diameter less than 10 μm) [**4**]. AQI is classified into into 6 levels of severity based on the value:

- Level 1: Excellent. AQI ranges from $0 \sim 50$.
- Level 2: Good. AQI ranges from $51 \sim 100$.
- Level 3: Lightly Polluted. AQI ranges from $101 \sim 150$.
- Level 4: Moderately Polluted. AQI ranges from $151 \sim 200$.
- Level 5: Heavily Polluted. AQI ranges from $201 \sim 300$.
- Level 6: Severely Polluted. AQI > 300.

This paper uses Beijing AQI data in year 2015. For better interpretation and visualization, Beijing city is divided into 6 spatial locations by circles (see Figure 1). Location of each 12 national Environmental Protection Monitoring Centers is also provided (see Figure 2). The station monitors air quality features and reports the information hourly.

A small sample of the dataset is given (see Figure 3) to illustrate what the dataset looks like. Attribute description of the dataset is then followed in Table 1.

Machine Learning techniques, Random Forest (RF) and Support Vector Machine (SVM) are implemented to predict the AQI value and classify the pollution level. Before applying the Machine Learning models, the dataset is being cleaned and pre-processed through the following steps:

- Cleaned missing values
- Converted categorical data to numerical (if necessary)
- Splitted dataset into training set and test set
- Feature scaling (if necessary)

