# Classification

- Logistic Regression
- Decison Tree classifier
- Random Forest classifier
- Naive Bayes
- Support Vector Machine
- K-Nearest Neighbour
- Artificial Neural Network

# Regression

- Simple Linear Regression
- Multilinear Regression
- Polynimial Regression
- Decison Tree Regressor
- Random Forest Regressor
- K-Nearest Neighbor
- Support Vactor machine
- Artificial Neural Network

In [1]:

```
# 1 import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# 2 read data set
df= pd.read_csv("C:\\Users\\Hp-\\OneDrive\\Desktop\\global.csv" , encoding="latin-1")
```

In [3]:

```
df
```

Out[3]:

| | Country | City | AQI Value | AQI Category | CO AQI Value | CO AQI Category | Ozone AQI Value | Ozone AQI Category | NO2 AQI Value | N Ca |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Russian Federation | Praskoveya | 51 | Moderate | 1 | Good | 36 | Good | 0 | |
| **1** | Brazil | Presidente Dutra | 41 | Good | 1 | Good | 5 | Good | 1 | |
| **2** | Italy | Priolo Gargallo | 66 | Moderate | 1 | Good | 39 | Good | 2 | |
| **3** | Poland | Przasnysz | 34 | Good | 1 | Good | 34 | Good | 0 | |
| **4** | France | Punaauia | 22 | Good | 0 | Good | 22 | Good | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **23458** | India | Gursahaiganj | 184 | Unhealthy | 3 | Good | 154 | Unhealthy | 2 | |
| **23459** | France | Sceaux | 50 | Good | 1 | Good | 20 | Good | 5 | |
| **23460** | India | Mormugao | 50 | Good | 1 | Good | 22 | Good | 1 | |
| **23461** | United States of America | Westerville | 71 | Moderate | 1 | Good | 44 | Good | 2 | |
| **23462** | Malaysia | Marang | 70 | Moderate | 1 | Good | 38 | Good | 0 | |

23463 rows × 12 columns

In [4]:

```
df.info(
)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23463 entries, 0 to 23462
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Country             23036 non-null  object
 1   City                23462 non-null  object
 2   AQI Value           23463 non-null  int64 
 3   AQI Category        23463 non-null  object
 4   CO AQI Value        23463 non-null  int64 
 5   CO AQI Category     23463 non-null  object
 6   Ozone AQI Value     23463 non-null  int64 
 7   Ozone AQI Category  23463 non-null  object
 8   NO2 AQI Value       23463 non-null  int64 
 9   NO2 AQI Category    23463 non-null  object
 10  PM2.5 AQI Value     23463 non-null  int64 
 11  PM2.5 AQI Category  23463 non-null  object
dtypes: int64(5), object(7)
memory usage: 2.1+ MB
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
Country               427
City                    1
AQI Value               0
AQI Category            0
CO AQI Value            0
CO AQI Category         0
Ozone AQI Value         0
Ozone AQI Category      0
NO2 AQI Value           0
NO2 AQI Category        0
PM2.5 AQI Value         0
PM2.5 AQI Category      0
dtype: int64
```

In [6]:

```
df1=df.dropna()
df1
```

Out[6]:

| | Country | City | AQI Value | AQI Category | CO AQI Value | CO AQI Category | Ozone AQI Value | Ozone AQI Category | NO2 AQI Value | N Ca |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Russian Federation | Praskoveya | 51 | Moderate | 1 | Good | 36 | Good | 0 | |
| **1** | Brazil | Presidente Dutra | 41 | Good | 1 | Good | 5 | Good | 1 | |
| **2** | Italy | Priolo Gargallo | 66 | Moderate | 1 | Good | 39 | Good | 2 | |
| **3** | Poland | Przasnysz | 34 | Good | 1 | Good | 34 | Good | 0 | |
| **4** | France | Punaauia | 22 | Good | 0 | Good | 22 | Good | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **23458** | India | Gursahaiganj | 184 | Unhealthy | 3 | Good | 154 | Unhealthy | 2 | |
| **23459** | France | Sceaux | 50 | Good | 1 | Good | 20 | Good | 5 | |
| **23460** | India | Mormugao | 50 | Good | 1 | Good | 22 | Good | 1 | |
| **23461** | United States of America | Westerville | 71 | Moderate | 1 | Good | 44 | Good | 2 | |
| **23462** | Malaysia | Marang | 70 | Moderate | 1 | Good | 38 | Good | 0 | |

