In [1]:

```
import pandas as pd
data = pd.read_csv('data.csv', encoding='cp1252')
print(data)
```

```
       stn_code       sampling_date                       state   location
\
0         150.0  February - M021990              Andhra Pradesh  Hyderabad
1         151.0  February - M021990              Andhra Pradesh  Hyderabad
2         152.0  February - M021990              Andhra Pradesh  Hyderabad
3         150.0     March - M031990              Andhra Pradesh  Hyderabad
4         151.0     March - M031990              Andhra Pradesh  Hyderabad
...         ...                 ...                         ...        ...
435737     SAMP            24-12-15                 West Bengal   ULUBERIA
435738     SAMP            29-12-15                 West Bengal   ULUBERIA
435739      NaN                 NaN  andaman-and-nicobar-islands        NaN
435740      NaN                 NaN                 Lakshadweep        NaN
435741      NaN                 NaN                     Tripura        NaN

                                        agency  \
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN
...                                        ...
435737  West Bengal State Pollution Control Board
435738  West Bengal State Pollution Control Board
435739                                     NaN
435740                                     NaN
435741                                     NaN

                                    type   so2   no2   rspm  spm  \
0       Residential, Rural and other Areas   4.8  17.4    NaN  NaN
1                        Industrial Area   3.1   7.0    NaN  NaN
2       Residential, Rural and other Areas   6.2  28.5    NaN  NaN
3       Residential, Rural and other Areas   6.3  14.7    NaN  NaN
4                        Industrial Area   4.7   7.5    NaN  NaN
...                                  ...   ...   ...    ...  ...
435737                             RIRUO  22.0  50.0  143.0  NaN
435738                             RIRUO  20.0  46.0  171.0  NaN
435739                               NaN   NaN   NaN    NaN  NaN
435740                               NaN   NaN   NaN    NaN  NaN
435741                               NaN   NaN   NaN    NaN  NaN

              location_monitoring_station  pm2_5        date
0                                     NaN    NaN  1990-02-01
1                                     NaN    NaN  1990-02-01
2                                     NaN    NaN  1990-02-01
3                                     NaN    NaN  1990-03-01
4                                     NaN    NaN  1990-03-01
...                                   ...    ...         ...
435737  Inside Rampal Industries,ULUBERIA    NaN  2015-12-24
435738  Inside Rampal Industries,ULUBERIA    NaN  2015-12-29
435739                                NaN    NaN         NaN
435740                                NaN    NaN         NaN
435741                                NaN    NaN         NaN

[435742 rows x 13 columns]
```

```
C:\Users\prath\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:
3444: DtypeWarning: Columns (0) have mixed types.Specify dtype option on imp
ort or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [2]:

```
print(data.shape)
print(data.columns)
```

```
(435742, 13)
Index(['stn_code', 'sampling_date', 'state', 'location', 'agency', 'type',
       'so2', 'no2', 'rspm', 'spm', 'location_monitoring_station', 'pm2_5',
       'date'],
      dtype='object')
```

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
Data columns (total 13 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   stn_code                     291665 non-null  object
 1   sampling_date                435739 non-null  object
 2   state                        435742 non-null  object
 3   location                     435739 non-null  object
 4   agency                       286261 non-null  object
 5   type                         430349 non-null  object
 6   so2                          401096 non-null  float64
 7   no2                          419509 non-null  float64
 8   rspm                         395520 non-null  float64
 9   spm                          198355 non-null  float64
 10  location_monitoring_station  408251 non-null  object
 11  pm2_5                        9314 non-null    float64
 12  date                         435735 non-null  object
dtypes: float64(5), object(8)
memory usage: 43.2+ MB
```

In [4]:

```
data.isnull().sum()
```

Out[4]:

```
stn_code                     144077
sampling_date                     3
state                             0
location                          3
agency                       149481
type                           5393
so2                           34646
no2                           16233
rspm                          40222
spm                          237387
location_monitoring_station   27491
pm2_5                        426428
date                              7
dtype: int64
```

In [5]:

```
data.count()   #It results in a number of non null values in each column.
```

Out[5]:

```
stn_code                     291665
sampling_date                435739
state                        435742
location                     435739
agency                       286261
type                         430349
so2                          401096
no2                          419509
rspm                         395520
spm                          198355
location_monitoring_station  408251
pm2_5                          9314
date                         435735
dtype: int64
```

