PROJECT REPORT

ABSTRACT

The objective of the project is prediction of air pollution levels by predicting the PM10 levels in a city by using air pollution data. The project aims at applying different machine learning algorithms suitable with data and using these models to build a concrete and robust model for prediction of pollution levels. In the project we have used several different algorithms like regression analysis, SVM, ANN and deep learning and compared the performance of these models on standard data set. We have used openair package for the data set and R programming as the statistical programming language for coding.

INTRODUCTION

Air pollution is one of the important environmental problems in metropolitan cities. There are many air pollution indicators affecting human health. Some of the important indicators are particulate matter (PM_{10}), carbon monoxide (CO) and sulfur dioxide (SO_2) etc. Many national environmental agencies have set standards and air quality guidelines for allowable levels of these pollutants in the air. When the concentration levels of these indicators exceed the air quality guidelines, short term and chronic human health problems may occur.

An air pollutant is a substance in the air that can have adverse effects on humans and the ecosystem. The substance can be solid particles, liquid droplets, or gases. A pollutant can be of natural origin or man-made. Pollutants are classified as primary or secondary. Primary pollutants are usually produced from a process, such as ash from a volcanic eruption. Other examples include carbon monoxide gas from motor vehicle exhaust, or the sulfur dioxide released from factories.

Major Different Air Pollutants

- Sulfur oxides (SO_x) - particularly sulfur dioxide, a chemical compound with the formula SO_2. SO_2 is produced by volcanoes and in various industrial processes. Coal and petroleum often contain sulfur compounds, and their combustion generates sulfur dioxide. Further oxidation of SO_2, usually in the presence of a catalyst such as NO_2, forms H_2SO_4, and thus acid rain.

- Nitrogen oxides (NO_x) - Nitrogen oxides, particularly nitrogen dioxide, are expelled from high temperature combustion, and are also produced during thunderstorms by electric discharge. They can be seen as a brown haze dome above or aplume downwind of cities. Nitrogen dioxide is a chemical compound with the formula NO_2. It is one of several nitrogen oxides. One of the most prominent air pollutants, this reddish-brown toxic gas has a characteristic sharp, biting odor.
- Carbon monoxide (CO) - CO is a colorless, odorless, toxic yet non-irritating gas. It is a product by combustion of fuel such as natural gas, coal or wood. Vehicular exhaust is a major source of carbon monoxide.
- Volatile organic compounds (VOC) - VOCs are a well-known outdoor air pollutant. They are categorized as either methane (CH_4) or non-methane (NMVOCs). Methane is an extremely efficient greenhouse gas which contributes to enhanced global warming. Other hydrocarbon VOCs are also significant greenhouse gases because of their role in creating ozone and prolonging the life of methane in the atmosphere. This effect varies depending on local air quality.
- Particulates, alternatively referred to as particulate matter (PM 2.5 and PM10), atmospheric particulate matter, or fine particles, are tiny particles of solid or liquid suspended in a gas. In contrast, aerosol refers to combined particles and gas. Some particulates occur naturally, originating from volcanoes, dust storms, forest and grassland fires, living vegetation, and sea spray. Human activities, such as the burning of fossil fuels in vehicles, power plants and various industrial processes also generate significant amounts of aerosols. Averaged worldwide, anthropogenic aerosols—those made by human activities—currently account for approximately 10 percent of our atmosphere. Increased levels of fine particles in the air are linked to health hazards such as heart disease, altered lung function and lung cancer.

ABOUT DATASET

The main aim of the project was to run the algorithms on dataset of a Indian city mainly Delhi due to rising pollution levels in Delhi.

But we were not able to find any reliable dataset of any Indian city. So we applied the algorithms on European cities which were readily available in the R package –OPENAIR written by david carslaw http://www.openair-project.org/.

The **OPENAIR** project is a **Natural Environment Research Council (NERC)** knowledge exchange project that aims to provide a collection of open-source tools for the analysis of air pollution data. We used the dataset of London, UK as our pollution dataset.

OPENAIR:

Openair is one of the most successful package for analyzing air pollution levels.It has many features like wind roses and bipolar diagrams which depict the picture of air pollution more clearly.

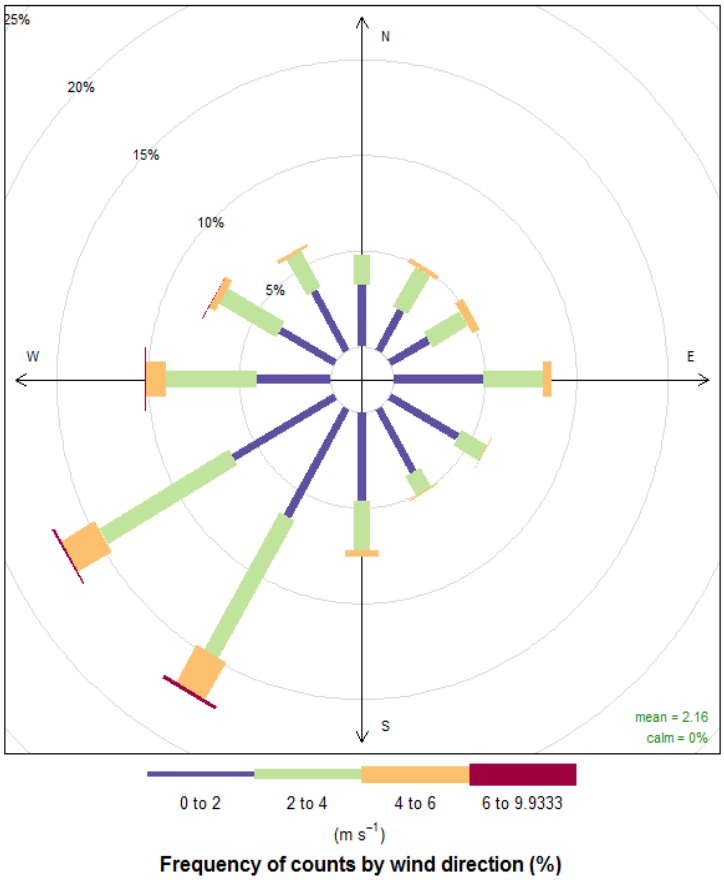
WindRoses:

Wind roses and pollution roses are the *'bread and butter'* of air pollution analysis, but it is surprisingly difficult to find software to produce these plots properly. **openair** comes with a dataset *'mydata'*, which provides several years of air pollution data from a site in London. The *'mydata'* dataset is useful for seeing how the functions work and all **openair** examples use it.

Given a data set with wind speed (ws) and wind direction (wd) data it is easy to produce a basic plot:

R command

```
windRose(mydata)
```

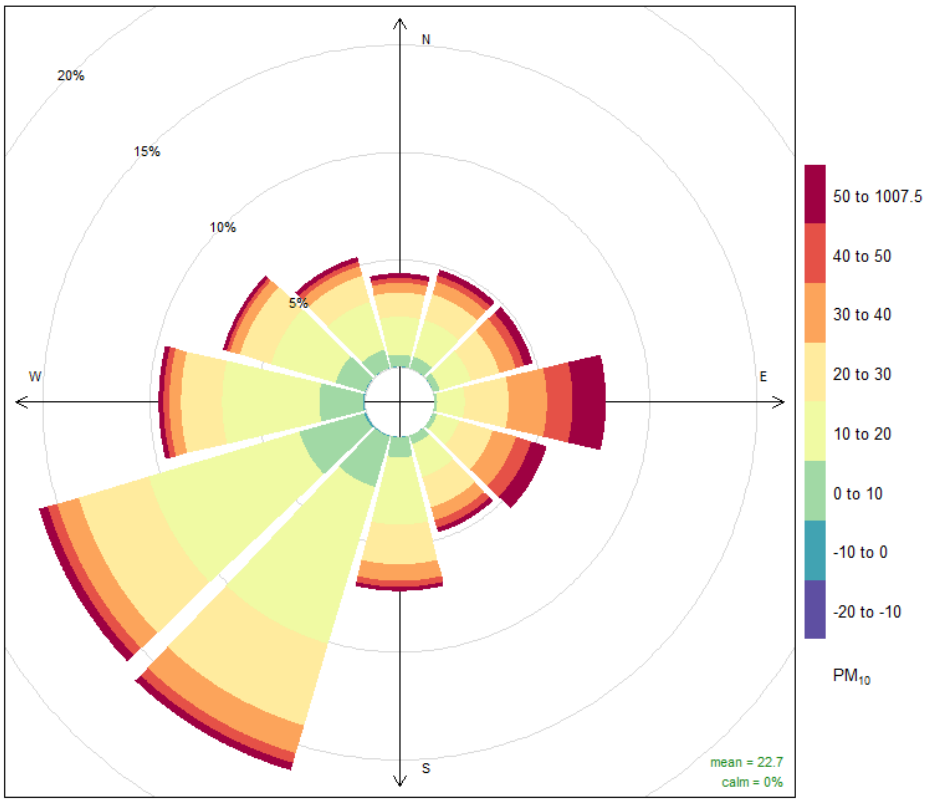


Pollution Rose:

The pollution rose is same as wind rose just pollution parameter is plotted instead of wind.

Rcommand:

```
pollutionRose(mydata, pollutant = 'o3', type = 'season')
```



Frequency of counts by wind direction (%)

BIVARIATE POLAR PLOT:

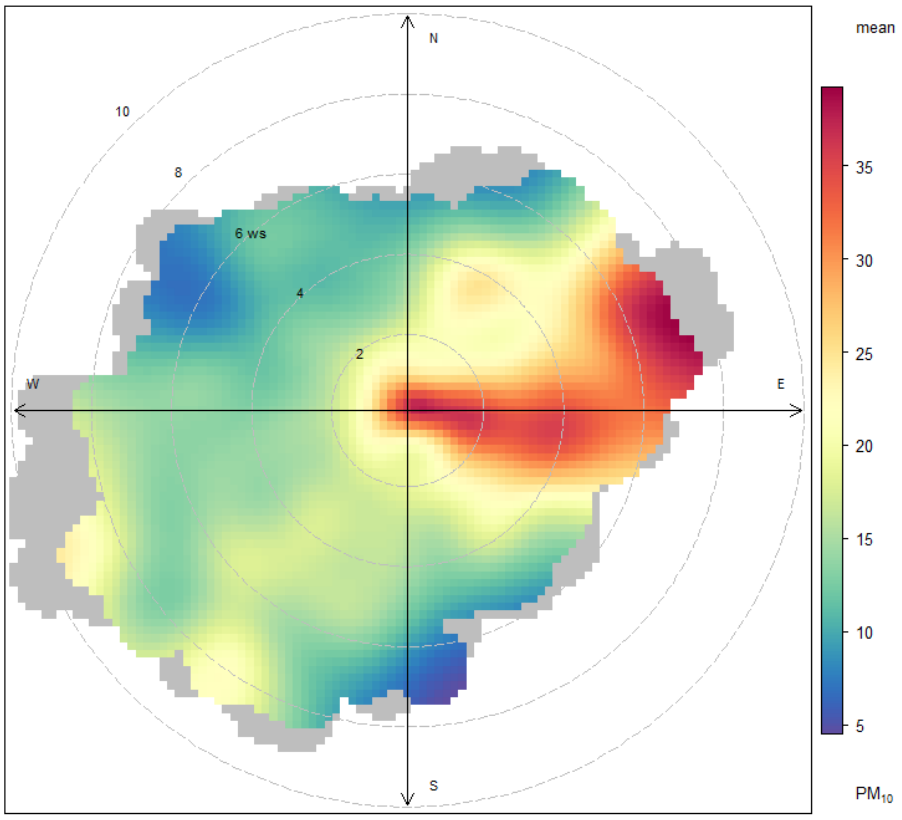
The bivariate polar plot (**polarPlot**) is a useful diagnostic tool for quickly gaining an idea of potential sources. Wind speed is one of the most useful variables to use to separate source types For example, ground-level concentrations resulting from buoyant plumes from chimney stacks tend to peak under higher wind speed conditions. Conversely, ground-level, non-buoyant plumes such as from road traffic, tend to have highest concentrations under low wind speed

