Perform the following operations using ‘python’ Language on the Air quality dataset.

1. Data cleaning
2. Data integration
3. Data transformation
4. Data model Building

In [1]:
```
import io
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

rawdata_df = pd.read_csv(r"C:\Users\yasha\Downloads\AirQualityUCI.csv")
rawdata_df
```

Out[1]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT
0	3/10/2004	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.
1	3/10/2004	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.
2	3/10/2004	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.
3	3/10/2004	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.
4	3/10/2004	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.
...	...	...	...	...	...	...	...	.
9466	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9467	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9468	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9469	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9470	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

9471 rows × 17 columns

In [2]:
```
rawdata_df.isnull().sum()
```

Out[2]:
```
Date              114
Time              114
CO(GT)            114
PT08.S1(CO)       114
NMHC(GT)          114
C6H6(GT)          114
PT08.S2(NMHC)     114
NOx(GT)           114
PT08.S3(NOx)      114
NO2(GT)           114
PT08.S4(NO2)      114
PT08.S5(O3)       114
T                 114
RH                114
AH                114
Unnamed: 15      9471
Unnamed: 16      9471
dtype: int64
```

In [3]:
```
rawdata_df.columns
```

Out[3]:
```
Index(['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(GT)',
       'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO
2)',
       'PT08.S5(O3)', 'T', 'RH', 'AH', 'Unnamed: 15', 'Unnamed: 16'],
      dtype='object')
```

In [108]:
```
# data cleaning
```

In [4]:

```
#we seleted only the columns having values
selected_columns=['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(
analysis_df = rawdata_df[selected_columns].copy()
analysis_df
```

Out[4]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT
0	3/10/2004	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.
1	3/10/2004	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.
2	3/10/2004	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.
3	3/10/2004	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.
4	3/10/2004	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.
...	...	...	...	...	...	...	...	.
9466	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9467	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9468	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9469	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
9470	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

9471 rows × 15 columns

In [5]:

```
analysis_df.shape
```

Out[5]:

```
(9471, 15)
```

In [6]:

```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9471 entries, 0 to 9470
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           9357 non-null   object 
 1   Time           9357 non-null   object 
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   float64
 5   C6H6(GT)       9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
 12  T              9357 non-null   float64
 13  RH             9357 non-null   float64
 14  AH             9357 non-null   float64
dtypes: float64(13), object(2)
memory usage: 1.1+ MB
```

In [7]:

```
#dropping the all null values
analysis_df=analysis_df.dropna()
analysis_df
```

Out[7]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT
0	3/10/2004	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.
1	3/10/2004	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.
2	3/10/2004	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.
3	3/10/2004	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.
4	3/10/2004	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.
...	...	...	...	...	...	...	...	.
9352	4/4/2005	10:00:00	3.1	1314.0	-200.0	13.5	1101.0	472.
9353	4/4/2005	11:00:00	2.4	1162.0	-200.0	11.4	1027.0	353.
9354	4/4/2005	12:00:00	2.4	1142.0	-200.0	12.4	1062.0	293.
9355	4/4/2005	13:00:00	2.1	1002.0	-200.0	9.5	960.0	234.
9356	4/4/2005	14:00:00	2.2	1071.0	-200.0	11.9	1047.0	265.

9357 rows × 15 columns

In [113]:

```
analysis_df.isnull().sum()
```

Out[113]:

```
Date             0
Time             0
CO(GT)           0
PT08.S1(CO)      0
NMHC(GT)         0
C6H6(GT)         0
PT08.S2(NMHC)    0
NOx(GT)          0
PT08.S3(NOx)     0
NO2(GT)          0
PT08.S4(NO2)     0
PT08.S5(O3)      0
T                0
RH               0
AH               0
dtype: int64
```

In [114]:

```
#data transformation
analysis_df['Date']
```

Out[114]:

```
0        3/10/2004
1        3/10/2004
2        3/10/2004
3        3/10/2004
4        3/10/2004
           ...
9352      4/4/2005
9353      4/4/2005
9354      4/4/2005
9355      4/4/2005
9356      4/4/2005
Name: Date, Length: 9357, dtype: object
```

In [115]:

```
analysis_df['Date'] = pd.to_datetime(analysis_df.Date)
analysis_df['Date']
```

```
C:\Users\yasha\AppData\Local\Temp\ipykernel_3164\2240121029.py:1: SettingWit
hCopyWarning:
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)
  analysis_df['Date'] = pd.to_datetime(analysis_df.Date)
```