23035 rows × 12 columns

In [7]:

```
df1.isnull().sum()
```

Out[7]:

```
Country               0
City                  0
AQI Value             0
AQI Category          0
CO AQI Value          0
CO AQI Category       0
Ozone AQI Value       0
Ozone AQI Category    0
NO2 AQI Value         0
NO2 AQI Category      0
PM2.5 AQI Value       0
PM2.5 AQI Category    0
dtype: int64
```

In [8]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23035 entries, 0 to 23462
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Country             23035 non-null  object
 1   City                23035 non-null  object
 2   AQI Value           23035 non-null  int64 
 3   AQI Category        23035 non-null  object
 4   CO AQI Value        23035 non-null  int64 
 5   CO AQI Category     23035 non-null  object
 6   Ozone AQI Value     23035 non-null  int64 
 7   Ozone AQI Category  23035 non-null  object
 8   NO2 AQI Value       23035 non-null  int64 
 9   NO2 AQI Category    23035 non-null  object
 10  PM2.5 AQI Value     23035 non-null  int64 
 11  PM2.5 AQI Category  23035 non-null  object
dtypes: int64(5), object(7)
memory usage: 2.3+ MB
```

In [9]:

```
df1.describe()
```

Out[9]:

| | AQI Value | CO AQI Value | Ozone AQI Value | NO2 AQI Value | PM2.5 AQI Value |
|---|---|---|---|---|---|
| **count** | 23035.000000 | 23035.000000 | 23035.000000 | 23035.000000 | 23035.000000 |
| **mean** | 72.344693 | 1.376254 | 35.233905 | 3.084741 | 68.883482 |
| **std** | 56.360992 | 1.844926 | 28.236613 | 5.281708 | 55.057396 |
| **min** | 6.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 39.000000 | 1.000000 | 21.000000 | 0.000000 | 35.000000 |
| **50%** | 55.000000 | 1.000000 | 31.000000 | 1.000000 | 54.000000 |
| **75%** | 80.000000 | 1.000000 | 40.000000 | 4.000000 | 79.000000 |
| **max** | 500.000000 | 133.000000 | 235.000000 | 91.000000 | 500.000000 |