In [6]:

```
data.describe()
```

Out[6]:

	so2	no2	rspm	spm	pm2_5
count	401096.000000	419509.000000	395520.000000	198355.000000	9314.000000
mean	10.829414	25.809623	108.832784	220.783480	40.791467
std	11.177187	18.503086	74.872430	151.395457	30.832525
min	0.000000	0.000000	0.000000	0.000000	3.000000
25%	5.000000	14.000000	56.000000	111.000000	24.000000
50%	8.000000	22.000000	90.000000	187.000000	32.000000
75%	13.700000	32.200000	142.000000	296.000000	46.000000
max	909.000000	876.000000	6307.033333	3380.000000	504.000000

Cleansing the dataset *In this step, we need to clean the data by adding and dropping the needed and unwanted data respectively. *

From the above dataset,

Dropping of less valued columns: stn_code, agency, sampling_date, location_monitoring_agency do not add much value to the dataset in terms of information. Therefore, we can drop those columns.

Changing the types to uniform format: When you see the dataset, you may notice that the ‘type’ column has values such as ‘Industrial Area’ and ‘Industrial Areas’ — both actually mean the same, so let’s remove such type of stuff and make it uniform.

Creating a year column To view the trend over a period of time, we need year values for each row and also when you see in most of the values in date column only has ‘year’ value. So, let’s create a new column holding year values.

1.stn_code, agency, sampling_date, location_monitoring_agency do not add much value to the dataset in terms of information. Therefore, we can drop those columns.

2.Dropping rows where no date is available.

In [7]:

```
data=data.drop(['stn_code', 'agency','sampling_date','location_monitoring_station'], axis =
```

In [8]:

```
data=data.dropna(subset=['date']) # dropping rows where no date is available
```

In [9]:

```
data.columns
```

Out[9]:

```
Index(['state', 'location', 'type', 'so2', 'no2', 'rspm', 'spm', 'pm2_5',
       'date'],
      dtype='object')
```

Changing the types to uniform format: Notice that the 'type' column has values such as 'Industrial Area' and 'Industrial Areas'—both actually mean the same, so let's remove them and make it uniform

In [10]:

```
data["type"].unique()
```

Out[10]:

```
array(['Residential, Rural and other Areas', 'Industrial Area', nan,
       'Sensitive Area', 'Industrial Areas', 'Residential and others',
       'Sensitive Areas', 'Industrial', 'Residential', 'RIRUO',
       'Sensitive'], dtype=object)
```

In [11]:

```
types = {
    "Residential": "R",
    "Residential and others": "RO",
    "Residential, Rural and other Areas": "RRO",
    "Industrial Area": "I",
    "Industrial Areas": "I",
    "Industrial": "I",
    "Sensitive Area": "S",
    "Sensitive Areas": "S",
    "Sensitive": "S",
    "NaN": "RRO"
}
data.type = data.type.replace(types)
```

In [12]:

```
data.head(5)
```

Out[12]:

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	NaN	NaN	NaN	1990-02-01
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	NaN	NaN	NaN	1990-02-01
3	Andhra Pradesh	Hyderabad	RRO	6.3	14.7	NaN	NaN	NaN	1990-03-01
4	Andhra Pradesh	Hyderabad	I	4.7	7.5	NaN	NaN	NaN	1990-03-01

Creating a year column To view the trend over a period of time, we need year values for each row and also when you see in most of the values in date column only has 'year' value. So, let's create a new column holding year values.

In [13]:

```
data['date'] = pd.to_datetime(data['date'], errors='coerce')
data.head(5)
```

Out[13]:

	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	NaN	NaN	NaN	1990-02-01
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	NaN	NaN	NaN	1990-02-01
3	Andhra Pradesh	Hyderabad	RRO	6.3	14.7	NaN	NaN	NaN	1990-03-01
4	Andhra Pradesh	Hyderabad	I	4.7	7.5	NaN	NaN	NaN	1990-03-01

In [14]:

```
data['year'] = data.date.dt.year
data.head(5)
```

Out[14]:

	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	NaN	NaN	NaN	1990-02-01	1990
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01	1990
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	NaN	NaN	NaN	1990-02-01	1990
3	Andhra Pradesh	Hyderabad	RRO	6.3	14.7	NaN	NaN	NaN	1990-03-01	1990
4	Andhra Pradesh	Hyderabad	I	4.7	7.5	NaN	NaN	NaN	1990-03-01	1990

Handling Missing Values The column such as SO2, NO2, rspm, spm, pm2_5 are the ones which contribute much to our analysis. So, we need to remove null from those columns to avoid inaccuracy in the prediction. We use the Imputer from sklearn.preprocessing to fill the missing values in every column with the mean.

