Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme	**: B.Tech**
Course Title	**: DATA VISUALIZATION**
Course Code	**: CSE3020**
Slot	**: D2**

Title: Exploratory Data Analysis and Prediction of India’s Air Quality Index

Team Members: MAHIR 20BCE1524
SANJIL K C 20BCE1855

Faculty: Dr. Joshan Athanesious J

Sign:
Date:

ABSTRACT

Examining and protecting air quality has become one of the most essential activities for the government in many industrial and urban areas today. The meteorological and traffic factors, burning of fossil fuels, and industrial parameters play significant roles in air pollution. With this increasing air pollution, we need implementing models which will record information about concentrations of air pollutants (so2, no2, etc.). The deposition of this harmful gasses in the air is affecting the quality of people's lives, especially in urban areas. Lately, many researchers began to use the Big Data Analytics approach as there are environmental sensing networks and sensor data available. In this paper, machine learning techniques are used to predict the concentration of SO_2, NO_2 in the environment. From the Machine Learning algorithms, we will come to know which algorithm gives us the best result to forecast the level of pollutants. This can be applied in real time data and the pollutants in each places be calculated. From this the government can implement policies related to the problems and the pollutant faced by that place

CONTENTS

Abstract ii

1 Introduction 4

1.1 Objective and goal of the project……………………………4

1.2 Problem Statement…………………………………………..5

1.3 Motivation…………………………………………………...5

1.4 Challenges…………………………………………………...6

2 Literature Survey…………………………………………………………7

3 Materials and Methods 9

3.1 Model……………………………………………………………...9

3.2 Dataset……………………………………………………………11

3.3 Architecture and Explanation ………………………………….11

4. Proposed Work 13

5 Results & Discussion 14

5.1 Result……………………………………………………………...14

5.2 Comparison……………………………………………………..25

5.3 Discussion…………………………………………………………25

6 Conclusion and Future Work……………………………………….26

7 References…………………………………………………………..27

8 Appendix……………………………………………………………28

1. Introduction

1.1 **Objective and goal of the project**

In developing countries like India, the rapid increase in population and economic upswing in cities have led to environmental problems such as air pollution, water pollution, noise pollution and many more. Air pollution has a direct impact on human health. There has been increased public awareness about the same in our country. Global warming, acid rains, increase in the number of asthma patients are some of the long-term consequences of air pollution. Precise air quality forecasting and thence producing proper control measures can reduce the effect of maximal pollution on the humans and biosphere as well. Accurate forecasting helps people plan, decreasing the effects on health and the costs associated. If people are aware of variations in the quality of the air they breathe, the effect of pollutants on health as well as concentrations likely to cause adverse effects and actions to curtail pollution.

Similar to forecasting weather, there are models to predict levels of air pollution and air quality. There are many forecast models that require more complexity than weather forecast models. These models are mathematical simulations of how airborne pollutants disperse in the air. Machine learning algorithms can help in predicting the AQI. Linear regression, LASSO regression, ridge regression and Random Forest for regression were used to forecast the AQI.
Sometimes it is convenient for media platforms like news reporters and forecast magazines to print the required content in a simply understandable Language. Hence, to make this problem a classification problem, the AQI is calculated into various spans and are labeled as “Good”, “Poor”, “Moderate”, “Unhealthy”, “Very Unhealthy” and “Hazardous”.
Classification Algorithms such as Logistic Regression, Random Forest Classifier, KNN, Weighted KNN, Ridge Classification, AdaBoost Classifier and XGBoost Classifier are used on the dataset. Almost all of the models have performed extremely well, the results have been shown later.

1.2 Problem Statement

The ever-increasing population all over the world is now a major concern to each and every government in the World. The increasing population brings in pollution as well.
Due to the various things' humans do, like smoking cigarettes, lighting bonfires, bursting crackers, using vehicles that let out harmful and toxic gases, pollutants in the air are rising to a very dangerous level.
Thus, the problem that we are dealing with right now is to know which state is contributing to the maximum pollution and suppressing the number of pollutants that get thrown out in the atmosphere. We all know that if you are doing a big job, then we must focus on dividing the job into small bits and distributing the work to various parties with the same goal in mind. This will allow efficient progress and faster results.
Thus, in our project we have gathered a dataset which contains the measurements of all the harmful pollutants that contribute to the pollution that are listed state wise and we are trying to predict the value of the air quality index and the air quality index range based on the measurements.

1.3 Motivation

The introduction of dangerous or excessive amounts of specific compounds into the atmosphere is the main source of air pollution. These substances include gases, liquid droplets, and solid particles. Primary and secondary pollutants are the two types of air pollutants. Primary pollutants are those that are directly emitted into the atmosphere from their source. Primary air pollutants can come from both natural and man-made sources, such as burning fossil fuels, leaking gas from appliances, and volcanic eruptions and sandstorms. Sulphur dioxide (SO2), nitrogen oxides (NOx), particulate matter (PM), and carbon monoxide are examples of primary pollutants (CO). When primary pollutants interact chemically or physically, secondary pollutants are created in the atmosphere. Photochemical oxidants and secondary particulate matter are examples of secondary pollutants.
Criteria pollutants are the most abundant pollutants and they correspond to the most common health risks. These include SO2, lead, NO2, ground-level ozone (O3), and PM. It has been shown that there is a link between brief exposure to these pollutants and health problems like inflamed respiratory tract in healthy individuals, increased respiratory symptoms in asthmatics, trouble meeting high oxygen demands during exercise, and critical respiratory situations,

especially in children and the elderly. Standards for acceptable air quality levels have been established by national organisations such as the EPA, EU, and many others. The levels of the air's criteria pollutants are shown by the air quality index (AQI). The highest AQI reading for each of the individual criterion pollutants makes up the total AQI. The health concerns connected to exposure to a specific air quality are also indicated by AQI levels. These medical symptoms may appear soon after being exposed to contaminated air or they may develop over time. Depending on the age and health status of the specific person being exposed, these symptoms may also differ. To prevent further rises in air pollution by on-demand pollution control systems or an emergency response, it is crucial that we have a system to foresee increases in air pollution levels. As a result, AQI would be easier to regulate to meet the demands of the population as a whole.

1.4 **Challenges**

The most prominent challenge before starting the project was the fact that we needed a very trustworthy source of data. Choosing a data set which had the least contributions to it, was the most challenging part. Our goal in mind was to stick to a very real-life critical problem. Thus, we chose air pollution. The next challenge was the data pre-processing. The attributes were very detailed, but feature engineering was very difficult on them. We had to pinpoint and identify the data that would be very beneficial for the prediction of the air quality index. Once identified, we needed to engineer them and convert them to appropriate data types that would be optimal for the models used.

Lastly, it was noted that there were a lot of missing values in the data. Hence, data amputation and handling the missing values was a very critical thing. Strategy that we followed while imputing the values of SO2, NO2 and other pollutants was that we used the mean of the pollutants filtered state wise and imputed them according to their respective state.

2. Literature Survey

Previous studies show the need to implement efficient air quality monitoring models which collect information about the concentration of air pollutants and provide assessment of air pollution in each area [3][2]. The aim of this research paper is to investigate various air quality forecasting methods.

In recent years, there has been a lot of research done on the impact of air pollution on health [4]. Increases in mortality and hospital admissions from respiratory and cardiovascular illness have been linked to exposure to pollutants including ozone and airborne particulate matter.

The impact of variations in air quality on human health has been studied through short- and long-term epidemiological studies as well as sporadic air pollution episodes like the infamous London fog in 1952. The link between air pollution and higher mortality and hospital admissions is a recurring finding [6].

Both short-term research that link daily changes in air pollution and health and long-term studies that have tracked cohorts of exposed people over time have discovered these impacts. Consequences have been observed at extremely low exposure levels, and it is uncertain whether particulate matter and ozone have a threshold concentration below which no adverse health effects are predicted [4].

There have been extensive studies conducted right from 1900s after recognizing the dire effects of the poor quality of air in the environment of an individual. Earlier studies were based on association with the direct relationship between the nature of a patient's symptoms and the carbon pollutants from sources like gasoline and coal. From national consensus for fog control and smoke abatement in 1900-1970s, global AQI standards have been set and also consideration for the future generations were being put in place with amendments for the current state of air quality and also to prevent further degradation of the air quality [7][8].

The nature of air is impacted by multi-faceted elements including area, time, and unsure factors. Past investigates show to think about elements, for example, of air contaminations [1] like NO2, CO, Ground level O3, SO2, PM2.5(particulate matter with diameter of 2.5*(10^-6) m), PM10, and meteorological data [3] like temperature, pressure, humidity, wind speed, wind direction.

The proposed algorithms in the current researches include ML techniques like Decision-Trees, Multiple Linear Regression (MLR), Multilayer Perceptron Artificial NN, Decision-Trees before ANN, random forest, back propagation and more [1][3][5].8].

Methods used for evaluating the predictive models (calculation of accuracies) were Root Mean Square Error, Normalized Root Mean Square Error, Mean Absolute Error, Symmetric Mean Absolute Percentage Error and Pearson correlation coefficient [1][5]. The results showed that compared to other methods, SLI-ESN (hybrid Scalable Link Interface-Echo state networks) performed better results [1].

It can be noted that most extensive research has been done in China which is leading this kind of works with 26 papers, followed by nations like Italy, Spain, USA, and more [1].

Given the high cost of further measures to reduce air pollution, and the many new findings which suggest that health effects can be seen at ever lower concentrations, the health effects of air pollution will need to receive much scientific and regulatory interest for years to come [4].

therefore, we conclude that

-Rather than utilizing straightforward AI strategies. Presently, the authors apply advanced and sophisticated techniques like gradient boost algorithms, random forest, Neural networks, back propagation [1][2][5].

-China was the leading main nation as far as such investigations are considered [3].

-Particulate matter with measurement equivalent to 2.5 micrometers was the fundamental expectation target [1].

-Because of the number of deaths due to pollutants like SO2, PM10 and PM2.5, health effects from air pollution have been estimated to be higher than effects from a long list of other environmental factors [4][6].

-The applications of air quality forecasting methods could be applied for air quality management purposes and protect public health.

3 Materials and Methods

3.1 Models

- **Linear Regression**
  In statistics, linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression.

```
#Linear regression

model=LinearRegression()

model.fit(X_train,y_train)
```

- **Random Forest Regression**
  Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

```
#Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

rf=RandomForestRegressor()

rf.fit(X_train,y_train)

rf_train_preds=rf.predict(X_train)
#predicting on test
rf_test_preds=rf.predict(X_test)
```

- **Logistic Regression**
  Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

```
#LOGISTIC REGRESSION

log_reg = LogisticRegression()
log_reg.fit(X_train2, Y_train2)


logreg_train_preds = log_reg.predict(X_train2)
print("Model accuracy on train is: ", accuracy_score(Y_train2,
logreg_train_preds))


logreg_test_preds = log_reg.predict(X_test2)
print("Model accuracy on test is: ", accuracy_score(Y_test2,
logreg_test_preds))

# Kappa Score.
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,logreg_test_preds))
```

- **Random Forest Classifier**
  The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

```
#RANDOMFOREST CLASSIFIER


RF=RandomForestClassifier()
RF.fit(X_train2,Y_train2)

RF_train_preds = RF.predict(X_train2)
print("Model accuracy on train is: ", accuracy_score(Y_train2,
RF_train_preds))

RF_test_preds = RF.predict(X_test2)
print("Model accuracy on test is: ", accuracy_score(Y_test2, RF_test_preds))

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,RF_test_preds))
```

- **KNN**
  K-Nearest Neighbors Algorithm. The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

```
#KNN

KNN = KNeighborsClassifier()
KNN.fit(X_train2,Y_train2)

knn_tr_preds = KNN.predict(X_train2)
print("Model accuracy on train is: ", accuracy_score(Y_train2, knn_tr_preds))

knn_test_preds = KNN.predict(X_test2)
print("Model accuracy on test is: ", accuracy_score(Y_test2, knn_test_preds))

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,knn_test_preds))
```

- **ANN (Artificial Neural Networks):**
  Artificial neural network (ANN) is a computational model that consists of several processing elements that receive inputs and deliver outputs based on their predefined activation functions.

```
#Neural Netwok (ANN) (Regression)
```

3.2 **Dataset**

The data set we used was obtained from Kaggle.
Link – https://www.kaggle.com/datasets/shrutibhargava94/india-air-quality-data

About the Dataset:
Using the above-mentioned dataset, India's air pollution levels can be explored at a more granular scale.
This data is combined (across the years and states) and largely clean version of the Historical Daily Ambient Air Quality Data released by the Ministry of Environment and Forests and Central Pollution Control Board of India under the National Data Sharing and Accessibility Policy (NDSAP).

3.3 **Architecture and Explanation**

- Data Analysis is going to be done using two approaches..
  - State-wise Data Visualization and Analysis.
  - Air Quality determinant sample wise Visualization and Analysis.

The State-wise Data Visualization will give results like the maximum polluted state and minimum polluted state, state containing highest concentration of SO2, NO2, etc.

Air Quality Determinant Samples included in the dataset are: SO2, NO2, RSPM, P2.5.

From these measurements we have calculated indexes of each of the samples and then from those have derived a global Air Quality Index of that particular record. To classify the index or to make it more reader friendly, we have calculated an AQI Range i.e Good, Healthy, etc.

After reviewing all of the attributes, it was determined that stn_code, location_monitoring_station, and sampling_date were unnecessary. As a result, were removed.
When the last three rows of the dataset are examined, it is discovered that they lack any information that would aid in analysis, so they are removed.

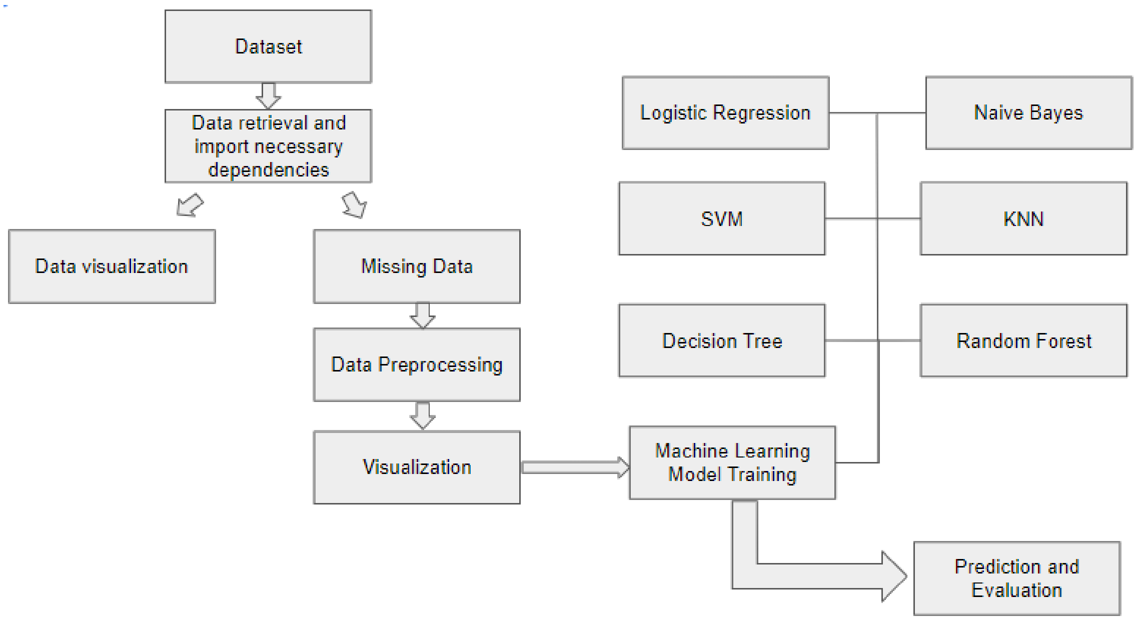
Categorical Data –
Missing values in 'agency' attribute were replaced by 'unknown'.
In the 'date' attribute, it is observed that it is a repetitive attribute in small range so, we felt replacing NA values in with last observed date would be the best action.
Similarly, in 'type' attribute too, we replaced NA values with previously observed type value.