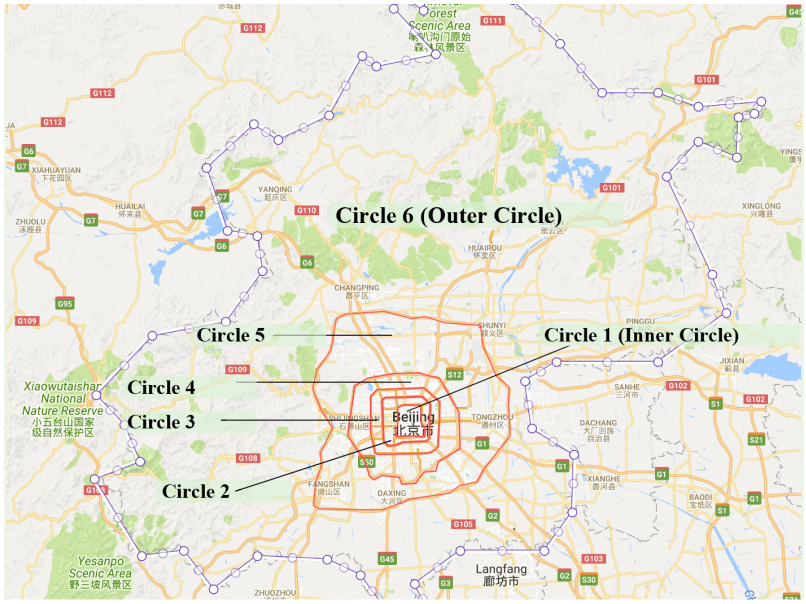

FIGURE 1. Circles of Beijing

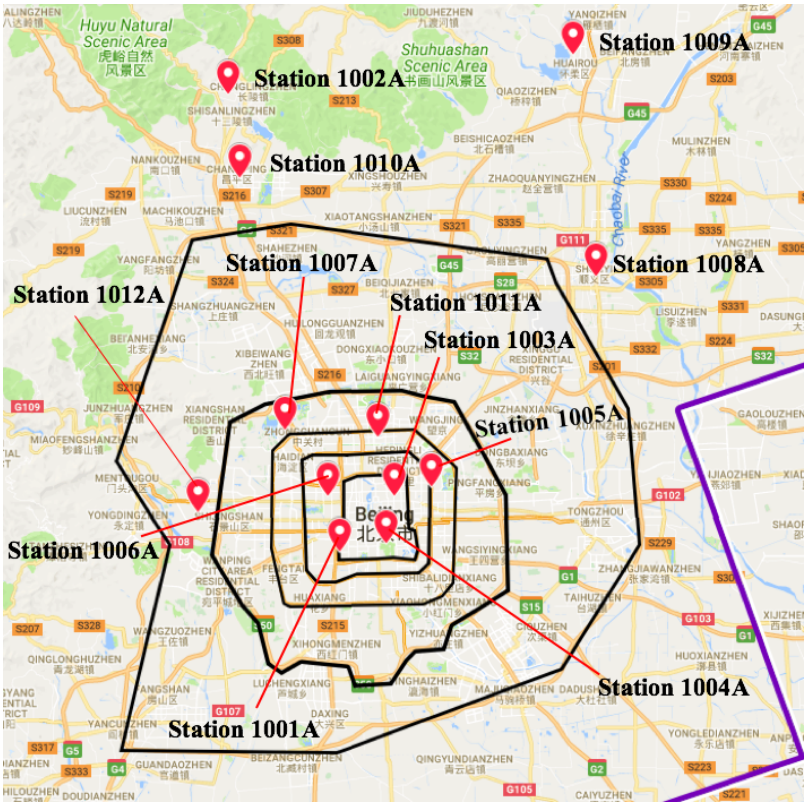

FIGURE 2. National Environmental Protection Monitoring Centers in Beijing

month	day	hour	station_code	longitude	latitude	circle	level_index	pollutions	so2	so2_24h	no2	no2_24h	co	co_24h	o3	o3_24h	o3_8h	o3_8h_24h	pm10	pm10_24h	pm2_5	pm2_5_24h	aqi
1	2	1	1001A	116.352	39.878	2	3	PM2.5	45	31	108	72	3	1.424	5	65	10	54	141	91	88	48	117
1	2	2	1001A	116.352	39.878	2	3	PM2.5	49	31	119	74	3	1.5	8	65	7	54	158	94	94	50	124
1	2	3	1001A	116.352	39.878	2	3	PM2.5	44	33	108	78	3	1.595	6	65	7	51	134	99	82	54	109
1	2	4	1001A	116.352	39.878	2	2	PM10	28	33	92	79	2	1.591	4	65	6	51	100	99	47	53	75
1	2	5	1001A	116.352	39.878	2	2	PM10	24	32	86	79	2	1.588	4	65	6	51	78	98	34	52	64
1	2	15	1001A	116.352	39.878	2	2	PM10	26	31	65	79	1	1.504	30	60	34	42	57	89	23	48	54
1	2	16	1001A	116.352	39.878	2	2	PM10	39	32	74	80	1	1.521	32	50	37	42	118	90	54	49	84
1	2	17	1001A	116.352	39.878	2	3	PM10	59	33	92	81	2	1.542	21	50	37	42	169	94	79	50	110
1	2	18	1001A	116.352	39.878	2	3	PM2.5	61	34	104	82	2	1.558	7	50	35	40	176	96	89	51	118
1	2	19	1001A	116.352	39.878	2	3	PM2.5	54	35	95	82	2	1.554	9	50	31	38	154	96	77	51	103
1	2	20	1001A	116.352	39.878	2	3	PM2.5	58	35	91	81	2	1.533	11	50	26	37	150	95	79	51	105
1	2	21	1001A	116.352	39.878	2	3	PM2.5	49	35	93	80	2	1.525	12	50	22	37	167	95	92	51	122
1	2	22	1001A	116.352	39.878	2	3	PM2.5	69	36	99	79	3	1.546	8	50	16	37	180	96	112	51	147
1	2	23	1001A	116.352	39.878	2	4	PM2.5	78	37	97	79	3	1.563	8	50	14	37	191	98	120	53	158

FIGURE 3. Small sample of the dataset

Attribute	Description
month	Month of the year (1 to 12)
day	Day of the month (1 to 31)
hour	Hour of the day (0 to 23)
station_code	Twelve stations in Beijing (1001 to 1012)
longitude	Longitude of the station
latitude	Latitude of the station
circle	Circle each station monitors (1 to 6)
level_index	AQI levels (1 to 6)
so2	SO_2 1-hour average (in $\mu g/m^3$)
so2_24h	SO_2 24-hour moving average (in $\mu g/m^3$)
no2	NO_2 1-hour average (in $\mu g/m^3$)
no2_24h	NO_2 24-hour moving average (in $\mu g/m^3$)
co	CO 1-hour average (in mg/m^3)
co_24h	CO 24-hour average (in mg/m^3)
o3	O_3 1-hour average (in $\mu g/m^3$)
o3_24h	O_3 Daily maximum 1-hour average (in $\mu g/m^3$)
o3_8h	O_3 8-hour moving average (in $\mu g/m^3$)
o3_8h_24h	O_3 Daily maximum 8-hour moving average (in $\mu g/m^3$)
pm10	PM_{10} 1-hour average (in $\mu g/m^3$)
pm10_24h	PM_{10} 24-hour moving average (in $\mu g/m^3$)
pm2_5	$PM_{2.5}$ 1-hour average (in $\mu g/m^3$)
pm2_5_24h	$PM_{2.5}$ 24-hour moving average (in $\mu g/m^3$)
aqi	Air Quality Index

TABLE 1. AQI Dataset Attribute Description

CHAPTER 2

Some Machine Learning Concepts

Before getting in depth discussion of machine learning, it is crucial to know some basic but important concepts in Machine Learning.