```
In [10]:
```

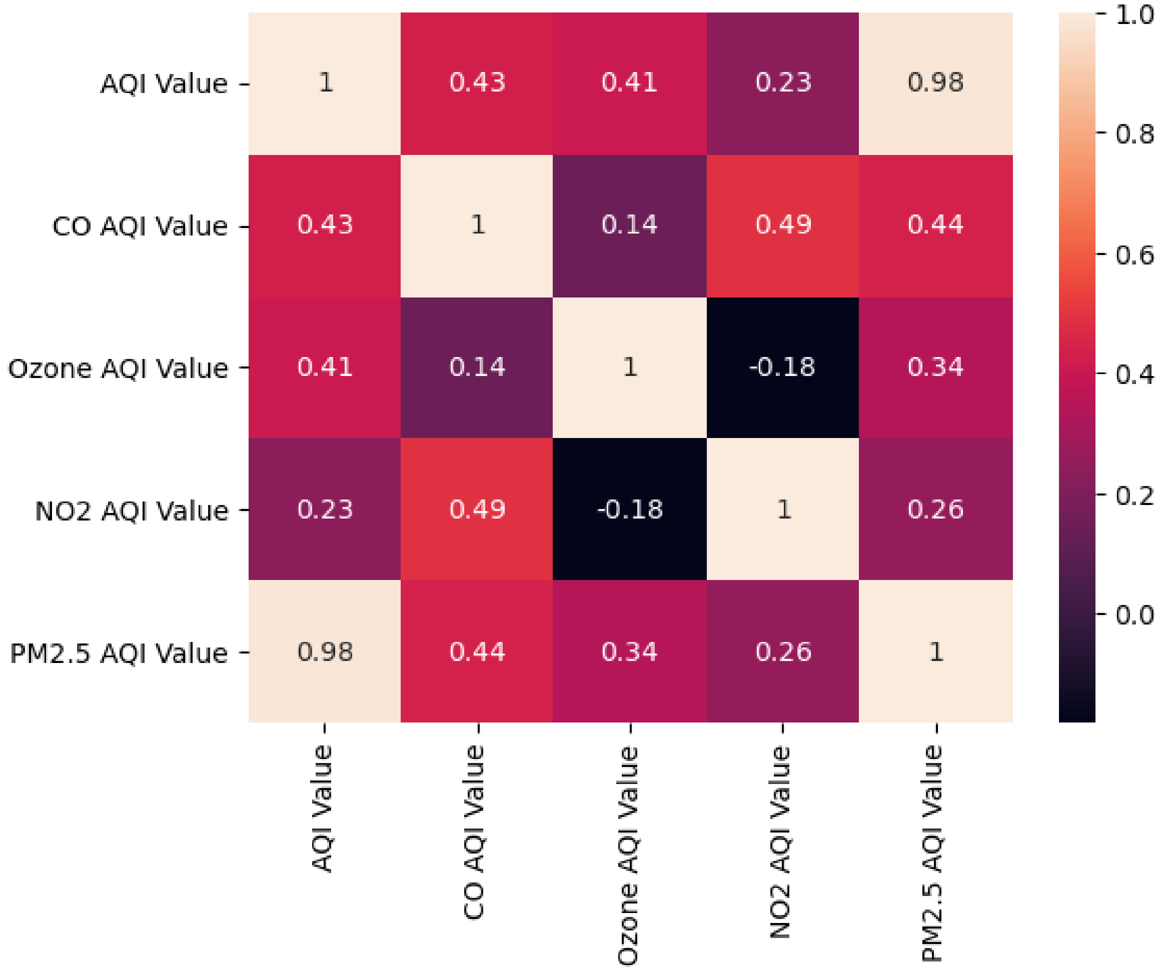
```
df1.corr()
```

```
Out[10]:
```

| | AQI Value | CO AQI Value | Ozone AQI Value | NO2 AQI Value | PM2.5 AQI Value |
|---|---|---|---|---|---|
| **AQI Value** | 1.000000 | 0.429643 | 0.405086 | 0.230845 | 0.984518 |
| **CO AQI Value** | 0.429643 | 1.000000 | 0.144838 | 0.487627 | 0.437751 |
| **Ozone AQI Value** | 0.405086 | 0.144838 | 1.000000 | -0.182934 | 0.340488 |
| **NO2 AQI Value** | 0.230845 | 0.487627 | -0.182934 | 1.000000 | 0.259084 |
| **PM2.5 AQI Value** | 0.984518 | 0.437751 | 0.340488 | 0.259084 | 1.000000 |

```
In [11]:
```

```
sns.heatmap(df1.corr(),annot=True)
plt.show()
```



# Classification algorithem

In [12]:

```
df1.head()
```

Out[12]:

| | Country | City | AQI Value | AQI Category | CO AQI Value | CO AQI Category | Ozone AQI Value | Ozone AQI Category | NO2 AQI Value | NO2 AQI Category |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Russian Federation | Praskoveya | 51 | Moderate | 1 | Good | 36 | Good | 0 | Good |
| **1** | Brazil | Presidente Dutra | 41 | Good | 1 | Good | 5 | Good | 1 | Good |
| **2** | Italy | Priolo Gargallo | 66 | Moderate | 1 | Good | 39 | Good | 2 | Good |
| **3** | Poland | Przasnysz | 34 | Good | 1 | Good | 34 | Good | 0 | Good |
| **4** | France | Punaauia | 22 | Good | 0 | Good | 22 | Good | 0 | Good |

In [13]:

```
df1.nunique()
```

Out[13]:

```
Country                  175
City                   23035
AQI Value                347
AQI Category               6
CO AQI Value              34
CO AQI Category            3
Ozone AQI Value          213
Ozone AQI Category         5
NO2 AQI Value             59
NO2 AQI Category           2
PM2.5 AQI Value          383
PM2.5 AQI Category         6
dtype: int64
```

# AQI Value AQI Category - classification

In [14]:

```
 # 4 lebel Encoding
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
```

In [15]:

```
df1['AQI Category']=lb.fit_transform(df1['AQI Category'])
```

```
C:\Users\Hp-\AppData\Local\Temp\ipykernel_35684\1468386655.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df1['AQI Category']=lb.fit_transform(df1['AQI Category'])
```

In [16]:

```
df1.head(2)
```

Out[16]:

| | Country | City | AQI Value | AQI Category | CO AQI Value | CO AQI Category | Ozone AQI Value | Ozone AQI Category | NO2 AQI Value | NO2 AQI Category |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Russian Federation | Praskoveya | 51 | 2 | 1 | Good | 36 | Good | 0 | Good |
| **1** | Brazil | Presidente Dutra | 41 | 0 | 1 | Good | 5 | Good | 1 | Good |

In [17]:

```
# 5 Define x and y as independent and dependent variable
```

In [18]:

```
df1.columns
```

Out[18]:

```
Index(['Country', 'City', 'AQI Value', 'AQI Category', 'CO AQI Value',
       'CO AQI Category', 'Ozone AQI Value', 'Ozone AQI Category',
       'NO2 AQI Value', 'NO2 AQI Category', 'PM2.5 AQI Value',
       'PM2.5 AQI Category'],
      dtype='object')
```

In [19]:

```
x=df1[['AQI Value']]
y=df1['AQI Category']
```

In [20]:

```
from sklearn.model_selection import train_test_split
```

In [21]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=.7)
```

In [22]:

```
x_train.head(),x_train.shape,x_test.head(),x_test.shape
```

Out[22]:

```
(       AQI Value
 20636         45
 23091         81
 12784        184
 10990         55
 18734         21,
 (16124, 1),
        AQI Value
 16920        180
 8105          15
 22292         66
 7989          56
 10069         67,
 (6911, 1))
```

In [23]:

```
y_train.head(),y_train.shape,y_test.head(),y_test.shape
```

Out[23]:

```
(20636    0
 23091    2
 12784    3
 10990    2
 18734    0
 Name: AQI Category, dtype: int32,
 (16124,),
 16920    3
 8105     0
 22292    2
 7989     2
 10069    2
 Name: AQI Category, dtype: int32,
 (6911,))
```

In [24]:

```
# import the Model
from sklearn.linear_model import LogisticRegression
logr=LogisticRegression()
```

In [25]:

```
# Train the model
logr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Out[25]:

```
LogisticRegression()
```

In [26]:

```
y_pred=logr.predict(x_test)
```

In [27]:

```
y_pred[:10],y_test.values[:10]
```

Out[27]:

```
(array([3, 0, 2, 2, 2, 0, 0, 2, 4, 0]), array([3, 0, 2, 2, 2, 0, 0, 2, 4,
0]))
```

In [28]:

```
logr.score(x_test,y_test)
```

Out[28]:

```
0.9972507596585154
```

In [29]:

```
logr.score(x_train,y_train)
```

Out[29]:

```
0.9972091292483255
```

In [30]:

```
y.value_counts()
```

Out[30]:

```
0    9688
2    9087
3    2215
4    1568
5     286
1     191
Name: AQI Category, dtype: int64
```

In [31]:

```
# evaluation
from sklearn.metrics import classification_report,confusion_matrix
```

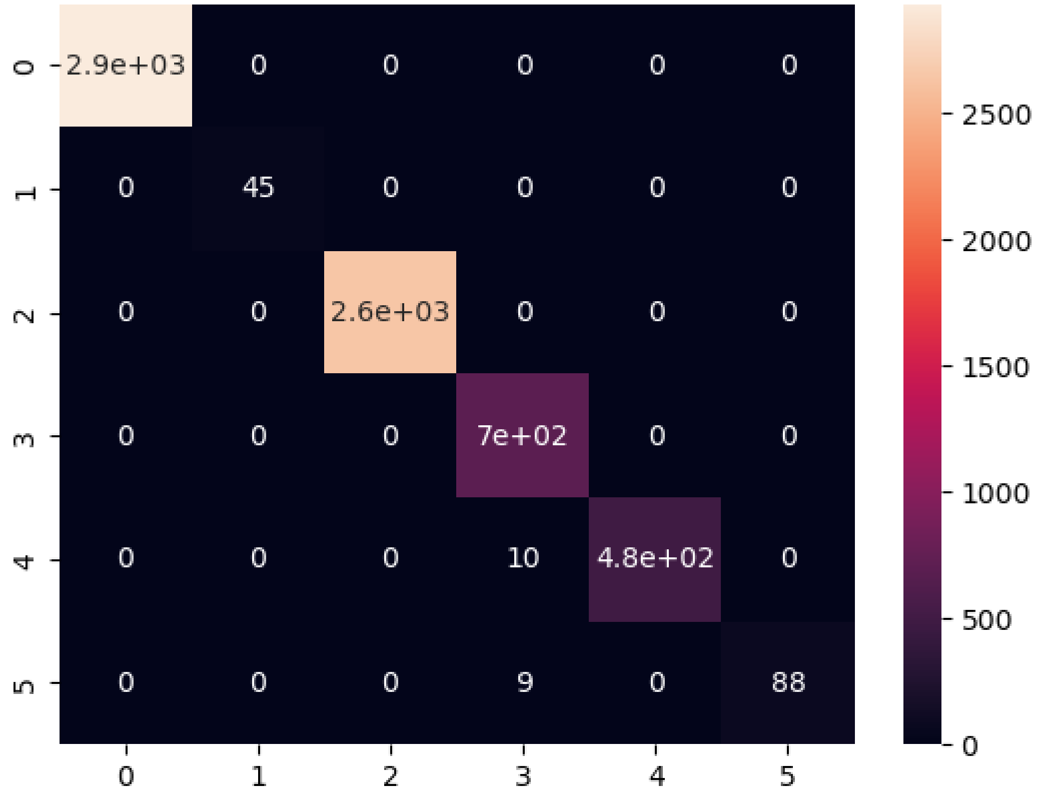
In [32]:

```
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[2934    0    0    0    0    0]
 [   0   45    0    0    0    0]
 [   0    0 2644    0    0    0]
 [   0    0    0  698    0    0]
 [   0    0    0   10  483    0]
 [   0    0    0    9    0   88]]
```

In [33]:

```
sns.heatmap(cm,annot=True)
plt.show()
```



In [34]:

```
cr=classification_report(y_test,y_pred)
print(cr)
```

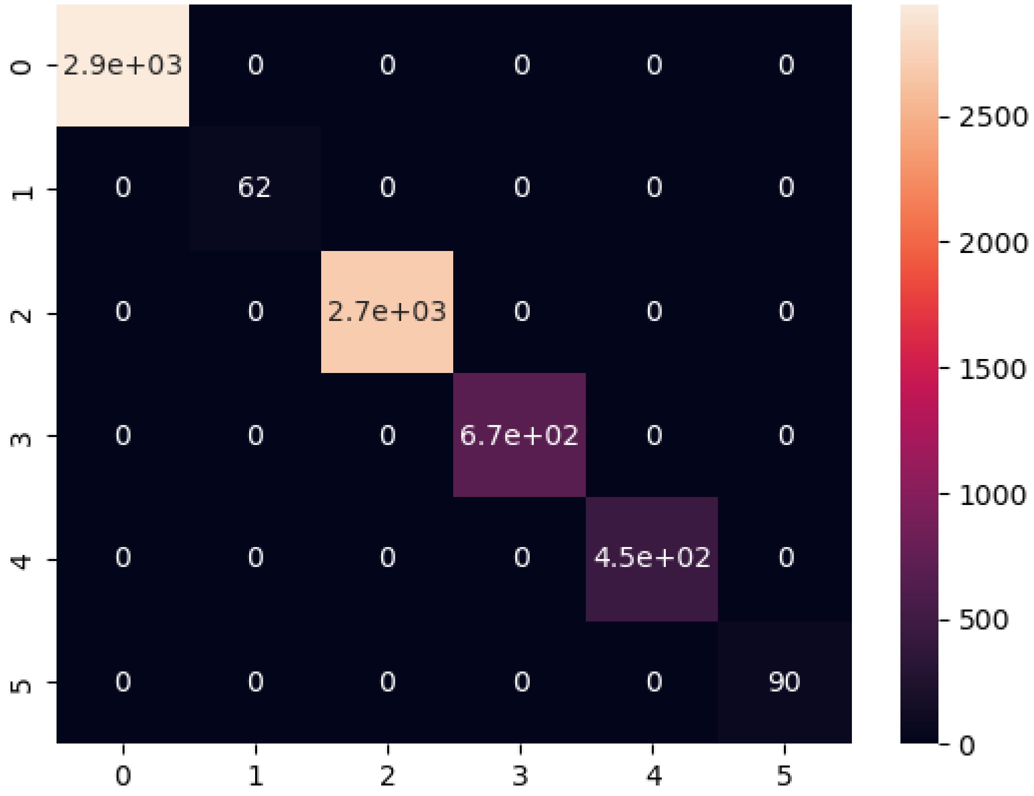
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2934
           1       1.00      1.00      1.00        45
           2       1.00      1.00      1.00      2644
           3       0.97      1.00      0.99       698
           4       1.00      0.98      0.99       493
           5       1.00      0.91      0.95        97

    accuracy                           1.00      6911
   macro avg       1.00      0.98      0.99      6911
weighted avg       1.00      1.00      1.00      6911
```