Explanation

1. Retrieve the dataset from Kaggle
2. Import it into the Jupyter Notebook
3. Using appropriate commands, read the csv file
4. Do data visualization of unprocessed data
5. Do Data Preprocessing in RStudio in R
6. Do Data Visualization in RStudio using packages like graphics and ggplot
7. Construct various Machine Learning models and evaluate the predictions for predicting AQI. (Regression)
8. Construct various Machine Learning models and evaluate the predictions for predicting AQI Range. (Classification)
9. Use Artificial Neural Networks and create a sequential model using various Layers like Dense Layers and Dropout Layers.

4 Proposed Work

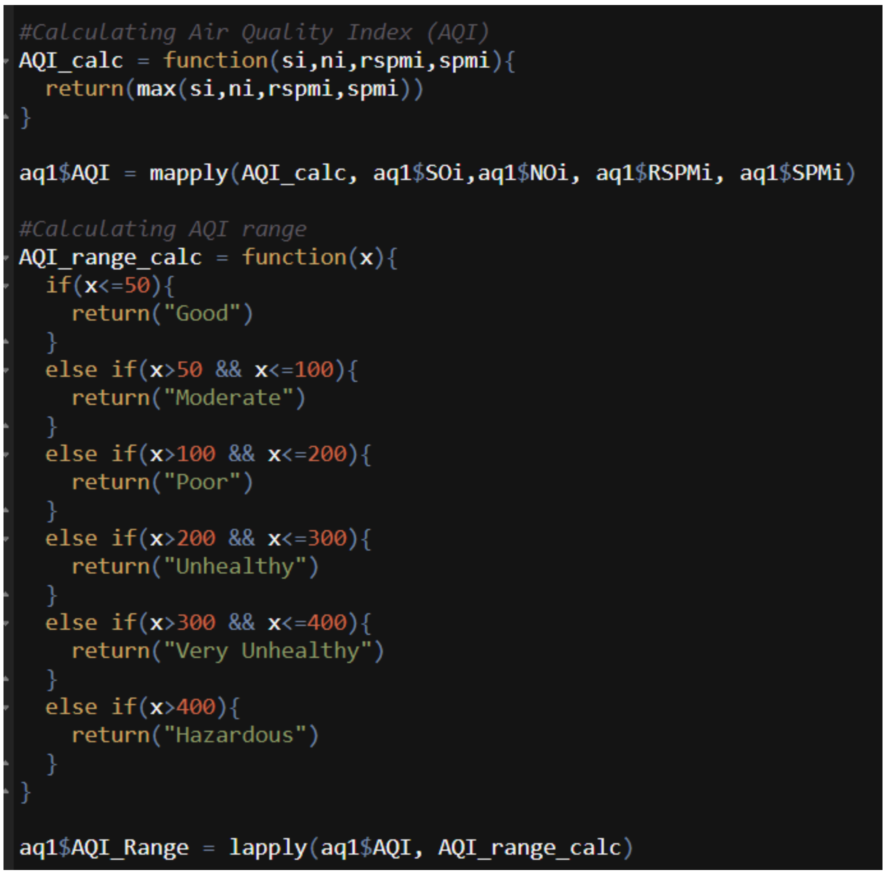
- Proposed system will consist of the full visualized version of the dataset and a number of Machine Learning Models that will be trained on the dataset using various algorithms.
- The visualization will allow the reader to judge the air quality in India statewise. The data also consists of the various measurement values. Thus we can also find out what gas is affecting the air quality the most and in which state.
- The highly severe regions can also be plotted on an Indian Map and be visualized.
- List of Modules:
    - Data Retrieval.
    - Data Visualization before handling missing values.
    - Data Preprocessing.
    - Data Visualization after preprocessing
    - Supervised Machine Learning (Regression).
    - Supervised Machine Learning (Classification).
    - Neural Network(ANN)
    - Evaluation of all the models using appropriate Metrics and plots.



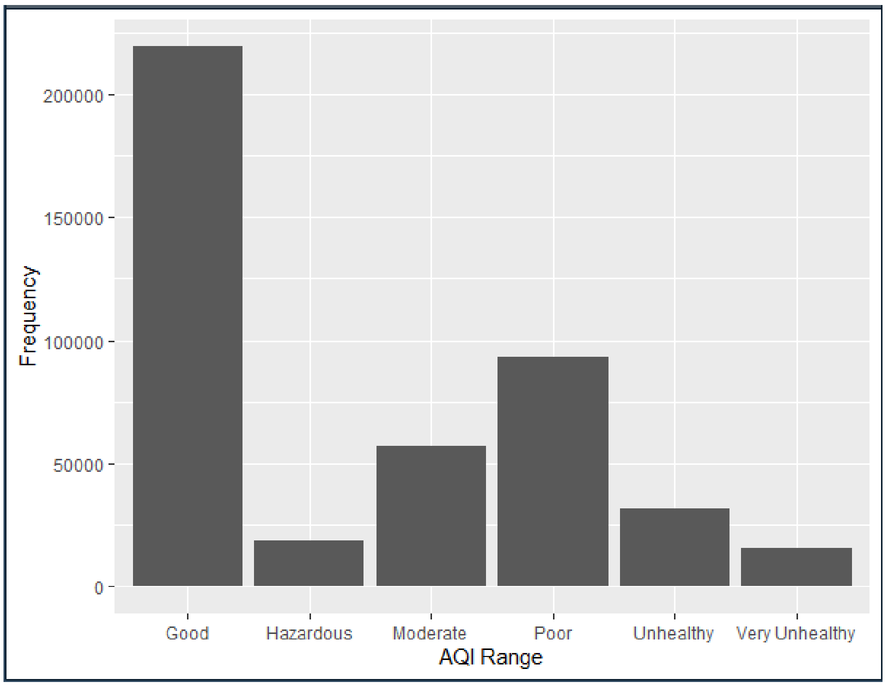
5 Result and Discussion

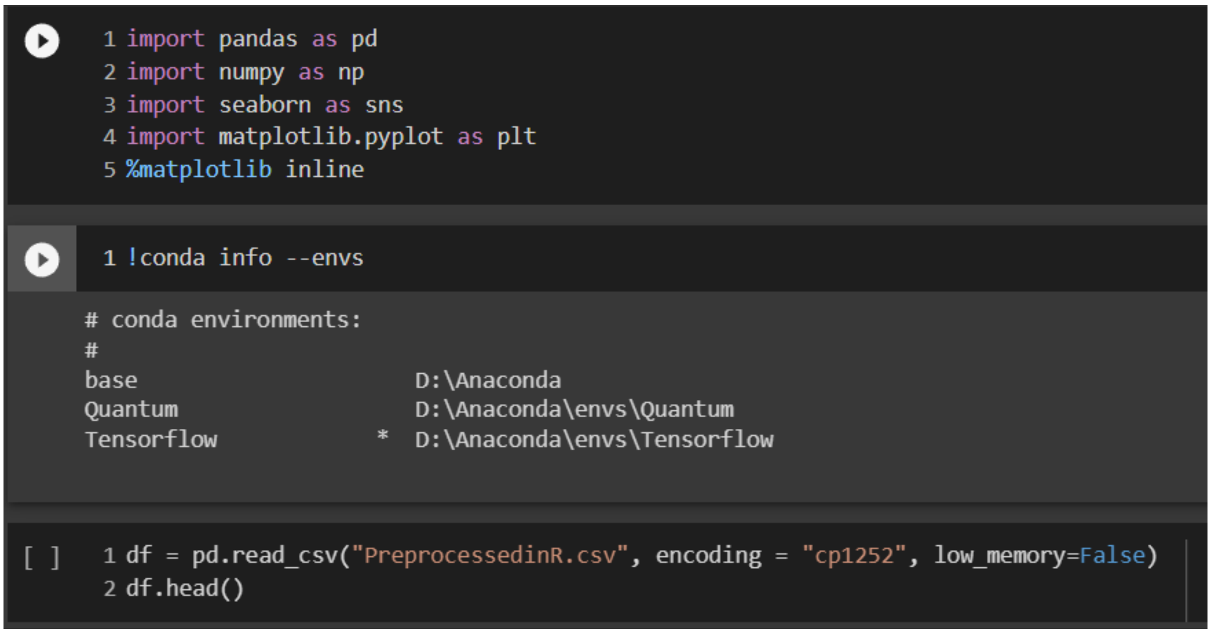
5.1 Result

1. Function to calculate AQI



```
#Calculating Air Quality Index (AQI)
AQI_calc = function(si,ni,rspmi,spmi){
  return(max(si,ni,rspmi,spmi))
}

aq1$AQI = mapply(AQI_calc, aq1$SOi,aq1$NOi, aq1$RSPMi, aq1$SPMi)

#Calculating AQI range
AQI_range_calc = function(x){
  if(x<=50){
    return("Good")
  }
  else if(x>50 && x<=100){
    return("Moderate")
  }
  else if(x>100 && x<=200){
    return("Poor")
  }
  else if(x>200 && x<=300){
    return("Unhealthy")
  }
  else if(x>300 && x<=400){
    return("Very Unhealthy")
  }
  else if(x>400){
    return("Hazardous")
  }
}

aq1$AQI_Range = lapply(aq1$AQI, AQI_range_calc)
```



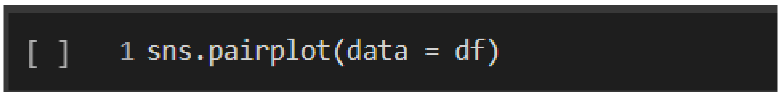


```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

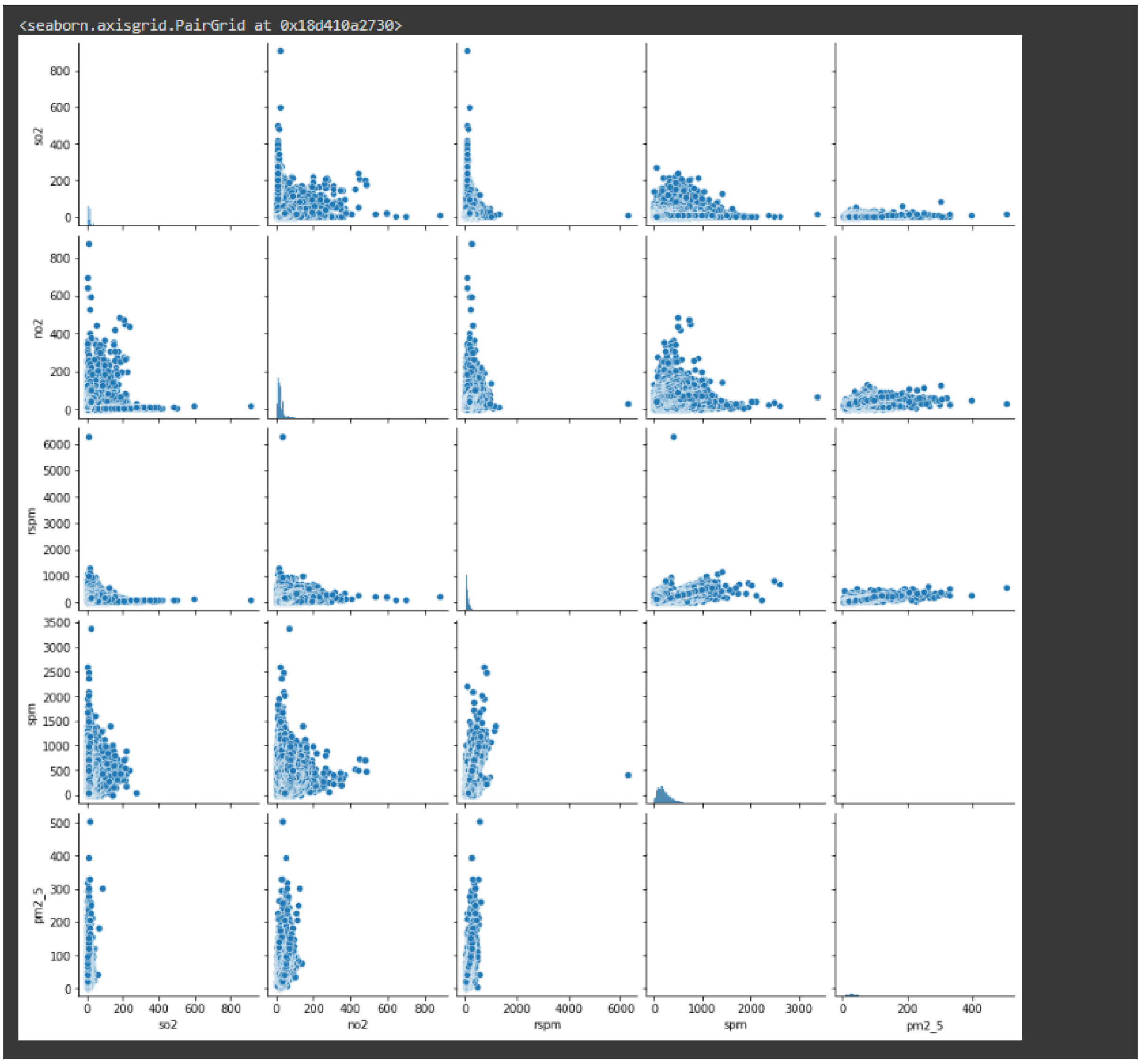
```
!conda info --envs
```

```
# conda environments:
#
base                     D:\Anaconda
Quantum                  D:\Anaconda\envs\Quantum
Tensorflow            *  D:\Anaconda\envs\Tensorflow
```

```python
df = pd.read_csv("PreprocessedinR.csv", encoding = "cp1252", low_memory=False)
df.head()
```



```python
sns.pairplot(data = df)
```



```
<seaborn.axisgrid.PairGrid at 0x18d410a2730>
```

so2
no2
rspm
spm
pm2_5

1. Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
#Linear regression

model=LinearRegression()

model.fit(X_train,y_train)
```

```
LinearRegression()
```