Definition 1. **Machine Learning** teaches computers to do what comes naturally to humans and animals: learn from experience. Machine learning algorithms use computational methods to learn information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases. [**12**]

Machine Learning can be used in many cases : Face detection, demand/supply optimization, disease identification, etc. Living in a world driven by data, people can use machine learning as a useful and robust tool to make business decisions.

Definition 2. **Supervised learning** entails learning a mapping between a set of input variables X and an output variable Y and applying this mapping to predict the outputs for unseen data. [**6**]

Definition 3. **Unsupervised learning** studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. [**7**]

In other way, in supervised learning, input predictors is used to train a model to classify or regress associated response variables. For unsupervised learning, there will be no response variables and the goal is to seek the relationship between variables. This paper will focus on supervised learning.

Definition 4. In machine learning, **hyperparameter optimization or tuning** is the problem of choosing a set of optimal hyperparameters for a learning algorithm. [**11**]

Choosing a better set of hyperparameters in a model is crucial for improving the model and getting higher accuracy.

CHAPTER 3

Methods & Models

3.1. Random Forest

Before getting to know **Random Forest (RF)**, we first have to know what a **Decision tree** is.

Decision tree is more well-known as Classification & Regression Trees (CART) methodology. The algorithm is a top-down,greedy approach that is known as ***recursive binary splitting***.[**14**] Starting from the top, all observations are in the same region. Then a series of questions are asked at each node (referred as ***internal nodes***) to split observations into subregions based on the result of the questions. The algorithm stops when pre-defined stopping criterion is met. A general criterion is when each node contains less than a specific number of observations. Now each observation falls into one subregion (known as ***terminal nodes*** or ***leaves***)[**14**].

Regression tree (decision tree for regression) uses variances reduction metric to choose the best splitting parameter. Each separation by the chosen splitting features should give the maximal information. In math, the algorithm chooses the feature and its level/value to minimize

$$\frac{1}{|S_t|^2}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i-x_j)^2+\frac{1}{|S_f|^2}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i-x_j)^2\ [\mathbf{15}]$$

where x represents the response value for each observation; S_t is the set of observations for which the splitting result is true and S_f is the set of observations for which the splitting result is false.

Now, the prediction is simply taking the mean of the response variable of all observation in each terminal node. The plot below shows how to use weather information to predict hours played:

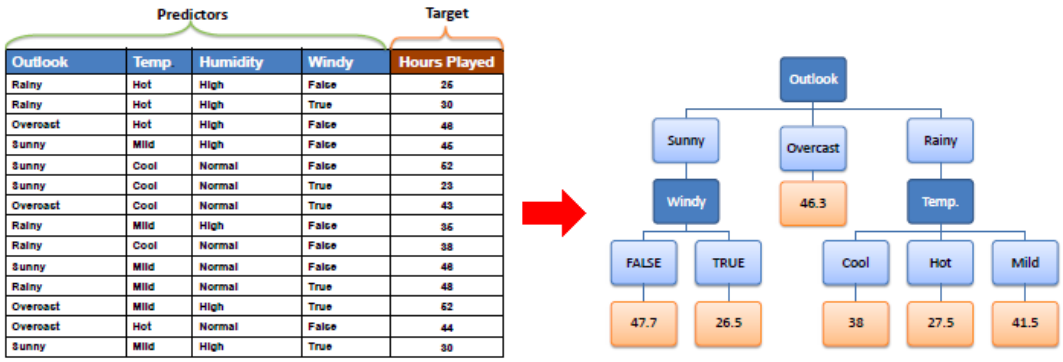

FIGURE 1. A simple regression tree example extracted from [**16**]

If we grow a very large tree, the model will have really high variance and might overfit, while a smaller tree model may result in less precise prediction. So we have to tune the tree size to make it appropriate. Generally, ***cost-complexity pruning*** method is used to reach this goal: [**9**]

First, we grow a large tree T_0, then we prune this tree to obtain any subtree $T \subset T_0$, and we index each terminal nodes by m and its respective sub-region as R_m, then we define:

$$|T| = \text{number of terminal nodes in T},$$

$$N_m = \#\ \{x_i \in \mathrm{R}_m\} \text{ where } x_i \text{ denotes the observation},$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i \text{ where } y_i \text{ is the respective response variable}$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

Then the cost complexity criterion is defined as

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|.$$

The main idea of this algorithm is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$. α is exactly the tuning parameter. It is intuitive to see that as α gets bigger, a smaller size subtree is required in order to minimize $C_\alpha(T)$. We use the ***weakest link pruning*** method to find T_α, that is, to successively collapse the internal node that produces the smallest per-node increase in $\sum_m N_m Q_m(T)$, and continue until we produce the single-node (root) tree. Estimation of α is achieved by five- or tenfold cross-validation: we choose the value $\hat{\alpha}$ to minimize the cross-validated sum of squares. Our final tree is $T_{\hat{\alpha}}$. [**9**]. A more detailed description of this algorithm could be found in Chapter 9 of the book **The Elements of Statistical Learning**. [**9**]

Though decision tree method is fast and could be easily visualized and interpreted, the prediction accuracy is not quite favorable compared to other regression approaches. **Random Forest (RF)** method is thus introduced to overcome the disadvantages of decision trees.

Random Forest is then simply a method that grows a collection of decision trees and gives the aggregated results. What is notable in the RF model is that it also implements a smart algorithm to decorrelate the trees: Instead of searching for the best splitting feature among the whole feature space, RF model selects the best splitting feature among a random sample of m features at each internal node. Typically, m is chosen to be the approximate of square root of total number of feature space.[**8**]

This algorithm perfectly mitigates the effect of having strong variables. Without this algorithm, decision trees in the forest will have similar structure due to the variance reduction metric and thus, the reduction in variance of the model might not be substantial. The implementation of this algorithm in the RF model gives each feature equal chance to be considered in the splitting step and make the model more appropriate.

3.2. Support Vector Machine

Some concepts are required to understand the algorithm of **Support Vector Machine (SVM)**:

DEFINITION 5. In a p-dimensional space, a **hyperplane** is a flat affine subspace of dimension $p-1$. In math, a p-dimensional hyperplane is define as:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0 \text{ [17]}$$

For parameters $\beta_0, \beta_1, ..\beta_p$, the above equation separates the observations in two hyperplanes.

DEFINITION 6. **Margin** is the perpendicular distance from each training observation to a given separating hyperplane.

DEFINITION 7. A **Maximal margin hyperplane** is the seperating hyperplane that is farthest from the training observation, that is, a hyperplane for which the margin is largest.

DEFINITION 8. The **Support vector classifier** classifies a test observation depending on which of a hyperplane it lies. It allows some observations to be on the wrong side of the margin or hyperplane (since some datasets are can not be perfectly separated by a maximal margin hyperplane. It is the solution to the following optimization problem (where M is the margin and C is the tuning parameter). An example of Support Vector Classifier is also given below.[17]

$$\begin{aligned}
&\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \; M \\
&\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \\
&y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1-\epsilon_i), \\
&\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C,
\end{aligned}$$

FIGURE 2. Optimization expression of Support Vector Classifier from [17]

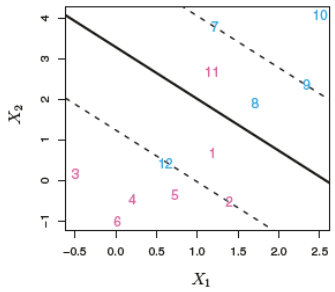

FIGURE 3. Support Vector Classifier Example from [17]

More detailed explanation of the above concepts could be found in Chapter 9 of the book **An Introduction to Statistical Learning with Application in R**. [17]

Then, Support Vector Machine is just an extension of Support Vector classifier. It uses **Kernels** to enlarge the feature space and include non-linear boundaries to classify.

DEFINITION 9. A **Kernel** is a function that takes two vectors X_i and X_j as arguments and returns the value of the inner product of their images $\phi(X_i)$ and $\phi(X_j)$:

$$K(X_1, X_2) = \phi(X_1)^T \phi(X_2)[?]$$

Now the support vector machine has the form:

$$f(x) = \beta_0 + \sum_{i \in S} K(x, x_i)$$

where S is the collection of indices of support vectors, the observations that lie on the margin. Here are some commonly used kernels [**18**]:

- Radial Kernel: $K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$
- linear Kernel: $K(x_i, x_{i'}) = 1 + \sum_{j=1}^{p} x_{ij} x_{i'j}$
- dth-degree polynomial: $K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$

Figure 4 and Figure 5 below show the Support Vector Machine classification using a degree 4 polynomial kernel and a radial kernel.

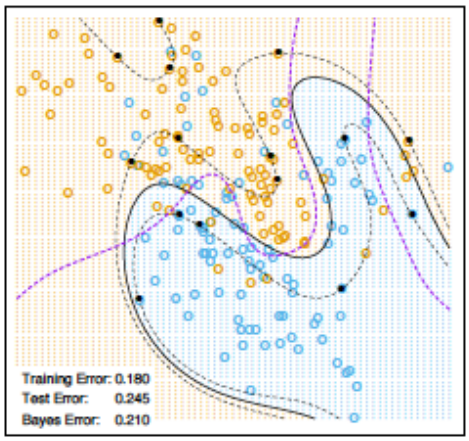

FIGURE 4. Degree 4 polynomial kernel from [**18**]

