In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
path2=r"C:\Users\JAYADEVA JAVALI\Downloads\archive (1)\city_day.csv"
df2=pd.read_csv(path2)
```

In [3]:
```
df2.head()
```

Out[3]:

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Tolu
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	1
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	1
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	1

In [4]:
```
df2.shape
```

Out[4]:
```
(29531, 16)
```

In [5]:
```
df2.isnull().sum()
```

Out[5]:
```
City             0
Date             0
PM2.5         4598
PM10         11140
NO            3582
NO2           3585
NOx           4185
NH3          10328
CO            2059
SO2           3854
O3            4022
Benzene       5623
Toluene       8041
Xylene       18109
AQI           4681
AQI_Bucket    4681
dtype: int64
```

In [6]:
```
df2['City'].value_counts()
```

Out[6]:
```
Chennai              2009
Delhi                2009
Lucknow              2009
Mumbai               2009
Bengaluru            2009
Ahmedabad            2009
Hyderabad            2006
Patna                1858
Gurugram             1679
Visakhapatnam        1462
Amritsar             1221
```

```
Jorapokhar            1169
Jaipur                1114
Thiruvananthapuram    1112
Amaravati              951
Brajrajnagar           938
Talcher                925
Kolkata                814
Guwahati               502
Coimbatore             386
Shillong               310
Chandigarh             304
Bhopal                 289
Kochi                  162
Ernakulam              162
Aizawl                 113
Name: City, dtype: int64
```

In [7]:
```
df_bangalore = df2[df2['City']=='Bengaluru']
```

In [8]:
```
df_bangalore.head()
```

Out[8]:

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Tol
4294	Bengaluru	2015-01-01	NaN	NaN	3.26	17.33	10.88	20.36	0.33	3.54	10.73	0.56	
4295	Bengaluru	2015-01-02	NaN	NaN	6.05	19.73	14.14	23.74	1.35	3.97	22.77	0.65	
4296	Bengaluru	2015-01-03	NaN	NaN	11.91	19.88	20.72	4.32	17.40	13.61	12.03	0.53	
4297	Bengaluru	2015-01-04	NaN	NaN	7.45	21.61	16.88	0.87	5.05	6.52	17.70	0.55	
4298	Bengaluru	2015-01-05	NaN	NaN	9.52	22.17	21.76	31.38	1.83	4.71	12.72	0.40	

In [9]:
```
df_bangalore.isnull().sum()
```

Out[9]:
```
City             0
Date             0
PM2.5          146
PM10           360
NO               6
NO2              6
NOx              4
NH3            203
CO              11
SO2              6
O3             144
Benzene        266
Toluene         93
Xylene        2009
AQI             99
AQI_Bucket      99
dtype: int64
```

In [10]:
```
for c in df_bangalore.columns:
    if df_bangalore[c].isna().sum() > 0:
        if df_bangalore[c].dtype == 'float64':
            df_bangalore[c] = df_bangalore[c].fillna(df_bangalore[c].mean())
        elif df_bangalore[c].dtype == 'object':
            df_bangalore[c] = df_bangalore[c].fillna(df_bangalore[c].value_counts().
```

```
<ipython-input-10-9761a85f9dbf>:4: SettingWithCopyWarning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_bangalore[c] = df_bangalore[c].fillna(df_bangalore[c].mean())
<ipython-input-10-9761a85f9dbf>:6: SettingWithCopyWarning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_bangalore[c] = df_bangalore[c].fillna(df_bangalore[c].value_counts().index[0])
```

In [11]:
```
df_bangalore.isnull().sum()
```

Out[11]:
```
City             0
Date             0
PM2.5            0
PM10             0
NO               0
NO2              0
NOx              0
NH3              0
CO               0
SO2              0
O3               0
Benzene          0
Toluene          0
Xylene        2009
AQI              0
AQI_Bucket       0
dtype: int64
```

In [12]:
```
df_bangalore.drop(['City' ,'Date','NO','NOx','Benzene' , 'Toluene', 'Xylene','AQI_Bu
```

```
E:\anaconda\lib\site-packages\pandas\core\frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

In [13]:
```
df_bangalore.head()
```

Out[13]:

	PM2.5	PM10	NO2	NH3	CO	SO2	O3	AQI
4294	35.819828	83.243287	17.33	20.36	0.33	3.54	10.73	94.318325
4295	35.819828	83.243287	19.73	23.74	1.35	3.97	22.77	94.318325
4296	35.819828	83.243287	19.88	4.32	17.40	13.61	12.03	94.318325
4297	35.819828	83.243287	21.61	0.87	5.05	6.52	17.70	94.318325
4298	35.819828	83.243287	22.17	31.38	1.83	4.71	12.72	94.318325