In [35]:

```
 # 4 lebel Encoding
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()

df1['AQI Category']=lb.fit_transform(df1['AQI Category'])

df1.head(2)

# 5 Define x and y as independent and dependent variable

df1.columns

x=df1[['AQI Value']]
y=df1['AQI Category']

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=.7)

x_train.head(),x_train.shape,x_test.head(),x_test.shape

y_train.head(),y_train.shape,y_test.head(),y_test.shape

# import the Model
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()

# Train the model
model.fit(x_train,y_train)

y_pred=model.predict(x_test)

print(y_pred[:10],y_test.values[:10])

print(model.score(x_test,y_test))

print(model.score(x_train,y_train))

y.value_counts()

# evaluation
from sklearn.metrics import classification_report,confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print(cm)

sns.heatmap(cm,annot=True)
plt.show()

cr=classification_report(y_test,y_pred)
print(cr)
```

```
[2 2 2 4 0 0 0 2 0 2] [2 2 2 4 0 0 0 2 0 2]
1.0
1.0
[[2945    0    0    0    0    0]
 [   0   62    0    0    0    0]
 [   0    0 2696    0    0    0]
 [   0    0    0  669    0    0]
 [   0    0    0    0  449    0]
 [   0    0    0    0    0   90]]
```

```
C:\Users\Hp-\AppData\Local\Temp\ipykernel_35684\1634432991.py:5: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df1['AQI Category']=lb.fit_transform(df1['AQI Category'])
```



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2945
           1       1.00      1.00      1.00        62
           2       1.00      1.00      1.00      2696
           3       1.00      1.00      1.00       669
           4       1.00      1.00      1.00       449
           5       1.00      1.00      1.00        90

    accuracy                           1.00      6911
```

```
   macro avg       1.00      1.00      1.00      6911
weighted avg       1.00      1.00      1.00      6911
```

In [36]:

```
x.head(2),y.head(2)
```

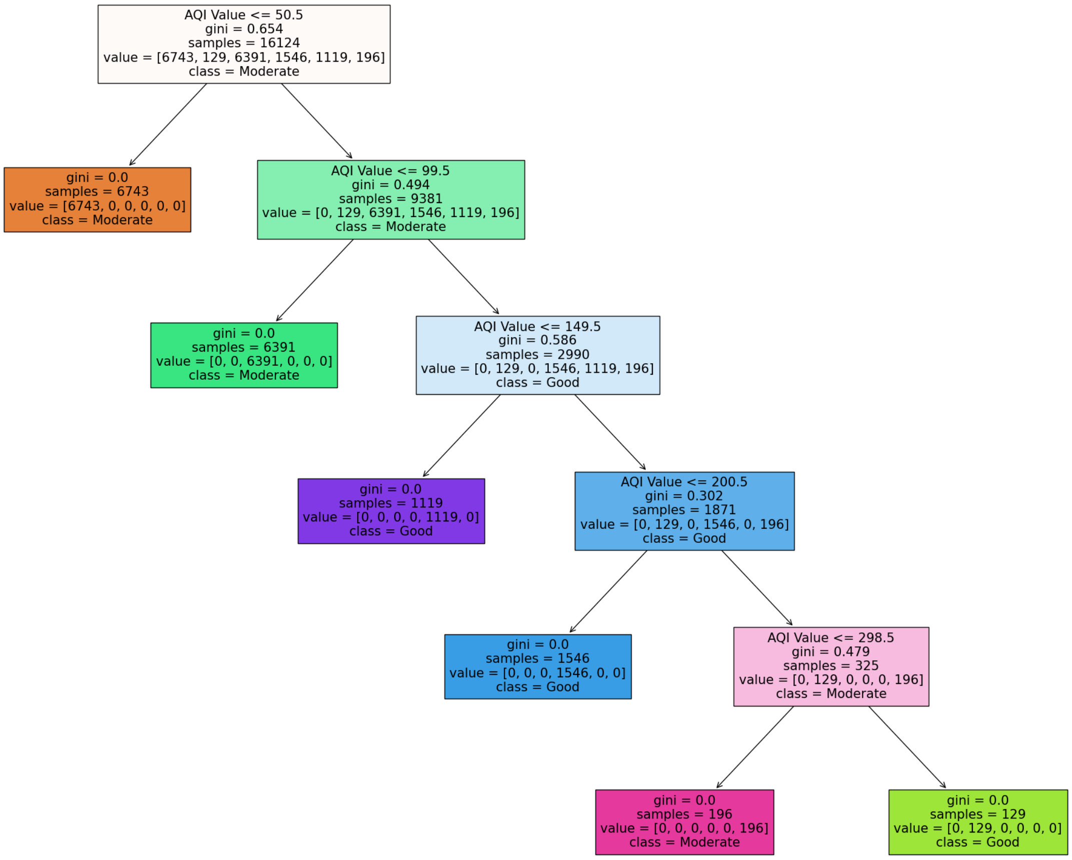
Out[36]:

```
(   AQI Value
 0         51
 1         41,
 0    2
 1    0
 Name: AQI Category, dtype: int64)
```

In [37]:

```
model.score(x_test,y_test)
from sklearn import tree
fn = x.columns
tn = df['AQI Category']
print(fn)
print(tn)
fig = plt.figure(figsize=(25,20))
tree.plot_tree(model, feature_names=fn,class_names=tn,filled=True)
plt.show()
```

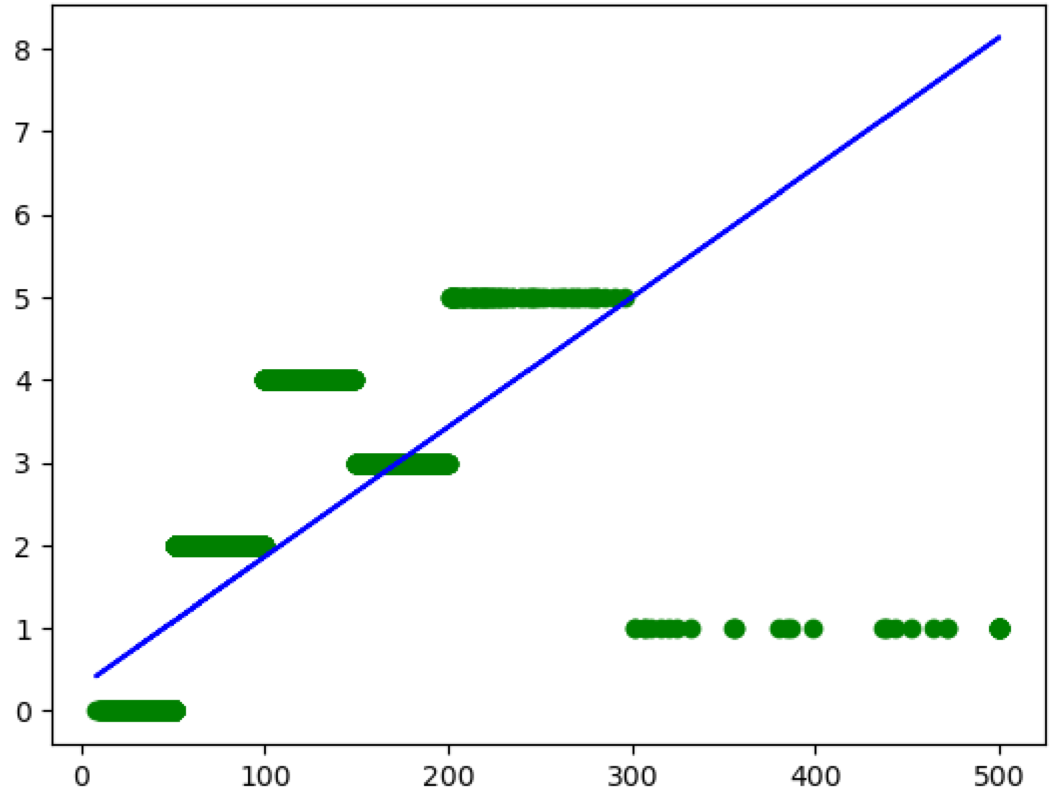
```
Index(['AQI Value'], dtype='object')
0         Moderate
1             Good
2         Moderate
3             Good
4             Good
           ...
23458    Unhealthy
23459         Good
23460         Good
23461     Moderate
23462     Moderate
Name: AQI Category, Length: 23463, dtype: object
```



In [38]:

```
from sklearn.linear_model import LinearRegression
slr=LinearRegression()

slr.fit(x_train,y_train)
y_pred=slr.predict(x_test)

print(slr.score(x_train,y_train))
print(slr.score(x_test,y_test))

from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score
print("Mean Absolute Error:- ",mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:- ",mean_squared_error(y_test,y_pred))
print("r2_score:- ",r2_score(y_test,y_pred))

a=slr.coef_
b=slr.intercept_

plt.scatter(x_test['AQI Value'],y_test,color='g')
plt.plot(x_test['AQI Value'],y_pred,color='b')

plt.show()
```

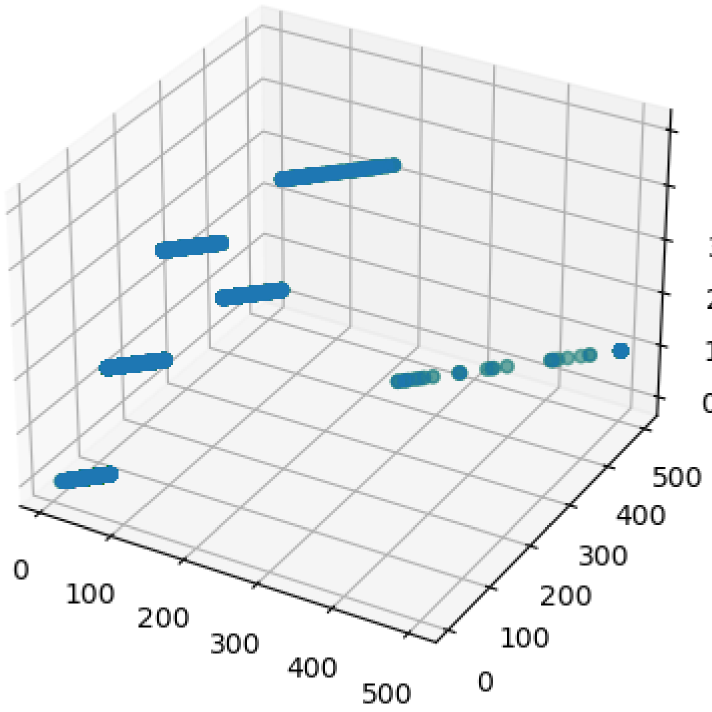
```
0.4150687205522784
0.3825910520522813
Mean Absolute Error:-  0.8373566685462246
Mean Squared Error:-  1.1316170219657296
r2_score:-  0.3825910520522813
```



In [39]:

```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()

print(dtr.fit(x_train, y_train))

y_pred_dtr=dtr.predict(x_test)

print(y_pred_dtr[:5])
print(y_test.values[:5])

print('------accuracy score--------')
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score
from math import sqrt

print('mean_absolute_error:-',mean_absolute_error(y_test,y_pred_dtr))
print('mean_squared_error:-', mean_squared_error(y_test,y_pred_dtr))

mse=mean_squared_error(y_test,y_pred_dtr)

print('r2_score',r2_score(y_test,y_pred_dtr))
print('MODEL SCORE', dtr.score(x_test,y_test))

print(sqrt(mse))

ax = plt.axes (projection ='3d')
ax.scatter3D(x_test['AQI Value'],x_test['AQI Value'],y_test,color='g')
ax.scatter3D(x_test['AQI Value'],x_test['AQI Value'],y_pred_dtr, 'b')
plt.show()
```

```
DecisionTreeRegressor()
[2. 2. 2. 4. 0.]
[2 2 2 4 0]
------accuracy score--------
mean_absolute_error:- 0.0
mean_squared_error:- 0.0
r2_score 1.0
MODEL SCORE 1.0
0.0
```


0
1
2
3
4
5
0
100
200
300
400
500
0
100
200
300
400
500