Out[115]:

```
0      2004-03-10
1      2004-03-10
2      2004-03-10
3      2004-03-10
4      2004-03-10
          ...
9352   2005-04-04
9353   2005-04-04
9354   2005-04-04
9355   2005-04-04
9356   2005-04-04
Name: Date, Length: 9357, dtype: datetime64[ns]
```

In [116]:
```
analysis_df['Year'] = pd.DatetimeIndex(analysis_df['Date']).year
analysis_df['month'] = pd.DatetimeIndex(analysis_df['Date']).month

analysis_df
```

```
C:\Users\yasha\AppData\Local\Temp\ipykernel_3164\1901626825.py:1: SettingWit
hCopyWarning:
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)
  analysis_df['Year'] = pd.DatetimeIndex(analysis_df['Date']).year
C:\Users\yasha\AppData\Local\Temp\ipykernel_3164\1901626825.py:2: SettingWit
hCopyWarning:
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)
  analysis_df['month'] = pd.DatetimeIndex(analysis_df['Date']).month
```

Out[116]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	F
0	2004-03-10	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	
1	2004-03-10	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	
2	2004-03-10	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	
3	2004-03-10	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	
4	2004-03-10	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	
...	...	...	...	...	...	...	...	...	
9352	2005-04-04	10:00:00	3.1	1314.0	-200.0	13.5	1101.0	472.0	
9353	2005-04-04	11:00:00	2.4	1162.0	-200.0	11.4	1027.0	353.0	
9354	2005-04-04	12:00:00	2.4	1142.0	-200.0	12.4	1062.0	293.0	
9355	2005-04-04	13:00:00	2.1	1002.0	-200.0	9.5	960.0	234.0	
9356	2005-04-04	14:00:00	2.2	1071.0	-200.0	11.9	1047.0	265.0	

9357 rows × 17 columns

In [117]:

```
analysis_df['month_apha'] = pd.to_datetime(analysis_df['month'], format='%m')
analysis_df
```

```
C:\Users\yasha\AppData\Local\Temp\ipykernel_3164\675708319.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)
  analysis_df['month_apha'] = pd.to_datetime(analysis_df['month'], format='%
m').dt.month_name()
```

Out[117]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	F
0	2004-03-10	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	
1	2004-03-10	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	
2	2004-03-10	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	
3	2004-03-10	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	
4	2004-03-10	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	
...	...	...	...	...	...	...	...	...	
9352	2005-04-04	10:00:00	3.1	1314.0	-200.0	13.5	1101.0	472.0	
9353	2005-04-04	11:00:00	2.4	1162.0	-200.0	11.4	1027.0	353.0	
9354	2005-04-04	12:00:00	2.4	1142.0	-200.0	12.4	1062.0	293.0	
9355	2005-04-04	13:00:00	2.1	1002.0	-200.0	9.5	960.0	234.0	
9356	2005-04-04	14:00:00	2.2	1071.0	-200.0	11.9	1047.0	265.0	

9357 rows × 18 columns

In [118]:

```
#integrate
data1 = analysis_df[['Date','Time','CO(GT)']].loc[0:15]
data1
```

Out[118]:

	Date	Time	CO(GT)
0	2004-03-10	18:00:00	2.6
1	2004-03-10	19:00:00	2.0
2	2004-03-10	20:00:00	2.2
3	2004-03-10	21:00:00	2.2
4	2004-03-10	22:00:00	1.6
5	2004-03-10	23:00:00	1.2
6	2004-03-11	0:00:00	1.2
7	2004-03-11	1:00:00	1.0
8	2004-03-11	2:00:00	0.9
9	2004-03-11	3:00:00	0.6
10	2004-03-11	4:00:00	-200.0
11	2004-03-11	5:00:00	0.7
12	2004-03-11	6:00:00	0.7
13	2004-03-11	7:00:00	1.1
14	2004-03-11	8:00:00	2.0
15	2004-03-11	9:00:00	2.2

In [119]:
```
data2 = analysis_df[['Date','Time','CO(GT)']].loc[16:30]
data2
```

Out[119]:

	Date	Time	CO(GT)
16	2004-03-11	10:00:00	1.7
17	2004-03-11	11:00:00	1.5
18	2004-03-11	12:00:00	1.6
19	2004-03-11	13:00:00	1.9
20	2004-03-11	14:00:00	2.9
21	2004-03-11	15:00:00	2.2
22	2004-03-11	16:00:00	2.2
23	2004-03-11	17:00:00	2.9
24	2004-03-11	18:00:00	4.8
25	2004-03-11	19:00:00	6.9
26	2004-03-11	20:00:00	6.1
27	2004-03-11	21:00:00	3.9
28	2004-03-11	22:00:00	1.5
29	2004-03-11	23:00:00	1.0
30	2004-03-12	0:00:00	1.7

In [120]:
```
integrate=pd.merge(data1,data2,on='Date',how='inner')
integrate
```

Out[120]:

	Date	Time_x	CO(GT)_x	Time_y	CO(GT)_y
0	2004-03-11	0:00:00	1.2	10:00:00	1.7
1	2004-03-11	0:00:00	1.2	11:00:00	1.5
2	2004-03-11	0:00:00	1.2	12:00:00	1.6
3	2004-03-11	0:00:00	1.2	13:00:00	1.9
4	2004-03-11	0:00:00	1.2	14:00:00	2.9
...	...	...	...	...	...
135	2004-03-11	9:00:00	2.2	19:00:00	6.9
136	2004-03-11	9:00:00	2.2	20:00:00	6.1
137	2004-03-11	9:00:00	2.2	21:00:00	3.9
138	2004-03-11	9:00:00	2.2	22:00:00	1.5
139	2004-03-11	9:00:00	2.2	23:00:00	1.0

140 rows × 5 columns

In [23]:

```
analysis_df['AH'] = analysis_df['AH'].apply(lambda x:0 if x<=0 else 1)
analysis_df
```

```
C:\Users\yasha\AppData\Local\Temp\ipykernel_6980\768665957.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)
  analysis_df['AH'] = analysis_df['AH'].apply(lambda x:0 if x<=0 else 1)
```

Out[23]:

Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT0
3/10/2004	18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	
3/10/2004	19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	
3/10/2004	20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	
3/10/2004	21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	
3/10/2004	22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	
...	...	...	...	...	...	...	...	
4/4/2005	10:00:00	3.1	1314.0	-200.0	13.5	1101.0	472.0	
4/4/2005	11:00:00	2.4	1162.0	-200.0	11.4	1027.0	353.0	
4/4/2005	12:00:00	2.4	1142.0	-200.0	12.4	1062.0	293.0	
4/4/2005	13:00:00	2.1	1002.0	-200.0	9.5	960.0	234.0	
4/4/2005	14:00:00	2.2	1071.0	-200.0	11.9	1047.0	265.0	

rows × 15 columns

In [24]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [25]:
```
X = analysis_df.drop(['AH','Date','Time'],axis=1)
X
```

Out[25]:

	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NC
0	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	
1	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	
2	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	
3	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	
4	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	
...	...	...	...	...	...	...	...	
9352	3.1	1314.0	-200.0	13.5	1101.0	472.0	538.0	
9353	2.4	1162.0	-200.0	11.4	1027.0	353.0	604.0	
9354	2.4	1142.0	-200.0	12.4	1062.0	293.0	603.0	
9355	2.1	1002.0	-200.0	9.5	960.0	234.0	702.0	
9356	2.2	1071.0	-200.0	11.9	1047.0	265.0	654.0	

9357 rows × 12 columns

In [26]:
```
Y = analysis_df['AH']
Y
```

Out[26]:
```
0       1
1       1
2       1
3       1
4       1
       ..
9352    1
9353    1
9354    1
9355    1
9356    1
Name: AH, Length: 9357, dtype: int64
```

In [27]:
```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.3)
```

In [28]:
```
logreg = LogisticRegression()
logreg.fit(X_train,Y_train)
```

Out[28]:
```
▾ LogisticRegression
LogisticRegression()
```

In [29]:

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

In [30]:

```
y_pred = logreg.predict(X_test)
```

In [31]:

```
print(confusion_matrix(Y_test, y_pred))
```

```
[[ 114    0]
 [   0 2694]]
```

In [32]:

```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       114
           1       1.00      1.00      1.00      2694

    accuracy                           1.00      2808
   macro avg       1.00      1.00      1.00      2808
weighted avg       1.00      1.00      1.00      2808
```