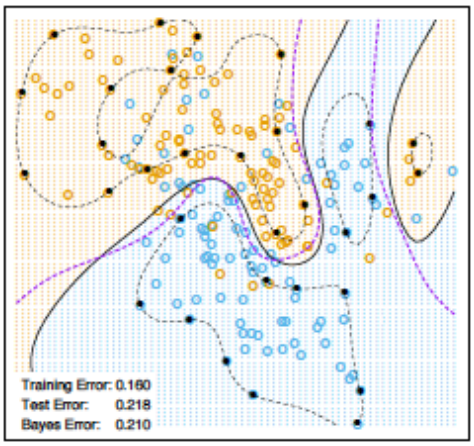

FIGURE 5. Radial kernel from [**18**]

Usually, SVM is applied for binary classification, but it can also be used to perform multi-classes classification by **One-Versus-One Classification** or **One-Versus-All Classification**:

Suppose we have K classes to be classified. One-Versus-One Classification basically chooses $\binom{K}{2}$ binary classes combination and assigns each observation to one of the classes. Eventually, the most frequently allocated class for each observation is the classification result.

A detailed Algorithm for One-Versus-All Classification is given in pseudocode:

Inputs:

- L, a learner (training algorithm for binary classifiers)
- samples X
- labels y where $y_i \in \{1, \dots K\}$ is the label for the sample X_i

Output:

- a list of classifiers f_k for $k \in \{1, \dots, K\}$

Procedure:

- For each k in $\{1, \dots, K\}$
  - Construct a new label vector z where $z_i = 1$ if $y_i = k$ and $z_i = 0$ otherwise
  - Apply L to X, z to obtain f_k

Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score:

$$\hat{y} = \underset{k \in \{1 \dots K\}}{\operatorname{argmax}} f_k(x)$$

FIGURE 6. Pseudo code for One-Versus-All Classification from [**19**]

CHAPTER 4

Experimental Results

4.1. Random Forest

The RF method fits the training dataset well and gives a 99 percent prediction accuracy. Since the dataset and the random forest are large, a regression tree pruned to the depth of 3 levels is presented below:

A table of the variable importance is also given:

Feature	**numerical importance**
level_index	0.93
pm2_5	0.05
pm10	0.01
pollutions_None	0.01
Other features	roughly zero

TABLE 1. Variable importance

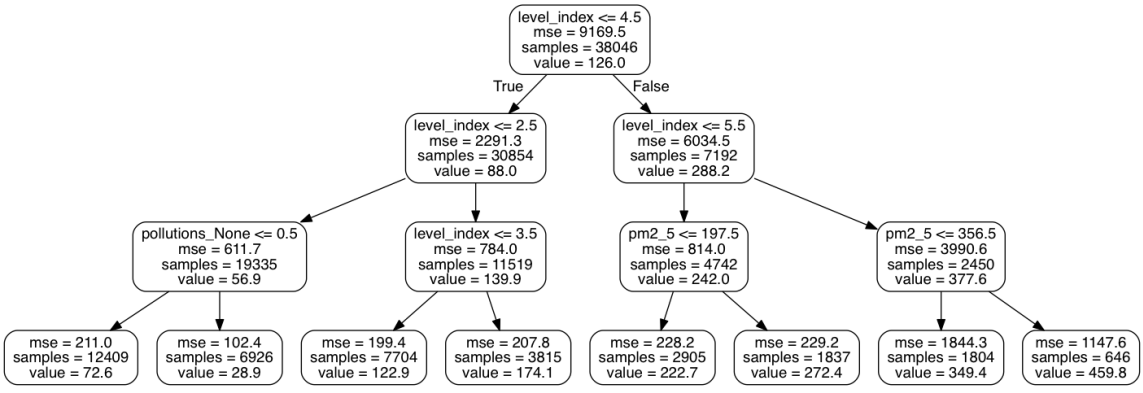

FIGURE 1. Regression tree of depth 3

Eventually, I use Random Grid Search to tune the hyperparameters:

Regressor Hyperparameter	**Selections**
bootstrap	[True, False]
max_depth	[10,20,30,40,50,60,70,80,90,100,110,None]
max_features	[auto,sqrt]
min_sample_leaf	[1,2,4]
min_sample_split	[2,5,10]
n_estimators	[20, 40, 60, 80, 100, 120, 140, 160, 180, 200]

TABLE 2. Hyperparameter tuning

The best hyperparameter selection is:

Regressor Hyperparameter	**Selection**
bootstrap	True
max_depth	100
max_features	auto
min_sample_leaf	1
min_sample_split	2
n_estimators	140

TABLE 3. Best hyperparameter selection

The accuracy improvement is minor since our original model already has really high accuracy.

4.2. Support Vector Machine

We use SVM model to classify the AQI index level. The default hyperparameters setting for the 3 kernels (radial,polynomial and linear) and the classification accuracy on the testdata are:

Kernel	Cost	Gamma	Degree	Classification Accuracy
Radial	1	0.009174312	—	0.9377155
Polynomial	1	—	3	0.6090913
Linear	1	—	1	0.9343915

TABLE 4. Default kernels hyperparameter setting Classification Accuracy

Since the dataset is too large and it will take too much time in the hyperparameter tuning part, feature importance is ranked firstly in this classification cases based on mean decrease of Gini Index. The result is shown below:

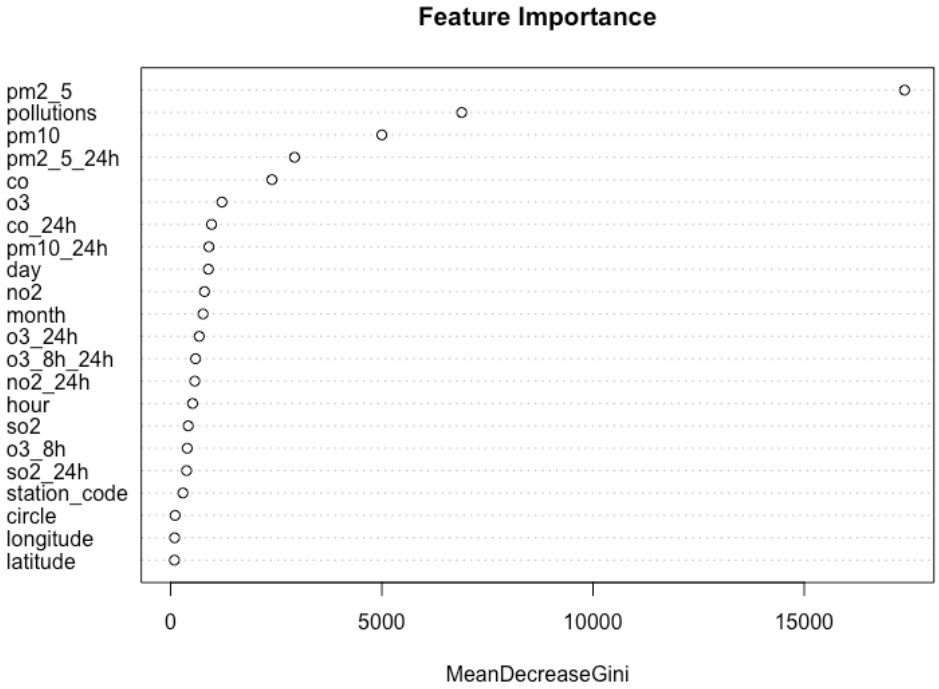

FIGURE 2. Feature Importance Ranking

Then, 5 most important features, pm2_5, pollutions, pm10,pm2_5_24h and co, are used to tune the hyperparameters.

Again, I use grid search with 5-fold cross validation to tune the hyperparameters.

The result for Radial Basis Function (RBF) kernel is shown below in Table 5:

Cost	Gamma	Accuracy
2^{-5}	2^{-9}	0.67628
2^{-5}	2^{-7}	0.76777
2^{-5}	2^{-5}	0.88082
2^{-5}	2^{-3}	0.92949
2^{-5}	2^{-1}	0.94557
2^{-3}	2^{-9}	0.76770
2^{-3}	2^{-7}	0.87971
2^{-3}	2^{-5}	0.93208
2^{-3}	2^{-3}	0.96100
2^{-3}	2^{-1}	0.96767
2^{-1}	2^{-9}	0.87706
2^{-1}	2^{-7}	0.92572
2^{-1}	2^{-5}	0.95777
2^{-1}	2^{-3}	0.97496
2^{-1}	2^{-1}	0.97856
1	2^{-9}	0.90482
1	2^{-7}	0.93887
1	2^{-5}	0.96569
1	2^{-3}	0.97982
1	2^{-1}	0.98208

TABLE 5. Radial kernel hyperparameter tuning

Cost	degree	Accuracy
2^{-5}	1	0.87229
2^{-5}	2	0.75749
2^{-5}	3	0.65925
2^{-5}	4	0.62625
2^{-3}	1	0.91687
2^{-3}	2	0.85650
2^{-3}	3	0.74909
2^{-3}	4	0.67377
2^{-1}	1	0.93577
2^{-1}	2	0.92661
2^{-1}	3	0.84444
2^{-1}	4	0.70409
1	1	0.94001
1	2	0.94254
1	3	0.88141
1	4	0.74808

TABLE 6. Polynomial kernel hyperparameter tuning

The best choice of Cost and Gamma is when Cost = 1 and Gamma = 0.5. The new SVM model scores 98.19255% accuracy, which is around 4.4% more accurate than the default model.