conditions. Other sources such as from aircraft engines also show differing characteristics by wind speed.

Furthermore, given the very flexible 'type' option in **openair** functions, it is very easy to generate plots by season, site and so on – the possibilities are enormous and can be tailored to the question at hand.

R command:

```
polarPlot(mydata, pollutant = 'so2', type = 'weekend')
```



The package contains data of different cities of counties in Europe. Different dataset contains different predictors(like NOx,SOx,CO2,wind speed, Temperature) according to the city.So we tried to pick the city with the maximum number of reliable predictors to make the model more robust.

The link of the datsets used in the refernces part of the report.

ALGORITHMS USED:

1. Multivariate Linear Regression.

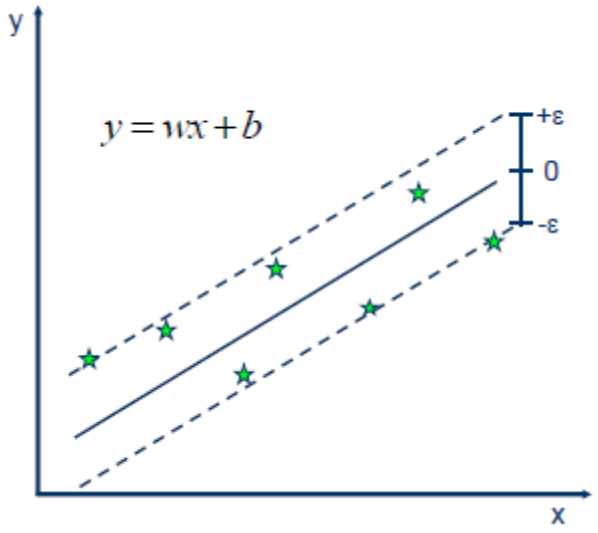
Regression analysis is used to predict the value of one or more responses from a set of predictors. It can also be used to estimate the linear association between the predictors and responses. Predictors can be continuous or categorical or a mixture of both.

The dependent variable *y* is specified as the sum of an intercept term α plus the product of the estimated β value and the *x* values for each of the *i* features. An error term (denoted by the Greek letter *epsilon*) has been added here as a reminder that the predictions are not perfect. This represents the **residual** term.

Let z1, z2, …… , zn be a set of n predictors believed to be related to a response variable Y .
The linear regression model for the jth sample unit has the form
Yj = a0 + a1*z1 + a2*z2 +……+_n*zn + j;
Where j is the residual term.

2. Support Vector Machines(regression).

A version of SVM for regression was proposed in 1996 by Vladimir N. Vapnik, Harris Drucker, Christopher J. C. Burges, Linda Kaufman and Alexander J. Smola.[26] Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities.In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.



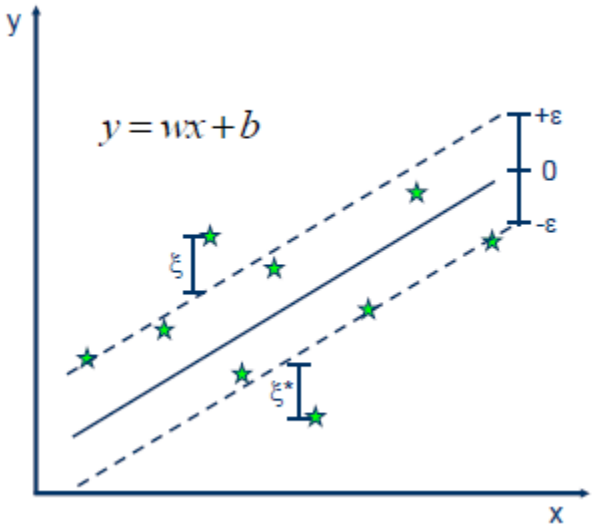
- Solution:

$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon$$

$$wx_i + b - y_i \le \varepsilon$$



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

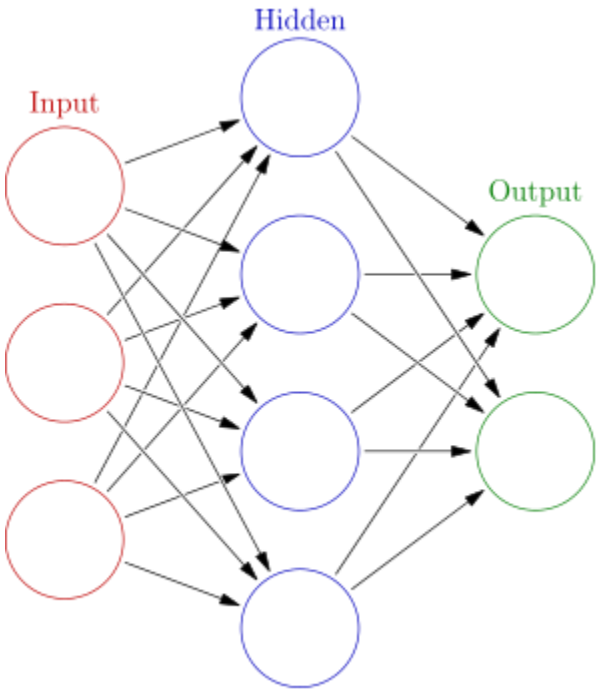
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

3.Artificial Neural Networks

ANNs are processing devices (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the mamalian cerebral cortex but on much smaller scales. A large ANN might have hundreds or thousands of processor units, whereas a mamalian brain has billions of neurons with a corresponding increase in magnitude of their overall interaction and emergent behavior.

Neural neworks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'.



Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with.

Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one; the delta rule. The delta rule is often utilized by the most common class of ANNs called **'backpropagational neural networks' (BPNNs).** Backpropagation is an abbreviation for the backwards propagation of error.

With the delta rule, as with other types of backpropagation, 'learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights.

Once a neural network is 'trained' to a satisfactory level it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation.

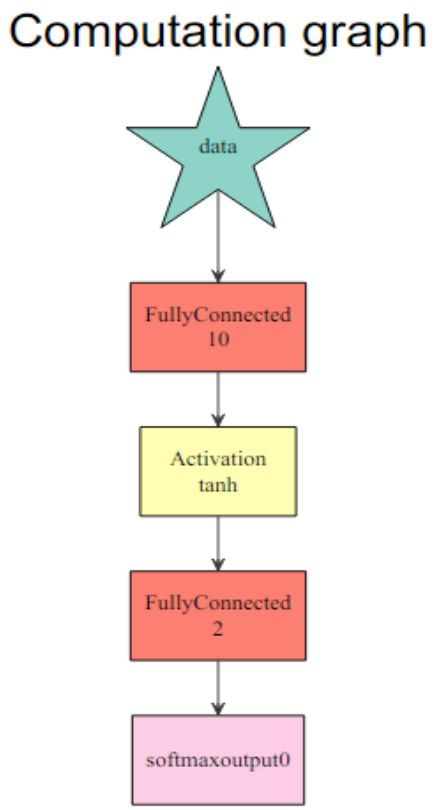
4. Deep Learning

Deep learning is a newly developed technique to predict and classify the data and its an improvement to existing neural networks model.

To apply the Deep Learning we have used the MXNET package in R.

In deep learning by MXNET we use the multilayer perceptron model for training the model and then we use the simple regression model for prediction of the results.

Here in the floe chart the layers of the deep learning are shown and the tanh and softmax is the activation function.



Deep Learning can be understood as an algorithm which is composed of hidden layers of multiple neural networks. It works on unsupervised data and is known to provide accurate results than traditional ML algorithms.

Input data is passed through this algorithm, which is then passed through several non-linearities before delivering output. This algorithm allows us to go ‘deeper’ (higher level of abstraction) in the network without ending up writing lot of duplicated code, unlike ‘shallow’ algorithms. As it goes deeper and deeper, it filter the complex features and combines with those of previous layer, thus better results.

PROCEDURE:

1. Data Mining:

As we took a standard data set from the Openair package in R, so the there was no need for checkeing the validity of the dataset as it is already obtained from a standasrd data source.

2.Data Preprocessing And Splitting:

For the data preprocessing step we basically used the 3 steps :

1.**Missing Values**

First we estimated the missing values which were present in the data set by using the mean of the data values till the data point.

Actually we ran the algorithm on different datasets in some dataset we estimated the missing values by mean of the data and in some dataset we totally removed the data [point with the missing values.

2. **Identifying Correlated Predictors:**

While there are some models that thrive on correlated predictors (such as pls), other models may benefit from reducing the level of correlation between the predictors.

For identifying the correlated parameters in the data set we used the PCA method i.e. Principle Component Analysis

##R CODE

nearZeroVar(data)

 # When predictors should be removed, a vector of integers is

 # returned that indicates which columns should be removed.

correlations <- cor(data)

dim(correlations)

correlations[1:4, 1:4]

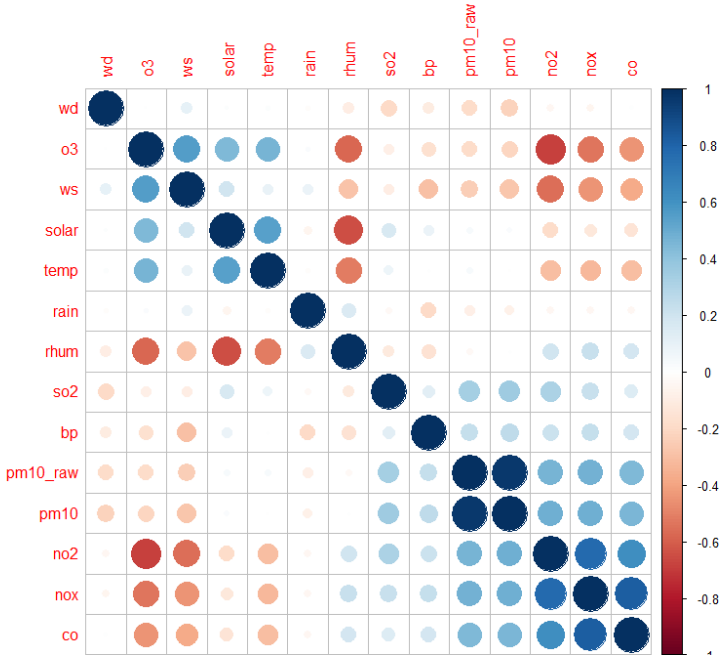
library(corrplot)

corrplot(correlations, order = "hclust")

highCorr <- findCorrelation(correlations, cutoff = .75)

length(highCorr)

head(highCorr)



