Contents

Input Variable Info Table			
Variable Name	**Details**	**Type**	**Example**
No	row number	Categorical	1
year	year of data in this row	Categorical	2010
month	month of data in this row	Categorical	1
day	day of data in this row	Categorical	1
hour	hour of data in this row	Categorical	0
season	season of data in this row	Categorical	4
DEWP	Dew Point (Celsius Degree)	Continuous	-6
TEMP	Temperature (Celsius Degree)	Continuous	59.48
HUMI	Humidity (%)	Continuous	1026.1
PRES	Pressure (hPa)	Continuous	1
cbwd	Combined wind direction	Categorical	cv
lws	Cumulated wind speed (m/s)	Continuous	1
precipitation	hourly precipitation (mm)	Continuous	0
lprec	Cumulated precipitation (mm)	Continuous	0

Figure 1: Attribute Info Table

1. Motivation

Air Pollution PM2.5 refers to atmospheric particulate matter (PM) that have a diameter of less than 2.5 micrometers, which is about 3% the diameter of a human hair (1). Since they are so small and light, fine particles tend to stay longer in the air than heavier particles and this increases the chances of humans and animals inhaling them into the bodies and raising serious health problems. Particulate matter (including soot) is emitted during the combustion of solid and liquid fuels, such as for power generation, domestic heating and in vehicle engines. It is bound to draw people's attention to this air quality problem, especially in Northern China. Nowadays, the concerns of Pm2.5 also is raising in southern part of China. Finding an efficient and accurate model to predict pollution PM2.5 for a specific climate and environment area would be helpful guide local citizens.

2. Introduction

This report aims to find a most effect and accurate model predict hourly Pm2.5 in year 2015 of Guangzhou, China, which is the largest industrial and commercial city and the largest important and export port in South China. There are seven parts come in later of this report, which are this, data introduction, data preprocessing, prediction models (smoothing, random forest, gradient boosting), statistical conclusions, conclusions in the context, future work and appendix.

3. Data Outline

The raw dataset comes from UCI machine learning repository and the link of dataset attach at the appendix part. The raw data attribute information. See figure 1.

Row number, "No" , will be discarded since it is not helpful for prediction.

For the following analysis, its subset, pm 2.5 in year 2015 of Guangzhou City, will be used, so we don't need to concern about variable "year". To avoid influences of too many time variables, "year", "month" and "day" will also be combined as one single time variable, simply using as a timestamp for each record.

The variable "season" have 1 represnts Spring from March to May, 2 represnts Summer from June to August, 3 represnts fall from September to November, and 4 represents winter from December to February. However, these integers (1-4) will be treated as levels of factors in later analysis. The variable "cwbd" is combined wind direction. According to original data collector Chen's paper, the weather data had 16 wind directions. The study shows that the directions can be grouped into five broad categories: northwest (NW), which

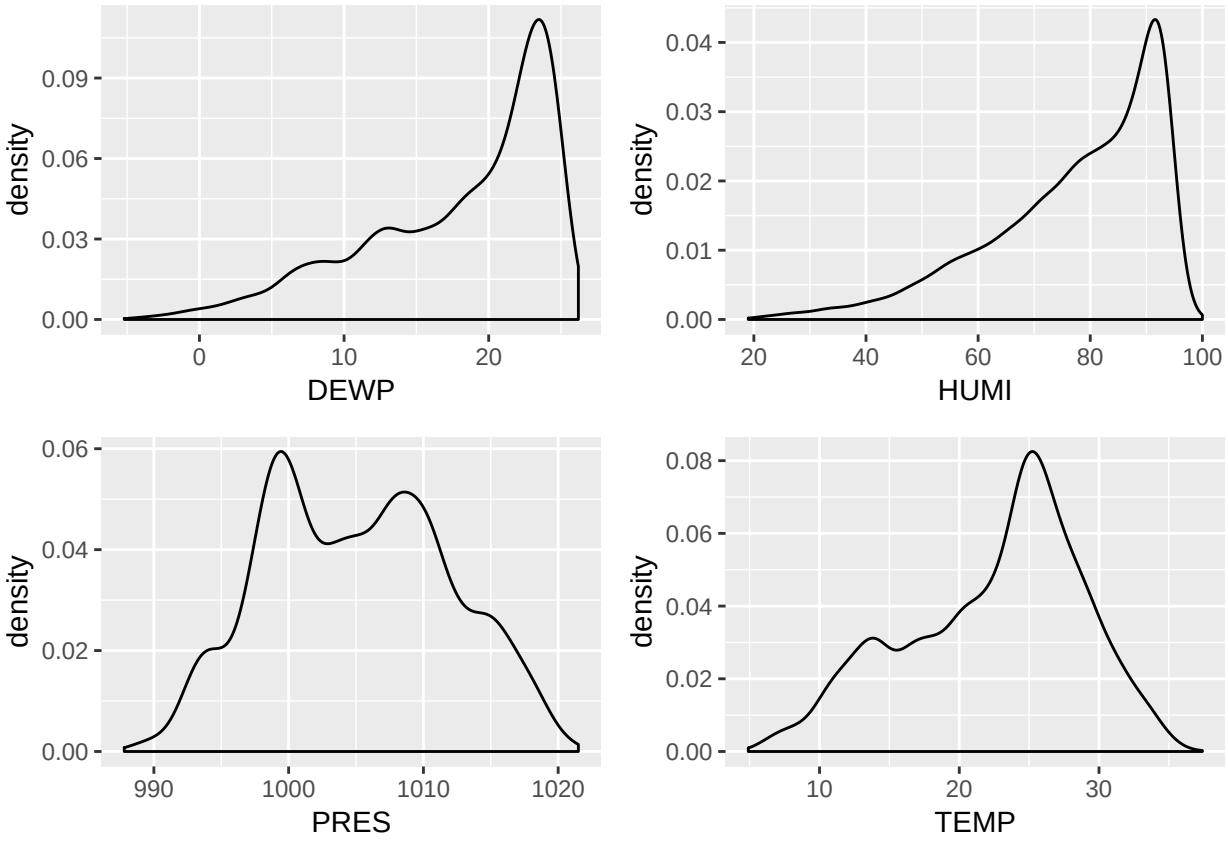
Count of Combined Wind Direction Table				
CV	NE	NW	SE	SW
196	2758	3227	1678	900