The tuning result for polynomial kernel is shown above in Table 6. Cost = 1 and degree = 2 is the best hyperparameter combination. The new SVM model gets 94.46961% accuracy, which is around 1.03% more accurate than the default model.

CHAPTER 5

Code

The following code is for **Random Forest:**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
df2 = pd.read_csv('Dataset_for_R.csv')
X = df2.iloc[:,:-1]
y = df2.iloc[:,-1:]
X = pd.get_dummies(X)
y = pd.DataFrame(y['aqi'].apply(np.log))
from sklearn.preprocessing import StandardScaler
X.iloc[:,6:21] = StandardScaler().fit_transform(X.iloc[:,6:21])
labels = np.array(y['aqi'])
feature_list = list(X.columns)
features = np.array(X)
from sklearn.model_selection import train_test_split
train_features, test_features, train_labels, test_labels =
train_test_split(features,
labels, test_size = 0.25, random_state = 42)


# Train the Model and try on test set.
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 100, random_state = 42)
rf.fit(train_features, train_labels);

predictions = rf.predict(test_features)
errors = abs(predictions - test_labels)
mape = 100 * (errors / test_labels)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
importances = list(rf.feature_importances_)
feature_importances = [(feature,
round(importance, 2)) for feature,
importance in zip(feature_list, importances)]
feature_importances = sorted(feature_importances, key = lambda x: x[1],
reverse = True)
[print('Variable: {:20} Importance: {}'.format(*pair)) for
pair in feature_importances];


# Hyperparameter tuning
from sklearn.model_selection import GridSearchCV
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
rf = RandomForestRegressor()
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
grid_search.fit(train_features, train_labels)
grid_search.best_params_
best_grid = grid_search.best_estimator_
grid_accuracy = evaluate(best_grid, test_features, test_labels)
print('Improvement: of {:0.2f}%.'.format(100 *
(grid_accuracy-base_accuracy)/base_accuracy))
```

The following code is for **Support Vector Machine:**

```
# Default SVM classificatin using R
setwd("~/Desktop/Thesis/Final")
df = read.csv('Dataset_for_R.csv')
df2 = df[,-24]
level = df2$level_index
df2$level = level
df3 = df2[,-8]
View(df3)
str(df3)
df3$level = factor(df3$level)
df3$circle = factor(df3$circle)
df3$month = factor(df3$month)
df3$day = factor(df3$day)
df3$hour = factor(df3$hour)
set.seed(1000)
train = sample(1:80224,56157,replace = FALSE)
traindata = df3[train,]
testdata = df3[-train,]
library('e1071')
fitdefaultradial = svm(level~.,data = traindata,kernel = 'radial')
fitdefaultpoly = svm(level~.,data = traindata,kernel = 'polynomial')
fitdefaultlinear = svm(level~.,data = traindata,kernel = 'linear')
summary(fitdefaultradial)
summary(fitdefaultpoly)
summary(fitdefaultlinear)
preddefaultradial = predict(fitdefaultradial,testdata)
table(preddefaultradial,testdata$level)  #22568/24067 = 93.77155%
predefaultpoly = predict(fitdefaultpoly,testdata)
table(predefaultpoly,testdata$level) #14659/24067 = 60.90913%
predefaultlinear = predict(fitdefaultlinear,testdata)
table(predefaultlinear,testdata$level) # 22488/24067 = 93.43915%


#Feature selection using R
set.seed(7)
library(mlbench)
library(caret)
control = trainControl(method="repeatedcv", number=10, repeats=3)
model = train(level~., data=df3, method="lvq",
preProcess="scale", trControl=control)
```

```
# Estimate variable importance
importance = varImp(model, scale=FALSE)
print(importance)
plot(importance)


# Hyperparameter tuning using python
import pandas as pd
df = pd.read_csv('Dataset_for_R.csv')
df2 = df.loc[:, ['pollutions','co','pm2_5','pm10','pm2_5_24h']]
level2 = df['level_index']
df2['level'] = level2
from sklearn.preprocessing import LabelEncoder
le_pollutions = LabelEncoder()
df2['pollutions_encoded'] = le_pollutions.fit_transform(df2.pollutions)
from sklearn.preprocessing import OneHotEncoder
pollutions_ohe = OneHotEncoder()

X = pollutions_ohe.fit_transform(df2.pollutions_encoded.values.
reshape(-1,1)).toarray()
dfOneHot = pd.DataFrame(X, columns = ["pollutions_"+str(int(i)) for i
in range(X.shape[1])])
df = pd.concat([df2, dfOneHot], axis=1)
set(df2['pollutions'])
df3 = df.drop(['pollutions', 'level','pollutions_encoded'], axis=1)
df4 = df3.copy()
df4['level'] = level2
X = df4.iloc[:,:-1]
y = df4.iloc[:,17]

from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,
test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from sklearn.grid_search import GridSearchCV
def grid_search_cv(clf, x, y, params, cv = 5):
    gs = GridSearchCV(clf, param_grid = params, cv = cv)
```

```
    gs.fit(x, y)
    print ("BEST", gs.best_params_, gs.best_score_, gs.grid_scores_)
    best_estimator = gs.best_estimator_
    return best_estimator


from sklearn import svm
params = {'C' : [2**-5,2**-3,2**-1,1],
'gamma' : [2**-9,2**-7,2**-5,2**-3,2**-1]}
clf = svm.SVC(kernel = 'rbf')
clf = grid_search_cv(clf, X_train, y_train, params)

params = {'C' : [2**-5,2**-3,2**-1,1], 'degree' : [1,2,3,4]}
clf2 = svm.SVC(kernel = 'poly')
clf2 = grid_search_cv(clf2, X_train, y_train, params)


# Refit the model with the best hyperparameter using R
fitradial_3 = svm(level2~.,data = traindata,cost = 1,gamma = 0.5)
summary(fitradial_3)
preddefaultradial_3 = predict(fitradial_3,testdata)
table(preddefaultradial_3,testdata$level2)     ###0.9819255

fitpoly = svm(level2~.,data = traindata,kernel = 'polynomial',
cost = 1, degree = 2)
summary(fitpoly)
predpoly = predict(fitpoly,testdata)
table(predpoly,testdata$level2) #0.9446961
```

Bibliography

[1] Wei, B. (2016). 2015 Report on the State of the Environment in China (Full article). Retrieved from http://cn.chinagate.cn/environment/2016-06/07/content_38617610_2.htm

[2] Du, X., Kong, Q., Ge, W., Zhang, S., Fu, L. (2010). Characterization of personal exposure concentration of fine particles for adults and children exposed to high ambient concentrations in Beijing, China. Journal of Environmental Sciences, 22(11), 1757-1764. doi:10.1016/s1001-0742(09)60316-8

[3] Rohde, R. A., Muller, R. A. (2015). Air Pollution in China: Mapping of Concentrations and Sources. Plos One, 10(8). doi:10.1371/journal.pone.0135749

[4] People's Republic of China Ambient air quality standards [PDF]. (2012, February 29).

[5] People's Republic of China Technical Regulation on Ambient Air Quality Index (on trial) [PDF]. (2012, February 29).

[6] Cord, M. (2008). Machine learning techniques for multimedia: Case studies on organization and retrieval ; 20 tables (pp. 21-49). Berlin: Springer.

[7] Dayan, P. (1999). Unsupervised Learning. The MIT Encyclopedia of the Cognitive Science.

[8] James, G. (2015). An introduction to statistical learning: With applications in R.(pp. 303-335) New York: Springer.

[9] Hastie, T., Tibshirani, R., Friedman, J. H. (2016). The elements of statistical learning data mining, inference, and prediction. (pp. 295-336). New York, NY: Springer.

[10] Drakos, N., Moore, R. (2016, August 19). Kernel Mapping. Retrieved from http://fourier.eng.hmc.edu/e161/lectures/svm/node8.html

[11] Hyperparameter optimization. (2018, April 11). Retrieved from https://en.wikipedia.org/wiki/Hyperparameter_optimization

[12] MathWorks Introducing Machine Learning [PDF]. (n.d.).

[13] Cortez, P., Morais, A. (2007). A Data Mining Approach to Predict Forest Fires using Meteorological Data.

[14] James, G. (2015). An introduction to statistical learning: With applications in R. (pp. 305-306) New York: Springer.

[15] Decision tree learning. (2018, May 16). Retrieved from https://en.wikipedia.org/wiki/Decision_tree_learning#Metrics

[16] Decision Tree Regression. (n.d.). Retrieved from http://www.saedsayad.com/decision_tree_reg.htm

[17] James, G. (2015). An introduction to statistical learning: With applications in R.(pp. 337-372) New York: Springer.

[18] Hastie, T., Tibshirani, R., Friedman, J. H. (2016). The elements of statistical learning data mining, inference, and prediction. (pp. 417-458). New York, NY: Springer.

[19] Multiclass classification. (2018, May 16). Retrieved from`https://en.wikipedia.org/wiki/Multiclass_classification#One-vs.-rest`