3. Centering And Scaling Of Data:

The most straightforward and common data transformation is to center scale the predictor variables. To center a predictor variable, the average predictor value is subtracted from all the values. As a result of centering, the predictor has a zero mean.

Similarly, to scale the data, each value of the predictor variable is divided by its standard deviation. Scaling the data coerce the values to have a common standard deviation of one.

These manipulations are generally used to improve the numerical stability of some calculations. Some models, such as PLS benefit from the predictors being on a common scale. The only real downside to these transformations is a loss of interpretability of the individual values since the data are no longer in the original units.

We used the max and min scaling method to scale the data.

```
## R CODE

index <- sample(1:nrow(data),round(0.25*nrow(data)))

maxs <- apply(mydata11, 2, max)

mins <- apply(mydata11, 2, min)

scaled <- as.data.frame(scale(mydata11, center = mins, scale = maxs - mins))

train_ <- scaled[1:20000,]

test_ <- scaled[20001:42890,]


## R CODE

normalize <- function(x) { return((x - min(x)) / (max(x) - min(x)))}
```

Applicationaof algorithms:

1.Multivariate Linear Regression:

```
##R CODE:

index <- sample(1:nrow(data),round(0.25*nrow(data)))

train <- mydata11[1:35000,]

test <- mydata11[35001:70000,]

lm.fit <- glm(pm10~., data=train)
```

summary(lm.fit)

pr.lm <- predict(lm.fit,test)

MSE.lm <- sum((pr.lm - test$pm10)^2)/nrow(test)

summary(pr.lm)

summary(MSE.lm)

2.Support Vector Machines(regression):

R CODE
library(kernlab)

airdata_model<- ksvm(pm10 ~ws+wd+nox+no2+o3+so2+co+rhum+bp+temp+rain+solar+pm10_raw,data= airdata_train, kernel = "vanilladot")

model_results <- predict(airdata_model, airdata_test[c(1,2,3,4,5,7,8,9)])

cor(model_results,airdata_test$pm10)

3.Artificial Neural Networks:

##R CODE

mydata11<-data

maxs <- apply(mydata11, 2, max)

mins <- apply(mydata11, 2, min)

scaled <- as.data.frame(scale(mydata11, center = mins, scale = maxs - mins))

train_ <- scaled[1:35000,]

test_ <- scaled[35001:70000,]

library(neuralnet)

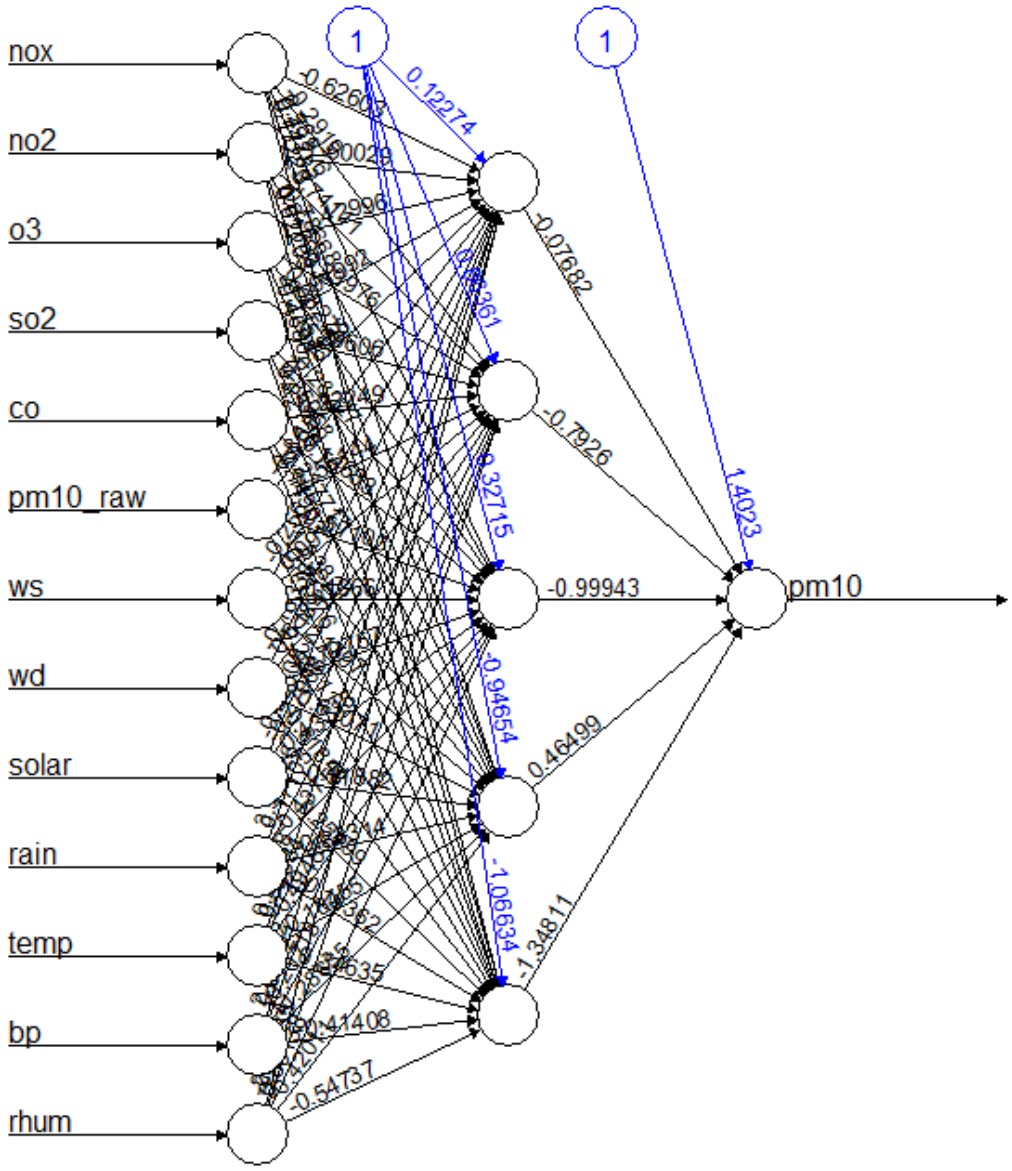
n <- names(train_)