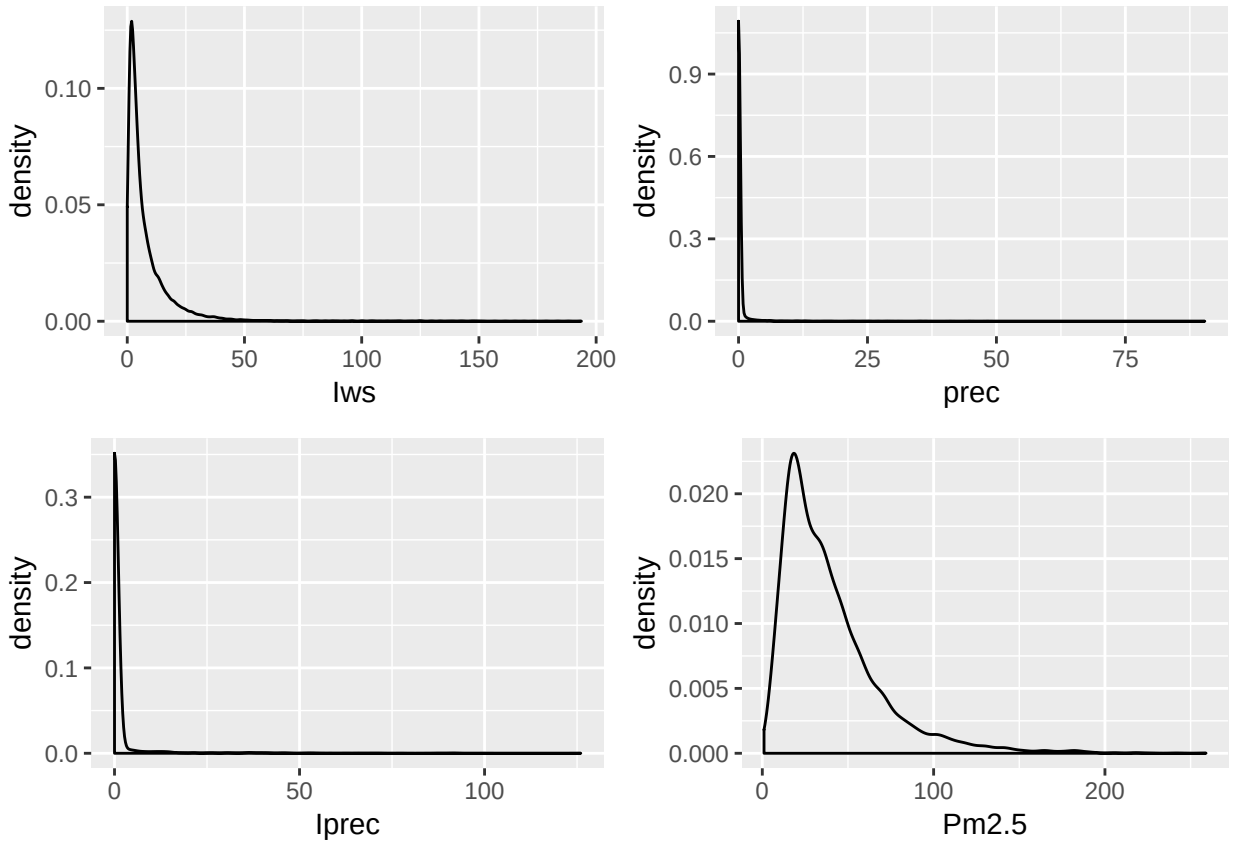
Figure 2: Attributes Table

includes W, WNW, NW, NNW and N; northeast (NE), for NNE, NE and ENE; southeast (SE), covering E, ESE, SE, SSE and S; southwest (SW), having SSW, SW and WSW; and calm and variable (CV). "cwbd" has the following count table:

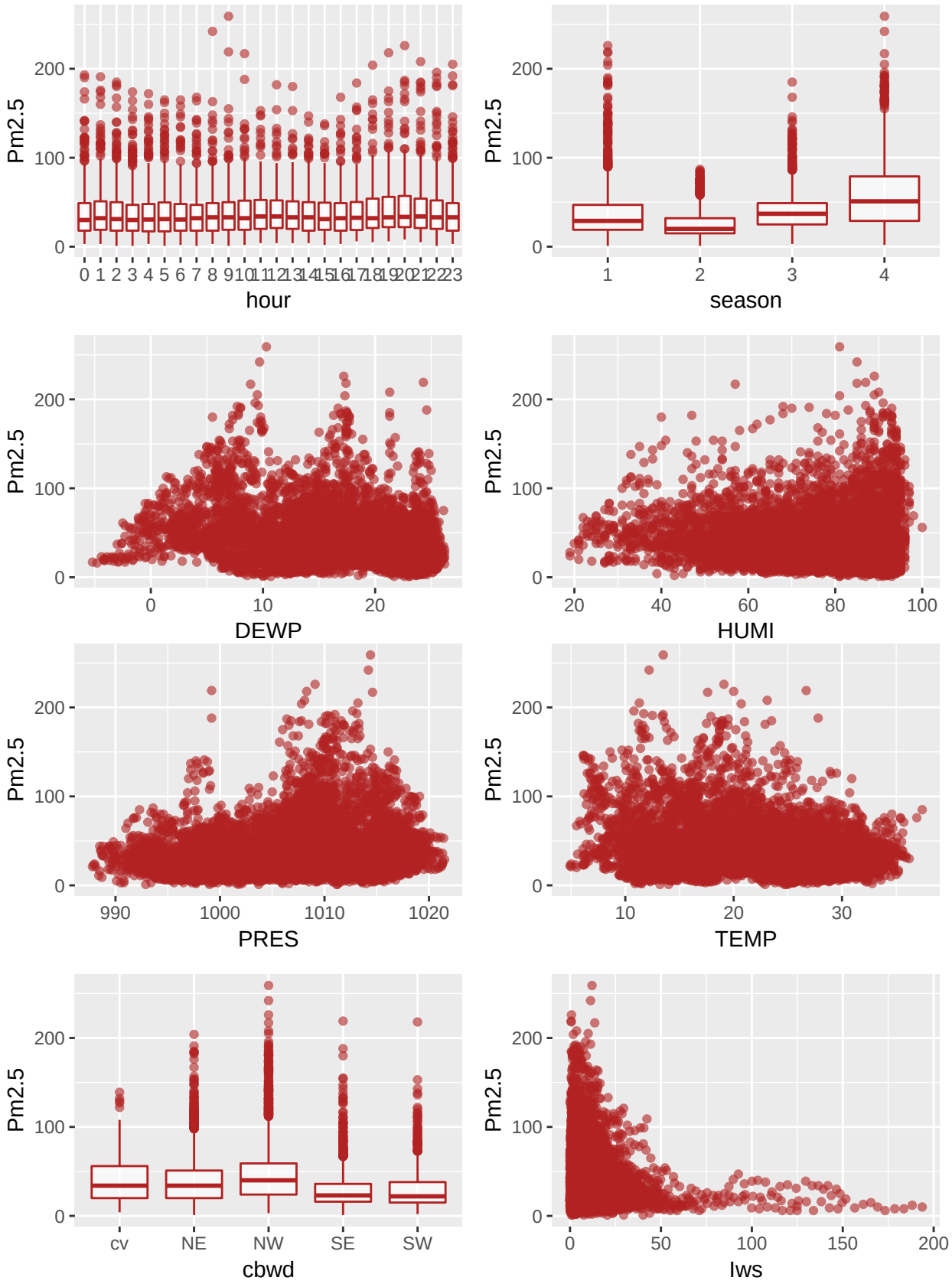
Iws" is a measure of the accumulated wind speed (in summed m/s) in the identified sustained direction (presumably by cbwd). According to the authors, this measurement takes into account wind velocity in a particular direction. It reflects the fact that it is sustained wind from a fixed direction that reduces or increases the pollution. The variable "precipitation" records the hourly precipitation while "Iprec" records the cumulated precipitation until the combined wind direction changes.