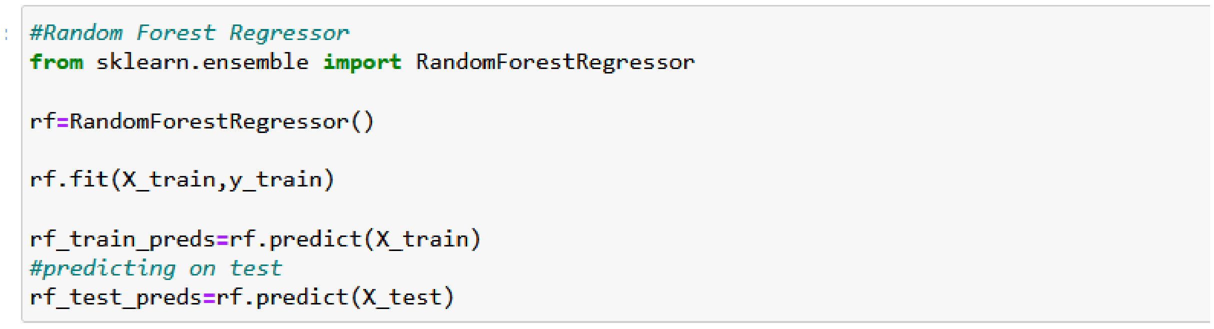
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
#predicting train
train_pred=model.predict(X_train)
#predicting on test
test_pred=model.predict(X_test)
```

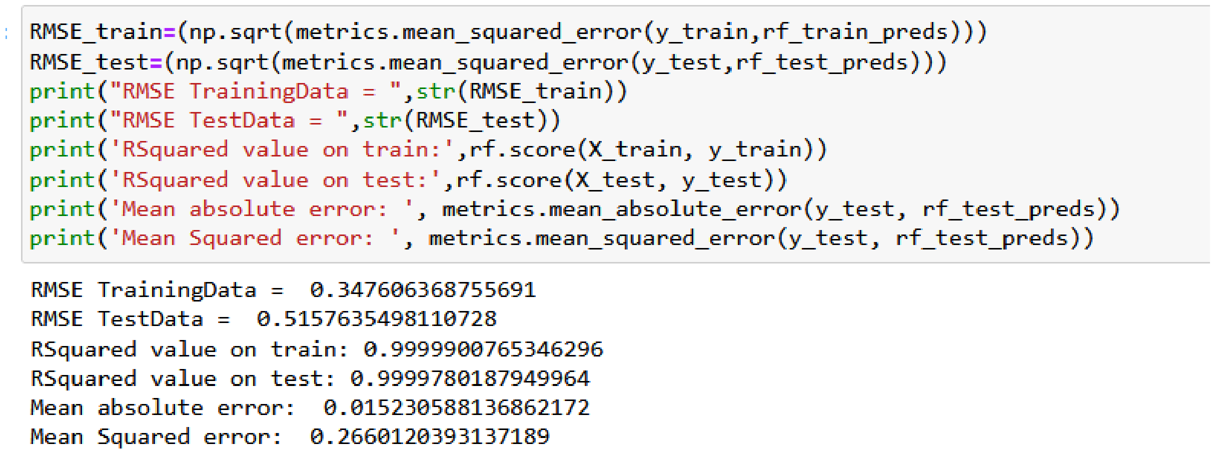
```
RMSE_train=(np.sqrt(metrics.mean_squared_error(y_train,train_pred)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(y_test,test_pred)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('RSquared value on train:',model.score(X_train, y_train))
print('RSquared value on test:',model.score(X_test, y_test))
print('Mean absolute error: ', metrics.mean_absolute_error(y_test, test_pred))
print('Mean Squared error: ', metrics.mean_squared_error(y_test, test_pred))
```

```
RMSE TrainingData =  13.355928487132466
RMSE TestData =  13.461230341980844
RSquared value on train: 0.985350052149406
RSquared value on test: 0.9850266245121643
Mean absolute error:  9.006269757689305
Mean Squared error:  181.2047223198657
```

2. RandomForest Regressor



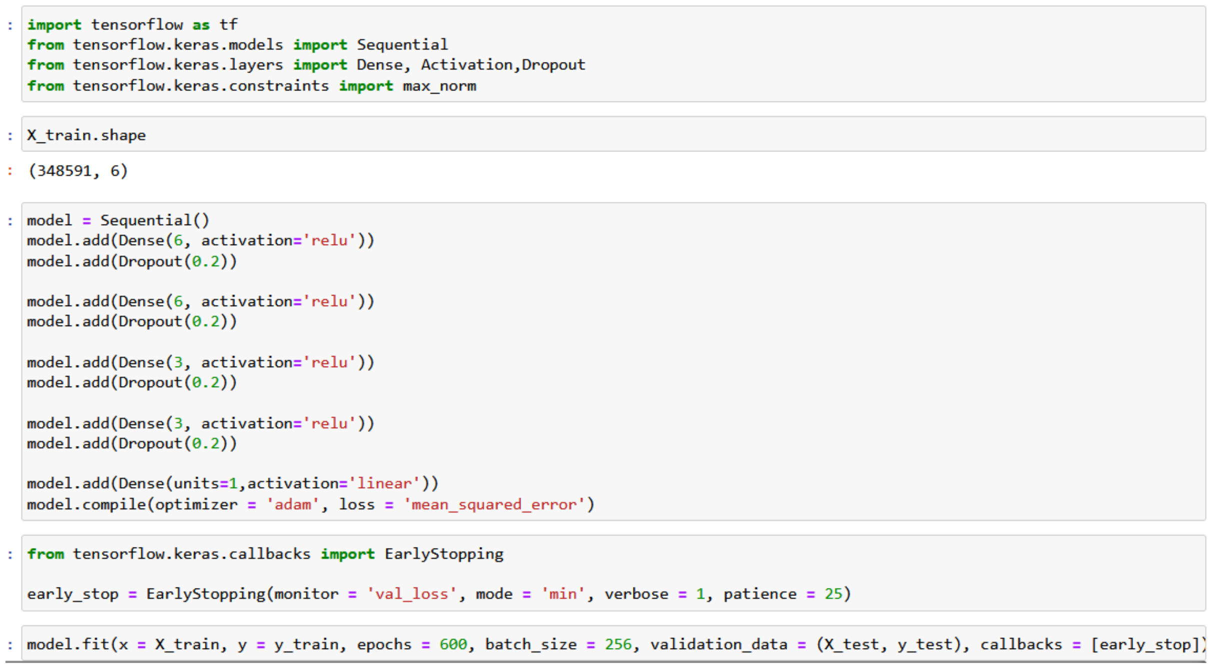
```
#Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

rf=RandomForestRegressor()

rf.fit(X_train,y_train)

rf_train_preds=rf.predict(X_train)
#predicting on test
rf_test_preds=rf.predict(X_test)
```



```
RMSE_train=(np.sqrt(metrics.mean_squared_error(y_train,rf_train_preds)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(y_test,rf_test_preds)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('RSquared value on train:',rf.score(X_train, y_train))
print('RSquared value on test:',rf.score(X_test, y_test))
print('Mean absolute error: ', metrics.mean_absolute_error(y_test, rf_test_preds))
print('Mean Squared error: ', metrics.mean_squared_error(y_test, rf_test_preds))
```

```
RMSE TrainingData =  0.347606368755691
RMSE TestData =  0.5157635498110728
RSquared value on train: 0.9999900765346296
RSquared value on test: 0.9999780187949964
Mean absolute error:  0.015230588136862172
Mean Squared error:  0.2660120393137189
```

3. ANN



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout
from tensorflow.keras.constraints import max_norm
```

```
X_train.shape
```

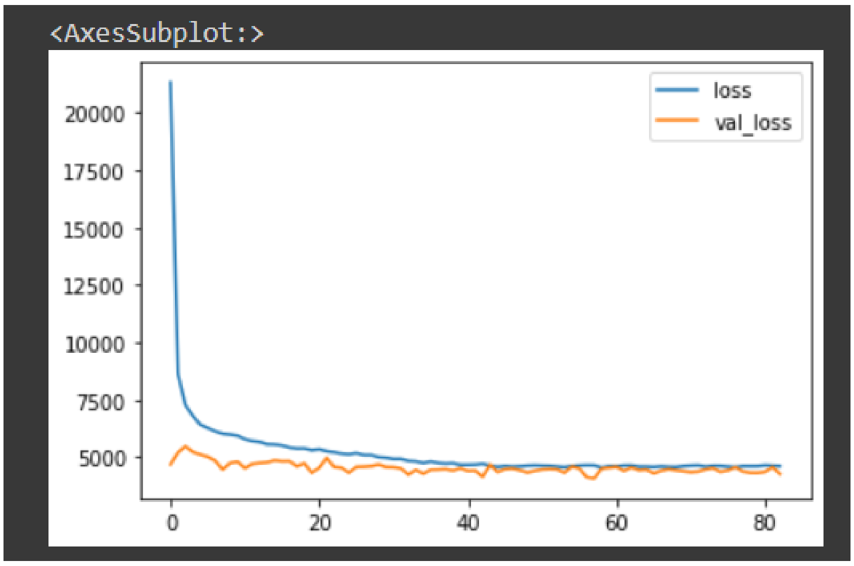
```
(348591, 6)
```

```
model = Sequential()
model.add(Dense(6, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(6, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(3, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(3, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(units=1,activation='linear'))
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 25)
```

```
model.fit(x = X_train, y = y_train, epochs = 600, batch_size = 256, validation_data = (X_test, y_test), callbacks = [early_stop]
```

<AxesSubplot:>



```
preds = model.predict(X_test)
preds_tr = model.predict(X_train)
```

```
RMSE_train=(np.sqrt(metrics.mean_squared_error(y_train,preds_tr)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(y_test,preds)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('RSquared value on train:',rf.score(X_train, y_train))
print('RSquared value on test:',rf.score(X_test, y_test))
print('Mean absolute error: ', metrics.mean_absolute_error(y_test, preds))
print('Mean Squared error: ', metrics.mean_squared_error(y_test, preds))
```

```
RMSE TrainingData =  59.91982045576715
RMSE TestData =  59.78213543878392
RSquared value on train: 0.9999900765346296
RSquared value on test: 0.9999780187949964
Mean absolute error:  44.395896896662585
Mean Squared error:  3573.903717621104
```

4. ANN (Classification)

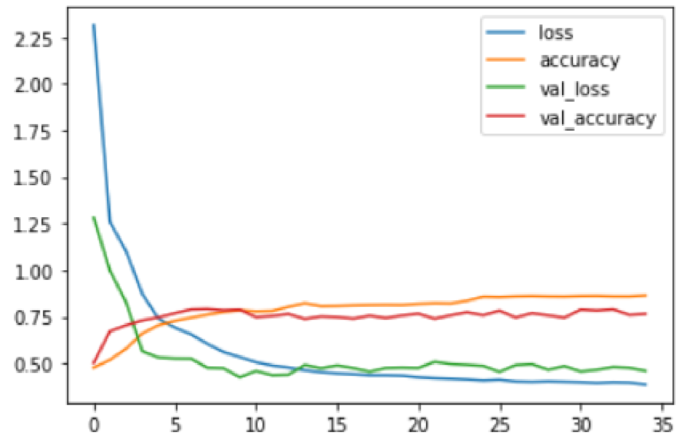
```
model2 = Sequential()
model2.add(Dense(10, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(10, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(10, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(5, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(5, activation='relu'))
model2.add(Dropout(0.2))

model2.add(Dense(units=6,activation='softmax'))
model2.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy',metrics = ['accuracy'])
```

```
print(df_new.AQI_Range.unique())
```

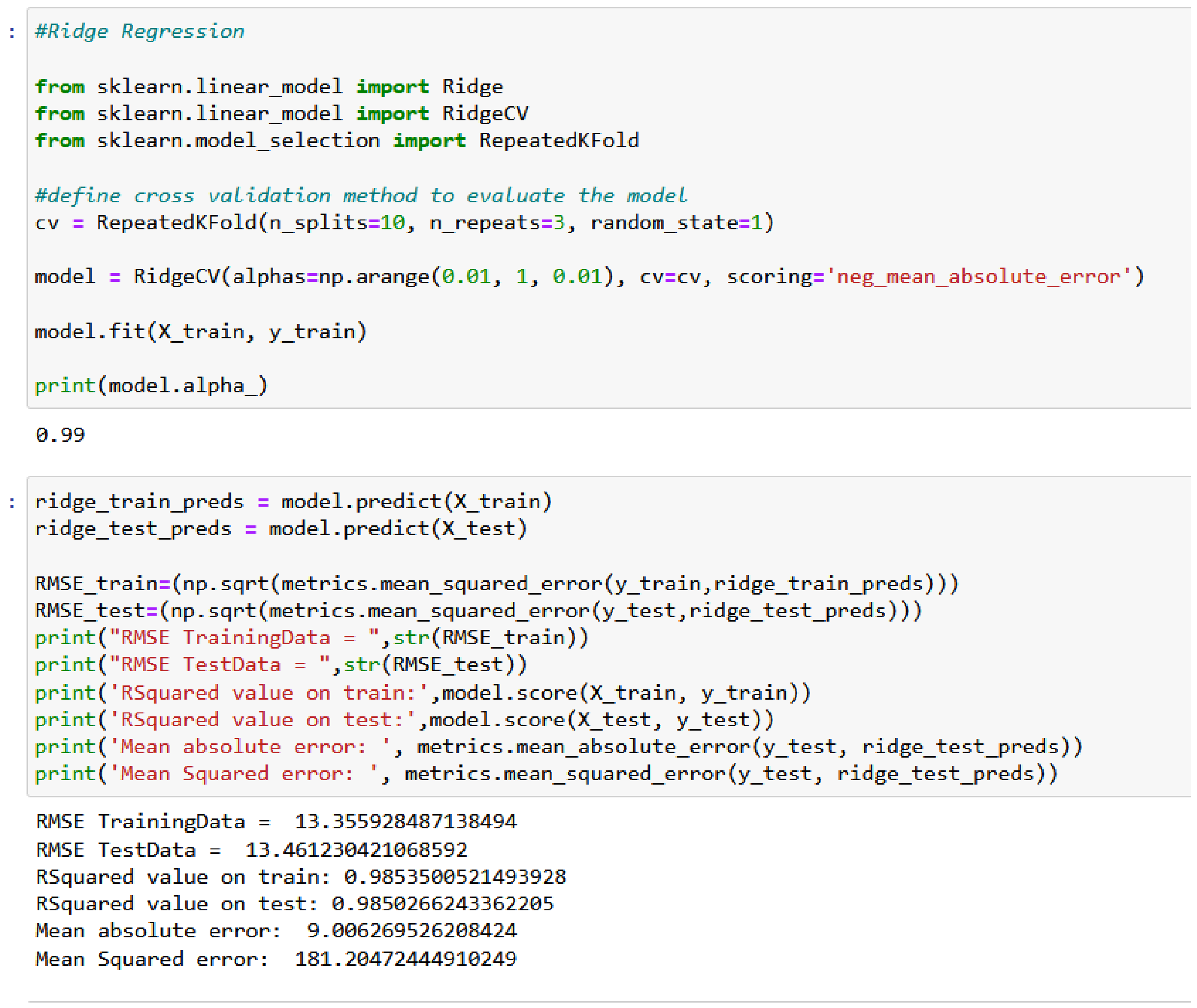
```
['Good' 'Poor' 'Moderate' 'Unhealthy' 'Very Unhealthy' 'Hazardous']
```

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 25)
```

model2.fit(x = X_train2, y = Y_train2, epochs = 600, batch_size = 256, validation_data = (X_test2, Y_test2), callbacks = [early_stop])



5. Ridge Regression



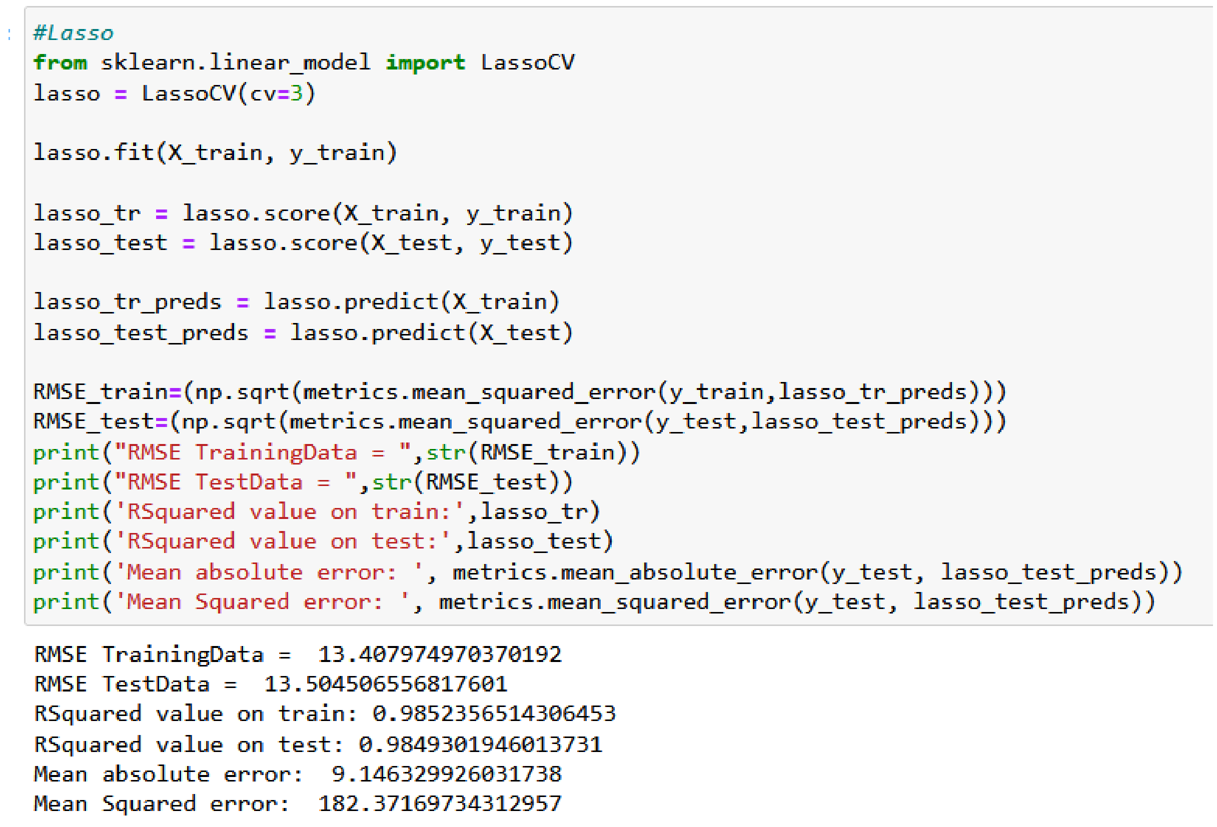
```
#Ridge Regression

from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
from sklearn.model_selection import RepeatedKFold

#define cross validation method to evaluate the model
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

model = RidgeCV(alphas=np.arange(0.01, 1, 0.01), cv=cv, scoring='neg_mean_absolute_error')

model.fit(X_train, y_train)

print(model.alpha_)
```

```
0.99
```

```
ridge_train_preds = model.predict(X_train)
ridge_test_preds = model.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(y_train,ridge_train_preds)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(y_test,ridge_test_preds)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('RSquared value on train:',model.score(X_train, y_train))
print('RSquared value on test:',model.score(X_test, y_test))
print('Mean absolute error: ', metrics.mean_absolute_error(y_test, ridge_test_preds))
print('Mean Squared error: ', metrics.mean_squared_error(y_test, ridge_test_preds))
```

```
RMSE TrainingData =  13.355928487138494
RMSE TestData =  13.461230421068592
RSquared value on train: 0.9853500521493928
RSquared value on test: 0.9850266243362205
Mean absolute error:  9.006269526208424
Mean Squared error:  181.20472444910249
```

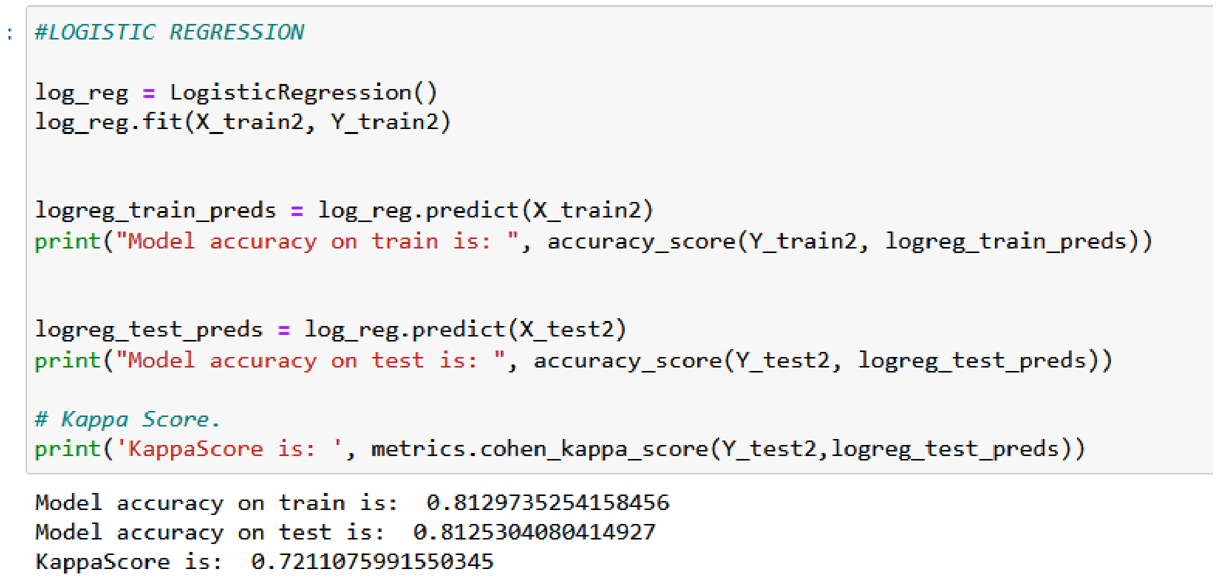
6. Lasso Regression



```
#Lasso
from sklearn.linear_model import LassoCV
lasso = LassoCV(cv=3)

lasso.fit(X_train, y_train)

lasso_tr = lasso.score(X_train, y_train)
lasso_test = lasso.score(X_test, y_test)

lasso_tr_preds = lasso.predict(X_train)
lasso_test_preds = lasso.predict(X_test)

RMSE_train=(np.sqrt(metrics.mean_squared_error(y_train,lasso_tr_preds)))
RMSE_test=(np.sqrt(metrics.mean_squared_error(y_test,lasso_test_preds)))
print("RMSE TrainingData = ",str(RMSE_train))
print("RMSE TestData = ",str(RMSE_test))
print('RSquared value on train:',lasso_tr)
print('RSquared value on test:',lasso_test)
print('Mean absolute error: ', metrics.mean_absolute_error(y_test, lasso_test_preds))
print('Mean Squared error: ', metrics.mean_squared_error(y_test, lasso_test_preds))
```

```
RMSE TrainingData =  13.407974970370192
RMSE TestData =  13.504506556817601
RSquared value on train: 0.9852356514306453
RSquared value on test: 0.9849301946013731
Mean absolute error:  9.146329926031738
Mean Squared error:  182.37169734312957
```

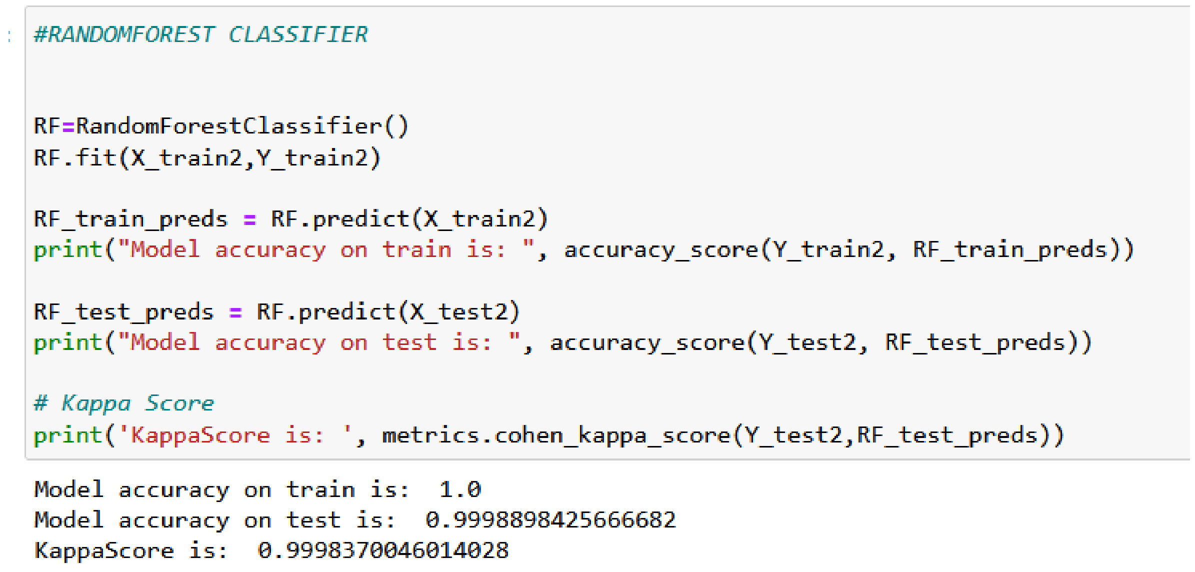
7. Logistic Regression



```
#LOGISTIC REGRESSION

log_reg = LogisticRegression()
log_reg.fit(X_train2, Y_train2)


logreg_train_preds = log_reg.predict(X_train2)
print("Model accuracy on train is: ", accuracy_score(Y_train2, logreg_train_preds))


logreg_test_preds = log_reg.predict(X_test2)
print("Model accuracy on test is: ", accuracy_score(Y_test2, logreg_test_preds))

# Kappa Score.
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,logreg_test_preds))
```

```
Model accuracy on train is:  0.8129735254158456
Model accuracy on test is:  0.8125304080414927
KappaScore is:  0.7211075991550345
```

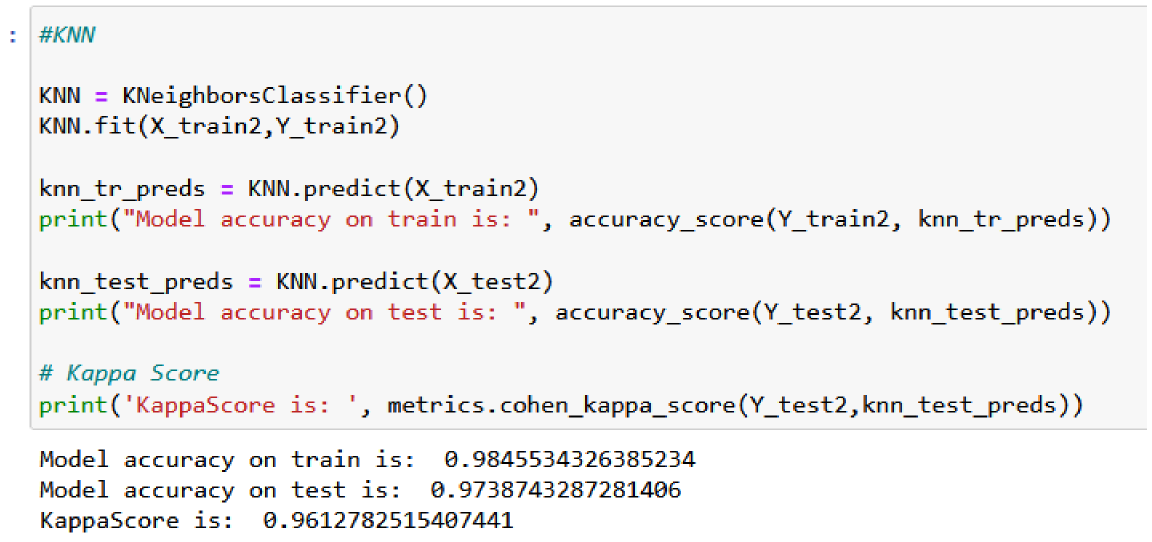
8. Random Forest Classifier



```
#RANDOMFOREST CLASSIFIER


RF=RandomForestClassifier()
RF.fit(X_train2,Y_train2)

RF_train_preds = RF.predict(X_train2)
print("Model accuracy on train is: ", accuracy_score(Y_train2, RF_train_preds))

RF_test_preds = RF.predict(X_test2)
print("Model accuracy on test is: ", accuracy_score(Y_test2, RF_test_preds))

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,RF_test_preds))
```

```
Model accuracy on train is:  1.0
Model accuracy on test is:  0.9998898425666682
KappaScore is:  0.9998370046014028
```

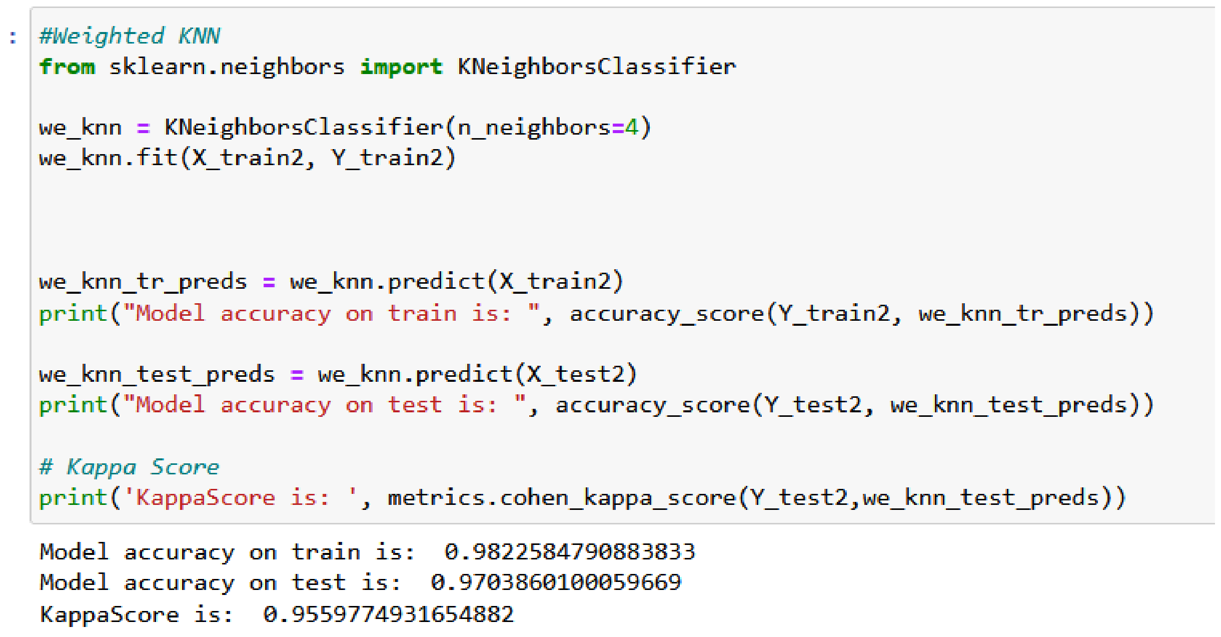
9. KNN



```
#KNN

KNN = KNeighborsClassifier()
KNN.fit(X_train2,Y_train2)

knn_tr_preds = KNN.predict(X_train2)
print("Model accuracy on train is: ", accuracy_score(Y_train2, knn_tr_preds))

knn_test_preds = KNN.predict(X_test2)
print("Model accuracy on test is: ", accuracy_score(Y_test2, knn_test_preds))

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,knn_test_preds))
```

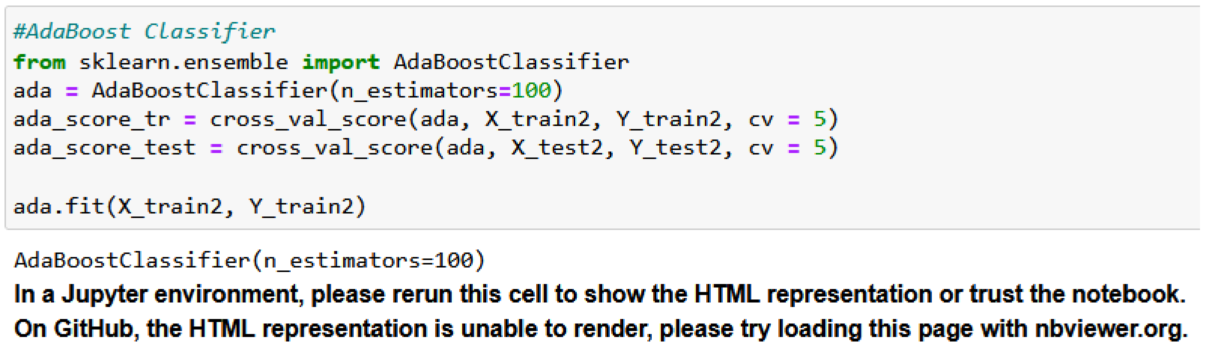
```
Model accuracy on train is:  0.9845534326385234
Model accuracy on test is:  0.9738743287281406
KappaScore is:  0.9612782515407441
```

10. Weighted KNN

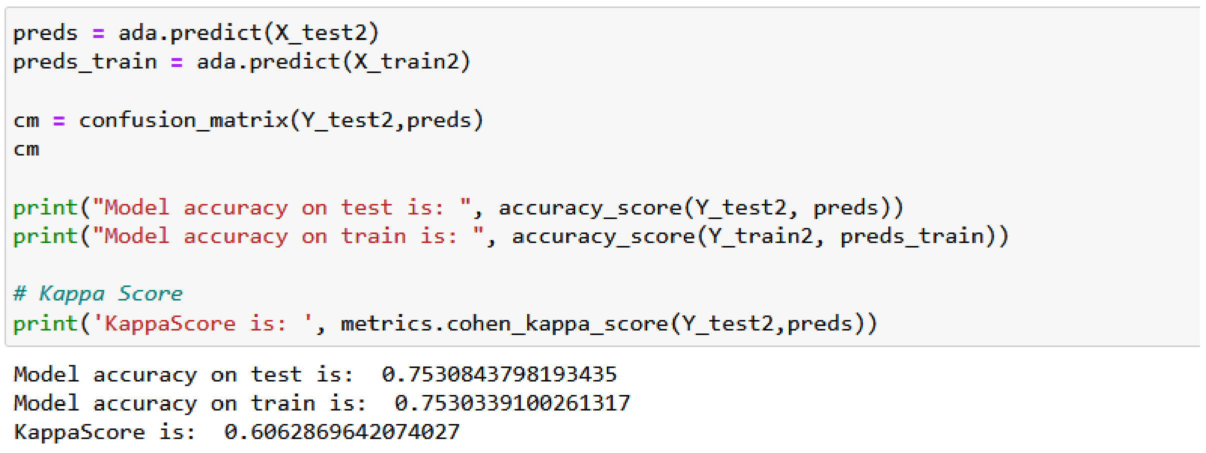

```
: #Weighted KNN
  from sklearn.neighbors import KNeighborsClassifier

  we_knn = KNeighborsClassifier(n_neighbors=4)
  we_knn.fit(X_train2, Y_train2)



  we_knn_tr_preds = we_knn.predict(X_train2)
  print("Model accuracy on train is: ", accuracy_score(Y_train2, we_knn_tr_preds))

  we_knn_test_preds = we_knn.predict(X_test2)
  print("Model accuracy on test is: ", accuracy_score(Y_test2, we_knn_test_preds))

  # Kappa Score
  print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,we_knn_test_preds))
```

```
Model accuracy on train is:  0.9822584790883833
Model accuracy on test is:  0.9703860100059669
KappaScore is:  0.9559774931654882
```

11. AdaBoost Classifier

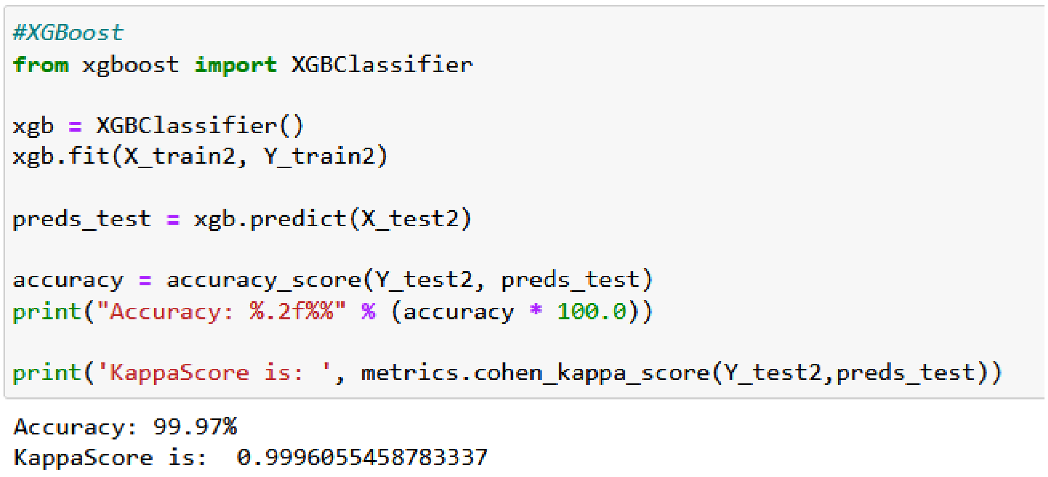

```
#AdaBoost Classifier
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(n_estimators=100)
ada_score_tr = cross_val_score(ada, X_train2, Y_train2, cv = 5)
ada_score_test = cross_val_score(ada, X_test2, Y_test2, cv = 5)

ada.fit(X_train2, Y_train2)
```

```
AdaBoostClassifier(n_estimators=100)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.


```
preds = ada.predict(X_test2)
preds_train = ada.predict(X_train2)

cm = confusion_matrix(Y_test2,preds)
cm

print("Model accuracy on test is: ", accuracy_score(Y_test2, preds))
print("Model accuracy on train is: ", accuracy_score(Y_train2, preds_train))

# Kappa Score
print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,preds))
```

```
Model accuracy on test is:  0.7530843798193435
Model accuracy on train is:  0.7530339100261317
KappaScore is:  0.6062869642074027
```

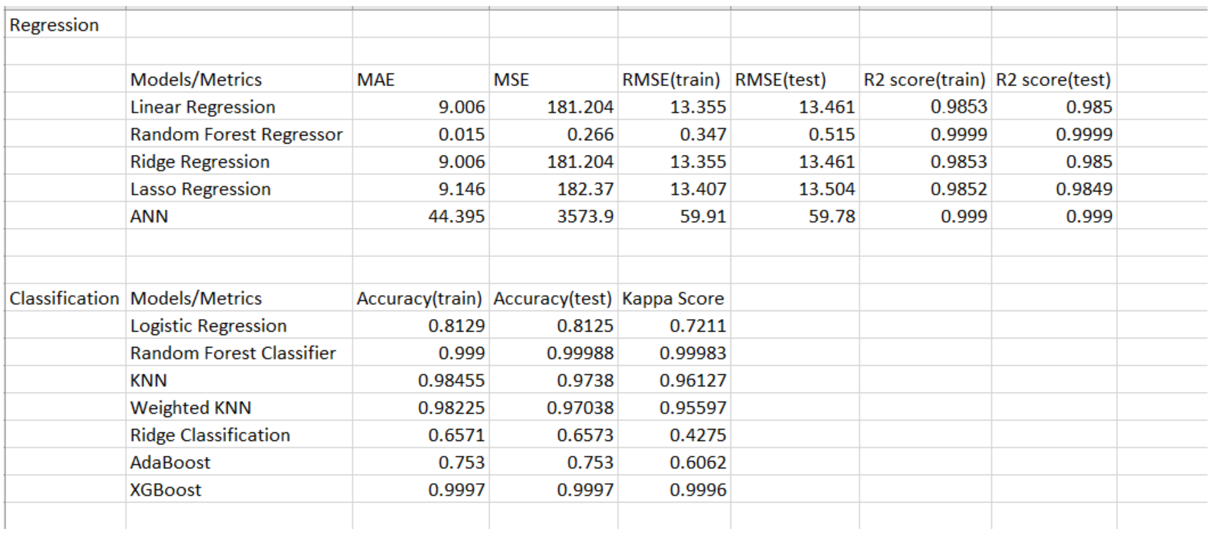
12. XGBoost Classifier



```
#XGBoost
from xgboost import XGBClassifier

xgb = XGBClassifier()
xgb.fit(X_train2, Y_train2)

preds_test = xgb.predict(X_test2)

accuracy = accuracy_score(Y_test2, preds_test)
print("Accuracy: %.2f%%" % (accuracy * 100.0))

print('KappaScore is: ', metrics.cohen_kappa_score(Y_test2,preds_test))
```

```
Accuracy: 99.97%
KappaScore is:  0.9996055458783337
```

5.2 Model Comparison:



Regression							
	Models/Metrics	MAE	MSE	RMSE(train)	RMSE(test)	R2 score(train)	R2 score(test)
	Linear Regression	9.006	181.204	13.355	13.461	0.9853	0.985
	Random Forest Regressor	0.015	0.266	0.347	0.515	0.9999	0.9999
	Ridge Regression	9.006	181.204	13.355	13.461	0.9853	0.985
	Lasso Regression	9.146	182.37	13.407	13.504	0.9852	0.9849
	ANN	44.395	3573.9	59.91	59.78	0.999	0.999
Classification	Models/Metrics	Accuracy(train)	Accuracy(test)	Kappa Score			
	Logistic Regression	0.8129	0.8125	0.7211			
	Random Forest Classifier	0.999	0.99988	0.99983			
	KNN	0.98455	0.9738	0.96127			
	Weighted KNN	0.98225	0.97038	0.95597			
	Ridge Classification	0.6571	0.6573	0.4275			
	AdaBoost	0.753	0.753	0.6062			
	XGBoost	0.9997	0.9997	0.9996			

5.3 Discussion:

From the above table we can deduce that for predicting AQI, all the algorithms work very well with the dataset. It maybe because the data is structured very well with very good records. However, if that even was not the case, the preprocessing done on the dataset fixes the missing values and makes the data optimal for passing to any model. For Prediction of AQI Range. We can see that Ridge Classification performs very badly. Even though ridge algorithm is made for regression and classification problems, ridge regression outperforms ridge classification.

Following Ridge, AdaBoost surprisingly underperforms without any hyperparameter tuning. Logistic Regression performs well than AdaBoost by having an accuracy 81%. All the other models have very good accuracy which is above 95% and is thus recommended to use for this dataset.

6. Conclusion and Future Work

The future work that remains for this dataset might be to see why algorithms like AdaBoost, Logistic Regression, Ridge Classification are not giving that good of an accuracy compared to the other models that are applied.
By using Hyper Parameter Tuning, one can run a series of experiments and based on Trial and Error, one can find out the best fitting parameters so that they also fit to the data very well.

The conclusion of our work is that:
The task of forecasting pollutant levels is inherently hard because of the volatile and dynamic nature of the data and its variability in space and time. However, the task of forecasting pollutant levels has been increasing in importance due to the effects of pollution on the population and the environment. In this work we have use algorithmic techniques like Linear Regression, Random Forest Regression, Logistic Regression, Random Forest Classifier, KNN, ANN (Artificial Neural Networks) for forecasting levels of pollutants like NO2, SO2, PM2.5 and Air Quality Index (AQI), using publicly available data for India.

GitHub Link For Project:

https://github.com/sanjil2/Exploratory-Data-Analysis-and-Prediction-of-India-s-Air-Quality-Index-

7. REFERENCES

The data set we used was obtained from Kaggle.

Link – https://www.kaggle.com/datasets/shrutibhargava94/india-air-quality-data

1. Ditsuhi Iskandaryan, Francisco Ramos and Sergio Trilles," Air Quality Prediction in Smart Cities Using Machine Learning Technologies Based on Sensor Data: A Review",Institute of New Imaging Technologies (INIT), Universitat Jaume I, Spain,Received: 07 February 2020; Accepted: 25 March 2020; Published: 1 April 2020

2 Mrs. A. Gnana Soundari MTech, (PhD) Associate Professor,Mrs. J. Gnana Jeslin M.E, (PhD) Assistant Professor,Akshaya A.C's,"INDIAN AIR QUALITY PREDICTION AND ANALYSIS USING MACHINE LEARNING",Jeppiaar Engineering College

3. Gaganjot Kaur Kang, Jerry Zeyu Gao, Sen Chiao, Shengqiang Lu, and Approaches",Gang Xie,"Air Quality Prediction: Big Data and Machine Learning

4. Brunekreef B, Holgate ST. Air pollution and health. Lancet. 2002 Oct 19;360(9341):1233-42. doi: 10.1016/S0140-6736(02)11274-8. PMID: 12401268.

5. Jamal A, Nabizadeh Nodehi R. PREDICTING AIR QUALITY INDEX BASED ON METEOROLOGICAL DATA: A COMPARISON OF REGRESSION ANALYSIS, ARTIFICIAL NEURAL NETWORKS AND DECISION TREE. JAPH. 2017;2(1).

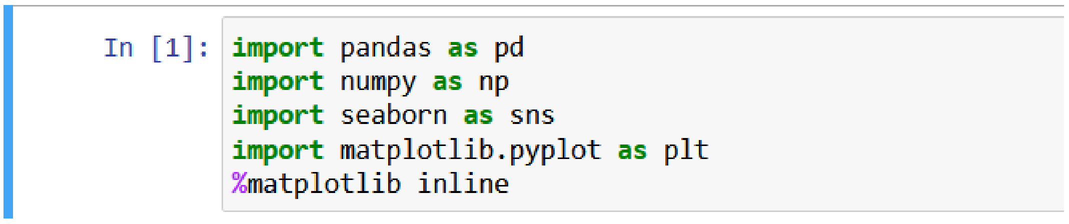
6. Marilena Kampa and Elias Castanas',"Human health effects of air pollution"Laboratory of Experimental Endocrinology, University of Crete, School of Medicine, P.O. Box 2208, Heraklion, 71003, Greece. Received 4 June 2007, Accepted 10 June 2007, Available online 23 July 2007.

7 Yang Zhang, Marc Bocquet, Vivien Mallet, Christian Seigneur, Alexander Baklanov,Real-time air quality forecasting, part I: History, techniques, and current status, Atmospheric Environment,Volume 60,2012,Pages 632-655,ISSN 1352-2310

1. John Bachmann (2007) Will the Circle Be Unbroken: A History of the U.S. National Ambient Air Quality Standards, Journal of the Air & Waste Management Association, 57:6, 652-697, DOI: 10.3155/1047-3289.57.6.652

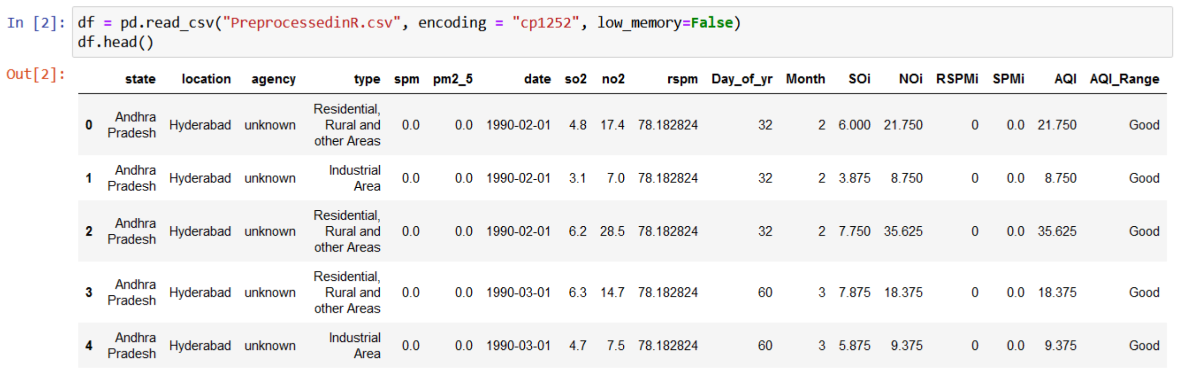
8. Appendix

1.Import Libraries



```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

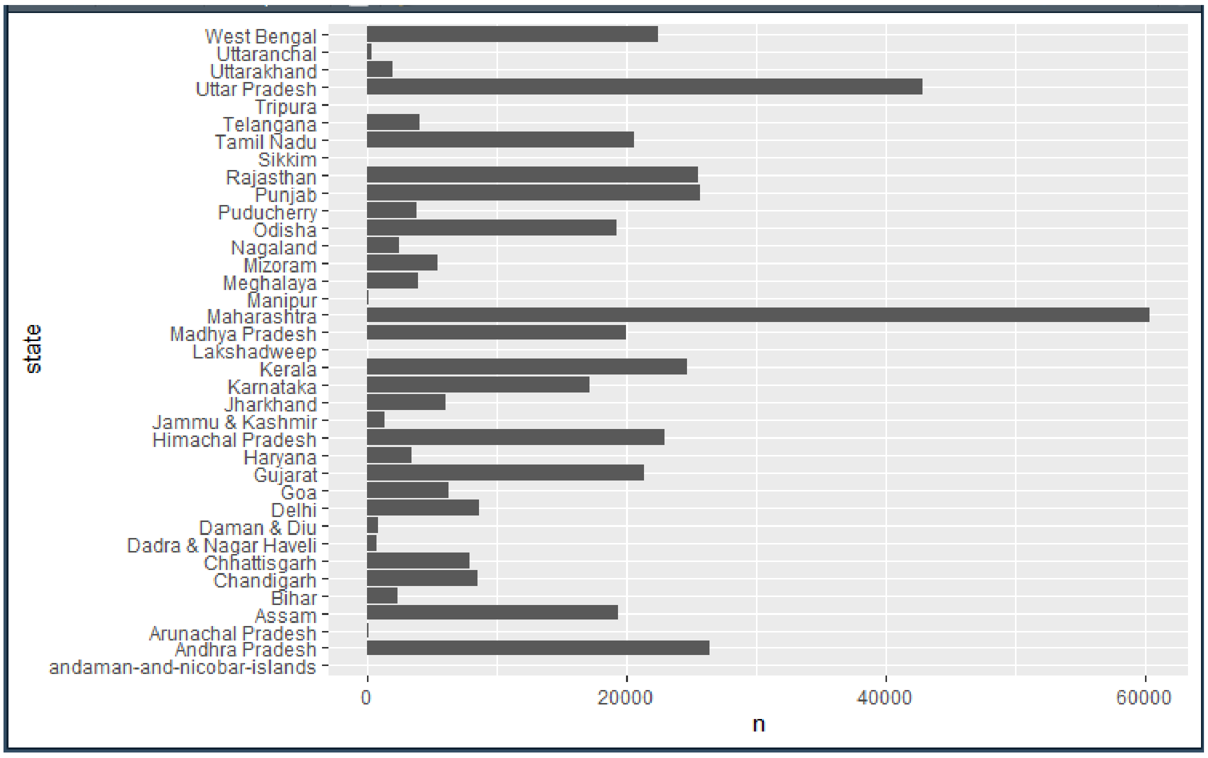
2. Read Data



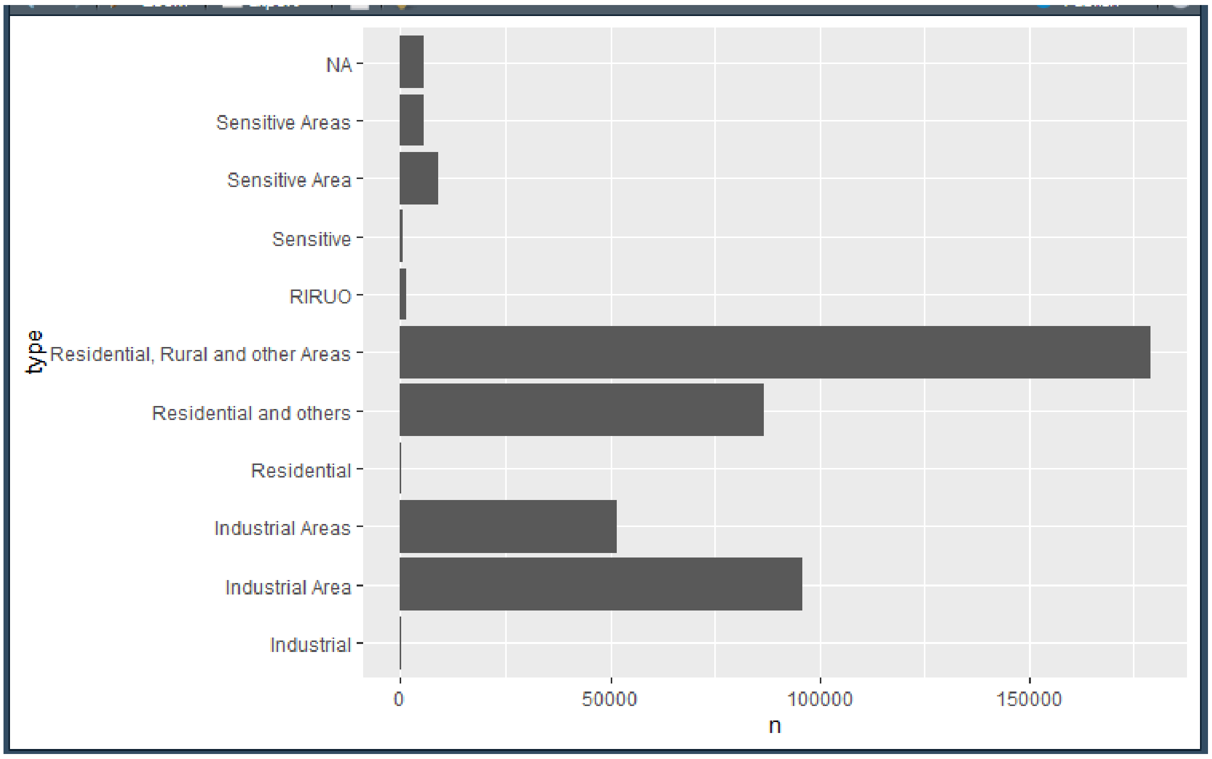
```
In [2]: df = pd.read_csv("PreprocessedinR.csv", encoding = "cp1252", low_memory=False)
        df.head()
Out[2]:
```

	state	location	agency	type	spm	pm2_5	date	so2	no2	rspm	Day_of_yr	Month	SOi	NOi	RSPMi	SPMi	AQI	AQI_Range
0	Andhra Pradesh	Hyderabad	unknown	Residential, Rural and other Areas	0.0	0.0	1990-02-01	4.8	17.4	78.182824	32	2	6.000	21.750	0	0.0	21.750	Good
1	Andhra Pradesh	Hyderabad	unknown	Industrial Area	0.0	0.0	1990-02-01	3.1	7.0	78.182824	32	2	3.875	8.750	0	0.0	8.750	Good
2	Andhra Pradesh	Hyderabad	unknown	Residential, Rural and other Areas	0.0	0.0	1990-02-01	6.2	28.5	78.182824	32	2	7.750	35.625	0	0.0	35.625	Good
3	Andhra Pradesh	Hyderabad	unknown	Residential, Rural and other Areas	0.0	0.0	1990-03-01	6.3	14.7	78.182824	60	3	7.875	18.375	0	0.0	18.375	Good
4	Andhra Pradesh	Hyderabad	unknown	Industrial Area	0.0	0.0	1990-03-01	4.7	7.5	78.182824	60	3	5.875	9.375	0	0.0	9.375	Good

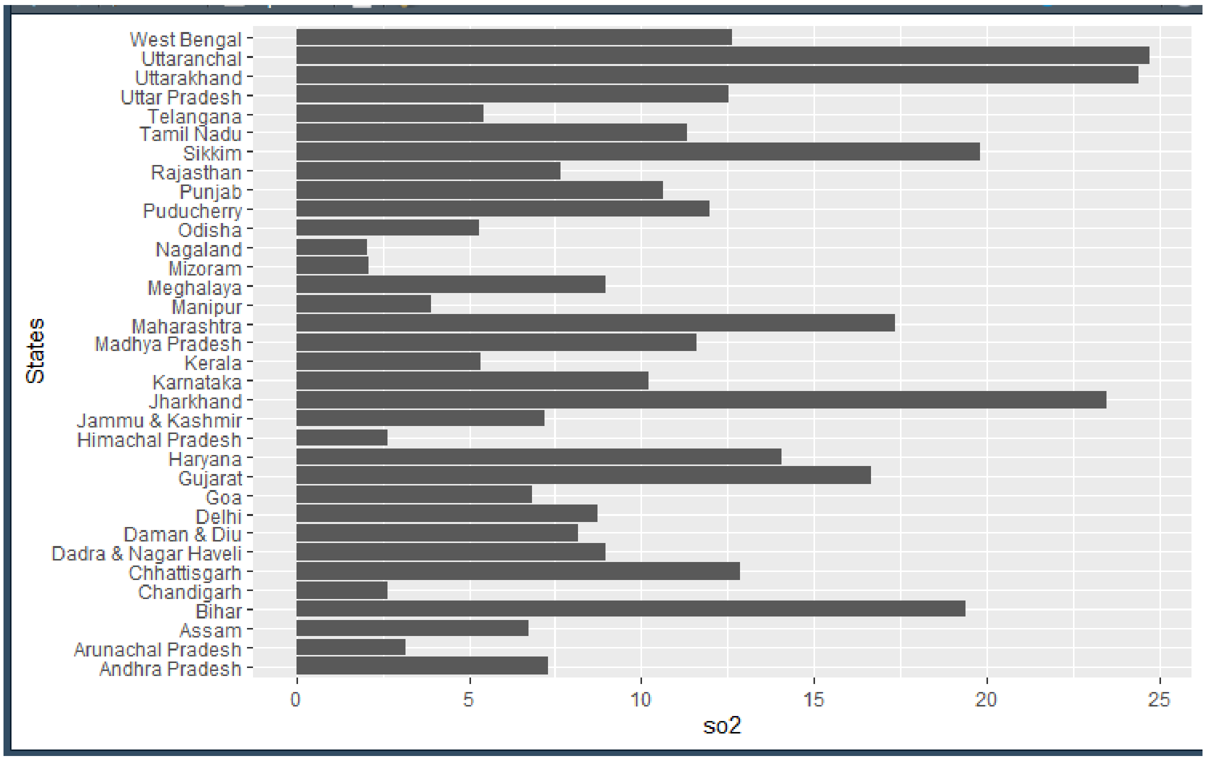
3. Visualization before preprocessing (state wise records available)



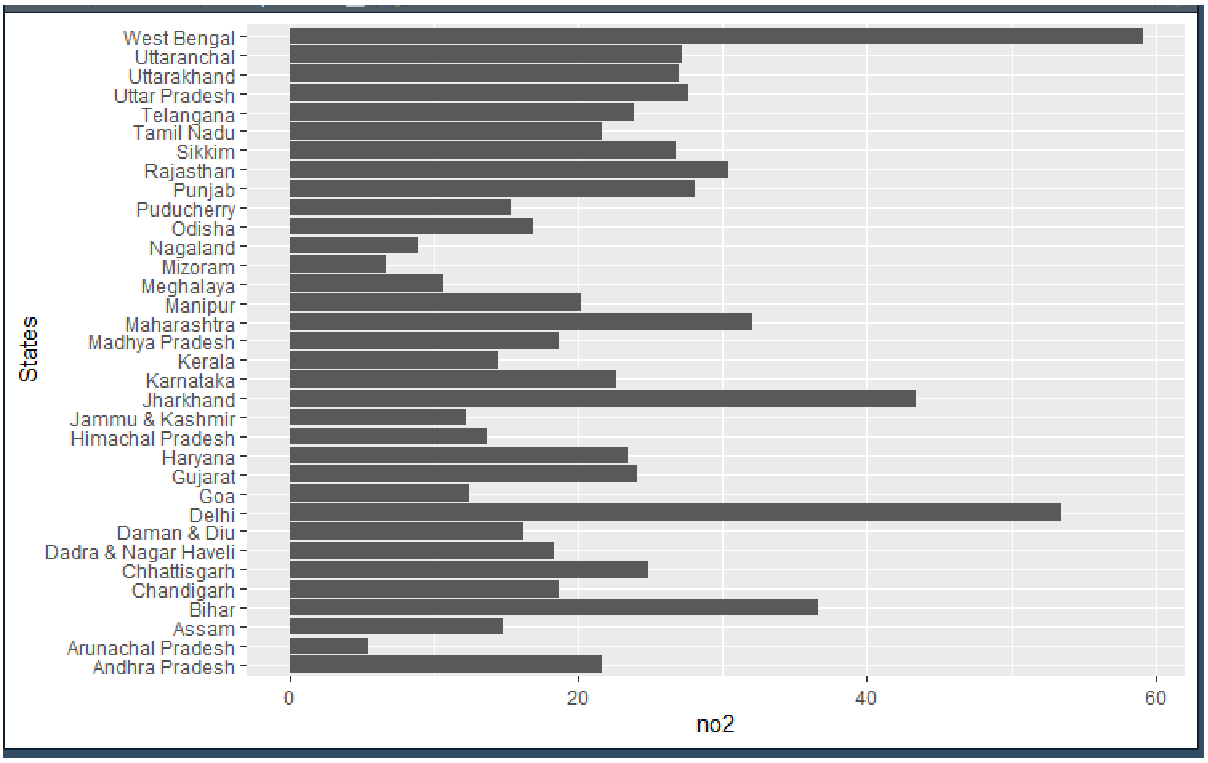
4. No of records for every location



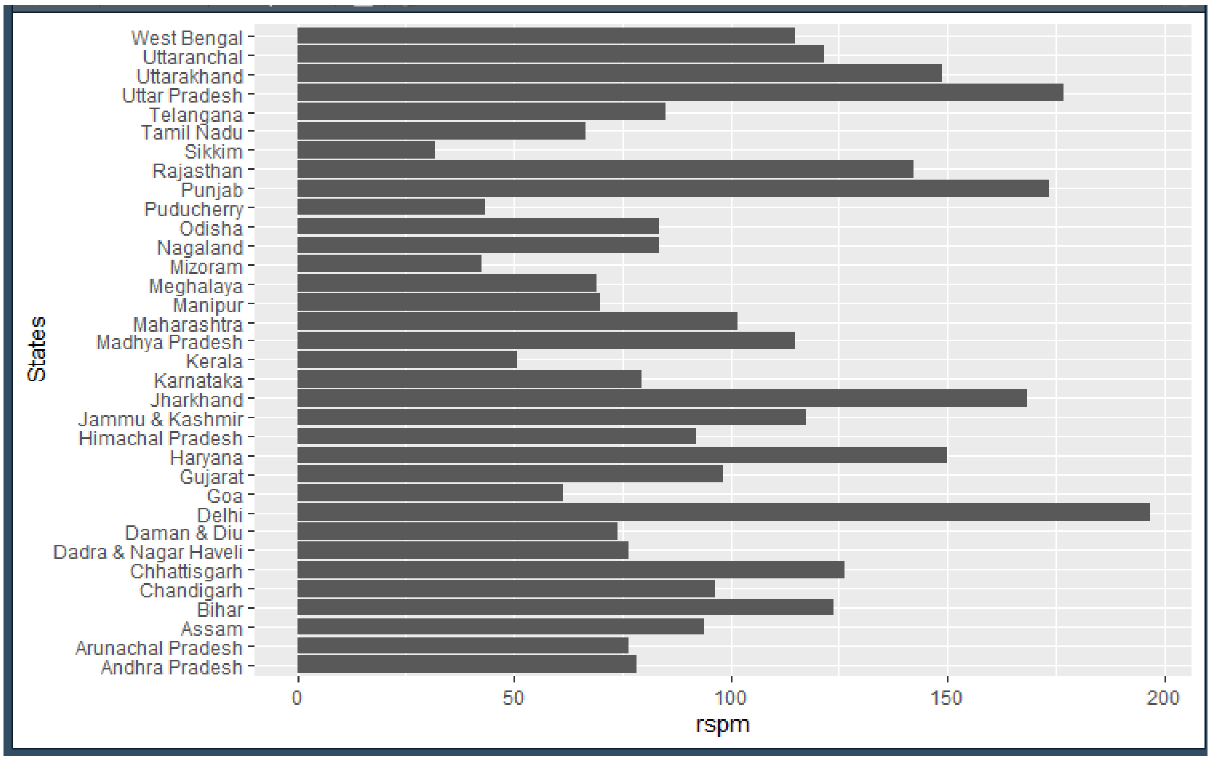
5. State-wise SO2



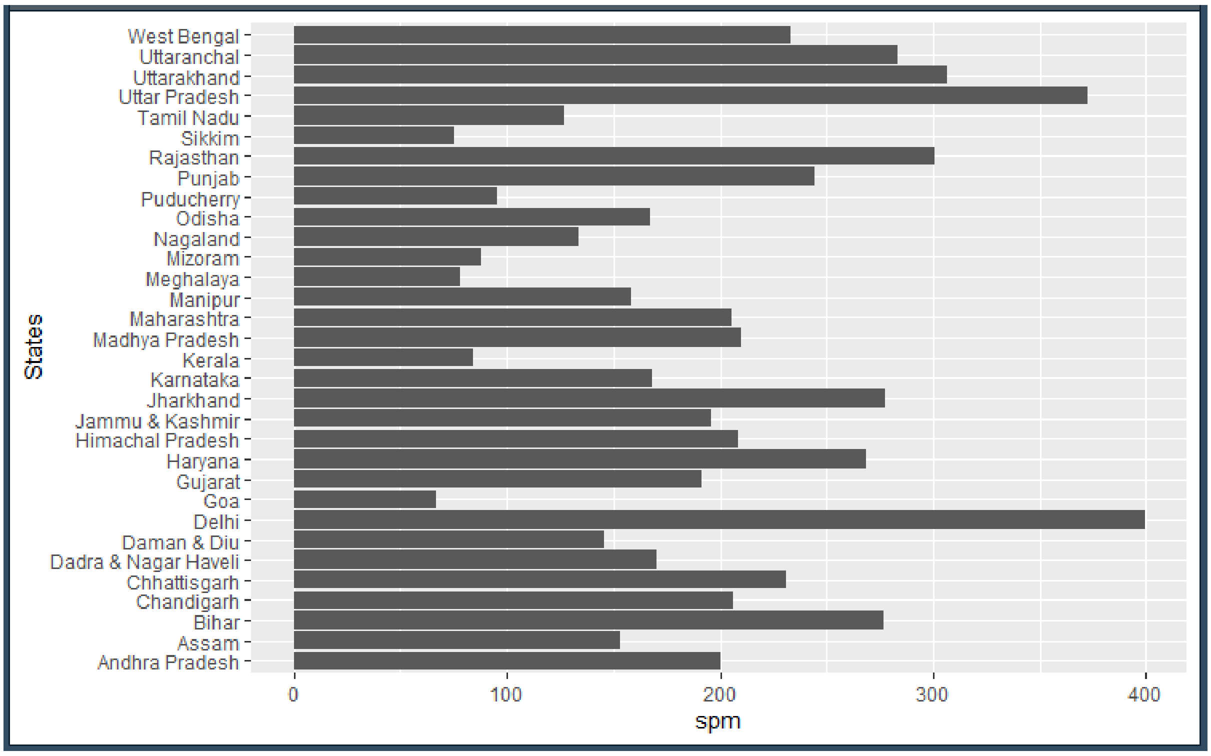
6. State-Wise NO2



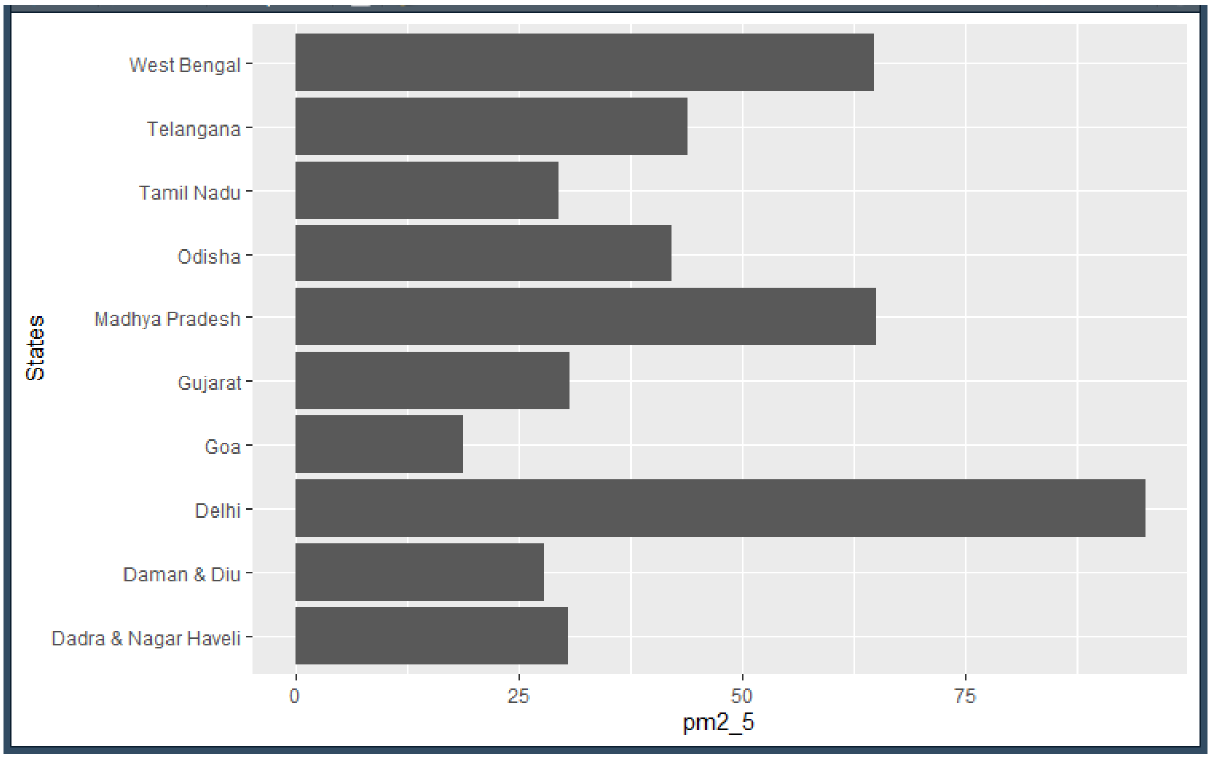
7. State-wise RSPM



8. State-wise SPM



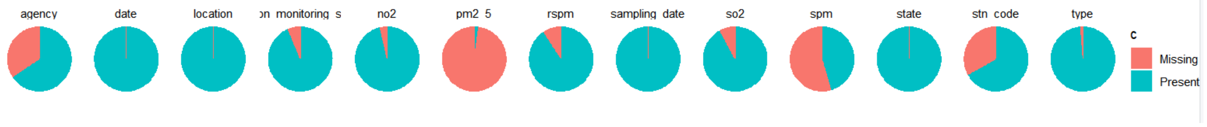
9. State-wise PM2_5



10. Missing Value Percentage

```
nullvalues = df.isnull().sum().sort_values(ascending=False)
null_values_percentage = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending=False)
missing_data_with_percentage = pd.concat([nullvalues, null_values_percentage], axis=1, keys=['Total', 'Percent'])
missing_data_with_percentage
```

	Total	Percent
pm2_5	426428	97.862497
spm	237387	54.478797
agency	149481	34.304933
stn_code	144077	33.064749
rspm	40222	9.230692
so2	34646	7.951035
location_monitoring_station	27491	6.309009
no2	16233	3.725370
type	5393	1.237659
date	7	0.001606
sampling_date	3	0.000688
location	3	0.000688
state	0	0.000000

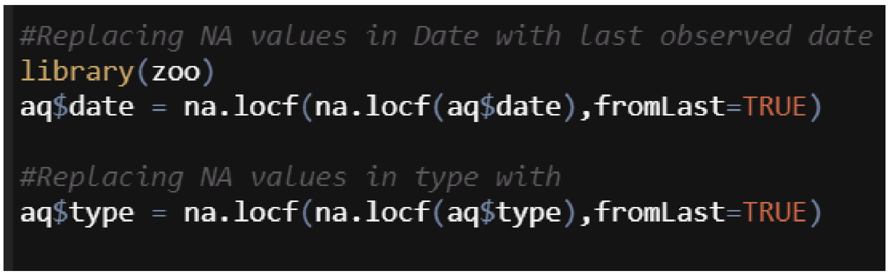


11. Strategy to deal with missing values

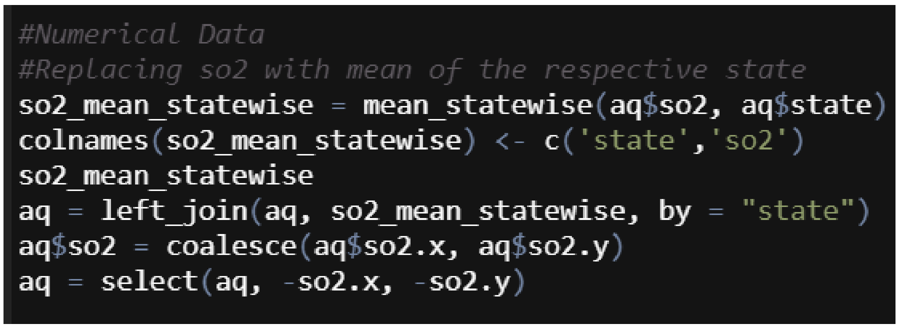
```
click to expand output; double click to hide output
In [84]: #dealing with missing values..

         #from above table, we can see that missing percent data of pm2_5 is 97 percent, we can thus drop the column.

         #For categorical columns:
         #Agency - we create a new category called "Unknown"
         #Stn_code - we can delete the column as it is not needed for prediction
         #Location_monitoring_station - we can delete the column
         #type - we can impute column with the mode (maxximum occuring data) in the column.
         #date - Firstly, fill the NA values with last observed value and then we can remove the column
                 #after adding a new column of day of the year after imputing the mode.
         #sampling date - we can remove the sampling date
         #location - impute with mode
         #state - no missing values


         #For numerical columns:
         #pm2_5 - impute with 0
         #spm - impute with 0
         #so2 - impute with mean of that state
         #no2 - impute with mean of that state
         #rspm - impute with mean of that state
```

```
aq = airq[,-c(1,2,11)] #Removing stn_code, location_monitoring_station, sampling_date data as they are not required
```

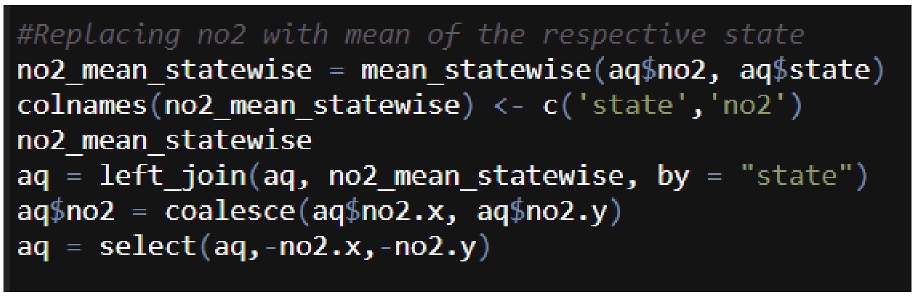
```
#Replacing NA values in agency to "unknown"
aq <- aq %>%
  replace_na(list(agency = 'unknown'))
aq
```



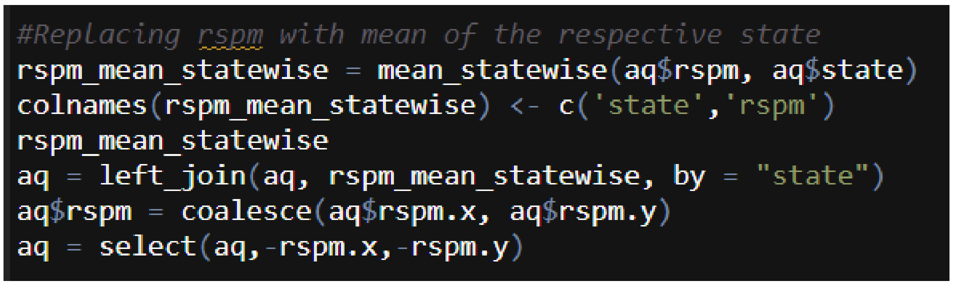
```
#Replacing NA values in Date with last observed date
library(zoo)
aq$date = na.locf(na.locf(aq$date),fromLast=TRUE)