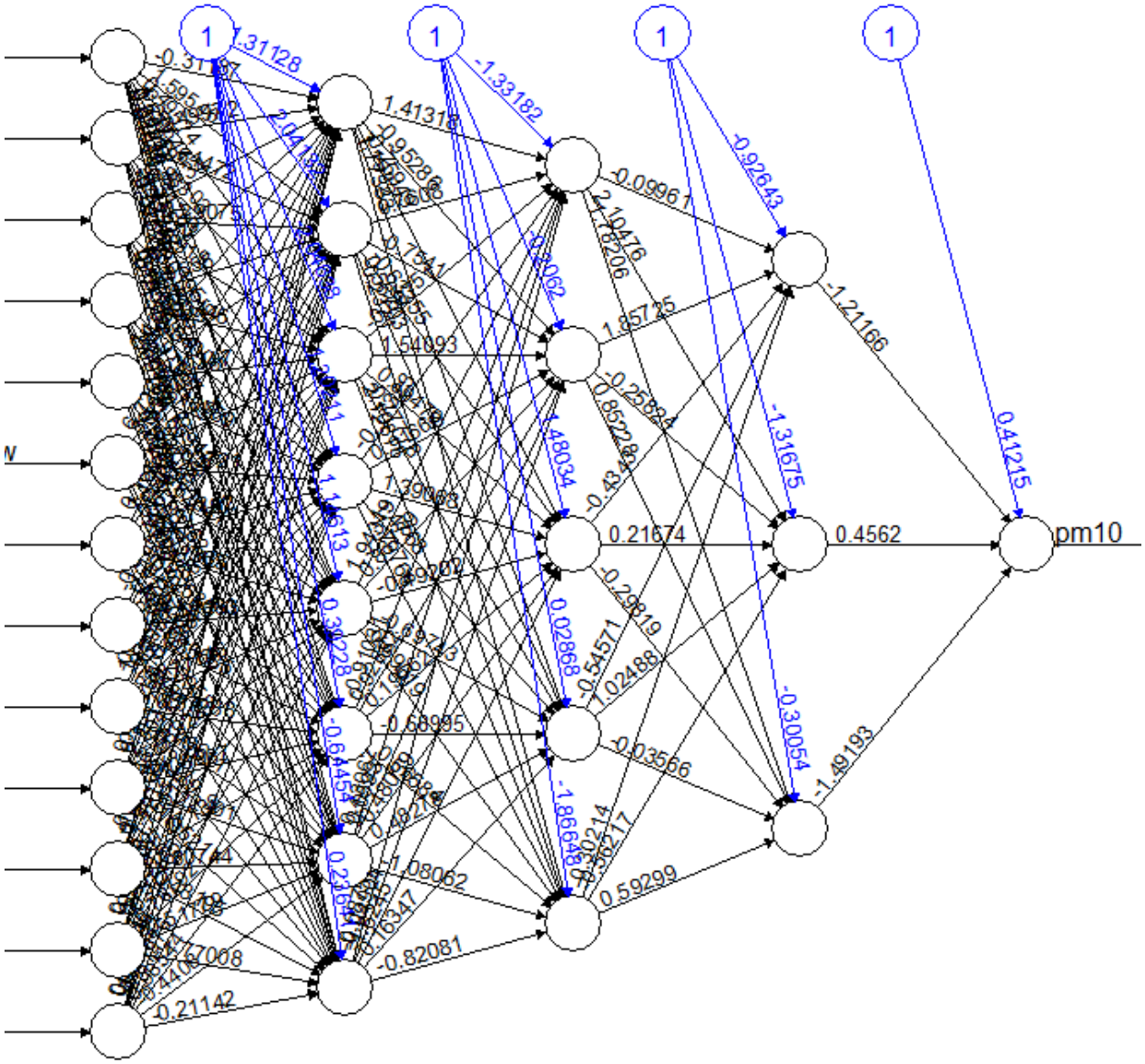
f <- as.formula(paste("pm10 ~", paste(n[!n %in% "pm10"], collapse = " + ")))

nn <- neuralnet(f,data=train_,hidden=c(5),linear.output=T)

plot(nn)

summary(nn)