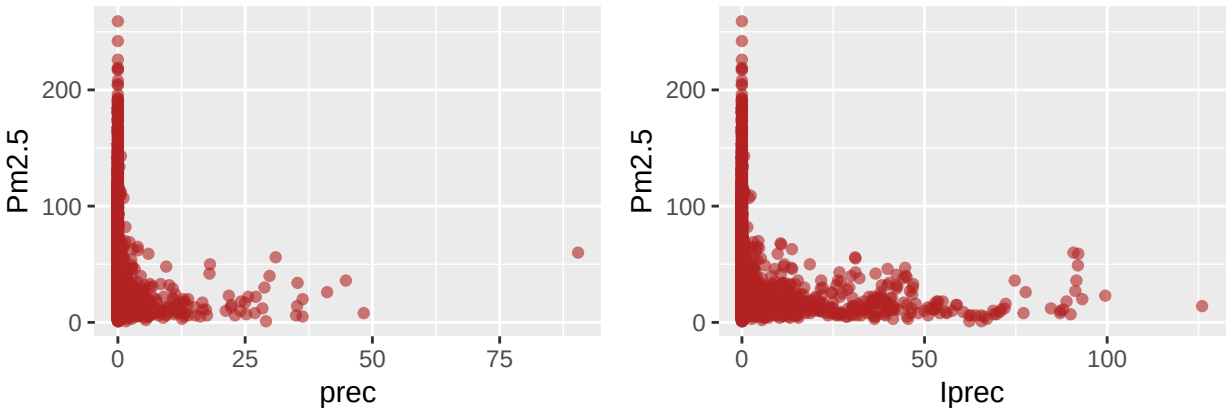
See how these variables distribute via density plots follows:



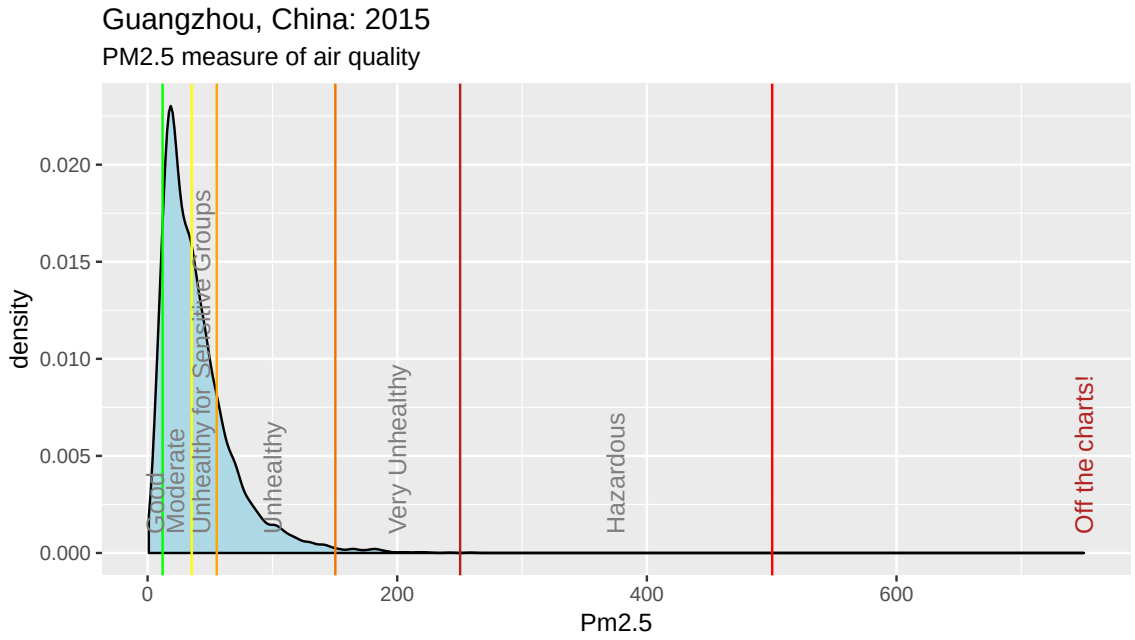


Also, to get a general idea of how covariates related to the response variables Pm2.5 values by some scatterplots.


Pm2.5
hour
season
DEWP
HUMI
PRES
TEMP
cbwd
Iws
cv
NE
NW
SE
SW



There are only 1 missing value for each covariates and 244 missing values for the response variable, we are confident to remove all missing values, then we get 8515 recrods to next step analysis.



It looks like most time the air quaility is moderate to unhealthy. Check current air quality in Guangzhou (Apr,13, 2019) according to `http://aqicn.org/city/guangzhou/`.

Compare to same time (apr,13,2019) air quaility in Toronto and Kitchener.

4 Data Preprocessing

4.1 Input Variables Modification

Then what left are numberical variables “DEWP”, “TEMP”, “HUMI”, “PRES”, “Iws”, “precipitation”, “Iprec” and categorical variables “hour”, “season” and “cwbd”. For the categorical variables, the variable “hour”, it is recorded by number “0-24” to describle 24 hours in a day and here is to treat as category variable.

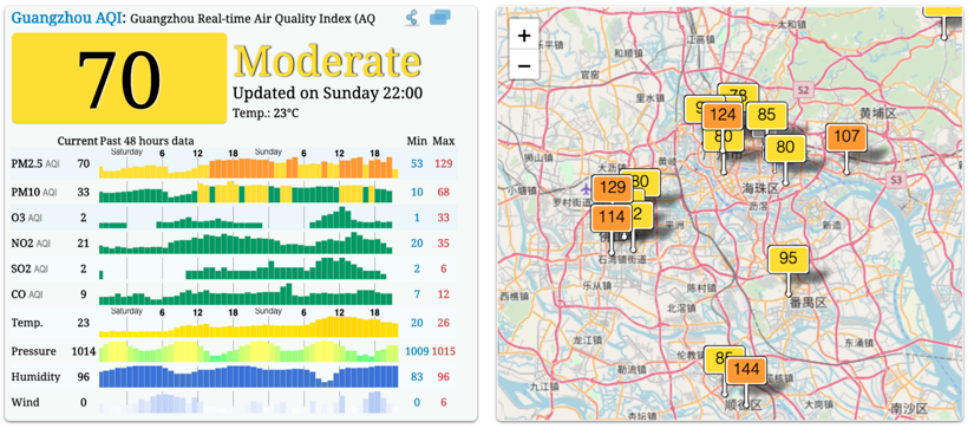


Figure 3: Guangzhou Pm2.5 - Apr 13, 2019

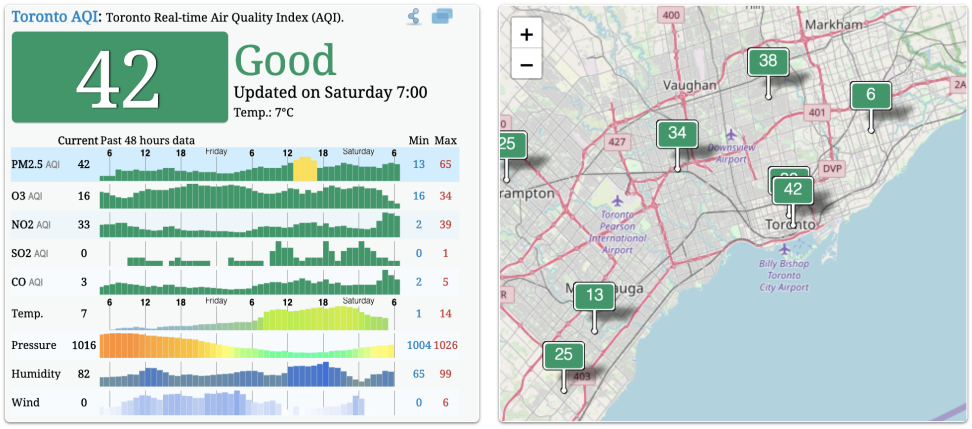


Figure 4: Toronto Pm2.5 - Apr 13, 2019

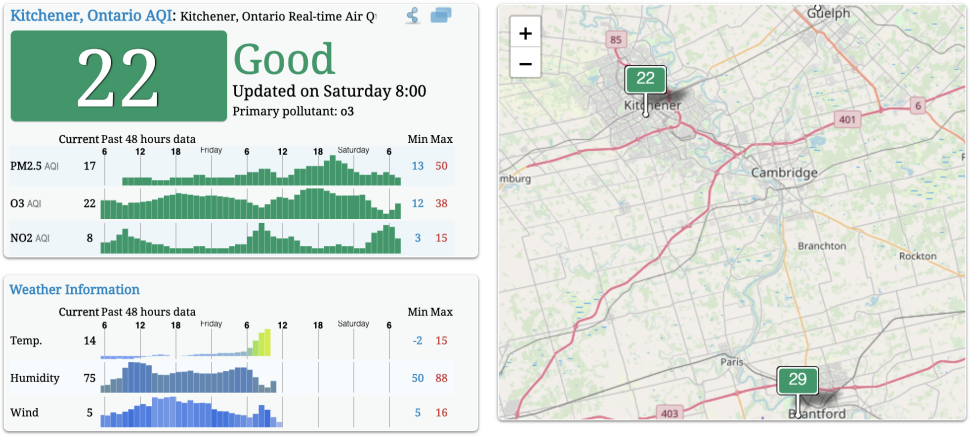
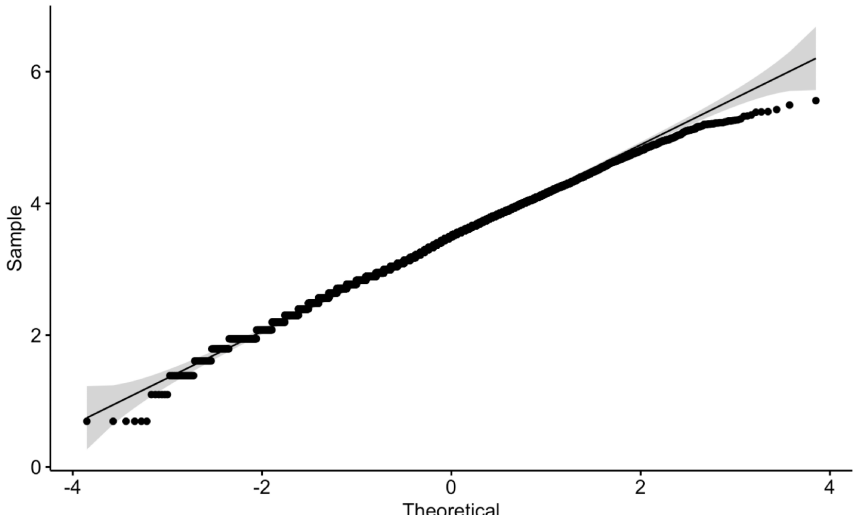


Figure 5: Kitchner Pm2.5 - Apr 13, 2019

4.2 Response Variable Transformation

We also notice our response variable is skewed distribution from previous density plots. We may also want to do log transformation before next step analysis. After transformation:



We may want to log transformation for convenience for later analysis.

4.3 Missing Values

We have 8760 records in total, check missing values by summary our dataset. We can directly omit rows that missing response values, since there are only 1 missing value for each covariate and 244 missing values for response variables, we are confident to remove all missing values, then we get 8515 recrod to next step analysis.

5. Smoothing

5.1 GAM without interaction effect

The first type model is generalized additive model with integrated smoothness estimation without any interaction. It starts with full model without interaction effects using gam () function inside of mgcv library. Using numerical term for smoothing function and categorical variables for simply additive.
Note: GAM refers prediction error based on 5-folod cross validation. The result is mse = 0.3247.

According to summary of no-interaction model, all seven smoothing term ae significant enough in our model. We use all numerical variables in one-interaction effect model.

5.2 GAMs with one interaction effect (numerical variables)

Here are seven numerical variables “DEWP”, “HUMI”, “PRES”, “TEMP”, “Iws”, “prec”, and “Iprec”. Choose two of these variables as one type interaction added to the baseline smoothing model. There are 21 one interaction effect model. Prediction error and mse table for each model, from model 2 to model 22, show as follow,

Model Name	CV-mse-GAM	Interaction	
m (baseline)	0.3247	-	-
m2	0.3205	DEWP	HUIMI
m3	0.3201	DEWP	PRES
m4	0.3178	DEWP	TEMP
m5	0.3244	DEWP	Iws
m6	0.2989	DEWP	prec
m7	0.2924	DEWP	Iprec
m8	0.3192	HUMI	PRES
m9	0.3228	HUMI	TEMP
m10	0.3233	HUMI	Iws
m11	0.322	HUMI	prec
m12	0.3202	HUMI	Iprec
m13	0.3173	PRES	TEMP
m14	0.3231	PRES	Iws
m15	0.2981	PRES	prec
m16	0.3057	PRES	Iprec
m17	0.3239	TEMP	Iws
m18	0.2975	TEMP	prec
m19	0.2937	TEMP	Iprec
m20	0.335	Iws	prec
m21	0.3256	Iws	Iprec
m22	0.2975	prec	Iprec

From the table, we notice that best one interaction model is model 7, which is interaction of "DEWP" and "Iprec" colored in loght green colored and error is 0.2924. The second good model is model 19 with interaction of "TEMP" and "Iprec", and the error is 0.2937. So, our best model for one numerical interaction model is model 7.

5.3 GAMs with two more interactions (numerical variables)

Here, we add another interaction to our beat one interaction model and no longer consider relative less important variables but only "DEWP", "Iprec" and "TEMP". Then we add the rest pair combinations to our best one-interaction effect model. Then we get our best two-interaction effect model with interaction of "DEWP, Iprec" and "DEWP,Temp", with mse equals to 0.2879. Also try three interaction effects DEWP, Iprec" and "DEWP, TEMP", and "TEMP, Iprec" and put best no-interaction/one-interaction/two-interactions/three interactions model together.

Model	CV-mse-GAM	Interaction 1	Interaction 2	Interaction 3
m (baseline)	0.3247	-	-	-
m7	0.2924	DEWP,Iprec	-	-
m26	0.2879	DEWP,Iprec	TEMP, Iprec	-
m28	0.2876	DEWP,Iprec	Iprec, TEMP	DEWP, TEMP

5.4 GAM with one interaction effect (number variables and factor variables)

According to later random forest's indication, the factor variable "cbwd" would be the most important variable for prediction purpose. We take this variable to consider interaction of numerical vaiable and factor

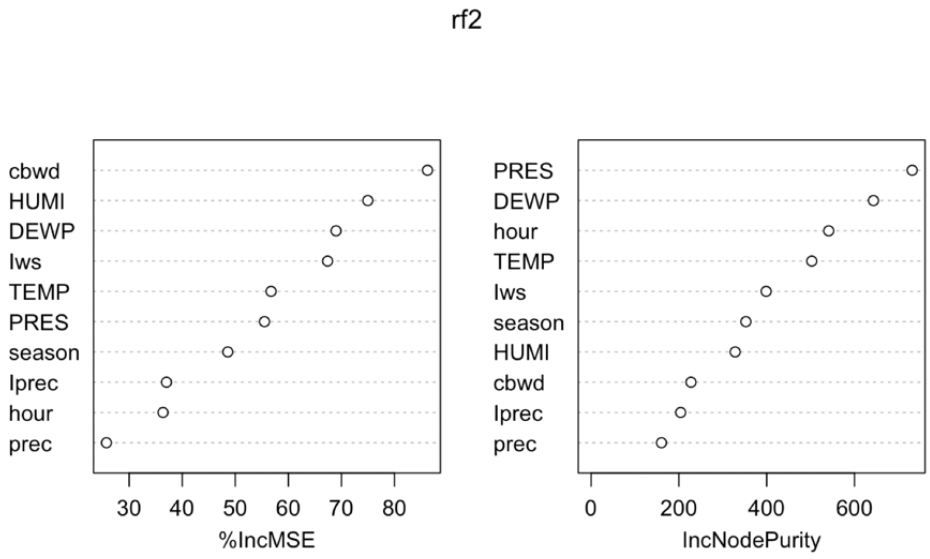
variable. However, the result of the simplest model by only adding to one numerical variable also take up twice longer than regular numerical smoothing models above with a little bit better accuracy then the baseline model. The details have been attached at the appendix part and more exploration would be done in the future work.

6. Random Forest

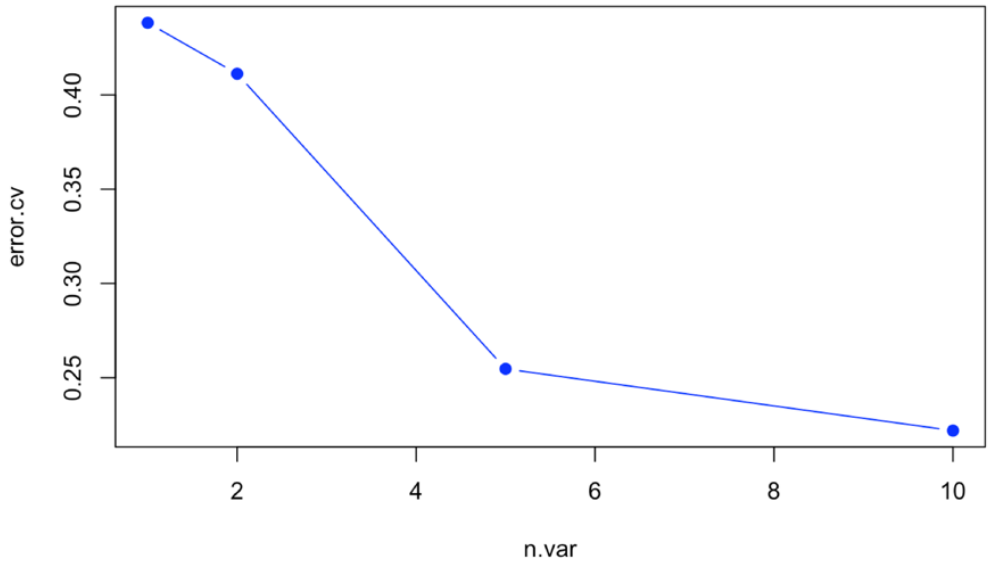
Random Forest is a popular ensemble technique is used to improve the predictive performance of Decision Trees by reducing the variance in the Trees by averaging them. In Random Forests the idea is to decorrelate the several trees which are generated on the different bootstrapped samples from training Data. Every time of constructing a tree, we only consider a Random subset of predictors (m) each time we do a split on training examples. So, by doing this trick of throwing away Predictors, we have de-correlated the Trees and the resulting average seems a little better.

First, I create a baseline for comparison by doing grid search for each parameter, the number of variables randomly sampled as candidates at each split, and number of trees to grow, ntree. There are two types importance of variables, type 1 is based om mean decrease in accuracy and type 2 is based on mean decrease in node impurity. The node impurity is a measure of the homogeneity of the labels at the node. For regression tree, it is actually the mean square error.

After grid searching, Mtry = 33.316625 is the optimal number of variables randomly sampled for each split, which is actually close to square root of number of predictors, and 500 trees are used here. Print out importance of each variable to decide which variables should be used for a better random forest. Two types of importance of variables as follows:



We notice importance ranks are not quite same in two importance metrics. For prediction purpose, we need to use “cbwd”, “HUMI”, “DEWP”,”Iws” and “TEMP” and “PRES”. While, if we want to consider to control more on RSS, we would take “PRES”, “DEWP”, “hour”, “TEMP”, “Iws” and “season” as most important six variables. Overall, “DEWP”, “PRES” and “Iws” are important variables for both measure metrices. To see how many variables should be used, we are using Random Forest Cross-Validation for feature selection to determine final model. The relationship between number of variables used and cross validation error shows as below:

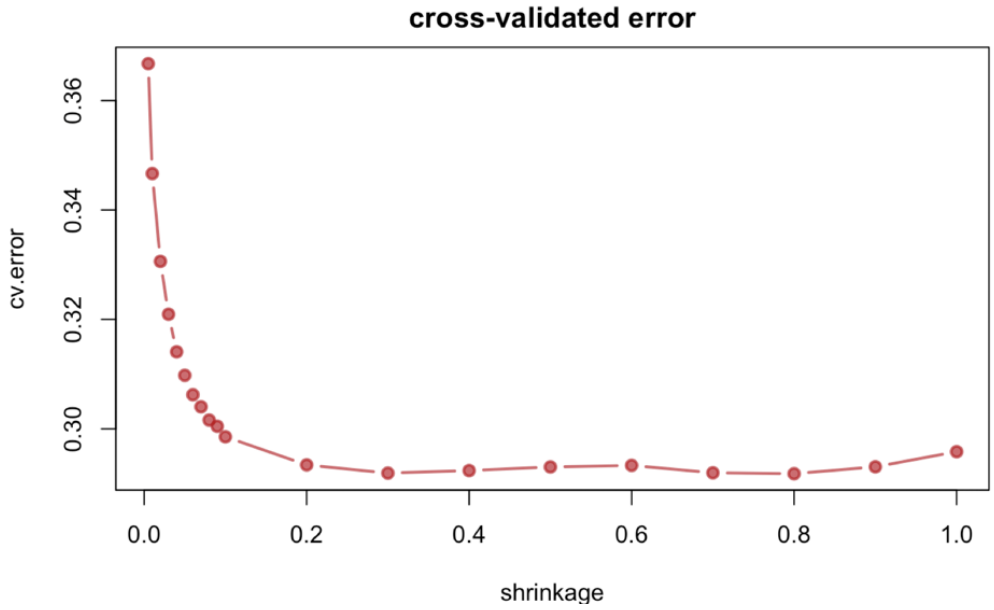


n.var	10	5	2	1
cv.error	0.221943	0.2546964	0.4112192	0.4382604

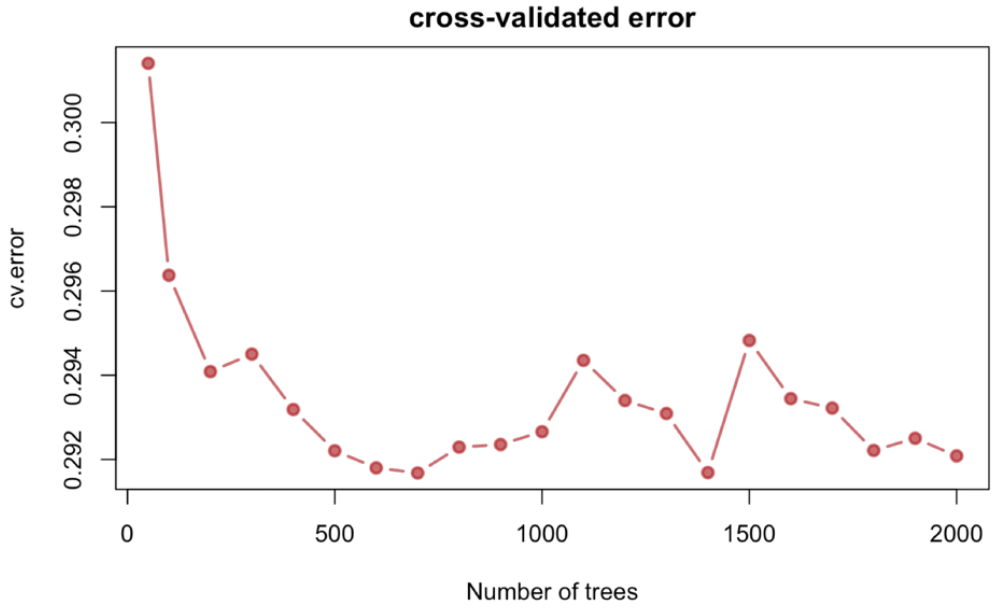
It indicates that we should all 10 predicators to build our random forest to get minimal cross validation error, so we do not need to decide which importance rank to use. So, the 5-fold cv error for random forest would be 0.221943.

7. Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. We need to choose best shrinkage and number of trees to split. The shrinkage parameter is applied to each tree in the expansion, also known as the learning rate or step-size reduction. Setting other parameters unchanged, decide which best shrinkage parameter to use, we do 5-fold cross validation on bunch of shrinkage parameter list $\{0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1\}$.

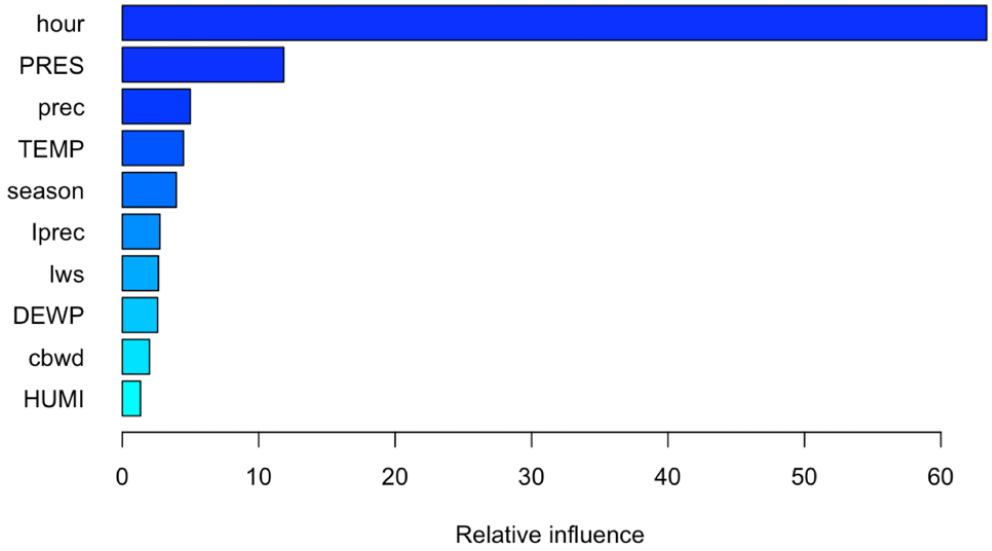


Our best shrinkage rate is 0.8. Then use the best shrinkage parameter to choose best number of trees from potential list. The relationship between numbers of trees and cross validation error plot shows as below.



We find the smallest cross validation error is corresponding to the 700 trees. Then use the best shrinkage parameter and tree numbers to run our best gradient boosting model. The results return a 5-fold cross validation error for each tree. We get the minimum cross validation error is 0.2974171. Which is larger than smoothing method and random forest method used above. Also investigate into relative influences of each variables, the details are displayed as follows. We notice that "hour" becomes the most significant variable in this method, compared to it is the one of the least important variables in random forest. Meanwhile, the important of variable graph still indicates that top importance of variables "PRES" and" TEMP".

	var <fctr>	rel.inf <dbl>
hour	hour	63.354300
PRES	PRES	11.837855
prec	prec	4.989437
TEMP	TEMP	4.495917
season	season	3.969611
Iprec	Iprec	2.760453
Iws	Iws	2.652089
DEWP	DEWP	2.598999
cbwd	cbwd	2.002111
HUMI	HUMI	1.339226



In gradient boosting, "hour", "PRES", "prec", "TEMP","season " and "Iprec" would be top six important variables. This ranks is different from random forest's measure, but also have common importance of "PRES" and "TEMP".

8. Comparison and Statistical Conclusions

This part will discuss best model among three based on prediction error, running time and interpretation purposes. The following table attached for later context reference.

Model Name		5-fold cv error	Computation Time
GAM	No interaction : m	0.3247	17.986
	One Ineraction : m7	0.2924	28.191
	Two Interactions : m24	0.2879	94.278
	Three Interactions :m 25	0.2876	194.407
Random Forest	-	0.2219	302.544
Gradient Boosting	-	0.2974	10.192

8.1 Five-Fold Cross Validation Prediction Error

According to table above, if we only want to focus on prediction accuracy, we would use random forest model for sure since it gives us than best gam model (three interactions ones) and best gradient model. While gradient boosting is similar to GAM models depends how much computation resource you have and how much complexity you want when running GAM.

8.2 Computation Time

According to the comparison table, we notice that random forest cost much more time than gradient boosting as well as GAMs. Running time of random forest is almost 30 times slower than gradient boost and over 20 times than baseline GAM model. Depends how much time constraint, best GAM model running time can also range from 17.986 seconds to 194.407 seconds. If we want a balanced tradeoff between complexity and accuracy, we may only consider up to one interaction for GAM model.

8.3 Difficulty of Interpretation

Generalized additive model with smoothing method would be the easiest way to interpret since it is similar to regression model. Random Forest and Gradient Boosting are two popular ensemble methods and predict by combining the outputs from individual trees (we use tree-based GBM here). Random forest trains each tree independently, using a random sample of the data. This randomness helps to make the model more robust than a single decision tree, and less likely to overfit on the training data. In our case, random forest performs best for predictive purposes, cbwd seems to be the most important according to random forest's importance of variables. The cross-validation suggests all ten variates remain in the model. Even with these tools, much has been lost in interpretability compared to a single tree.

8.4 Importance Variables Ranking

In this PM2.5 dataset, importance of variables varies among three models, It may indicates that all these variables are important and more vital variables are not given or missing in this data, e.g, Pm10, ozone etc,. However, all three methods claim to remain all variables in models as well as some common top important variables do have intersection such as "DEWP", "PRES" and "TEMP".

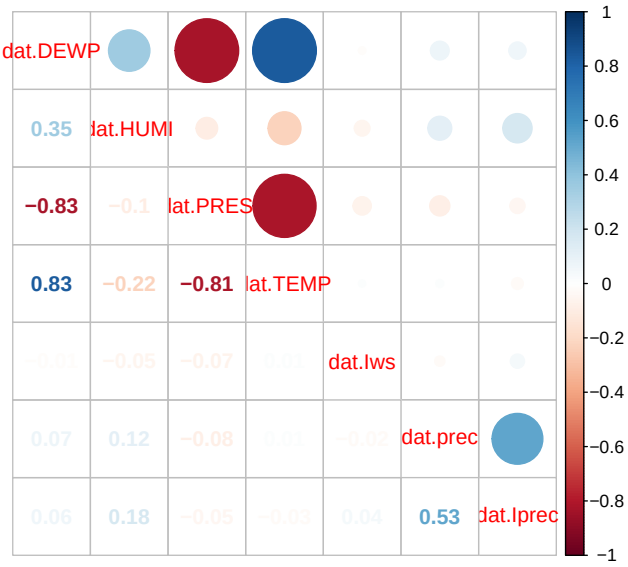
8.5 Ease of Use: Terms of Tuning Parameters

Random Forest would be the easiest one to use since even you do not choose which parameters to tuning or which interaction you want to add, you still fairly well result. For gradient boosting, there are lot potential parameters you can tune, in this report, only shrinkage and number of trees are considered. Theoretically, carefully tuning parameter can make boosting better than random forest, according to data science blog (3). For GAMs, we find it would be hard to handle when we have plenty of variables,

9. Conclusion Based on Context

PRES (pressure), TEMP (temperature), cbwd (combined wind direction), DEWP (dew point) and hour are important for predicting this dataset. These variables are actually associated a lot. First, dew point depends on temperature. Temperature is also related to pressure. Pm 2.5 also tends to be server during morning time and evening time, which may surprise some readers and afternoon tends to be the lowest Pm2.5 pollution during the day. This may because a lot traffic happens during morning peak and off work peak. Intuitively, we can notice all these environment factors are closely related to temperature. This is actually

to be noticed during data preprocessing time, all covariates are highly related to each other, the correlation plot for numerical variables are attached as follows:



10. Future Work

10.1 Investigation of Interaction GAMs

This report does not fully investigate all possible interaction especially of categorical-categorical interaction and numerical - categorical interaction. If time enough, more models can be implemented to investigate.

10.2 More Variables Needed

We only have seven numerical variables and three variables here and most of them are natural environment factors. However, the air pollution Pm2.5 tends to be caused by human being behavior like industrial spills, vehicle exhaust and other pollution causes. So other air pollution variables such Pm10 value, carbon dioxide (CO2), carbon monoxide (CO), nitrogen oxides (NO2), nitrogen monoxide (NO), or ozone content can be much more useful. Also, weekday can be considered if time permits and weekday can be directly calculated from datetime.

10.3 Potential Machine Learning Models

Other machine learning models such as XGBoost as suggested in guildline. Also, Encoder-Decoder model, Long-Short Term Memory as purposed in Bu, Li, and Cha's paper "A Deep Learning Approach for Forecasting Air Pollution in South Korea Using LSTM".

Appendix

Data

The data collector is Song Xi Chen, (csx '@' gsm.pku.edu.cn), Guanghua School of Management, Center for Statistical Science, Peking University. The datset is actually a combined dataset the Pm2.5 values are released by U.S. embessy, and two local monitor stations in Guangzhou. In this report use U.S. embessy's reslese as preiction targert.

Literature

1. "Public Health: Sources and Effects of PM2.5." LAQM Support, laqm.defra.gov.uk/public-health/pm25.html.
2. Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., ... Chen, S. X. (2015). Assessing Beijing's PM2.5 pollution: severity, weather impact, APEC and winter heating. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471(2182), [20150257]. https://doi.org/10.1098/rspa.2015.0257

3.Bui, T., Le, V., Cha, S. K., A Deep Learning Approach for Forecasting Air Pollution in South Korea Using LSTM., Department of Electrical and Computer Engineering, Seoul National University. https://arxiv.org/pdf/1804.07891.pdf

