Journal of Asian Scientific Research

journal homepage: http://aessweb.com/journal-detail.php?id=5003

COMPARISON OF THREE CLASSIFICATION ALGORITHMS FOR PREDICTING PM2.5 IN HONG KONG RURAL AREA

Yin Zhao

School of Mathematical Sciences, Universiti Sains Malaysia, Penang, Malaysia

Yahya Abu Hasan

School of Mathematical Sciences, Universiti Sains Malaysia, Penang, Malaysia

ABSTRACT

Data mining is an approach to discover knowledge from large data. Pollutant forecasting is an important problem in the environmental sciences. This paper tries to use data mining methods to forecast fine particles (PM2.5) concentration level in a new town of Hong Kong rural area. There are several classification algorithms available in data mining, such as Artificial Neural Network (ANN), Boosting, k-Nearest Neighbours (k-NN), and so forth. All of them are popular machine learning algorithms in various data mining areas, which including environmental data mining, educational data mining, financial data mining, etc. This paper builds PM2.5 concentration level predictive models based on ANN, Boosting (i.e. AdaBoost.M1), k-NN by using R packages. The data set includes 2009 to 2011 period meteorological data and PM2.5 data. The PM2.5 concentration is divided into 2 levels: low and high. The critical point is 25μg/m^3 (24 hours mean. The parameters of both models are selected by multiple cross validation. According to 100 replications of 10-fold cross validation, the testing accuracy of AdaBoost.M1 is around 0.846~0.868, which is the best result among three algorithms in this paper.

Keywords: Artificial neural network (ANN), AdaBoost.M1, *k*-Nearest Neighbours (*k*-NN), PM2.5 prediction, Data mining, Machine learning

INTRODUCTION

Air pollution is a major problem in the world, especially in some developing countries and business cities. One of the most important pollutants is particulate matter. Particulate matter (PM) can be defined as a mixture of fine particles and droplets in the air and this can be characterized by their sizes. PM2.5 refers to particulate matter whose size is 2.5 micrometers or smaller. Due to its effect

on health, it is crucial to prevent the pollution getting worse in a long run. According to WHO's report, the mortality in cities with high levels of pollution exceeds that observed in relatively cleaner cities by 15–20% (WHO, 2011).

Data mining provides many methods for building predictive models in various areas, which including regression, classification, cluster analysis, association analysis, and so on. Data mining projects are often structured around the specific needs of an industry sector or even tailored and built for a single organization. A successful data mining project starts from a well defined question or need. PM2.5 prediction models can be divided into two groups: one is to build a regression or related model in order to capture the exact numeric value in future (i.e. next day or hours); another one is to use classification method building a model for predicting the level of concentration. We will use the latter one in this paper, that is, classification model. Classification result will tell people what is the PM2.5 concentration level in the next day instead of the concrete value, this should be helpful for us to understand the pollution situation rather than the exact number.

Forecasting of air quality is much needed in a short term so that necessary preventive action can be taken during episodes of air pollution. Considering that our target data set is from a rural area of Hong Kong, so we try to find a strict standard of PM2.5 as the split criterion. WHO's Air Quality Guideline (AQG) says the mean of PM2.5 concentration in 24-hour level should be less than 25μg/m^3 (WHO, 2005), although Hong Kong's proposed Air Quality Objectives (AQOs) is 75μg/m^3 right now (AQO, 2012). As a result, we use WHO's critical value as our standard point. The number of particulate at a particular time is dependent on many environmental factors, especially the meteorological data, say, air pressure, rainfall, humidity, air temperature, wind speed, and so forth.

In this paper, we try to build models for predicting next day's PM2.5 mean concentration level by using three popular machine learning algorithms, which are, Artificial Neural Network (ANN), *k*-Nearest Neighbours (*k*-NN), and Boosting (i.e. AdaBoosting.M1). ANN is inspired by attempts to simulate biological neural system. It may contain many intermediary layers between its input and output layers and may use types of activation functions (Rojas, 1996). *k*-NN is one of the simplest techniques in classification problems, that is, it only specifies the training data and then predicts the class of a new value by looking for the *k* observations in the training set that are closest to the new value. Boosting based on decision tree method which is a basic classifier in many tree-based algorithms (e.g. Random Forest, C5.0, etc.). The simple decision tree is a weak classifier; hence a smart technique is to combine them to be a stronger one. For instance, assuming the error rate of a weak classifier is 0.49, but if we put 1001 same weak learners to be one learner and make them vote by weighted (simple majority rule), the error rate will be $1-\binom{1001}{500}(0.49)^{500}(0.51)^{501}=0.26$ (i.e. binomial distribution). Thus, Boosting can be considered as the learner of a mass of decision trees being combined and weighted voting.