In [14]:
```
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
SEED=42
```

In [15]:
```
X = df_bangalore.drop(["AQI"], axis = 1).copy()
y = df_bangalore["AQI"].copy()
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=
```

Linear Regression

In [17]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
prediction = regressor.predict(X_test)
rmse_Lreg = np.sqrt(mean_squared_error(y_test, prediction))
print('RMSE value is = {}'.format(rmse_Lreg))
r2_Lreg = r2_score(y_test, prediction)
print('R-squared value is {}'.format(r2_Lreg))
```

```
RMSE value is = 28.837558642976433
R-squared value is 0.45406941719671523
```

RandomForest Regressor

In [18]:
```
RFreg_model = RandomForestRegressor()
RFreg_model.fit(X_train,y_train)
prediction2 = RFreg_model.predict(X_test)
rmse_RFreg = np.sqrt(mean_squared_error(y_test, prediction2))
print('RMSE value is = {}'.format(rmse_RFreg))
r2_RFreg = r2_score(y_test, prediction2)
print('R-squared value is {}'.format(r2_RFreg))
from sklearn.metrics import mean_absolute_error
rmse_rfreg1=mean_absolute_error(y_test,prediction2)
print('Mean Absolute Error value is {}'.format(rmse_rfreg1))
```

```
RMSE value is = 23.038329366985145
R-squared value is 0.6515644903068056
Mean Absolute Error value is 13.86730071689256
```

Decision Tree Regressor

In [19]:
```
regressor1 = DecisionTreeRegressor(random_state = 0)
regressor1.fit(X_train, y_train)
prediction4 = regressor1.predict(X_test)
dt_reg = np.sqrt(mean_squared_error(y_test, prediction4))
print('RMSE value is = {}'.format(dt_reg))
r2_dt_reg = r2_score(y_test, prediction4)
print('R-squared value is {}'.format(r2_dt_reg))
```

```
RMSE value is = 40.16733109512667
R-squared value is -0.05917176470374441
```

XGBoost Regressor

In [20]:
```
xgbr = XGBRegressor(learning_rate = 0.1, n_estimators = 200, random_state = SEED)
xgbr.fit(X_train,y_train)
prediction5 = xgbr.predict(X_test)
xgbr_reg = np.sqrt(mean_squared_error(y_test, prediction5))
print('RMSE value is = {}'.format(xgbr_reg))
r2_xgbr_reg = r2_score(y_test, prediction5)
print('R-squared value is {}'.format(r2_xgbr_reg))
```

```
RMSE value is = 26.03309425188638
R-squared value is 0.5550901467188109
```

Polynomial Regressor

In [21]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree = 4)
x_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(x_poly, y_train)
prediction3 = regressor.predict(poly_reg.transform(X_test))

ploy_reg = np.sqrt(mean_squared_error(y_test, prediction3))
print('RMSE value is = {}'.format(ploy_reg))
r2_poly_reg = r2_score(y_test, prediction3)
print('R-squared value is {}'.format(r2_poly_reg))
```

```
RMSE value is = 272.66661012258635
R-squared value is -47.80725246724345
```

In [22]:
```
Result= pd.DataFrame({'Actual AQI':y_test,'Predicted AQI By LinearRegression':predic
Result
```

Out[22]:

	Actual AQI	Predicted AQI By LinearRegression	Predicted AQI By RandomForest	Predicted AQI By DecisionTreeRegressor
4584	140.0	89.910457	127.929099	166.000000
5637	78.0	78.125859	72.043183	79.000000
5463	77.0	76.684229	76.356366	92.000000
5827	118.0	114.686694	116.460000	146.000000
5558	69.0	81.357502	71.200000	48.000000
...	...	...	...	...
5009	134.0	103.884138	103.779550	163.000000
5708	141.0	107.581695	128.760000	352.000000
6040	72.0	67.679778	67.280000	69.000000
4909	60.0	74.129877	63.133183	94.318325
4492	156.0	134.334912	163.860000	309.000000

603 rows × 4 columns

In [23]:
```
Result= pd.DataFrame({'Actual Price':y_test,'Predicted AQI By LinearRegression':pred
Result
```

Out[23]:

	Actual Price	Predicted AQI By LinearRegression	Predicted AQI By RandomForest	Predicted AQI By PolynomialRegressor	Predicted AQI By DecisionTreeRegressor	Predicte XgbReg
4584	140.0	89.910457	127.929099	116.336539	166.000000	126.9
5637	78.0	78.125859	72.043183	73.720825	79.000000	71.9
5463	77.0	76.684229	76.356366	78.588618	92.000000	73.5
5827	118.0	114.686694	116.460000	96.527714	146.000000	114.1
5558	69.0	81.357502	71.200000	157.870045	48.000000	87.1
...	...	...	...	...	...	
5009	134.0	103.884138	103.779550	168.928260	163.000000	100.7
5708	141.0	107.581695	128.760000	111.140106	352.000000	125.8

	Actual Price	Predicted AQI By LinearRegression	Predicted AQI By RandomForest	Predicted AQI By PolynomialRegressor	Predicted AQI By DecisionTreeRegressor	Predicte XgbReg
6040	72.0	67.679778	67.280000	66.246164	69.000000	66.6
4909	60.0	74.129877	63.133183	87.775968	94.318325	61.4
4492	156.0	134.334912	163.860000	129.398762	309.000000	191.0

603 rows × 6 columns

In [24]:
```
final=pd.DataFrame({'Actual AQI':y_test,'Predicted AQI by RF':prediction2})
```

In [25]:
```
final
```

Out[25]:

	Actual AQI	Predicted AQI by RF
4584	140.0	127.929099
5637	78.0	72.043183
5463	77.0	76.356366
5827	118.0	116.460000
5558	69.0	71.200000
...	...	...
5009	134.0	103.779550
5708	141.0	128.760000
6040	72.0	67.280000
4909	60.0	63.133183
4492	156.0	163.860000

603 rows × 2 columns

In [26]:
```
quality_list = []
```

In [27]:
```
for i in final['Actual AQI']:
    if 50 >= i >= 0:
        quality_list.append("GOOD")
    elif 100 >= i > 50:
        quality_list.append("SATISFACTORY")
    elif 200>= i > 100:
        quality_list.append("MODERATE")
    elif 300 >= i > 200:
        quality_list.append("POOR")
    elif 400>= i > 300:
        quality_list.append("VERY POOR")
    elif 500 >= i > 400:
        quality_list.append("SEVERE")
final['AQI Pool'] = quality_list
```

In [28]:
```
final['AQI Pool'].unique()
```

Out[28]:
```
array(['MODERATE', 'SATISFACTORY', 'GOOD', 'POOR'], dtype=object)
```

In [29]:
```
quality_list1=[]
```

In [30]:
```
for i in final['Predicted AQI by RF']:
    if 50 >= i >= 0:
        quality_list1.append("GOOD")
    elif 100 >= i > 50:
        quality_list1.append("SATISFACTORY")
    elif 200>= i > 100:
        quality_list1.append("MODERATE")
    elif 300 >= i > 200:
        quality_list1.append("POOR")
    elif 400>= i > 300:
        quality_list1.append("VERY POOR")
    elif 500 >= i > 400:
        quality_list1.append("SEVERE")
final['Predicted AQI Pool'] = quality_list1
```

In [31]:
```
final
```

Out[31]:

	Actual AQI	Predicted AQI by RF	AQI Pool	Predicted AQI Pool
4584	140.0	127.929099	MODERATE	MODERATE
5637	78.0	72.043183	SATISFACTORY	SATISFACTORY
5463	77.0	76.356366	SATISFACTORY	SATISFACTORY
5827	118.0	116.460000	MODERATE	MODERATE
5558	69.0	71.200000	SATISFACTORY	SATISFACTORY
...	...	...	...	...
5009	134.0	103.779550	MODERATE	MODERATE
5708	141.0	128.760000	MODERATE	MODERATE
6040	72.0	67.280000	SATISFACTORY	SATISFACTORY
4909	60.0	63.133183	SATISFACTORY	SATISFACTORY
4492	156.0	163.860000	MODERATE	MODERATE

603 rows × 4 columns

In [32]:
```
len(final)
```

Out[32]:
```
603
```

In [33]:
```
l1 = list(final['AQI Pool'])
l2 = list(final['Predicted AQI Pool'])
```

In [34]:
```
final['AQI Pool'].value_counts()
```

Out[34]:
```
SATISFACTORY    359
MODERATE        188
GOOD             41
POOR             15
Name: AQI Pool, dtype: int64
```

In [35]:
```
final['Predicted AQI Pool'].value_counts()
```

Out[35]:
```
SATISFACTORY    372
```

```
MODERATE        221
GOOD              8
POOR              2
Name: Predicted AQI Pool, dtype: int64
```