#Replacing NA values in type with
aq$type = na.locf(na.locf(aq$type),fromLast=TRUE)
```



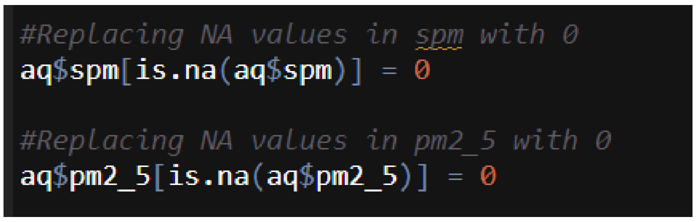
```
#Numerical Data
#Replacing so2 with mean of the respective state
so2_mean_statewise = mean_statewise(aq$so2, aq$state)
colnames(so2_mean_statewise) <- c('state','so2')
so2_mean_statewise
aq = left_join(aq, so2_mean_statewise, by = "state")
aq$so2 = coalesce(aq$so2.x, aq$so2.y)
aq = select(aq, -so2.x, -so2.y)
```



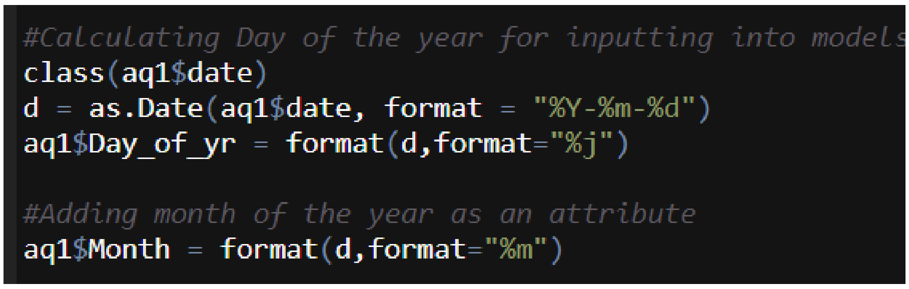
```
#Replacing no2 with mean of the respective state
no2_mean_statewise = mean_statewise(aq$no2, aq$state)
colnames(no2_mean_statewise) <- c('state','no2')
no2_mean_statewise
aq = left_join(aq, no2_mean_statewise, by = "state")
aq$no2 = coalesce(aq$no2.x, aq$no2.y)
aq = select(aq,-no2.x,-no2.y)
```



```
#Replacing rspm with mean of the respective state
rspm_mean_statewise = mean_statewise(aq$rspm, aq$state)
colnames(rspm_mean_statewise) <- c('state','rspm')
rspm_mean_statewise
aq = left_join(aq, rspm_mean_statewise, by = "state")
aq$rspm = coalesce(aq$rspm.x, aq$rspm.y)
aq = select(aq,-rspm.x,-rspm.y)
```

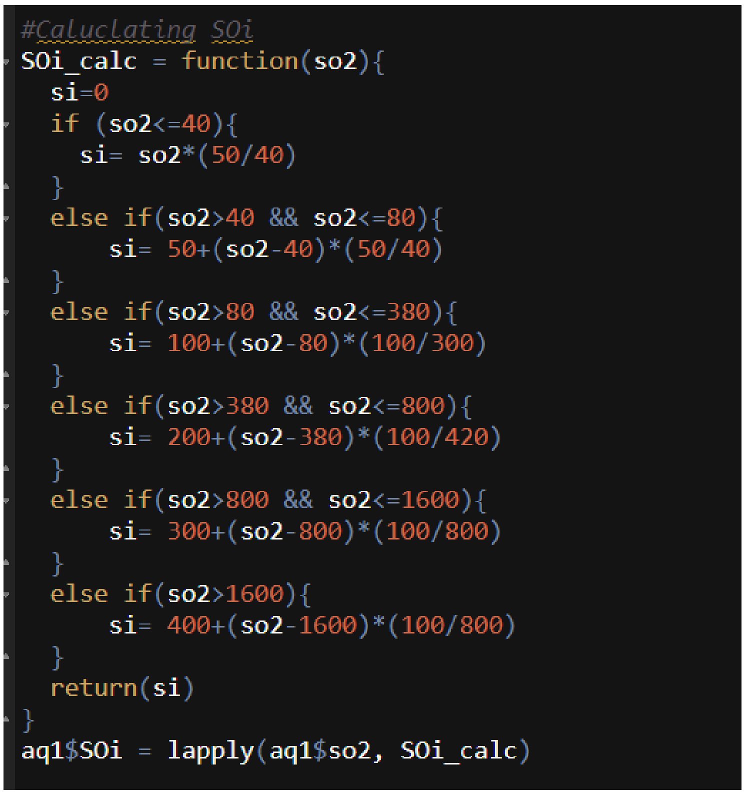


```
#Replacing NA values in spm with 0
aq$spm[is.na(aq$spm)] = 0

#Replacing NA values in pm2_5 with 0
aq$pm2_5[is.na(aq$pm2_5)] = 0
```



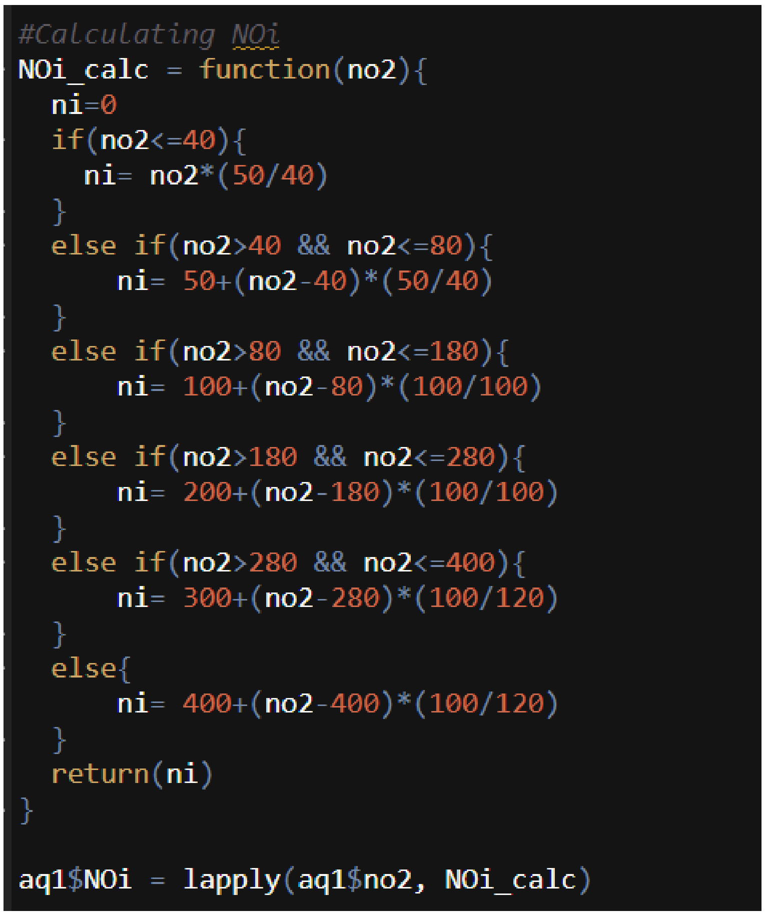
```
#Calculating Day of the year for inputting into models
class(aq1$date)
d = as.Date(aq1$date, format = "%Y-%m-%d")
aq1$Day_of_yr = format(d,format="%j")

#Adding month of the year as an attribute
aq1$Month = format(d,format="%m")
```

12. Function to calculate SO2 index



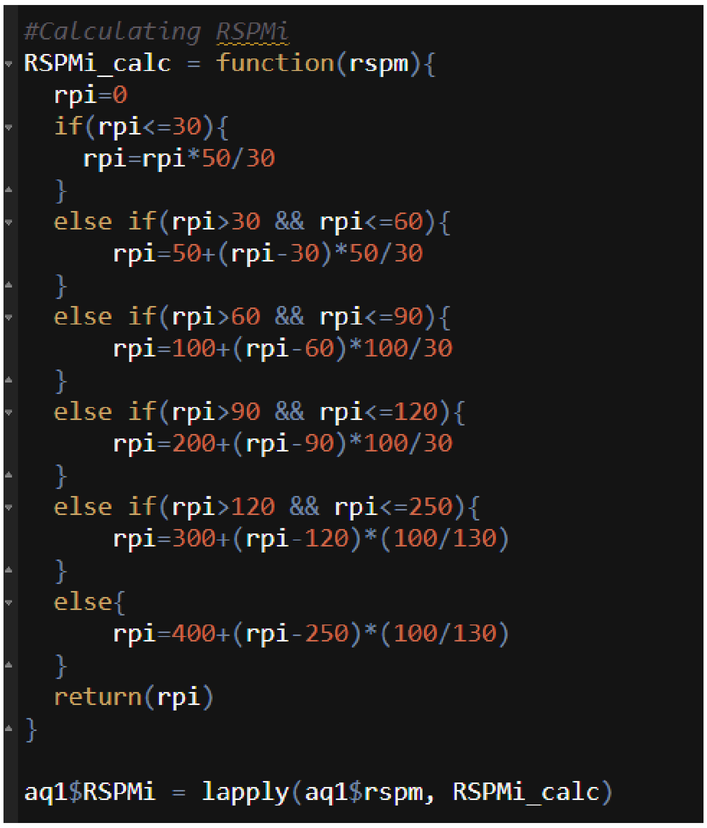
```r
#Caluclating SOi
SOi_calc = function(so2){
  si=0
  if (so2<=40){
    si= so2*(50/40)
  }
  else if(so2>40 && so2<=80){
      si= 50+(so2-40)*(50/40)
  }
  else if(so2>80 && so2<=380){
      si= 100+(so2-80)*(100/300)
  }
  else if(so2>380 && so2<=800){
      si= 200+(so2-380)*(100/420)
  }
  else if(so2>800 && so2<=1600){
      si= 300+(so2-800)*(100/800)
  }
  else if(so2>1600){
      si= 400+(so2-1600)*(100/800)
  }
  return(si)
}
aq1$SOi = lapply(aq1$so2, SOi_calc)
```

13. Function to calculate NO2 index



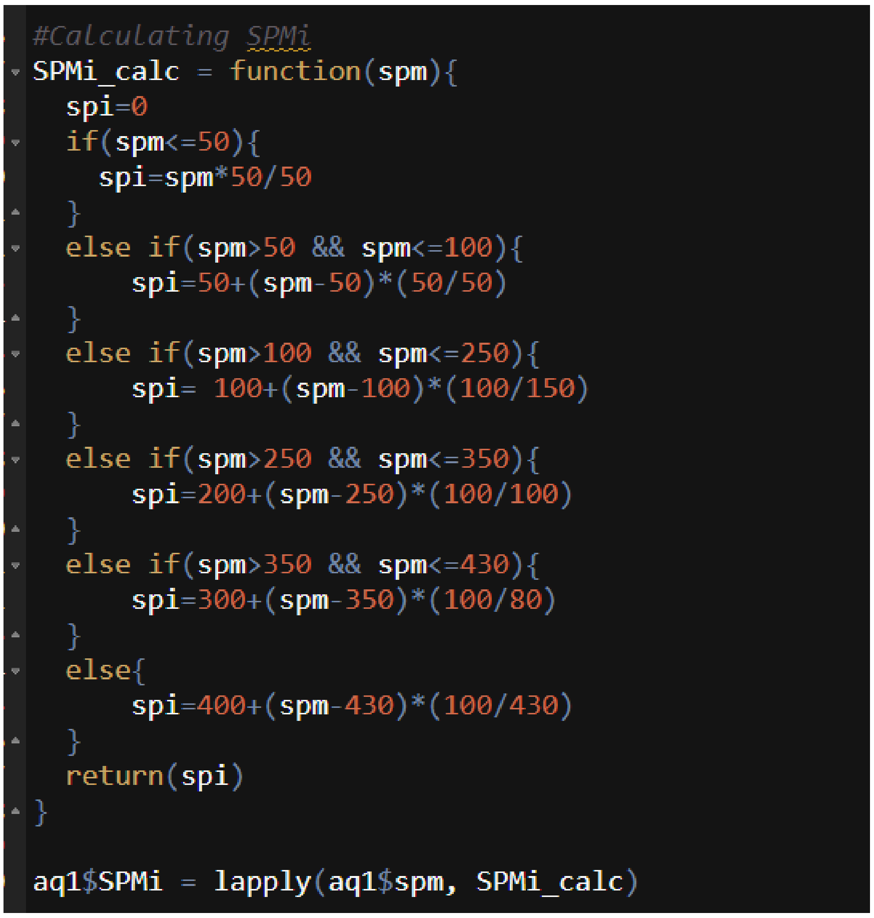
```r
#Calculating NOi
NOi_calc = function(no2){
  ni=0
  if(no2<=40){
    ni= no2*(50/40)
  }
  else if(no2>40 && no2<=80){
      ni= 50+(no2-40)*(50/40)
  }
  else if(no2>80 && no2<=180){
      ni= 100+(no2-80)*(100/100)
  }
  else if(no2>180 && no2<=280){
      ni= 200+(no2-180)*(100/100)
  }
  else if(no2>280 && no2<=400){
      ni= 300+(no2-280)*(100/120)
  }
  else{
      ni= 400+(no2-400)*(100/120)
  }
  return(ni)
}

aq1$NOi = lapply(aq1$no2, NOi_calc)
```

14. Function to calculate RSPM index



```
#Calculating RSPMi
RSPMi_calc = function(rspm){
  rpi=0
  if(rpi<=30){
    rpi=rpi*50/30
  }
  else if(rpi>30 && rpi<=60){
      rpi=50+(rpi-30)*50/30
  }
  else if(rpi>60 && rpi<=90){
      rpi=100+(rpi-60)*100/30
  }
  else if(rpi>90 && rpi<=120){
      rpi=200+(rpi-90)*100/30
  }
  else if(rpi>120 && rpi<=250){
      rpi=300+(rpi-120)*(100/130)
  }
  else{
      rpi=400+(rpi-250)*(100/130)
  }
  return(rpi)
}

aq1$RSPMi = lapply(aq1$rspm, RSPMi_calc)
```

15. Function to calculate SPM index



```
#Calculating SPMi
SPMi_calc = function(spm){
  spi=0
  if(spm<=50){
    spi=spm*50/50
  }
  else if(spm>50 && spm<=100){
      spi=50+(spm-50)*(50/50)
  }
  else if(spm>100 && spm<=250){
      spi= 100+(spm-100)*(100/150)
  }
  else if(spm>250 && spm<=350){
      spi=200+(spm-250)*(100/100)
  }
  else if(spm>350 && spm<=430){
      spi=300+(spm-350)*(100/80)
  }
  else{
      spi=400+(spm-430)*(100/430)
  }
  return(spi)
}

aq1$SPMi = lapply(aq1$spm, SPMi_calc)
```