An important issue in data mining is not only to analyse data but also to see them, so we choose R (Ihaka and Gentleman, 1996) as our analysis tool in this paper. R is an open source programming language and software environment for statistical computing and graphics. It is widely used for data analysis and statistical computing projects. In this paper, we will use some R packages as our analysis tools, namely "nnet" package (Ripley, 2013), "kknn" package (Schliep and Hechenbichler, 2013), and "RWeka" package (Hornik *et al.*, 2013). Moreover, we also use some packages for plotting figures, such as "reshape2" package and "ggplot2" package (Wickham, 2013a; 2013b)

The remainder of the paper is organized as follow: Section 2 gives brief reviews of these three algorithms. Section 3 and 4 will describe the data and the experiments. At last, the conclusion and discussion will be given in Section 5.

METHODOLOGY

Artificial Neural Network (ANN)

ANN is formed by a set of computing units (i.e. the neurons) linked to each other. Each neuron executes two consecutive calculations: a linear combination of its inputs, followed by a nonlinear computation of the result to obtain its output value that is then fed to other neurons in the network. Each of the neuron connections has an associated weight. Constructing an artificial neural network consists of establishing architecture for the network and then using an algorithm to find the weights of the connections between the neurons. The network may contain many intermediary layers between its input and output layers. Such intermediary layers are called hidden layers and the nodes embedded in these layers are called hidden nodes. The network may use types of activation functions, such as linear, sigmoid (logistic), and hyperbolic tangent functions, etc. In R "nnet" package, the sigmoid function is default for classification model. Its expression is shown below:

$$f(x)=\frac{e^x}{1+e^x}$$

The back-propagation (i.e. BP) algorithm is used in layered feed-forward ANN (Hagan *et al.*, 1996). The BP algorithm uses supervised learning, which means that we provide the algorithm with examples of the inputs and outputs we want the network to compute, and then the error is calculated.

Let $D = \{(x_i, y_i) \mid i=1, 2, \ldots, N$ be the set of training examples. The goal of the ANN learning algorithm is to determine a set of weights w that minimize the total sum of squared errors.

$$E(w)=\frac{1}{2}\cdot\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$$

where $\hat{y}_i$ is the output value by performing a weighted sum on its input.

And the weight update formula used by the gradient descent method:

$$w_j \leftarrow w_j - \lambda \frac{\partial E(w)}{\partial w_j}$$

where λ is the learning rate.

There are two phases in each iteration of the BP algorithm:

♦ The forward phase. During the forward phase, outputs of the neurons at level k are computed prior to computing the outputs at level k+1.

♦ The backward phase. During the backward phase, the weights at level k+1 are updated before the weights at level k are updated.

This BP algorithm allows us to use the error for neurons at layer k+1 to estimate the errors for neurons at layer k.

k-Nearest Neighbours (*k*-NN)

k-NN (Cover and Hart, 1967) is one of the simplest data mining algorithms and it belongs to the class of so-called lazy learners. *k*-NN does not actually obtain a model from training data but simply store the data sets. Its main work happens at prediction time. Given a new test case, its prediction is obtained by searching for similar cases in the training data that was stored. The k most similar training cases (i.e. neighbours) are used to obtain the prediction for the given test case. When we talk about neighbours we are implying that there is a distance or proximity measure that we can compute between samples based on the independent variables. There are many distance functions, but a rather frequent selection is the Minkowski distance function that is defined as

$$d(x,y) = (\sum_{k=1}^{n} |x_k - y_k|^r)^{1/r}$$

where n is the number of dimensions (i.e. attributes) and x_k and y_k are, respectively, the k^{th} attributes of x and y. There are two important particular cases:

♦ r=1. Manhattan distance. A common example is the Hamming distance, which is the number of bits that are different between two objects that have only binary attributes. It is the method of calculating the distance among nominal variables in *k*-NN algorithm.

♦ r=2. Euclidean distance. It is used to calculate the distance among numeric variables in *k*-NN algorithm.

Once the nearest-neighbours list is obtained, the test example is classified based on the majority class of its nearest neighbours. This approach makes the algorithm sensitive to the choice of k if every neighbour has the same impact. An alternative way to reduce the impact of k is to weight the influence of each nearest neighbour x_i according to its distance: $w_i = 1/d(x', x_i)^2$. Thus, training examples that are located far away from a given test example $z=(x', y')$ have a weaker impact on the classification compared to those that are located close to z. Using the distance-weighted voting scheme, the class label can be determined as follows:

Distance-Weighted Voting: $y' = \arg\max_{v} \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$

where D_z is the set of *k* closest training examples to *z*, *v* is a class label, y_i is the class label for one of the nearest neighbours, and *I* is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

Boosting (AdaBoost.M1)

Boosting is an ensemble classification method. Firstly, it uses voting to combine the output of individual models. Secondly, it combines models in the same type, that is, decision tree or stump. However, boosting assigns a weight to each example and may adaptively change the weight at the end of each boosting round. There are many variants on the idea of boosting algorithm, and we will describe a widely used method called AdaBoost.M1 (Freund and Schapire, 1995) which is designed specifically for classification problem.

AdaBoost stands for "**ada**ptive **boost**ing", it increases the weights of incorrectly classified examples and decreases the ones of those classified correctly. The AdaBoost.M1 algorithm is summarized below:

Given: (x_1, y_1), (x_2, y_2), …, (x_m, y_m) where $x_i \in X, y_i \in Y = \{-1, +1\}$.

- Initialize $D_1(i) = 1/m$.
- For t=1, 2, …, T:
  - ♦ Train weak learner using distribution D_t.
  - ♦ Get weak hypothesis h_t: $X \longrightarrow -1, +1$ with error

$$\varepsilon_t = \Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$$

If $\varepsilon_i > 0.5$, then the weights $D_t(i)$ are reverted back to their original uniform values 1/*m*.

  - ♦ Choose $\alpha_t = ln((1 - \varepsilon_t) / \varepsilon_t)$.
  - ♦ Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if\ h_t(x_i) = y_i \\ 1 & if\ h_t(x_i) \neq y_i \end{cases}$$

where Z_t is a normalization factor, that is, D_{t+1} will be a probability distribution.

- Output the final hypothesis:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

DATA PREPARATION

All of data for the 2009-2011 period were obtained from Hong Kong Environmental Protection Department (HKEPD) and Hong Kong Met-online. The air monitoring station is Tung Chung Air Monitoring Station (Latitude 22°17'19"N, Longitude 113°56'35"E) which is in a new town of Hong Kong, and the meteorological monitoring station is Hong Kong International Airport Weather Station (Latitude 22°18'34"N, Longitude 113°55'19"E) which is the nearest station from Tung Chung. As mentioned in Section 1, accurately predicting high PM2.5 concentration is of most value from a public health standpoint, thus, the response variable has two classes, which are,

"Low" indicating the daily mean concentration of PM2.5 is equal or below 25μg/m^3, and "High" representing above 25μg/m^3. **Figure 1** shows that the days of two levels in 2009-2011. The best situation is in 2010 which has the most days of low level, while the worst is in 2011. **Figure 2** shows the box plot of these three years. We can learn that there are many outliers in both 2009 and 2010, which means there are many serious pollution days. This situation is from various reasons, for instance, the China mainland pollution influenced or appearing a pollution point in a certain time, etc. On the other hand, there is no obvious outlier in 2011 and it has the largest IQR.

We convert all hourly air data to daily mean values, additionally, we add some other related values, which are the Open value (i.e. at 0 o'clock), the Close value (i.e. at 23 o'clock), the Low value (i.e. the lowest value of a day), and the High value (i.e. the highest value of a day). The meteorological data is the original daily data and some of them including low value, high value and mean value. We certainly cannot ignore the effects of seasonal changes and human activities; hence we add two time variables, namely the month (**Figure 3**) and the day of week (**Figure 4**). Figure 3 clearly shows that PM2.5 concentration reaches a low level from May to September, during which is the summer or the rainy season in Hong Kong. But from October to next April the pollutant is serious, especially in October, December and January. We should know that the rainfall may not be an important factor in the experiment as the response variable is the next day's PM2.5, and it is easy to understand that rainy season includes variant meteorological factors. Figure 4 presents the trends of people's activities in some sense. We learn that the air pollution is serious on Saturday and Sunday, while the lowest level appears on Tuesday. This situation is difficult to explain exactly, a proper reason may be the monitoring station locates at a living district in rural area and human activities are mostly at the weekend. But the "Week" factor is not enough satisfied as its trend is too smooth, that is, classification tools prefer to classify much waved variables than the smooth ones.

At last, there are 1065 observations by deleting all NAs and 18 predictor variables (**Table 1**) and 1 response variable which is the next day's PM2.5 concentration level. In summary, the percentage of "Low" level is around 45.8% and "High" level is around 54.2%, respectively. Note that the goal of the predictive model is to obtain the prediction accuracy above randomly guessing, namely 54.2% in this project, otherwise it will be failure.

EXPERIMENTS

The experiments include four parts: the first three parts will respectively show how to train and test each model by multiple times of 10-fold cross validation (10-fold CV), we will choose the best parameter using in the last section which will test the performance and stability of each model by 100 replications of 10-fold CV.

ANN

"nnet" package is the R package we used in this paper. One of the most important parameters of ANN is to select the number of nodes in the hidden layer. There is no stable theory to calculate the

nodes, so the best way is to search it in a proper range. Generally speaking, the number of nodes should not be more than the predictor variables. We use 10 replications of 10-fold CV to calculate the training and testing accuracy when the nodes are from 1 to 20 and the number of iteration is 500. The result is shown in **Table 2**. We learn that the best number is *size* = 3, whose testing accuracy is 0.845. **Figure 5** shows the trends of accuracy by changing the number of nodes in the hidden layer from 1 to 20 and the testing method is still 10 replications of 10-fold CV. We find that either training accuracy or testing accuracy increases no stable, that is, the best accuracy on testing set appearing at *size* = 3 while the training set at *size* = 19. ANN may appear over-fitting when the hidden nodes is large, which means when the training accuracy increasing and the testing accuracy decreasing rapidly. But it does not appear over-fitting in our experiment, or say at least it has a proper performance within 20 hidden nodes. In summary, we will use *size* = 3 in the last section.

k-NN

We will use "kknn" package as *k*-NN analysis tool in this paper. An important issue in *k*-NN algorithm is how to select the proper number of nearest neighbours *k*, while there is no standard method to calculate it exactly. If *k* is too small, then the result can be sensitive to noise points. On the other hand, if *k* is too large, then the neighbourhood may include too many points from other classes. Unquestionably, an odd number for *k* is desirable, that is, the numbers of the nearest neighbours in the set {1, 3, 5, 7, … …}. Empirically, selecting *k* no more than the square root of the samples is a proper choice for *k*-NN. Similar as ANN, we will use 10 replications of 10-fold CV to select the best *k* from 1 to *round* ($\sqrt{1065}$) = 33. The result is shown in **Table 3**. And the trends of accuracy by changing the nearest neighbours is shown in **Figure 6**. We can learn that the accuracy is very close when *k* = 15, 17, and 19 (i.e. they are equivalent when remain 3 decimals). The testing accuracy line seems to be more smooth than ANN in Figure 5. In contrast, the training accuracy line decreases rapidly while *k* becomes larger. Again, there is no over-fitting in *k*-NN as we mentioned above. We choose *k* = 17 as the parameter in the last section (actually it is the highest accuracy when remain more decimals).

Boosting

We will use "RWeka" package for building AdaBoost.M1 model in this paper. Similar as ANN and *k*-NN, we try to obtain the best number of parameter at first. In AdaBoost.M1 we have to select the proper iterations which will influence the weighted power in the model. We set it from 1 to 100 and still use 10 replications of 10-fold CV. Some of the results are shown in **Table 4**. We find that the highest accuracy is at the 80^{th} iteration, whose testing accuracy is 0.862. **Figure 7** indicates the trends of accuracy by changing iterations in AdaBoost.M1. We can see that both the training and testing accuracy are low at the first iterations, that is, the basic classifier (i.e. stump) is weak. The model becomes more powerful when iteration increases. According to Figure 7 we learn that the training accuracy increases more stable than the testing set, the latter waves seriously. Alternatively, one can choose other basic classifiers in AdaBoost.M1 algorithm, for instance, C4.5 (i.e. J48 in "RWeka" package) and so forth. Generally speaking, if the basic classifier is enough powerful then the boosting model will be better, too.

Comparison

This section try to test the performance and the stability of three models, that is, a good algorithm should not only obtain high accuracy but also perform stable in process. We compare all algorithms by using 100 replications of 10-fold CV with the result shown in **Table 5**. We learn that AdaBoost.M1 obtains the best result and its accuracy is around 0.846~0.868. More precisely, its median accuracy is even better than the highest accuracy of either ANN or *k*-NN. **Figure 8** shows the violin plot of this result. A violin plot is a combination of a box plot and a kernel density plot. We can see ANN has a long tail which means its accuracy waves seriously. AdaBoost.M1 and *k*-NN are much more stable than ANN, especially AdaBoost.M1 performs much more better than others.

CONCLUSION

In this paper, we build PM2.5 concentration levels predictive models by using three popular machine learning algorithms, which are ANN, *k*-NN and AdaBoost.M1. The dataset, which is from a rural area in Hong Kong, includes 1065 rows and 19 columns by deleting all missing values. Based on all experiments, the conclusion is shown below.

Either of three algorithms needs to set proper parameters in the model by multiple times 10-fold CV (we set 10 replications in this paper), this can be maximum limit reducing random error in the model. For ANN, selecting the hidden nodes should not be more than the number of variables. For *k*-NN, one can search the suitable *k* among an odd set but no more than the square root of the samples. For Adaboost.M1, iteration is an important parameter related the weighted power and it should be searched in a wide range (e.g. 1 to 100). In order to avoid over-fitting, the selecting criterion should be the testing accuracy but not the training accuracy.

According to 100 replications of 10-fold CV, the best result is from AdaBoost.M1, which not only obtains the highest accuracy but also performs more stable than others. In practice, researchers can change other basic classifiers (e.g. C4.5) or add new parameters (e.g. rule based) in AdaBoost.M1 algorithm. ANN performs clearly unstable and it may not be a suitable tool for PM2.5 prediction models. *k*-NN is also a stable model though its accuracy is lower than AdaBoost.M1. A more accurate distance function may be enhancing *k*-NN's performance, for instance, high dimension functions. Additionally, more powerful weighted is also very important (such like boosting).

Figure-1. PM2.5 Concentration Levels in 2009-2011

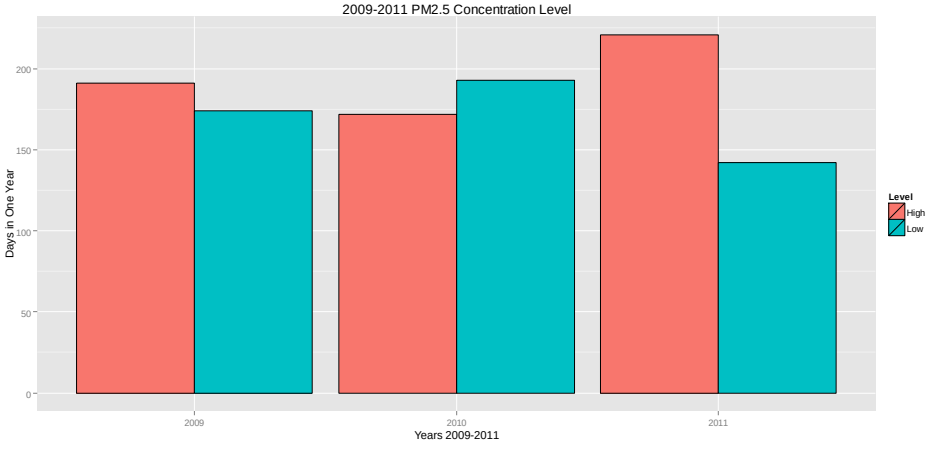


Figure-2. Box Plot of PM2.5 Concentration in 2009-2011

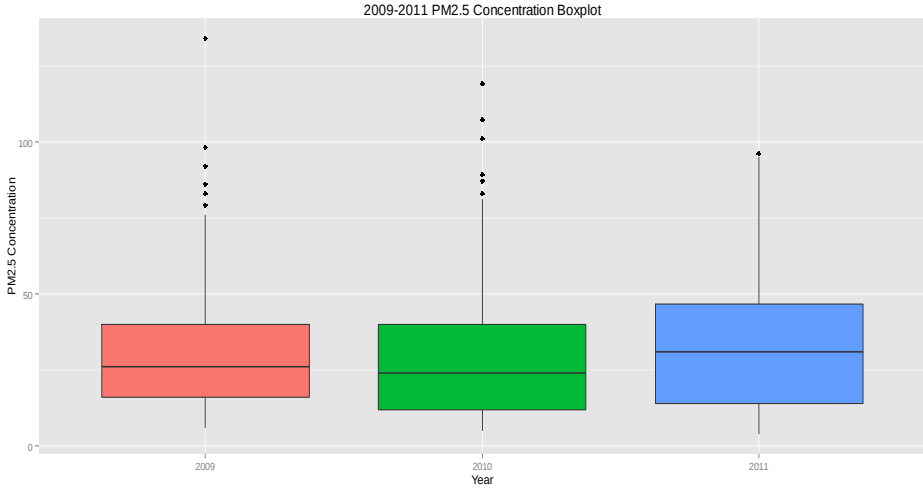


Figure-3. Monthly PM2.5 Median Concentration in 2009-2011

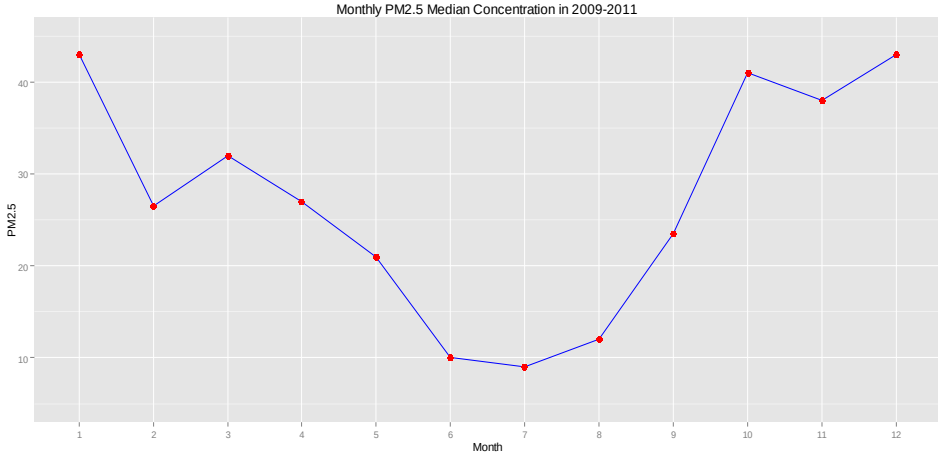


Figure-4. Weekly PM2.5 Median Concentration in 2009-2011

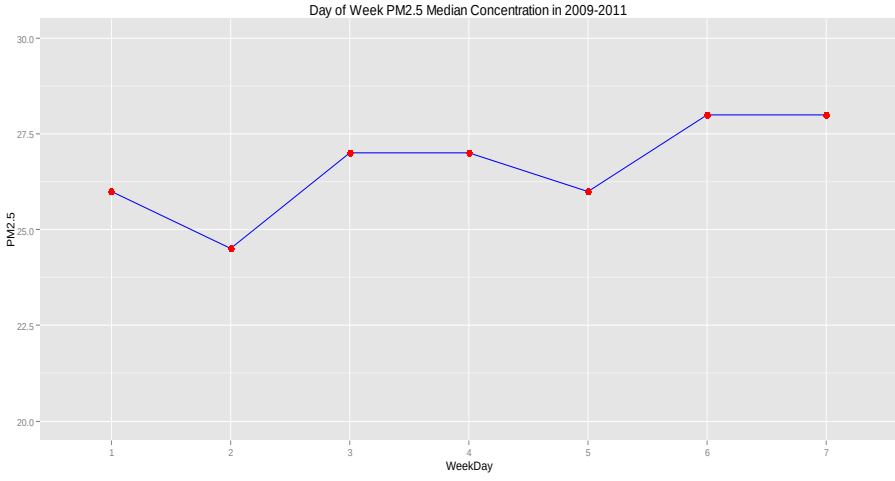


Table-1. Predictor Variables List

Notation	Description	Variable Class
MP	Mean Pressure	Numeric
AT1	Max Air Temperature	Numeric
AT2	Mean Air Temperature	Numeric
AT3	Min Air Temperature	Numeric
MDPT	Mean Dew Point Temperature	Numeric
RH1	Max Relative Humidity	Numeric
RH2	Mean Relative Humidity	Numeric
RH3	Min Relative Humidity	Numeric
TR	Total Rainfall	Numeric
PWD	Prevailing Wind Direction	Numeric
MWS	Mean Wind Speed	Numeric
Mean	Mean of PM2.5Concentration	Numeric
Open	PM2.5 Value at Midnight	Numeric
Close	PM2.5 Value at 23 o'clock	Numeric
Low	Lowest PM2.5 Value of a Day	Numeric
High	Highest PM2.5 Value of a Day	Numeric
MONTH	Month (Jan. to Dec.)	Nominal
WEEK	Day of week (Mon. to Sun.)	Nominal

Table-2. Accuracy of Different Number of Hidden Nodes in ANN

Nodes	Training Accuracy	Testing Accuracy
1	0.815	0.784
2	0.868	0.830
3	**0.882**	**0.845**
4	0.876	0.834
5	0.881	0.838
6	0.880	0.837
7	0.881	0.842
8	0.884	0.842
9	0.892	0.843
10	0.897	0.843
11	0.892	0.834
12	0.883	0.840
13	0.888	0.842

14	0.886	0.839
15	0.886	0.841
16	0.894	0.844
17	0.894	0.841
18	0.887	0.844
19	0.900	0.837
20	0.894	0.836

Figure-5. Trends of Accuracy by Changing Nodes in ANN

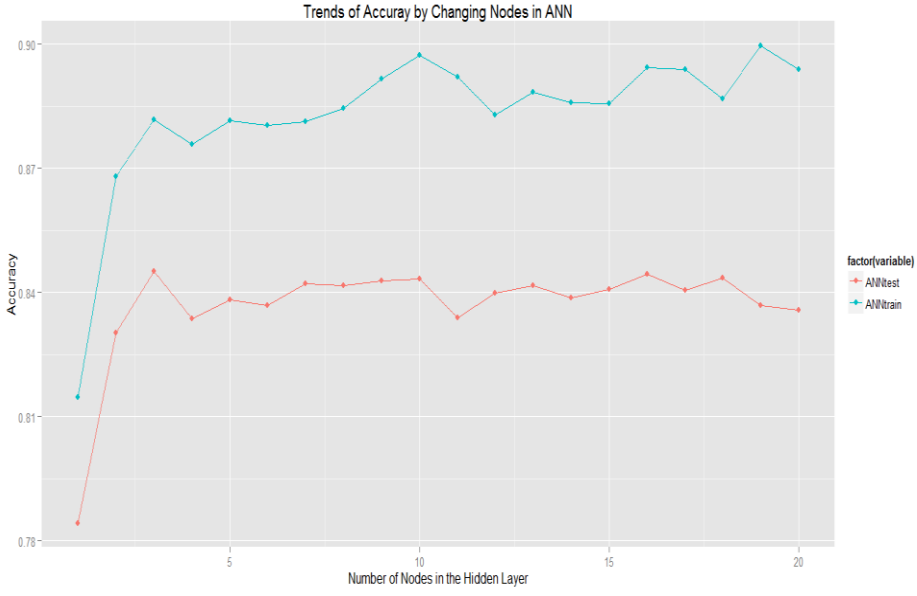


Table-3. Accuracy of Different Nearest Neighbors in *k*NN

Nearest Neighbors	**Training Accuracy**	**Testing Accuracy**
1	1.000	0.789
3	1.000	0.785
5	0.947	0.824
7	0.925	0.832
9	0.913	0.838
11	0.904	0.840
13	0.897	0.842
15	0.891	0.843
17	**0.888**	**0.843**
19	0.885	0.843
21	0.882	0.839
23	0.880	0.842
25	0.878	0.841
27	0.875	0.842
29	0.873	0.840
31	0.871	0.838
33	0.869	0.839

Figure-6. Trends of Accuracy by Changing Nearest Neighbors in *k*NN

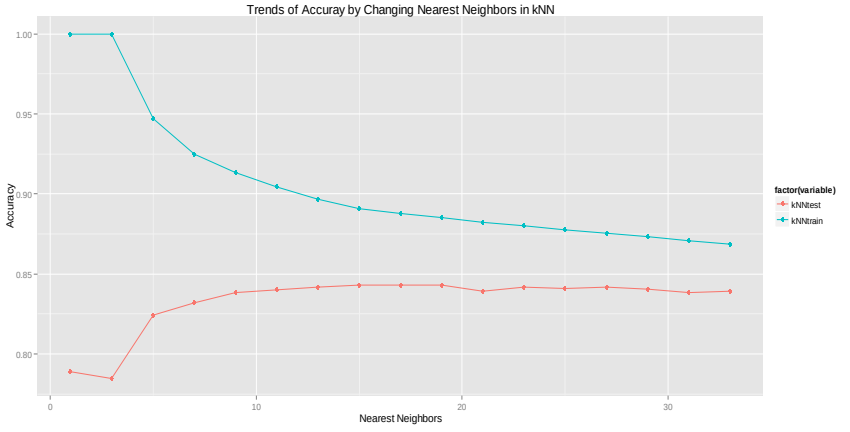


Table-4. Accuracy of Different Iterations in AdaBoost.M1

Iterations	Training Accuracy	Testing Accuracy
1	0.845	0.845
2	0.845	0.845
3	0.845	0.845
4	0.847	0.845
5	0.848	0.846
.	.	.
.	.	.
.	.	.
79	0.876	0.857
80	**0.876**	**0.862**
81	0.875	0.861
.	.	.
.	.	.
.	.	.
100	0.878	0.860

Figure-7. Trends of Accuracy by Changing Iterations in AdaBoost.M1

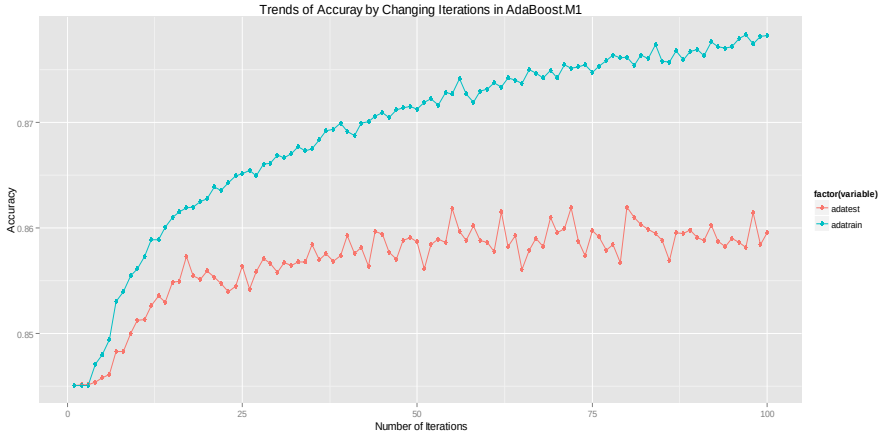
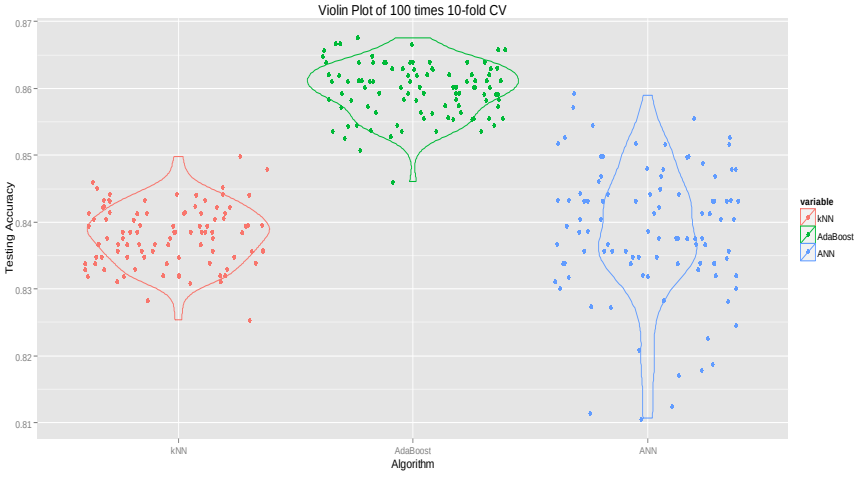


Table-5. The Result of 100 times 10-fold CV

	Maximum	Minimum	Median
ANN	0.859	0.811	0.839
kNN	0.850	0.825	0.838
AdaBoost.M1	0.868	0.846	0.860

Figure-8. Violin Plot of 100 times 10-fold CV



ACKNOWLEDGEMENTS

Thanks to Hong Kong Environmental Protection Department (HKEPD) and Hong Kong Met-online provide all related data in this paper.

REFERENCE

AQO, 2012. Proposed new aqos for hong kong. Available from http://www.epd.gov.hk/epd/english/environmentinhk/air/air_quality_objectives/files/proposed_newAQOs_eng.pdf.

Cover, T.M. and P.E. Hart, 1967. Nearest neighbor pattern classification, institute of electrical and electronics engineers transactions on information theory. 13: 21-27.

Freund, Y. and R. Schapire, 1995. A decision-theoretic generalization of on-line learning and an application to boosting, european conference on computational learning theory. pp: 23-37.

Hagan, M.T., H.B. Demuth and M.H. Beale, 1996. Neural network design. Boston London: Pws Pub.

Hornik, K., C. Buchta, T. Hothorn, A. Karatzoglou, D. Meyer and A. Zeileis, 2013. R/weka interface, "rweka" package, version 0. 4-17. Available from http://cran.r-project.org/web/packages/RWeka/RWeka.pdf.

Ihaka, R. and R. Gentleman, 1996. R: A language for data analysis and graphics. Journal of computational and graphical statistics, 5(3): 299-314.

Ripley, B., 2013. Feed-forward neural networks and multinomial log-linear models, "nnet" package, version 7.3.6. Available from http://cran.r-project.org/web/packages/nnet/nnet.pdf.

Rojas, R., 1996. Neural networks: A systematic introduction. Springer.

Schliep, K. and K. Hechenbichler, 2013. Weighted k-nearest neighbors, "kknn" package, version 1.2.2. Available from http://cran.r-project.org/web/packages/kknn/kknn.pdf.

WHO, 2005. Air quality guidelines for particulate matter, ozone, nitrogen dioxide and sulfur dioxide (global update 2005), world health organization. Available from whqlibdoc.who.int/hq/2006/WHO_SDE_PHE_OEH_06.02_eng.pdf.

WHO, 2011. Air quality and health. Available from http://www.who.int/mediacentre/factsheets/fs313/en/index.html.

Wickham, H., 2013a; 2013b. Flexibly reshape data: A reboot of the reshape package , reshape2 package, version 1.2.2. Available from http://cran.r-project.org/web/packages/reshape2/reshape2.pdf.