1
1
1
1
1.31128
-1.33182
-0.92643
1.41318
-0.09961
-1.21166
0.41215
-1.31675
0.21674
0.4562
pm10
-0.68995
-0.64454
-1.08062
-0.82081
-0.21142
0.59299
-1.49193

4.Deep Learning:

```
## R CODE
bx1<-normalize(bx1)
bx11<-bx1[-1]
bx11<-bx1[-5]
train.ind = seq(1, 15000, 3)
train.x = data.matrix(bx11[train.ind, -7])
train.y = bx1[train.ind, 7]
test.x = data.matrix(bx11[-train.ind, -7])
test.y = bx1[-train.ind, 7]
# Define the input data
data <- mx.symbol.Variable("data")
# A fully connected hidden layer
# data: input source
# num_hidden: number of neurons in this hidden layer
fc1 <- mx.symbol.FullyConnected(data, num_hidden=1)
# Use linear regression for the output layer
lro <- mx.symbol.LinearRegressionOutput(fc1)

mx.set.seed(0)
model <- mx.model.FeedForward.create(lro, X=train.x, y=train.y,
            ctx=mx.cpu(), num.round=50, array.batch.size=20,
            learning.rate=2e-6, momentum=0.9,         eval.metric=mx.metric.rmse)

demo.metric.mae <- mx.metric.custom("mae", function(label, pred) {
  res <- mean(abs(label-pred))
  return(res)})
mx.set.seed(0)
```

model1 <- mx.model.FeedForward.create(lro, X=train.x, y=train.y,

ctx=mx.cpu(), num.round=50, array.batch.size=20,

learning.rate=2e-6, momentum=0.9, eval.metric=demo.metric.mae)

RESULTS AND CONCLUSION:

1. Multivariate Linear Regression

Call:
glm(formula = pm10 ~ ., data = train)

Deviance Residuals:
Min	1Q	Median	3Q	Max
-0.035992	-0.000361	0.000044	0.000405	0.027453

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-0.0129030	0.0026686	-4.835	1.34e-06	***
nox	-0.0034025	0.0004857	-7.006	2.50e-12	***
no2	0.0068406	0.0015418	4.437	9.16e-06	***
o3	0.0046495	0.0010121	4.594	4.37e-06	***
so2	0.0125285	0.0016481	7.602	2.99e-14	***
co	0.5889010	0.0968538	6.080	1.21e-09	***
pm10_raw	0.9813481	0.0010819	907.077	< 2e-16	***
ws	-0.2491795	0.0182751	-13.635	< 2e-16	***
wd	-0.0018893	0.0001919	-9.843	< 2e-16	***
solar	0.0022650	0.0001409	16.072	< 2e-16	***
rain	0.5937769	0.1716504	3.459	0.000542	***
temp	-0.0858833	0.0035374	-24.278	< 2e-16	***
bp	0.0050608	0.0016657	3.038	0.002381	**
rhum	0.0138214	0.0020319	6.802	1.05e-11	***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 4.476747e-06)

Null deviance: 6.19478 on 34999 degrees of freedom
Residual deviance: 0.15662 on 34986 degrees of freedom
AIC: -331740

Number of Fisher Scoring iterations: 2

Min. 1st Qu. Median Mean 3rd Qu. Max.
0.001006 0.020820 0.024700 0.026460 0.029790 0.447500

Min. 1st Qu. Median Mean 3rd Qu. Max.
1.574e-05 1.574e-05 1.574e-05 1.574e-05 1.574e-05 1.574e-05

The model gives a accuracy of 78.912% on the dataset.

2.Support Vector Machines(Regression)

We got the following results for our Support Vector Machines(Regression)

1. Rmse 3.1413
2. Mse 6.31589
3. Accuracy 88.413%

3.Artificial Neural Networks:

##OUTPUT

	Length	*Class*	*Mode*
call	*5*	*-none-*	*call*
response	*35000*	*-none-*	*numeric*
covariate	*455000*	*-none-*	*numeric*
model.list	*2*	*-none-*	*list*
err.fct	*1*	*-none-*	*function*
act.fct	*1*	*-none-*	*function*
linear.output	*1*	*-none-*	*logical*
data	*14*	*data.frame*	*list*
net.result	*1*	*-none-*	*list*
weights	*1*	*-none-*	*list*
startweights	*1*	*-none-*	*list*
generalized.weights	*1*	*-none-*	*list*
result.matrix	*79*	*-none-*	*numeric*

The neural networks gave

4. Rmse 2.809243
5. Mse 5.023567
6. Accuracy 91.6734%

4. Deep Learning:

##OUTPUT:

Start training with 1 devices
[1] Train-rmse=0.0324828087894422
[2] Train-rmse=0.0322357936142398
[3] Train-rmse=0.0320144356671684
[4] Train-rmse=0.0317949821011668
[5] Train-rmse=0.0315774133157252
[6] Train-rmse=0.0313617162896698
[7] Train-rmse=0.0311478789637983
[8] Train-rmse=0.0309358864850154
[9] Train-rmse=0.0307257265153365
[10] Train-rmse=0.030517383818663
[11] Train-rmse=0.0303108482299818
[12] Train-rmse=0.030106104507999
[13] Train-rmse=0.0299031393095779
[14] Train-rmse=0.0297019412563302
[15] Train-rmse=0.0295024940282488
[16] Train-rmse=0.0293047875709911
[17] Train-rmse=0.0291088092665217
[18] Train-rmse=0.0289145450096055
[19] Train-rmse=0.0287219824769046
[20] Train-rmse=0.0285311094834895
[21] Train-rmse=0.0283419134608775
[22] Train-rmse=0.0281543825077083
[23] Train-rmse=0.027968503205177
[24] Train-rmse=0.0277842642601124
[25] Train-rmse=0.027601653011835
[26] Train-rmse=0.0274206584078134
[27] Train-rmse=0.0272412693491175
[28] Train-rmse=0.0270634734612795
[29] Train-rmse=0.0268872589312788
[30] Train-rmse=0.02671261556091
[31] Train-rmse=0.0265395295192749
[32] Train-rmse=0.0263679892360978
[33] Train-rmse=0.0261979856015501
[34] Train-rmse=0.026029507230944
[35] Train-rmse=0.0258625431973581
[36] Train-rmse=0.0256970809193035
[37] Train-rmse=0.0255331117417649
[38] Train-rmse=0.0253706232723056
[39] Train-rmse=0.025209603386662
[40] Train-rmse=0.0250500430568909

[41] Train-rmse=0.0248919320317004
[42] Train-rmse=0.024735258140536
[43] Train-rmse=0.0245800108998474
[44] Train-rmse=0.0244261831948971
[45] Train-rmse=0.0242737624157034
[46] Train-rmse=0.0241227366816529
[47] Train-rmse=0.0239730981230134
[48] Train-rmse=0.0238248353756341
[49] Train-rmse=0.0236779388167525
[50] Train-rmse=0.0235323985270086

Start training with 1 devices
[1] Train-mae=0.0302957312712244
[2] Train-mae=0.0300405745614815
[3] Train-mae=0.0298060379314644
[4] Train-mae=0.0295733350666938
[5] Train-mae=0.0293424457222107
[6] Train-mae=0.0291133566705976
[7] Train-mae=0.0288860557594045
[8] Train-mae=0.0286605277435738
[9] Train-mae=0.028436760075693
[10] Train-mae=0.0282147371396888
[11] Train-mae=0.0279944487497909
[12] Train-mae=0.0277758792937268
[13] Train-mae=0.0275590151165379
[14] Train-mae=0.027343844775029
[15] Train-mae=0.0271303513671271
[16] Train-mae=0.026918524862011
[17] Train-mae=0.0267083523995942
[18] Train-mae=0.0264998195202323
[19] Train-mae=0.0262929136349587
[20] Train-mae=0.0260876224126201
[21] Train-mae=0.0258839330122108
[22] Train-mae=0.0256818333140342
[23] Train-mae=0.0254813095705351
[24] Train-mae=0.0252823503556196
[25] Train-mae=0.025084942753613
[26] Train-mae=0.0248890755135799
[27] Train-mae=0.0246947374193463
[28] Train-mae=0.0245019157701172
[29] Train-mae=0.0243105986159295
[30] Train-mae=0.0241207755987532
[31] Train-mae=0.0239324325419962
[32] Train-mae=0.0237455575862434
[33] Train-mae=0.0235601417175494
[34] Train-mae=0.0233761732809245

[35] Train-mae=0.0231936412483919
[36] Train-mae=0.023012532637734
[37] Train-mae=0.0228328389031813
[38] Train-mae=0.0226545473227743
[39] Train-mae=0.0224776454511099
[40] Train-mae=0.0223021243371069
[41] Train-mae=0.0221279735147022
[42] Train-mae=0.0219551805811003
[43] Train-mae=0.0217837348463014
[44] Train-mae=0.0216136294509284
[45] Train-mae=0.021444851375185
[46] Train-mae=0.0212773884428665
[47] Train-mae=0.021111232942529
[48] Train-mae=0.0209463731859811
[49] Train-mae=0.0207827995543368
[50] Train-mae=0.020620501991082

The final accuracy given by deep learning is 95.023510%

CONCLUSION

We applied various machine learning algorithms on our dataset and we found out that deep learni ng gave best results and hence model prepared by deep learning was the most robust and concret e model. The accuracy was far better than the other algorithms applied.

FURTHER WORK

In future we intend to apply ensemble learning (bagging and boosting methods) to build our model more concrete. We also plan to combine two or more models to improve our prediction and make our model more concrete. We also plan to handle missing value more properly by using regression or inference-based tools using a Bayesian formalism, or decision tree induction.

REFERENCES

[1] Pandey, Gaurav, Bin Zhang, and Le Jian. "Predicting submicron air pollution indicators: a machine
learning approach." Environmental Science: Processes & Impacts 15.5 (2013): 996-1005.
[2] Athanasiadis, Ioannis N., et al. "Applying machine learning techniques on air quality data for real-time

decision support." First international NAISO symposium on information technologies in environmental
engineering (ITEE'2003), Gdansk, Poland. 2003.
[3] Ioannis N. Athanasiadis, Kostas D. Karatzas and Pericles A. Mitkas. "Classification techniques for air
quality forecasting." Fifth ECAI Workshop on Binding Environmental Sciences and Artificial
Intelligence, 17th European Conference on Artificial Intelligence, Riva del Garda, Italy, August 2006.
[4] M. Caselli & L. Trizio & G. de Gennaro & P. Ielpo. "A Simple Feedforward Neural Network for the
PM10 Forecasting: Comparison with a Radial Basis Function Network and a Multivariate Linear
Regression Model." Water Air Soil Pollut (2009) 201:365–377.
[5] S. Bordignon, C. Gaetan and F. Lisi, "Nonlinear models for ground- level ozone forecasting."
Statistical Methods and Applications, 11, 227-246, (2002).

[6] http://www.openair-project.org/