In [15]:

```
# defining columns of importance, which shall be used reguarly
COLS = ['so2', 'no2', 'rspm', 'spm', 'pm2_5']
```

In [16]:

```
import numpy as np
from sklearn.impute import SimpleImputer
# invoking SimpleImputer to fill missing values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
data[COLS] = imputer.fit_transform(data[COLS])
```

In [17]:

```
print(data.head(5))
print(data.info())
```

```
           state   location type  so2   no2        rspm        spm  \
0  Andhra Pradesh  Hyderabad  RRO  4.8  17.4  108.833091  220.78348
1  Andhra Pradesh  Hyderabad    I  3.1   7.0  108.833091  220.78348
2  Andhra Pradesh  Hyderabad  RRO  6.2  28.5  108.833091  220.78348
3  Andhra Pradesh  Hyderabad  RRO  6.3  14.7  108.833091  220.78348
4  Andhra Pradesh  Hyderabad    I  4.7   7.5  108.833091  220.78348

       pm2_5       date  year
0  40.791467 1990-02-01  1990
1  40.791467 1990-02-01  1990
2  40.791467 1990-02-01  1990
3  40.791467 1990-03-01  1990
4  40.791467 1990-03-01  1990
<class 'pandas.core.frame.DataFrame'>
Int64Index: 435735 entries, 0 to 435738
Data columns (total 10 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   state     435735 non-null  object
 1   location  435735 non-null  object
 2   type      430345 non-null  object
 3   so2       435735 non-null  float64
 4   no2       435735 non-null  float64
 5   rspm      435735 non-null  float64
 6   spm       435735 non-null  float64
 7   pm2_5     435735 non-null  float64
 8   date      435735 non-null  datetime64[ns]
 9   year      435735 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(1), object(3)
memory usage: 36.6+ MB
None
```

Data Transformation All machine learning algorithms are based on mathematics. So, we need to convert all the columns into numerical format.

Taking a broader perspective, data is classified into numerical and categorical data:

Numerical: As the name suggests, this is numeric data that is quantifiable. Categorical: The data is a string or non-numeric data that is qualitative in nature.

1. Encoding To address the problems associated with categorical data, we can use encoding. This is the process by which we convert a categorical variable into a numerical form. Here, we will look at three simple methods of encoding categorical data.
2. Replacing This is a technique in which we replace the categorical data with a number. This is a simple replacement and does not involve much logical processing. Let's look at an exercise to get a better idea of this.

Simple Replacement of Categorical Data with a Number

In [18]:

```
print(data.head(5))
data['type'].value_counts()
```

```
           state   location type  so2   no2        rspm        spm  \
0  Andhra Pradesh  Hyderabad  RRO  4.8  17.4  108.833091  220.78348
1  Andhra Pradesh  Hyderabad    I  3.1   7.0  108.833091  220.78348
2  Andhra Pradesh  Hyderabad  RRO  6.2  28.5  108.833091  220.78348
3  Andhra Pradesh  Hyderabad  RRO  6.3  14.7  108.833091  220.78348
4  Andhra Pradesh  Hyderabad    I  4.7   7.5  108.833091  220.78348

       pm2_5       date  year
0  40.791467 1990-02-01  1990
1  40.791467 1990-02-01  1990
2  40.791467 1990-02-01  1990
3  40.791467 1990-03-01  1990
4  40.791467 1990-03-01  1990
```

Out[18]:

```
RRO      179013
I        148069
RO        86791
S         15010
RIRUO      1304
R           158
Name: type, dtype: int64
```

In [19]:

```
data['type'].replace({"RRO":1, "I":2, "RO":3,"S":4,"RIRUO":5,"R":6}, inplace= True)
data['type']
```

Out[19]:

```
0         1.0
1         2.0
2         1.0
3         1.0
4         2.0
         ...
435734    5.0
435735    5.0
435736    5.0
435737    5.0
435738    5.0
Name: type, Length: 435735, dtype: float64
```

In [20]:

```
#Converting Categorical Data to Numerical Data Using Label Encoding
#print(data['state'].value_counts())
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
data["state"]=labelencoder.fit_transform(data["state"])
print(data)
```

```
        state   location  type   so2   no2        rspm        spm      pm2_5
\
0           0  Hyderabad   1.0   4.8  17.4  108.833091  220.78348  40.791467
1           0  Hyderabad   2.0   3.1   7.0  108.833091  220.78348  40.791467
2           0  Hyderabad   1.0   6.2  28.5  108.833091  220.78348  40.791467
3           0  Hyderabad   1.0   6.3  14.7  108.833091  220.78348  40.791467
4           0  Hyderabad   2.0   4.7   7.5  108.833091  220.78348  40.791467
...       ...        ...   ...   ...   ...         ...        ...        ...
435734     33   ULUBERIA   5.0  20.0  44.0  148.000000  220.78348  40.791467
435735     33   ULUBERIA   5.0  17.0  44.0  131.000000  220.78348  40.791467
435736     33   ULUBERIA   5.0  18.0  45.0  140.000000  220.78348  40.791467
435737     33   ULUBERIA   5.0  22.0  50.0  143.000000  220.78348  40.791467
435738     33   ULUBERIA   5.0  20.0  46.0  171.000000  220.78348  40.791467

             date  year
0      1990-02-01  1990
1      1990-02-01  1990
2      1990-02-01  1990
3      1990-03-01  1990
4      1990-03-01  1990
...           ...   ...
435734 2015-12-15  2015
435735 2015-12-18  2015
435736 2015-12-21  2015
435737 2015-12-24  2015
435738 2015-12-29  2015

[435735 rows x 10 columns]
```

In [21]:

```
#One Hot Encoding
dfAndhra=data[(data['state']==0)]
print(dfAndhra)
```

```
       state     location  type  so2   no2        rspm        spm      pm2_5
\
0          0    Hyderabad   1.0  4.8  17.4  108.833091  220.78348  40.791467
1          0    Hyderabad   2.0  3.1   7.0  108.833091  220.78348  40.791467
2          0    Hyderabad   1.0  6.2  28.5  108.833091  220.78348  40.791467
3          0    Hyderabad   1.0  6.3  14.7  108.833091  220.78348  40.791467
4          0    Hyderabad   2.0  4.7   7.5  108.833091  220.78348  40.791467
...      ...          ...   ...  ...   ...         ...        ...        ...
26363      0  Rajahmundry   2.0  7.0  13.0   71.000000  220.78348  40.791467
26364      0  Rajahmundry   2.0  7.0  18.0   77.000000  220.78348  40.791467
26365      0  Rajahmundry   2.0  8.0  23.0   64.000000  220.78348  40.791467
26366      0  Rajahmundry   2.0  7.0  19.0   61.000000  220.78348  40.791467
26367      0  Rajahmundry   2.0  6.0  17.0   71.000000  220.78348  40.791467

            date  year
0     1990-02-01  1990
1     1990-02-01  1990
2     1990-02-01  1990
3     1990-03-01  1990
4     1990-03-01  1990
...          ...   ...
26363 2015-12-13  2015
26364 2015-12-16  2015
26365 2015-12-19  2015
26366 2015-12-22  2015
26367 2015-12-25  2015

[26368 rows x 10 columns]
```

In [22]:

```
dfAndhra['location'].value_counts()
```

Out[22]:

```
Hyderabad         7764
Visakhapatnam     7108
Vijayawada        2093
Chittoor          1003
Tirupati           986
Kurnool            857
Patancheru         698
Guntur             629
Nalgonda           618
Ramagundam         554
Nellore            408
Khammam            385
Warangal           336
Ananthapur         324
Ongole             317
Kadapa             316
Srikakulam         315
Rajahmundry        311
Eluru              300
Vishakhapatnam     297
Kakinada           288
Vizianagaram       282
Sangareddy          85
Karimnagar          67
Nizamabad           27
Name: location, dtype: int64
```

In [23]:

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder=OneHotEncoder(sparse=False,handle_unknown='error',drop='first')
pd.DataFrame(onehotencoder.fit_transform(dfAndhra[["location"]]))
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26365	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26367	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

26368 rows × 24 columns