In [36]:
```
count=0
```

In [37]:
```
for i in range(len(l1)):
    if l1[i]==l2[i]:
        count+=1
print(count)
```

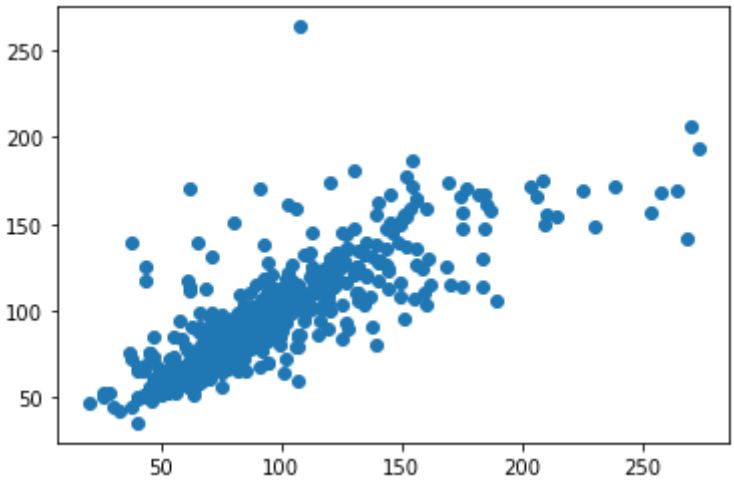
```
478
```

In [38]:
```
print("Accuracy:",count/len(final)*100)
```

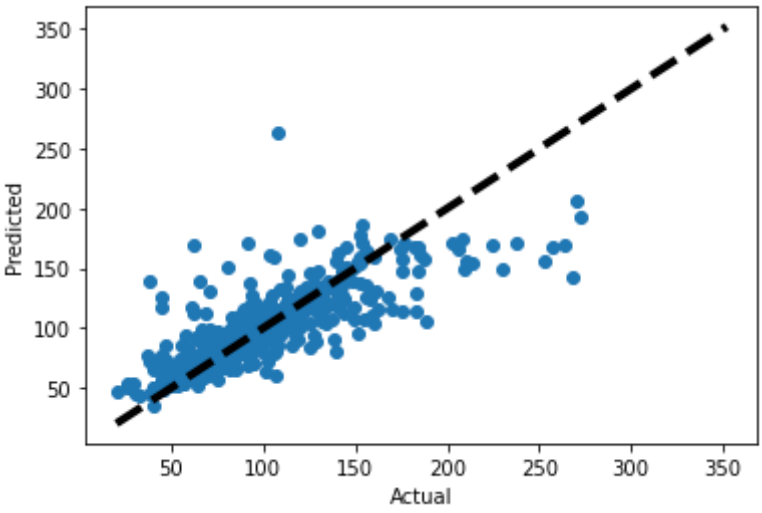
```
Accuracy: 79.27031509121062
```

In [39]:
```
import seaborn as sns
from matplotlib import pyplot as plt
plt.scatter(y_test,prediction2)
```

Out[39]:
```
<matplotlib.collections.PathCollection at 0x22812d73250>
```



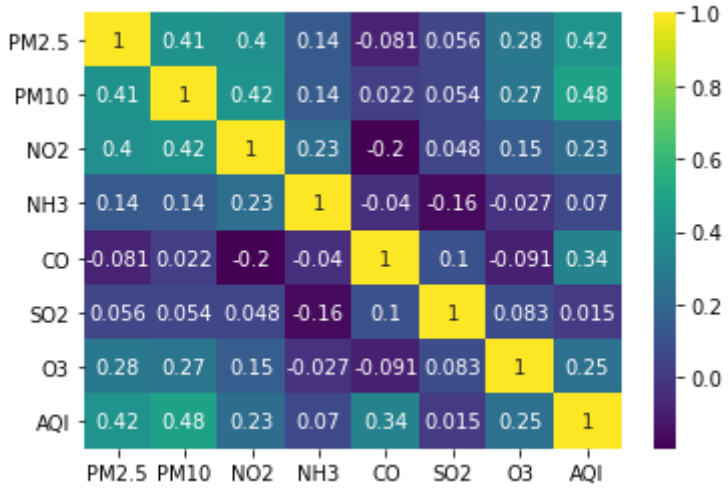
In [40]:
```
fig, ax = plt.subplots()
ax.scatter(y_test, prediction2)
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw = 4)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
plt.show()
```




```
sns.heatmap(df_bangalore.corr(),cmap="viridis",annot=True)
```

In [41]:

Out[41]: <AxesSubplot:>



In [ ]:

In [ ]: