AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF CSE

Data Warehousing and Data Mining

Summer 2022-2023

Section: A

Final Term Project On

Implementing the TDIDT algorithm from scratch, implementing it on a dataset and comparing various parameters all using the R language

Based On

Global Air Pollution Dataset

Supervised By:

Dr. Akinul Islam Jony

Associate Professor & Head (UG), Computer Science

Submitted By:

Name	ID
1. Muhammad Shahriar Zaman	20-41840-1
2. Sadia Khanam	20-44005-2
3. Ashraful Islam	20-42010-1

Date of Submission: **August 13th, 2023**

TABLE OF CONTENTS

Sections	Pages
• **Project Overview**	**3**
• **Dataset Overview**	**4-6**
• **Data Preprocessing and Visualization**	**7-14**
• **Decision Tree Evaluation**	**15-20**
• **Attribute Selection And Evaluation**	
• **Discussion and Conclusion**	**23**

Project Overview:

Air pollution is a major issue in the metropolitan life of modern times. Man-made causes are damaging air purity and in-tern harming the balance of our ecosystem. Effects of these can be noticed from the multitude of ailments caused by air pollution such as aggravated asthma, chronic obstructive pulmonary disease, lung cancer and many more.

Identifying the problem will be the first step in terms of solving it. This project will aim to create a machine learning model which can accurately predict air quality based on provided parameters.

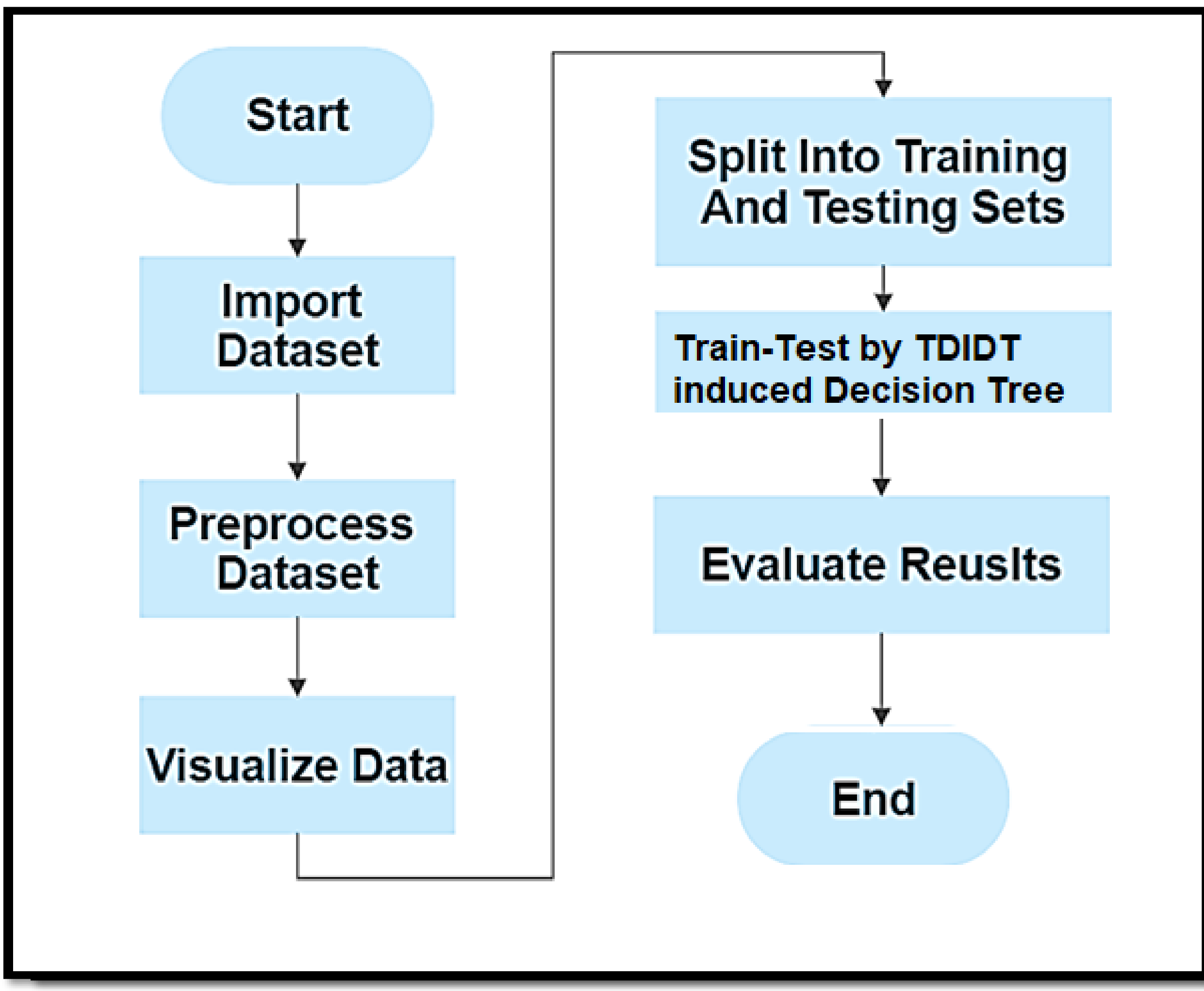


Fig.: Course of actions of this project

<u>Dataset Overview:</u> Our dataset originally has 12 attributes and 23,463 records.

Dataset source->
https://www.kaggle.com/datasets/hasibalmuzdadid/global-air-pollution-dataset?fbclid=IwAR0gVSQlEEJG8zWPPSmnJyoKkjtw90Q0uxBz2oWOwDp2_hkw4QWh5Rl95PQ

1	Country	City	AQI_Value	AQI_Category	CO_Value	CO_Category	Ozone_Value	Ozone_Category	NO2_Value	NO2_Category	PM_2.5_Value	PM2.5 AQI Category
2	Russian Fe	Praskoveya	51	Moderate	1	Good	36	Good	0	Good	51	Moderate
3	Brazil	Presidente Dutra	41	Good	1	Good	5	Good	1	Good	41	Good
4	Italy	Priolo Gargallo	66	Moderate	1	Good	39	Good	2	Good	66	Moderate
5	Poland	Przasnysz	34	Good	1	Good	34	Good	0	Good	20	Good
6	France	Punaauia	22	Good	0	Good	22	Good	0	Good	6	Good
7	United Sta	Punta Gorda	54	Moderate	1	Good	14	Good	11	Good	54	Moderate
8	Germany	Puttlingen	62	Moderate	1	Good	35	Good	3	Good	62	Moderate
9	Belgium	Puurs	64	Moderate	1	Good	29	Good	7	Good	64	Moderate
10	Russian Fe	Pyatigorsk	54	Moderate	1	Good	41	Good	1	Good	54	Moderate
11	Egypt	Qalyub	142	Unhealthy for Se	3	Good	89	Moderate	9	Good	142	Unhealthy for Sensitive Groups
12	China	Qinzhou	68	Moderate	2	Good	68	Moderate	1	Good	58	Moderate
13	Netherlan	Raalte	41	Good	1	Good	24	Good	6	Good	41	Good
14	India	Radaur	158	Unhealthy	3	Good	139	Unhealthy for Se	1	Good	158	Unhealthy
15	Pakistan	Radhan	158	Unhealthy	1	Good	50	Good	1	Good	158	Unhealthy
16	Republic o	Radovis	83	Moderate	1	Good	46	Good	0	Good	83	Moderate
17	France	Raismes	59	Moderate	1	Good	30	Good	4	Good	59	Moderate
18	India	Rajgir	154	Unhealthy	3	Good	100	Unhealthy for Se	2	Good	154	Unhealthy
19	Italy	Ramacca	55	Moderate	1	Good	47	Good	0	Good	55	Moderate
20	United Sta	Phoenix	72	Moderate	1	Good	4	Good	23	Good	72	Moderate

<u>Fig.:</u> Dataset to be used in this project (Global Air pollution)

The dataset we used, discusses the idea of air pollution and the different pollutants that it might contain. Here is a list of the main ideas:

When the atmosphere is contaminated by chemical, physical, or biological agents that change its natural properties, this is referred to as air pollution. Household combustion appliances, automobiles, industrial facilities, and natural occurrences like forest fires are some of the sources of air pollution. Nitrogen dioxide (NO2), ozone (O3), carbon monoxide (CO), and particulate matter (PM2.5) are the main pollutants of concern. The provided dataset offers geolocated information about the pollutants mentioned above (NO2, O3, CO, PM2.5). It likely includes data related to pollutant levels in different geographical locations, allowing for analysis and understanding of air pollution patterns and potential health implications.

Dataset Overview (Cont'd):

We have described each of the columns of our dataset below:

- **Country:** Name of the country from which the air will be studied
- **City:** Name of the cities(unique); represents each row.
- **AQI Value:** Overall AQI**(Air Quality Index)** value of the city, the lesser the better
- **AQI Category:** Overall AQI category of the city
- **CO Value:** AQI value of Carbon Monoxide of the city
- **CO Category:** AQI category of Carbon Monoxide**(Pollutant)** of the city
- **Ozone Value:** AQI value of Ozone**(Pollutant)** of the city
- **Ozone Category:** AQI category of Ozone of the city
- **NO2 Value:** AQI value of Nitrogen Dioxide**(NO2, works as a pollutant)** of the city
- **NO2 Category:** AQI category of Nitrogen Dioxide of the city
- **PM2.5 Value:** AQI value of Particulate Matter with a diameter of 2.5**(Type of pollutant)** micrometers or less of the city.
- **PM2.5 Category:** AQI category of Particulate Matter with a diameter of 2.5 micrometers or less of the city.

AQI or Air Quality Index is a frequently occurring phrase in this project. AQI is a scale of air pollution and lower values are always preferred. It can be calculated using the following equation:

$$\boldsymbol{AQI_{pollutant}} = \frac{\boldsymbol{reading_{pollutant}}}{\boldsymbol{standard\ value}} \times \mathbf{100} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\boldsymbol{i})$$

Importing our dataset:

Code:

```
rm(list = ls()) # Clearing all previous variables
dataset <-
read.csv("C:/Users/Asus/Desktop/Data Mining Project/global air pollution dataset.csv")
# Importing Dataset
View(dataset)
```

Result:

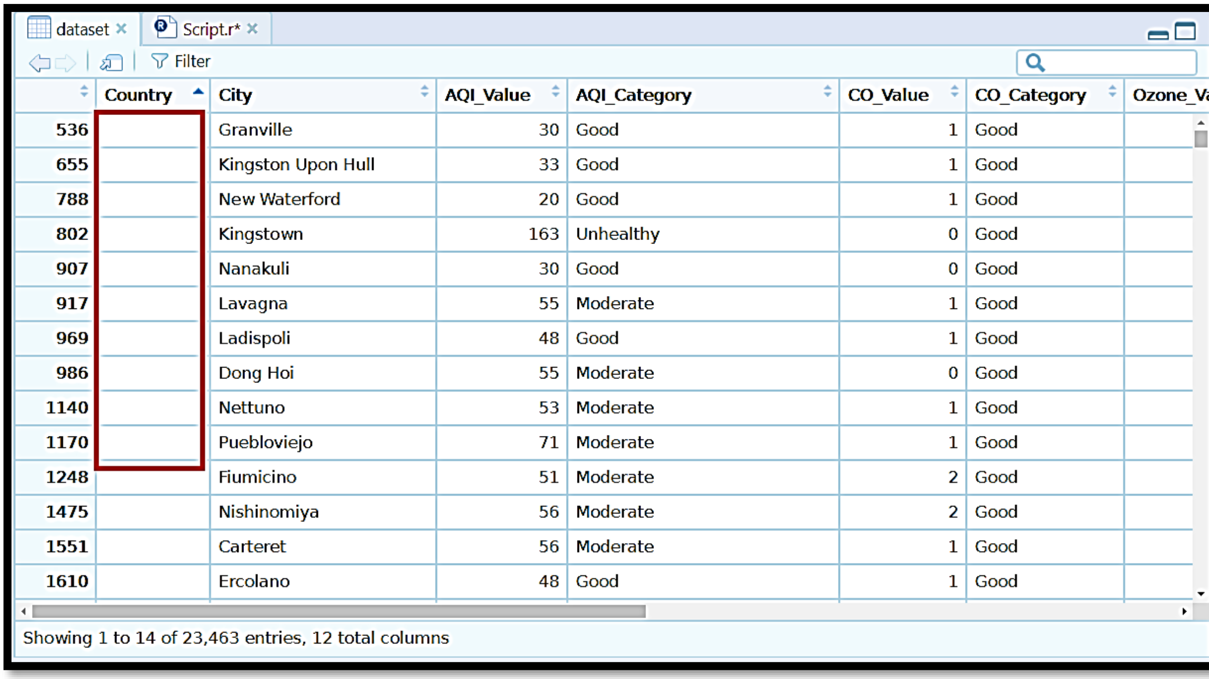
	Country	City	AQI_Value	AQI_Category	CO_Value
1	Russian Federation	Praskoveya	51	Moderate	
2	Brazil	Presidente Dutra	41	Good	
3	Italy	Priolo Gargallo	66	Moderate	
4	Poland	Przasnysz	34	Good	
5	France	Punaauia	22	Good	
6	United States of America	Punta Gorda	54	Moderate	
7	Germany	Puttlingen	62	Moderate	
8	Belgium	Puurs	64	Moderate	
9	Russian Federation	Pyatigorsk	54	Moderate	
10	Egypt	Qalyub	142	Unhealthy for Sensitive Groups	
11	China	Qinzhou	68	Moderate	
12	Netherlands	Raalte	41	Good	
13	India	Radaur	158	Unhealthy	
14	Pakistan	Radhan	158	Unhealthy	

Showing 1 to 14 of 23,463 entries, 12 total columns

Fig: Dataset imported and stored in a data frame(all columns are not visible)

<u>Dataset Preprocessing:</u> For training our decision tree machine learning model we have to work with a clean dataset, that is why we are preprocessing our data frame.

<u>1. Handling Missing Values:</u> Our data frame has some missing values in the country and city columns.



dataset × | Script.r* ×

Filter

	Country	City	AQI_Value	AQI_Category	CO_Value	CO_Category	Ozone_Va
536		Granville	30	Good	1	Good	
655		Kingston Upon Hull	33	Good	1	Good	
788		New Waterford	20	Good	1	Good	
802		Kingstown	163	Unhealthy	0	Good	
907		Nanakuli	30	Good	0	Good	
917		Lavagna	55	Moderate	1	Good	
969		Ladispoli	48	Good	1	Good	
986		Dong Hoi	55	Moderate	0	Good	
1140		Nettuno	53	Moderate	1	Good	
1170		Puebloviejo	71	Moderate	1	Good	
1248		Fiumicino	51	Moderate	2	Good	
1475		Nishinomiya	56	Moderate	2	Good	
1551		Carteret	56	Moderate	1	Good	
1610		Ercolano	48	Good	1	Good	

Showing 1 to 14 of 23,463 entries, 12 total columns

<u>Fig:</u> Dataset with missing values in country column

<u>Deleting Rows with Missing Values:</u>

<u>Code:</u>

```
# Deleting rows with blank values in Country/City column

dataset <- dataset[dataset$Country!="", ]
dataset <- dataset[dataset$City!="", ]
View(dataset)
```

(P.T.O)

Output:

dataset × Script.r* × Filter

	Country	City	AQI_Value	AQI_Category	CO_Valu
534	United States of America	Granger	57	Moderate	
535	United States of America	Grants Pass	29	Good	
537	Mexico	Guasave	35	Good	
538	Germany	Guben	37	Good	
539	Cameroon	Guider	76	Moderate	
540	China	Guilin	90	Moderate	
541	France	Guipavas	34	Good	
542	United States of America	Fort Walton Beach	65	Moderate	
543	France	Fosses	29	Good	
544	United States of America	Fountain Hills	37	Good	
545	United States of America	Fountain	72	Moderate	
546	Brazil	Francisco Morato	169	Unhealthy	
547	Germany	Freiberg	36	Good	
548	United States of America	Front Royal	46	Good	

Showing 533 to 547 of 23,035 entries, 12 total columns

Fig: Data frame with missing values removed

Dataset Visualization: Let us visualize the relation between various pollutants and air quality.

Code: (AQI vs Ozone Value)

```
library(ggplot2) #Plotting tools belong to this library
ggplot(data = dataset, mapping = aes(x = Ozone_Value, y = AQI_Value)) +
geom_point(color = "orange", alpha = .7, size = 2)+ #Specifying color, opacity and size

geom_smooth( method = "lm")+ #Specifying a linear model to be fitted
ggtitle("Plot of overall Air Quality Index vs Ozone Value in air") +
#Title/heading of the plot
xlab("Ozone Value") + #Label of x axis
ylab("AQI Value") #Label of y axis
```

Output:

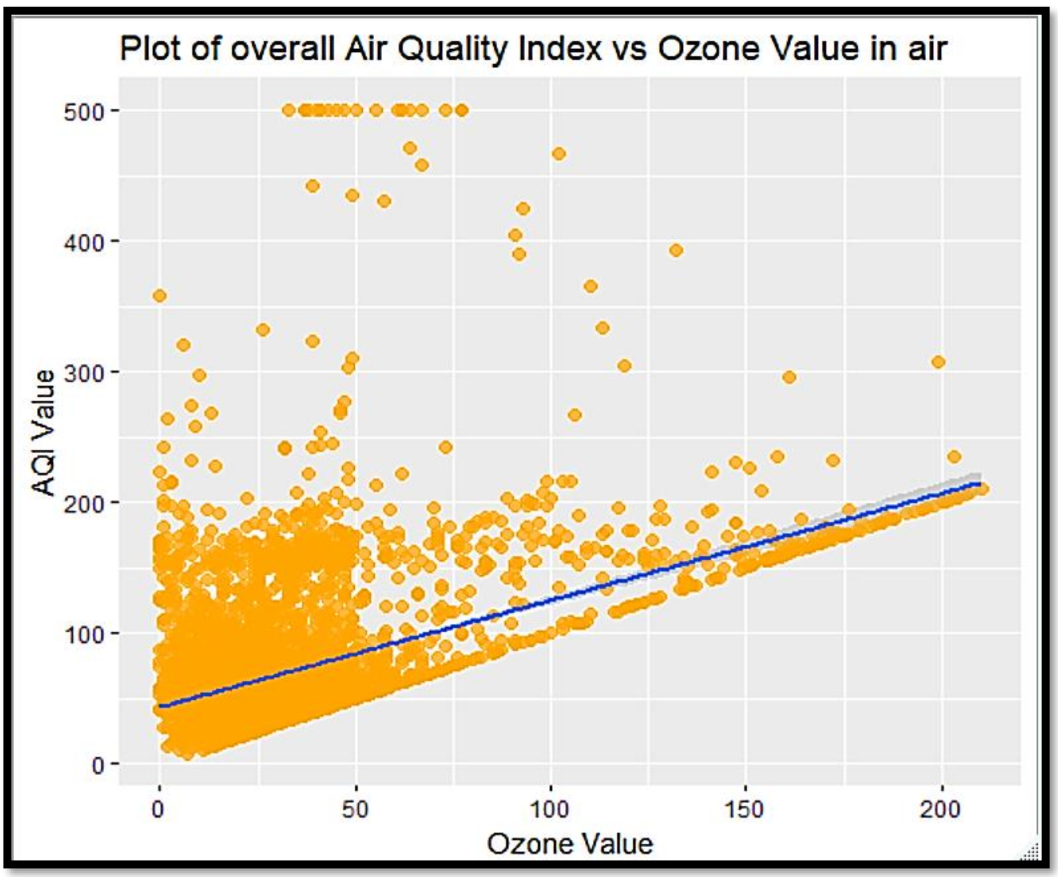


Fig.: AQI vs Ozone value plot

Code: (AQI vs Carbon-Monixide Value)

```
library(ggplot2) #Plotting tools belong to this library
ggplot(data = dataset, mapping = aes(x = CO_Value, y = AQI_Value)) +
geom_point(color = "green", alpha = 1, size = 2)+ #Specifying color, opacity and size
geom_smooth(method = "lm")+ #Specifying a linear model to be fitted
ggtitle("Plot of overall Air Quality Index vs Carbon-Monoxide Value in air") +
#Title/heading of the plot
xlab("Carbon Monoxide Value") + #Label of x axis
ylab("AQI Value") #Label of y axis
```

Output:

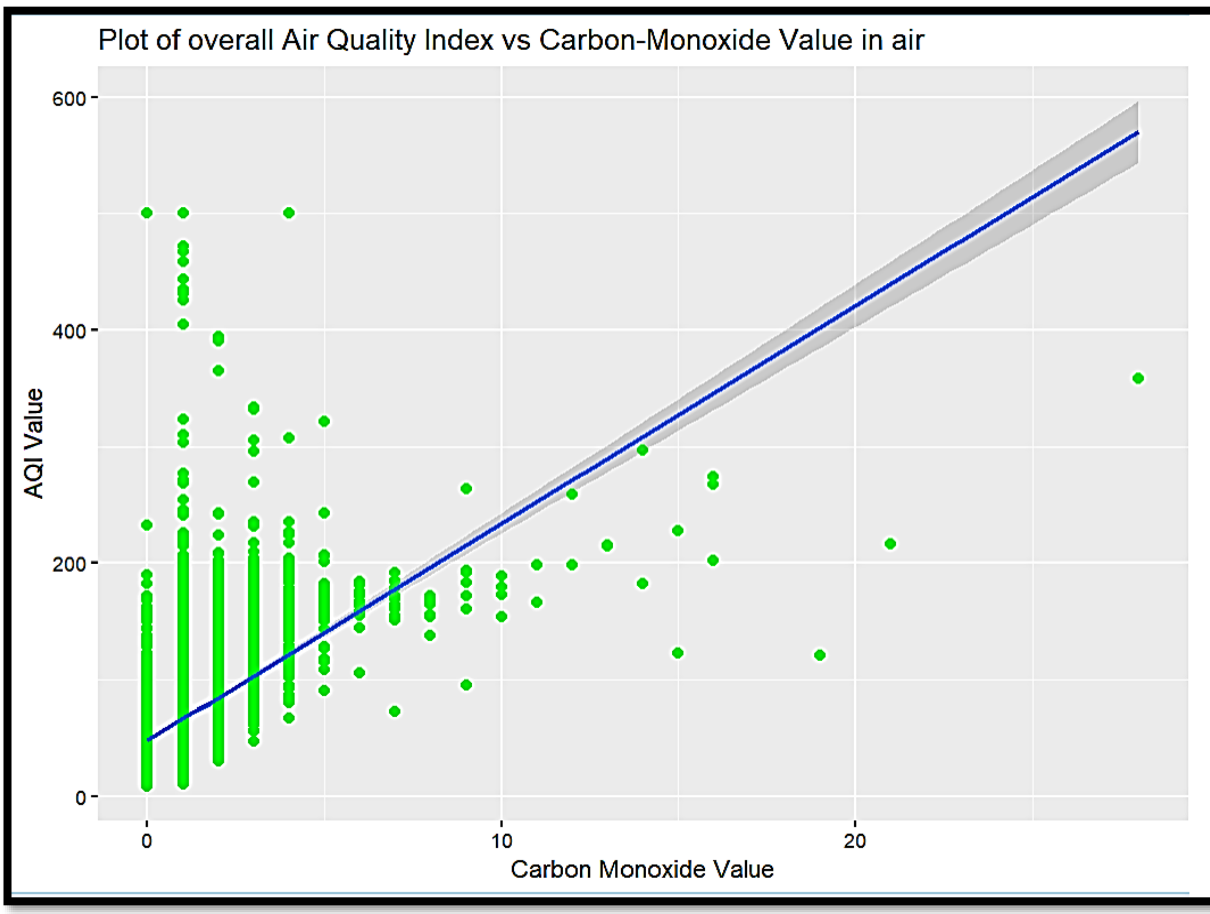


Fig.: AQI vs Carbon Monoxide value plot

Code: (AQI vs Nitrous Oxide Value)

```
library(ggplot2) #Plotting tools belong to this library
ggplot(data = dataset, mapping = aes(x = NO2_Value, y = AQI_Value)) +
geom_point(color = "brown", alpha = 1, size = 2)+ #Specifying color, opacity and size
geom_smooth(method = "lm")+ #Specifying a linear model to be fitted
ggtitle("Plot of overall Air Quality Index vs Nitrous Oxide Value in air") +
#Title/heading of the plot
xlab("Nitrous Oxide Value") + #Label of x axis
ylab("AQI Value") #Label of y axis
```

Output:

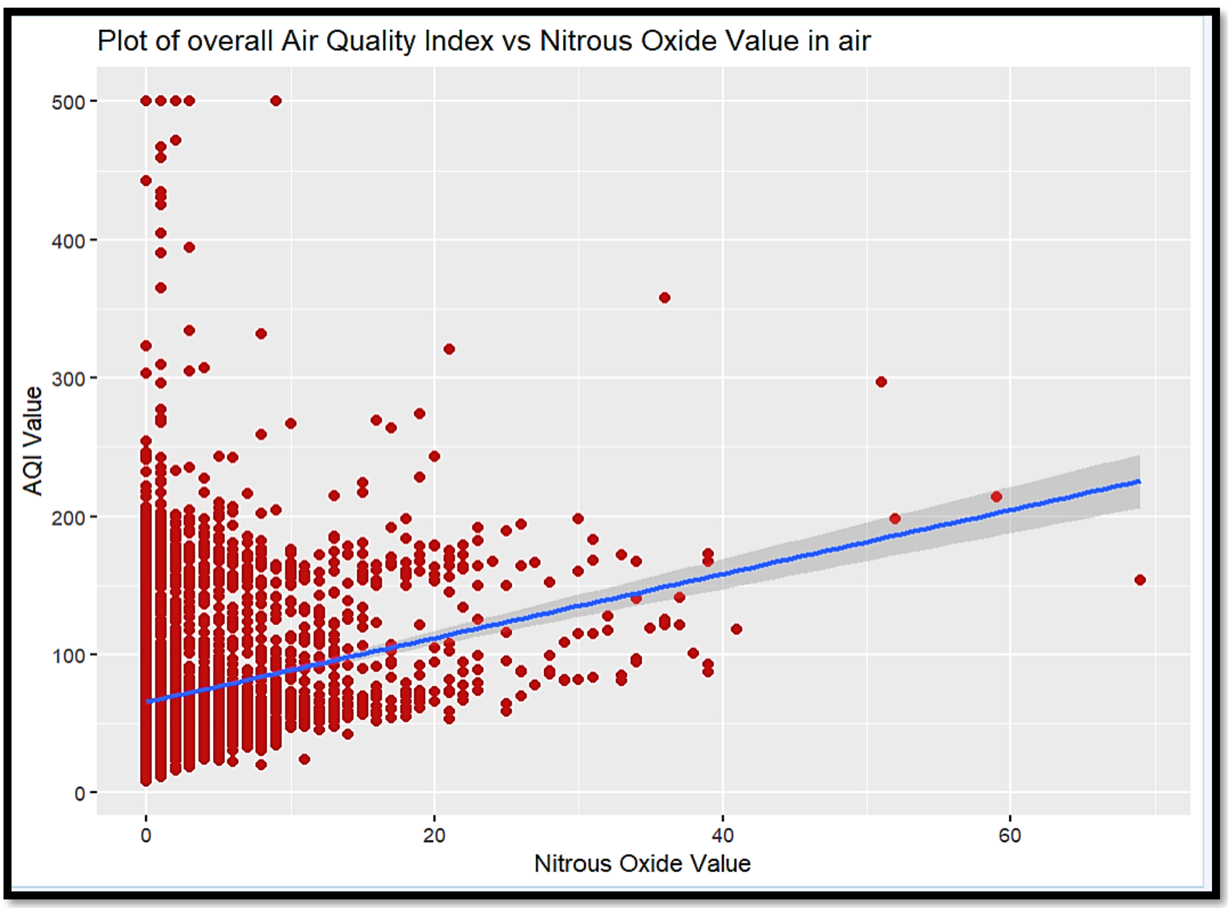


Fig.: AQI vs Nitrous Oxide value plot

Code: (AQI vs PM 2.5 Value)

```
library(ggplot2) #Plotting tools belong to this library
ggplot(data = dataset, mapping = aes(x = PM_2.5_Value, y = AQI_Value)) +
geom_point(color = "black", alpha = 1, size = 2)+ #Specifying color, opacity and size
geom_smooth(method = "lm")+ #Specifying a linear model to be fitted
ggtitle("Plot of overall Air Quality Index vs value of 2.5 µm particulates in air") +
#Title/heading of the plot
xlab("2.5 µm particulates value") + #Label of x axis
ylab("AQI Value") #Label of y axis
```

Output:

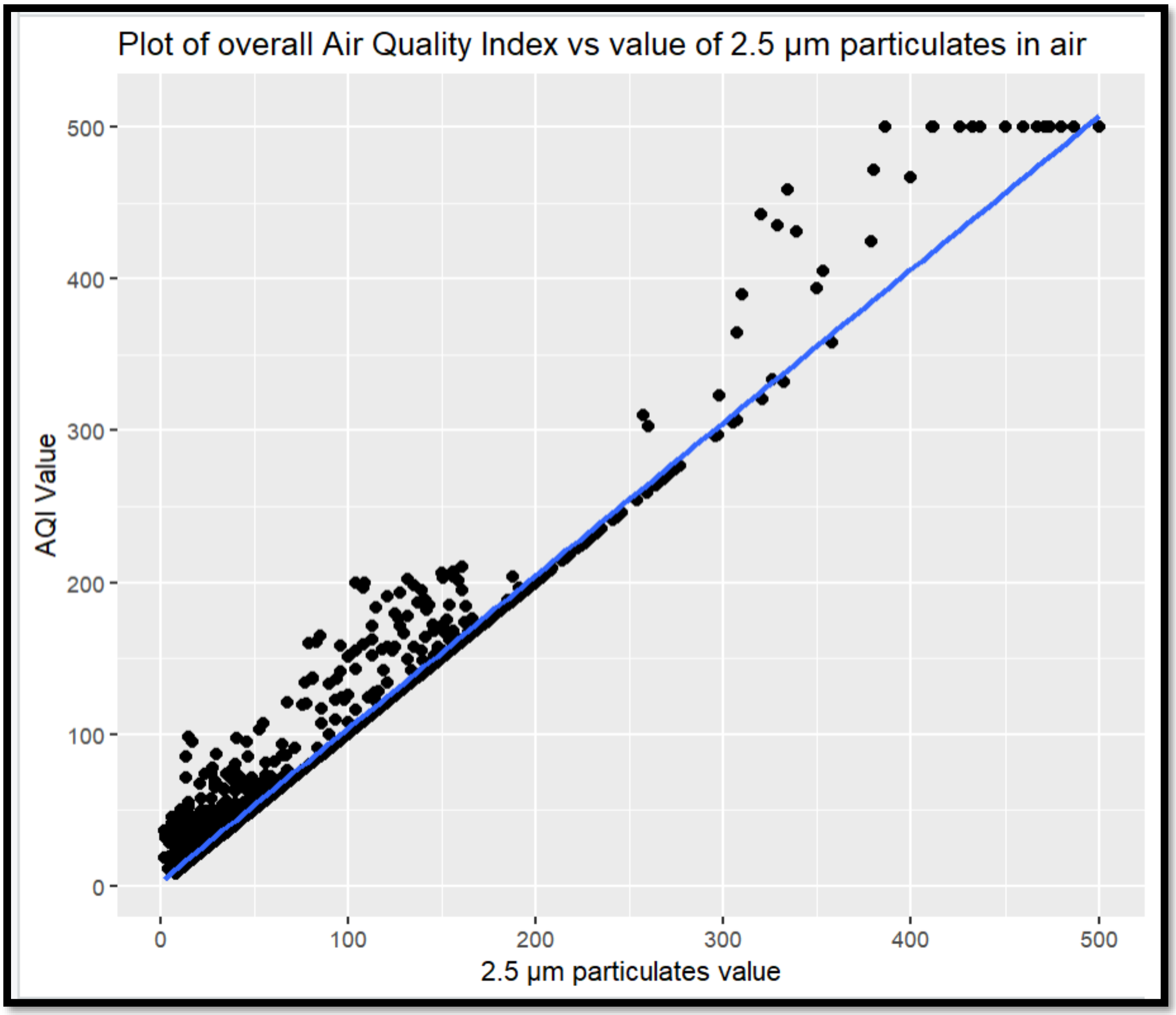


Fig.: AQI vs 2.5 µm particulates value plot

Now, if we consider AQI_Category as target label then let us check wich class/category has most records. Here each record represents a city.

Code: (Barchart for each class of label)

```
# Factoring our label from ‘Good’ to ‘Hazardous’
AQI_Category_list <-
factor(dataset$AQI_Category, levels=c('Good', 'Moderate', 'Unhealthy for Sensitive Groups',
'Unhealthy', 'Very Unhealthy', 'Hazardous'))

# Making a barplot

barplot(table(AQI_Category_list),
        main= "Number of cities divided by quality of air",
        xlab= "Grade",
        ylab= "Count",
        border= "red",
        col=c( "green" , "yellow" , "orange" , "red" , "brown" ,  "black" ))
```

Output:

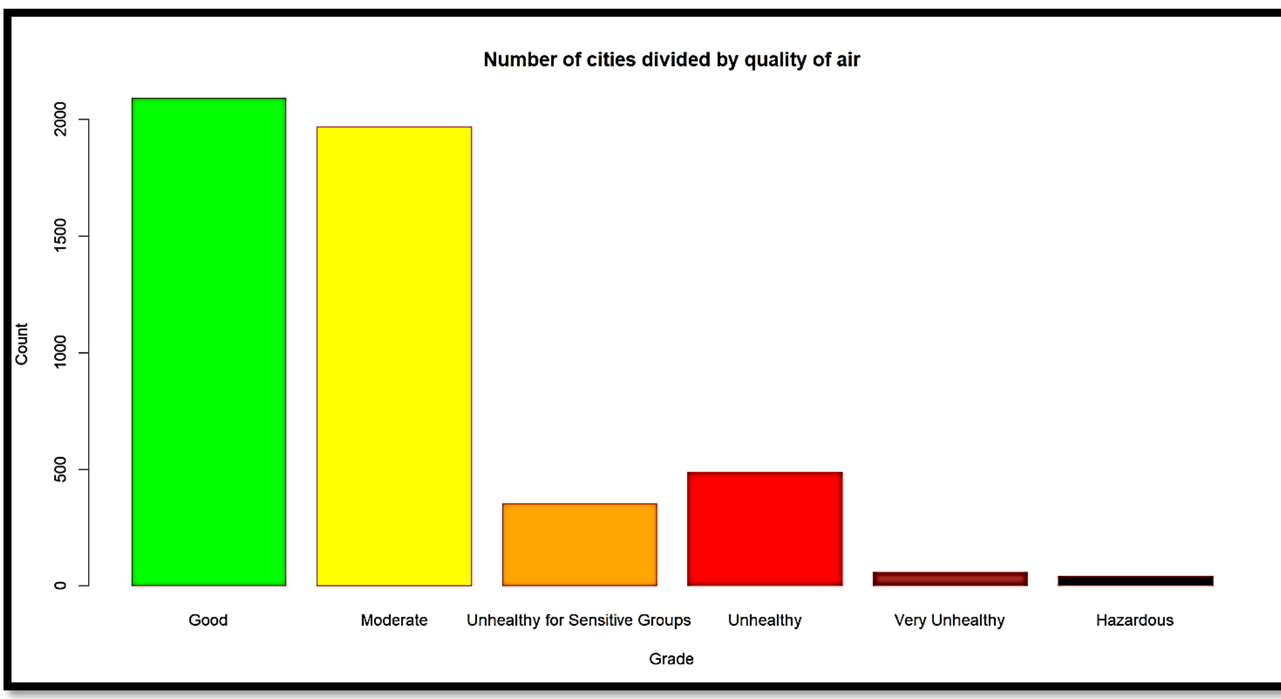


Fig.: Quality or air among cities of the world

2. Data Reduction and Transformation: Here we have 12 total columns, but 7 of them are redundant for our project. The country and city name columns cannot be used to determine air quality.

The attributes CO_Value, Ozone_Value, NO2_ Value and PM_2.5_Value are also obsolete because they are continuous and their values are already related to other categorical variables. We are excluding continuous values because our TDIDT algorithm shall work with discrete values.

So we will delete these 7 columns.

Code:

```
# Deleting unnecessary columns

dataset<-dataset[ , !(names(dataset) %in%
c('AQI_Value','Country','City','CO_Value','Ozone_Value','NO2_Value','PM_2.5_Value'))]

# Reordering
dataset<-dataset[, c(5, 3, 4, 2,1 )]

# Giving the label a simpler name

names(dataset)[5]='class'
```

Output:

	PM_2.5_Category	Ozone_Category	NO2_Category	CO_Category	class
1	Moderate	Good	Good	Good	Moderate
2	Moderate	Good	Good	Good	Moderate
3	Moderate	Good	Good	Good	Moderate
4	Moderate	Good	Good	Good	Moderate
5	Unhealthy for Sensitive Groups	Moderate	Good	Good	Unhealthy for Sensitive Groups
6	Moderate	Moderate	Good	Good	Moderate
7	Moderate	Good	Good	Good	Moderate
8	Unhealthy	Unhealthy for Sensitive Groups	Good	Good	Unhealthy
9	Moderate	Good	Good	Good	Moderate
10	Unhealthy	Moderate	Good	Good	Unhealthy
11	Unhealthy	Moderate	Good	Good	Unhealthy
12	Moderate	Good	Good	Good	Moderate
13	Moderate	Good	Good	Good	Moderate

Fig.: Data frame after reduction and transformation

Decision Tree Evaluation:

We have prepared the following code to induce the dataset into a tree-like data structure. Here our tree will be stored in a list of hash objects. Each hash object will represent an internal node of the tree.

Code:

```
# TDIDT tree training function
train_tdidt_tree<-function(DF, parent_name, parent_split_by){

if(names(DF[1])=='X')
{
this_column<-DF[2]
}

else{
this_column<-DF[1]
}
column_name<-names(this_column)
#print(column_name)

if(column_name=='class' || is.null(column_name)){return(0)}

dict<- hash(keys=c('name','parent_name','parent_split_by'), values=c(column_name,
parent_name, parent_split_by))



for(j in unique(this_column))
{
#print(names(this_column))
for(k in 1:length(j))
{
if(length(unique(subset(DF, DF[names(this_column)]==j[k])$class))==1)
{

dict[j[k]]=unique(DF[DF[names(this_column)]==j[k], ]$class)


}else{ #print('not unique')

dict[j[k]]=names(DF)[match(names(this_column),names(DF))+1]
```

```
list1<-append(list1,dict)

DFx<-DF[DF[names(this_column)]==j[k],]
DFx<-DFx[ , !(names(DFx) %in% c(names(this_column)))]



parent_name<-names(this_column)
parent_split_by<-j[k]

list1<-append(list1,train_tdidt_tree(DFx,parent_name,parent_split_by))


}

}

}

list1<-append(list1,dict)


  return(unique(list1))
}

# Splitter function
splitter = function(dataset, training_ratio)
{ # This function will split the dataset...
  # ...into training and testing sets and then return them in a list


  if(training_ratio>=1 | training_ratio<=0)
  {return("Training ratio has to be a fraction value i range: 0<x<1")


  }else
  {training_row_count<-round(training_ratio*nrow(dataset)) # No. Of rows
  testing_row_count<-round((1-training_ratio)*nrow(dataset))
  training_dataset<-head(dataset, training_row_count) # Splitting by head()
  testing_dataset<-tail(dataset, testing_row_count) # Splitting by tail()
  return_values <- list(training_dataset, testing_dataset) # Binding in a list
```

```
  return(return_values) # Returning the list
  }

}



# Function to count branches of the induced tree

branch_counter<-function(induced_tree)
{
  branch_count=0
  for(i in 1:length(induced_tree))
  {

    branch_count=branch_count+length(keys(induced_tree[i][[1]]))-3

  }

  return(branch_count)
}

# Calling the TDIDT tree training function
train_set<-data.frame(splitter(dataset,0.80)[1])
  test_set<-data.frame(splitter(dataset,0.80)[2])

  list1<-list()  # Tree list to store all the nodes

  induced_tree<-train_tdidt_tree(train_set,"", "")
  print(induced_tree)
  branch_counter(induced_tree)
```

Output:

```
[[1]]
<hash> containing 9 key-value pair(s).
  Good : Ozone_Category
  Hazardous : Hazardous
  Moderate : Ozone_Category
  name : PM_2.5_Category
  parent_name :
  parent_split_by :
  Unhealthy : Ozone_Category
  Unhealthy for Sensitive Groups : Ozone_Category
  Very Unhealthy : Very Unhealthy
[[2]]
<hash> containing 7 key-value pair(s).
  Good : Moderate
  Moderate : Moderate
  name : Ozone_Category
  parent_name : PM_2.5_Category
  parent_split_by : Moderate
  Unhealthy : Unhealthy
  Unhealthy for Sensitive Groups : Unhealthy for Sensitive Groups
[[3]]
<hash> containing 8 key-value pair(s).
  Good : Unhealthy for Sensitive Groups
  Moderate : Unhealthy for Sensitive Groups
  name : Ozone_Category
  parent_name : PM_2.5_Category
  parent_split_by : Unhealthy for Sensitive Groups
  Unhealthy : Unhealthy
  Unhealthy for Sensitive Groups : Unhealthy for Sensitive Groups
  Very Unhealthy : Very Unhealthy
```

[[4]]

<hash> containing 8 key-value pair(s).

Good : Unhealthy

Moderate : Unhealthy

name : Ozone_Category

parent_name : PM_2.5_Category

parent_split_by : Unhealthy

Unhealthy : Unhealthy

Unhealthy for Sensitive Groups : Unhealthy

Very Unhealthy : Very Unhealthy

[[5]]

<hash> containing 6 key-value pair(s).

Good : Good

Moderate : Moderate

name : Ozone_Category

parent_name : PM_2.5_Category

parent_split_by : Good

Unhealthy for Sensitive Groups : Unhealthy for Sensitive Groups

```
>   branch_counter(induced_tree)
[1] 23
>
```

Fig.: No. of branches

Attribute Selection: For attribute selection we will compare parameters Information Gain, Gain Ratio and Gini Index. Let us create functions for them.

Code:

```
# Function to generate E_start
get_E_start<-function(train_df)
{
  E_start=0
  for(i in 1:length(table(train_df$class)))
  {
    ratio<-(table(train_df$class)[[i]]/nrow(train_df))

    E_start=E_start+(-1)*ratio*log(ratio, base=2)

  }

  return(E_start)
}



# Function to generate E_new values for a column

get_E_new<-function(train_df,column_name)
{ E_new=0
for(i in 1:nrow(unique(train_set[column_name])))
{


  unique_value<-unique(train_df[column_name])[[1]][i]

  ratio<-table(train_set[column_name])[[i]]/nrow(train_set)
```

```
  log_result<-get_E_start(train_df[train_df[column_name]==unique_value,])

  E_new=E_new + ratio*log_result

}

return(E_new)

}


# Function to generate IG value for a column

get_IG<-function(train_df,column_name)
{ E_new=0
for(i in 1:nrow(unique(train_df[column_name])))
{


  unique_value<-unique(train_df[column_name])[[1]][i]

  ratio<-table(train_df[column_name])[[i]]/nrow(train_df)


  log_result<-get_E_start(train_df[train_df[column_name]==unique_value,])

  E_new=E_new + ratio*log_result

}
```

```
IG<-get_E_start(train_df)-E_new
return(IG)

}
# Function to generate split information for training set

get_SI<-function(train_df,column_name)
{
  SI=0
  for(i in 1:length(table(train_df[column_name])))
  {
    ratio<-(table(train_df[column_name])[[i]]/nrow(train_df))

    SI=SI+(-1)*ratio*log(ratio, base=2)

  }

  return(SI)
}

# Function to generate Gini_start for training set
get_Gini_start<-function(train_df)
{
  Gini_start=0
  for(i in 1:length(table(train_df$class)))
  {
    ratio<-(table(train_df$class)[[i]]/nrow(train_df))

    Gini_start=Gini_start+(ratio)*ratio

  }
```

```
  Gini_start=1-Gini_start

  return(Gini_start)
}



# Information gain for the columns
print('------------------')

get_IG(train_set,'PM_2.5_Category')

get_IG(train_set,'Ozone_Category')

get_IG(train_set,'NO2_Category')

get_IG(train_set,'CO_Category')
```

```
> get_IG(train_set,'PM_2.5_Category')
[1] 1.549021
>
> get_IG(train_set,'Ozone_Category')
[1] 0.3534173
>
> get_IG(train_set,'NO2_Category')
[1] 0.005423373
>
> get_IG(train_set,'CO_Category')
[1] 0.001846157
>
```

Fig.: Information gain for each attribute

```
> get_IG(train_set,'PM_2.5_Category')/get_SI(train_set,'PM_2.5_Category')
[1] 0.8903033
>
> get_IG(train_set,'Ozone_Category')/get_SI(train_set,'Ozone_Category')
[1] 0.5601278
>
> get_IG(train_set,'NO2_Category')/get_SI(train_set,'NO2_Category')
[1] 0.5460183
>
> get_IG(train_set,'CO_Category')/get_SI(train_set,'CO_Category')
[1] 0.4850303
```

Fig.: Gain ratio for each attribute

```
> get_Gini_new(train_set,'PM_2.5_Category')
[1] 0.06613896
>
> get_Gini_new(train_set,'Ozone_Category')
[1] 0.095772
>
> get_Gini_new(train_set,'NO2_Category')
[1] 0.1540385
>
> get_Gini_new(train_set,'CO_Category')
[1] 0.1541307
```

Fig.: Gini index for each attribute

From the above we can see that PM_2.5_Category is the best attribute to start with, other than this the order has remained the same. Now let us count the number of branches at this order.

```
>   branch_counter(induced_tree)
[1] 22
>
```

Fig.: Branches of our tree after attribute selection

Now, let us check the accuracy of our model

Code:

```
evaluate_accuracy(induced_tree, test_set)
```

Output:

```
[1] "Accuracy : "
[1] 99.75
```

Fig.: Accuracy for all attribute combinations

Now, let us create a confusion matrix for the model

```
test_set2<-evaluate_accuracy(induced_tree, test_set)
test_set2[test_set2 == "Unhealthy for Sensitive Groups"] <- "UFSG"

set.seed(123)
data <- data.frame(Actual = test_set2$class,
                   Prediction = test_set2$predicted_class
)
 table(data$Prediction, data$Actual)
```

	Good	Hazardous	Moderate	UFSG	Unhealthy	Very Unhealthy
Good	67	0	0	0	0	0
Hazardous	0	6	0	0	0	0
Moderate	7	0	226	0	0	0
UFSG	0	0	0	60	0	0
Unhealthy	0	0		0	35	0
Very Unhealthy	0	0	0	0	0	6

Fig.: Confusion Matrix

Discussion and Conclusion: From the results we can see that all attribute selection methods gave the same results. Our dataset can also be made smaller by taking samples for faster training and testing since in this case we had 23,000+ records. We believe our efforts have satisfied the project requirements and it is a success.

Note: Our project can be downloaded from the following link:
https://drive.google.com/drive/u/0/folders/1X-pfwRlYbfFP7Qn7aqDb2Ejog1RH7s53