Model Details

All R codes for attempted models are attached below.

```
#df <- 11
library(mgcv)
#dat$Pm2.5 = log(dat$Pm2.5
m = mgcv::gam(Pm2.5~ hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec),data=dat,method="GACV.Cp")
m2 =mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,HUMI),data=dat,method="GACV.Cp")
m3 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,PRES),data=dat,method="GACV.Cp")
m4 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,TEMP),data=dat,method="GACV.Cp")
m5 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,Iws),data=dat,method="GACV.Cp")

m6 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,prec),data=dat,method="GACV.Cp")
m7 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,Iprec),data=dat,method="GACV.Cp")



###HUMI
m8 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(HUMI,PRES),data=dat,method="GACV.Cp")
m9 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(HUMI,TEMP),data=dat,method="GACV.Cp")
```

```
m10 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(HUMI,Iws),data=dat,method="GACV.Cp")
m11 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(HUMI,prec),data=dat,method="GACV.Cp")
m12 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(HUMI,Iprec),data=dat,method="GACV.Cp")

##PRES
m13 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,TEMP),data=dat,method="GACV.Cp")
m14 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,Iws),data=dat,method="GACV.Cp")
m15 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,prec),data=dat,method="GACV.Cp")
m16 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,Iprec),data=dat,method="GACV.Cp")

##TEMP
m17 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(TEMP,Iws),data=dat,method="GACV.Cp")
m18 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(TEMP,prec),data=dat,method="GACV.Cp")
m19 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(TEMP,Iprec),data=dat,method="GACV.Cp")

#Iws
m20 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(Iws,prec),data=dat,method="GACV.Cp")
m21 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(Iws,Iprec),data=dat,method="GACV.Cp")

#prec
m22 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+
s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(prec,Iprec),data=dat,method="GACV.Cp")

library(gamclass)
gam_cv = CVgam(as.formula(m), data = dat, nfold = 5)
gam_cv2 = CVgam(as.formula(m2), data = dat, nfold = 5)
gam_cv3 = CVgam(as.formula(m3), data = dat, nfold = 5)
gam_cv4 = CVgam(as.formula(m4), data = dat, nfold = 5)
gam_cv5 = CVgam(as.formula(m5), data = dat, nfold = 5)
gam_cv6 = CVgam(as.formula(m6), data = dat, nfold = 5)
gam_cv7 = CVgam(as.formula(m7), data = dat, nfold = 5)
gam_cv8 = CVgam(as.formula(m8), data = dat, nfold = 5)
gam_cv9 = CVgam(as.formula(m9), data = dat, nfold = 5)
gam_cv10 =CVgam(as.formula(m10), data = dat, nfold = 5)
gam_cv11= CVgam(as.formula(m11), data = dat, nfold = 5)
gam_cv12= CVgam(as.formula(m12), data = dat, nfold = 5)
gam_cv13= CVgam(as.formula(m13), data = dat, nfold = 5)
gam_cv14= CVgam(as.formula(m14), data = dat, nfold = 5)
gam_cv15= CVgam(as.formula(m15), data = dat, nfold = 5)
gam_cv16= CVgam(as.formula(m16), data = dat, nfold = 5)
gam_cv17= CVgam(as.formula(m17), data = dat, nfold = 5)
```

```
gam_cv18= CVgam(as.formula(m18), data = dat, nfold = 5)
gam_cv19= CVgam(as.formula(m19), data = dat, nfold = 5)
gam_cv20= CVgam(as.formula(m20), data = dat, nfold = 5)
gam_cv21= CVgam(as.formula(m21), data = dat, nfold = 5)
gam_cv22= CVgam(as.formula(m22), data = dat, nfold = 5)


m23 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)
+s(HUMI)+s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,TEMP)+
ti(DEWP,PRES)+ti(DEWP,TEMP),data=dat,method="GACV.Cp")
gam_cv23= CVgam(as.formula(m23), data = dat, nfold = 5)

ptm <- proc.time()
m = mgcv::gam(Pm2.5~ hour + season+cbwd+s(DEWP)
+s(HUMI)+s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec),data=dat,method="GACV.Cp")
gam_cv = CVgam(as.formula(m), data = dat, nfold = 5)
proc.time() - ptm



ptm <- proc.time()
m7 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)
+s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP,Iprec),data=dat,method="GACV.Cp")
gam_cv7 = CVgam(as.formula(m7), data = dat, nfold = 5)
proc.time() - ptm


ptm <- proc.time()
m23 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+s(PRES)
+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,TEMP)+ti(DEWP,TEMP),data=dat,method="GACV.Cp")
gam_cv23= CVgam(as.formula(m23), data = dat, nfold = 5)
proc.time() - ptm

ptm <- proc.time()
m24 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+s(PRES)
+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,TEMP)+ti(DEWP,PRES),data=dat,method="GACV.Cp")
gam_cv24= CVgam(as.formula(m24), data = dat, nfold = 5)
proc.time() - ptm

ptm <- proc.time()
m25 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+s(PRES)
+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(PRES,TEMP)+ti(DEWP,TEMP)
+ti(DEWP,PRES),data=dat,method="GACV.Cp")
gam_cv25= CVgam(as.formula(m25), data = dat, nfold = 5)
proc.time() - ptm


ptm <- proc.time()
m26 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)
+s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEWP, Iprec)+ti(DEWP,TEMP),data=dat,method="GACV.Cp")
gam_cv26= CVgam(as.formula(m26), data = dat, nfold = 5)
proc.time() - ptm
```

```
ptm <- proc.time()
m27 = mgcv::gam(Pm2.5~hour + season+cbwd+s(DEWP)+s(HUMI)+s(PRES)+s(TEMP)+s(Iws)+s(prec)+s(Iprec)+ti(DEW
gam_cv27= CVgam(as.formula(m27), data = dat, nfold = 5)
proc.time() - ptm


ptm <- proc.time()
m30  = mgcv::gam(Pm2.5~ hour + season+cbwd+s(DEWP)+s(HUMI)+s(PRES)+s(TEMP, by = cbwd)
+s(Iws)+s(prec)+s(Iprec),data=dat,method="GACV.Cp")
gam_cv30 = CVgam(as.formula(m30), data = dat, nfold = 5)
proc.time() - ptm

ptm <- proc.time()
m31  = mgcv::gam(Pm2.5~ hour + season+cbwd+s(DEWP)+s(HUMI,by = cbwd)+s(PRES)
+s(TEMP, by = cbwd)+s(Iws)+s(prec)+s(Iprec),data=dat,method="GACV.Cp")
gam_cv31 = CVgam(as.formula(m31), data = dat, nfold = 5)
proc.time() - ptm


ptm <- proc.time()
m26 = mgcv::gam(Pm2.5~ hour + season+cbwd+s(DEWP)+s(HUMI, by =cbwd)+s(PRES)+s(TEMP, by = cbwd)+s(Iws)+s
gam_cv26 = CVgam(as.formula(m26), data = dat, nfold = 5)
proc.time() - ptm


ptm <- proc.time()
library(rgl)
library(ggplot2)
library(randomForest)
set.seed(101)
rf <- randomForest(dat$Pm2.5 ~.,
data = dat,mtry = 3, ntree=500,
importance = TRUE)
importance(rf, type=2)
importance(rf, type=1)
varImpPlot(rf)
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"
set.seed(101)
mtry <- sqrt(ncol(dat))
tunegrid <- expand.grid(.mtry=mtry)
rf_default <- train(Pm2.5~., data=dat, method="rf",  tuneGrid=tunegrid, trControl=control)
print(rf_default)
library(caret)
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=c(1:15))
rf_gridsearch <- train(Pm2.5~., data=dat, method="rf", metric=metric, tuneGrid=tunegrid, trControl=cont
print(rf_gridsearch)
plot(rf_gridsearch)
proc.time() - ptm


library(rgl)
library(ggplot2)
library(randomForest)

set.seed(101)
```

```
ptm <- proc.time()
rf <- randomForest(dat$Pm2.5 ~.,
data = dat,mtry = 3.316625, ntree= 500,
importance = TR
UE)


proc.time() - ptm


importance(rf, type=2)
importance(rf, type=1)
varImpPlot(rf)


plot(rf)

set.seed(101)
ptm <- proc.time()
rf <- randomForest(dat$Pm2.5 ~.,
data = dat,mtry = 3.316625, ntree= 500,
importance = TRUE)

get.explanatory_varnames <- function(formula){
f <- as.formula(formula)
terms <- terms(f)
attr(terms, "term.labels")
}#

get.explanatory_varnames(rf2)
trainy <- dat$Pm2.5
trainx <- dat[, get.explanatory_varnames(rf2)]
# Five fold cross-validataion
rfcv2 <- rfcv(trainx = trainx, trainy = trainy, cv.fold = 5,log=TRUE)
# We can plot the results
with(rfcv2, plot(n.var, error.cv, pch = 19, type="b", col="blue"))


proc.time() - ptm

ptm <- proc.time()
library(gbm)


M <- 500
k <- 5

# Shrinkage parameter values
alphas <- c(0.005, 0.01, 0.02, 0.03, 0.04,
0.05, 0.06, 0.07, 0.08, 0.09,
0.10, 0.20, 0.30, 0.40, 0.50,
0.60, 0.70, 0.80, 0.90, 1)
n_alphas <- length(alphas)
cverror <- numeric(length = n_alphas)
```

```
Mvals <- numeric(length = n_alphas)
fit <- list(length = n_alphas)

for (i in 1:n_alphas) {
fit[[i]] <- boost.shrink <- gbm(dat$Pm2.5~.,
data = dat,
distribution = "gaussian",
shrinkage = alphas[i],
n.trees = M,
bag.fraction = 1,
cv.folds = k
)
cverror[i] <- min(fit[[i]]$cv.error)
Mvals[i] <- which.min(fit[[i]]$cv.error)
}
plot(alphas, cverror, type = "b",
col = adjustcolor("firebrick", 0.7), pch=19, lwd=2,
main = "cross-validated error", xlab = "shrinkage", ylab="cv.error")
proc.time() - ptm

i <- which.min(cverror)

alpha <- alphas[i]
alpha
par(las=1)
summary(fit[[i]])

set.seed(101)
ptm <- proc.time()

alpha <- 0.8
k <- 5

# Shrinkage parameter values
Ms <- c(50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,1100,1200,1300,1400,1500,1600,1700,1800,19
n_Ms <- length(Ms)
cverror <- numeric(length = n_Ms)
Mvals <- numeric(length = n_Ms)
fit <- list(length = n_Ms)
for (i in 1:n_Ms) {
fit[[i]] <- boost.shrink <- gbm(Pm2.5~.,
data=dat,
distribution = "gaussian",
shrinkage = alpha,
n.trees = Ms[i],
bag.fraction = 1,
cv.folds = k
)
cverror[i] <- min(fit[[i]]$cv.error)
}
plot(Ms, cverror, type = "b",
col = adjustcolor("firebrick", 0.7), pch=19, lwd=2,
main = "cross-validated error", xlab = "Number of trees", ylab="cv.error")
```

```
proc.time() - ptm

#1100
j <- which.min(cverror)
j
Ms[j]
par(las=1)
summary(fit[[j]])
par(las=1)

library(gbm)
ptm <- proc.time()
best.boost.shrink <- gbm(Pm2.5~., data=dat,
shrinkage = 0.8, n.trees = 700,cv.folds = 5)
par(las=1)
summary(best.boost.shrink)
proc.time() - ptm

library(corrplot)
M = data.frame(dat$DEWP,dat$HUMI,dat$PRES, dat$TEMP, dat$Iws,dat$prec,dat$Iprec)
corrplot.mixed(cor(M))
```