

Explore your Data: Exploratory Data Analysis

Would you like to explore your data? Let's learn by analyzing India Air Quality dataset



Bala Kowsalya

[Follow](#)

Feb 26, 2019 · 8 min read

Are you the one, who have gathered loads of data and wondering how to unravel the hidden treasure of knowledge behind it?

No need to wait anymore, let's start exploring data!



Explore data to get the most out of it ~ Unsplash

Quick Fact: The article in Forbes states that ‘*The amount of data we produce every day is truly mind-boggling. There are 2.5 quintillion bytes of data created each day at our current pace*’. Omg! It’s really spellbinding.

. . .

A huge amount of data is being produced in every click and tap of the users. This data is helpful in giving customized service to each user with the help of data analysis. So, it’s really a big deal right?

The process of analyzing data is chunked down into several steps and there comes our EDA as a top of the list.

Exploratory Data Analysis is an approach analyzing data sets to summarize their main characteristics such as mean, standard deviation, and count, so on, often with visual methods.

Exploratory Data Analysis (EDA) is an approach to analyzing data. It’s where the researcher takes a bird’s eye view of the data and tries to make some sense of it. It’s often the first step in data analysis, implemented before any formal statistical techniques are applied.

. . .

India Air Quality Dataset

As always, learning by doing is the best practice to understand deeper. So now, we are going to make our hands dirt by analyzing **India Air Quality Dataset**(*not the recently updated one, but anyways we are just going to use it as an example*) downloaded from **Kaggle**.

Wherever you get the dataset, you first need to analyze the structure, domain, and contents of it thoroughly.



Air Pollution ~ Unsplash

Dataset: Basic Info

What is it about?

This data is released by the Ministry of Environment and Forests and Central Pollution Control Board of India under the National Data Sharing and Accessibility Policy (NDSAP), using this one can explore India's air pollution levels at a more granular scale.

What information it has?

The dataset has 13 columns which are,

- **stn_code**: Station Code
- **sampling_date**: Date of sampling (note how this is formatted)
- **state**: State
- **location**: Location of recording
- **agency**: Agency
- **type**: Type of area
- **so2**: Sulphur dioxide ($\mu\text{g}/\text{m}^3$)

- **no2**: Nitrogen dioxide ($\mu\text{g}/\text{m}^3$)
- **rspm**: Respirable Suspended Particulate Matter ($\mu\text{g}/\text{m}^3$)
- **spm**: Suspended Particulate Matter ($\mu\text{g}/\text{m}^3$)
- **location_monitoring_station**: Location of data collection
- **pm2_5**: PSI 2.5 ($\mu\text{g}/\text{m}^3$)
- **date**: Date of sampling

Why we need these details?

SPM, RSPM, PM2.5 values are the parameters used to measure the quality of air based on the number of particles present in it. Using these values, we are going to identify the air quality over the period of time in different states of India.

But, how?

This can only be answered once we analyze the data. 😊

. . .

Let's start: Import the dataset

Download the dataset from this link — [data.csv](#).

Create a new Jupyter Notebook and import the packages needed.

Read the .csv file into Pandas dataframe:

Now, read the data into pandas DataFrame using `read_csv()` and display the first few rows with `head()`, by default `head()` will return first 5 rows of the dataset, but you can specify any number of rows like `head(10)`.

	stn_code	sampling_date	state	location	agency	type	so2	no2	rspm	spm	location_monitoring_station	pm2_5	date
0	150	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	4.8	17.4	NaN	NaN	NaN	NaN	1990-02-01
1	151	February - M021990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	3.1	7.0	NaN	NaN	NaN	NaN	1990-02-01
2	152	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.2	28.5	NaN	NaN	NaN	NaN	1990-02-01
3	150	March - M031990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.3	14.7	NaN	NaN	NaN	NaN	1990-03-01

4	151	March - M031990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	4.7	7.5	NaN	NaN	NaN	NaN	1990-03-01
---	-----	-----------------	----------------	-----------	-----	-----------------	-----	-----	-----	-----	-----	-----	------------

Sample rows from India Air Quality dataset

Yes, we got the data now!

. . .

Check the dataset info

As discussed earlier, the dataset has 13 columns in it. So how many rows are there? Many of the cells are filled with NaN, which is an unknown value and cannot contribute to our analysis. So how many such types of values are there? and how can we get rid of those? let's find the answer to these questions.

To proceed with EDA, the initial level of investigation of data can be done using several commands,

data.shape

It returns a number of rows and columns in a dataset.

```
(435742, 13)
```

data.isnull().sum()

It returns a number of null values in each column.

```

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
dtype: int64
```

7

data.info()

It returns range, column, number of non-null objects of each column, datatype and memory usage.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
Data columns (total 13 columns):
stn_code                291665 non-null object
sampling_date           435739 non-null object
state                   435742 non-null object
location                435739 non-null object
agency                  286261 non-null object
type                    430349 non-null object
so2                     401096 non-null float64
no2                     419509 non-null float64
rspm                    395520 non-null float64
spm                     198355 non-null float64
location_monitoring_station 408251 non-null object
pm2_5                   9314 non-null float64
date                    435735 non-null object
dtypes: float64(5), object(8)
memory usage: 43.2+ MB
```

data.count()

It results in a number of non null values in each column.

```
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
```

Summarised details

Generate descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values.

	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

count: Count number of non-NA/null observations

mean: Mean of the values

std:Standard deviation of the observations

min: Minimum of the values in the object

max: Maximum of the values in the object

A subset of a DataFrame including/excluding columns based on their dtype.

. . .

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.

	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	Andhra Pradesh	Hyderabad	RRO	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	Andhra Pradesh	Hyderabad	I	4.7	7.5	108.833091	220.78348	40.791467	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**.

Now, check the number of null values in each column.

```
state      0
location   0
type       0
so2        0
no2        0
rspm       0
spm        0
pm2_5      0
date       0
dtype: int64
```


Yes! there are no missing values, we filled it using mean values. Now, our dataset looks like this.

	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	Andhra Pradesh	Hyderabad	RRO	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	Andhra Pradesh	Hyderabad	RRO	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	Andhra Pradesh	Hyderabad	RRO	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	Andhra Pradesh	Hyderabad	I	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01	1990

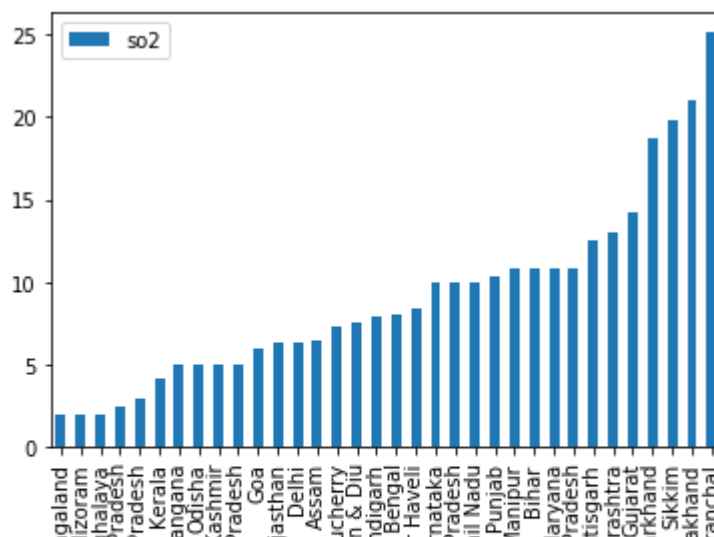
• • •

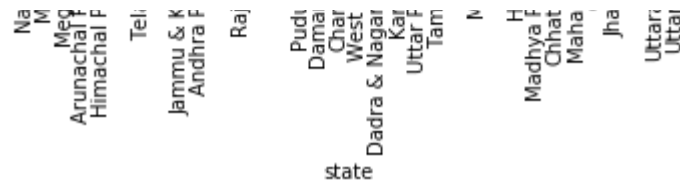
All set! Ready to go...

Every preprocessing step are done, let's find out some higher level information from it. As I said earlier, so2, no2, rspm, and spm are the parameters that determine air quality in a particular locality. Now, let's frame a question and get the answer from data.

Which is the state that has higher SO2 content?

Group the data based on states and find the median for so2 content over a period of time, sort it and we will get the states with higher and lower level SO2 content.

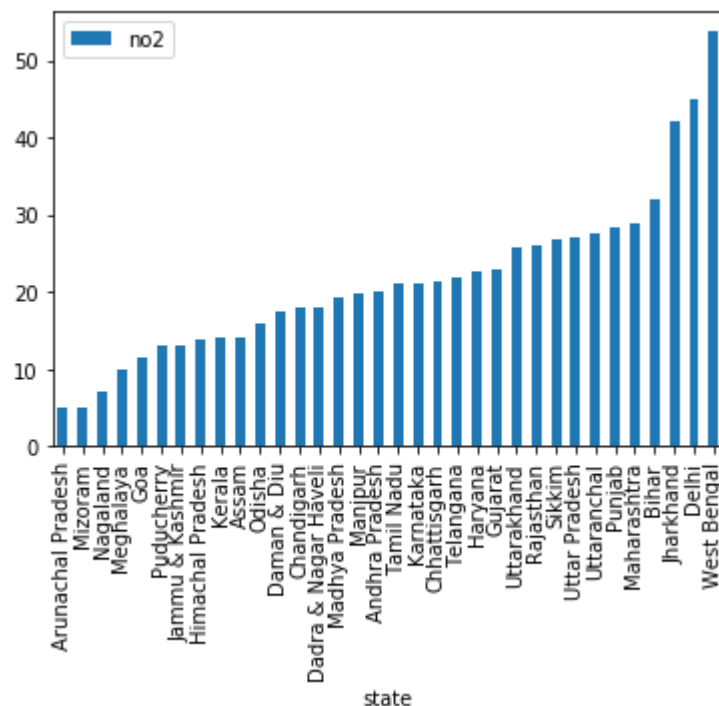




Observation: Plotting for SO₂, we can see that the **top** state is Uttaranchal, while the **bottom** state is Meghalaya.

Which is the state that has higher NO₂ content?

Again the same process, but now for NO₂ value, group the data based on states and find the median for NO₂ content over a period of time, sort it and we will get the states with higher and lower level NO₂ content.



Observation: Plotting for NO₂, we can see that the **top** state is West Bengal, while the **bottom** state is Mizoram.

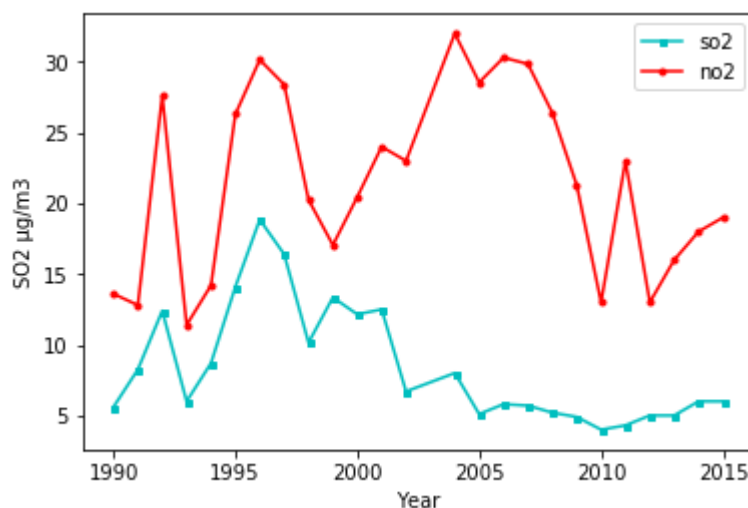
💡 **Exercise:** In the same way, as an exercise generate a graph for rspm and spm values and find the state with max and min rspm and spm value.

What is the yearly trend in a particular state, say 'Andhra Pradesh'?

We have created a new dataframe containing the NO₂, SO₂, rspm, and spm data regarding state 'Andhra Pradesh' only and group it by 'year'.

	so2	no2	rspm	spm
year				
1990	5.60	13.6	108.833091	179.000000
1991	8.25	12.8	108.833091	141.500000
1992	12.40	27.6	108.833091	192.000000
1993	6.00	11.4	108.833091	220.78348
1994	8.70	14.2	108.833091	220.78348

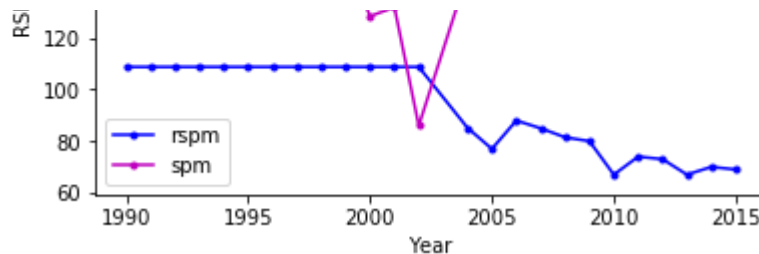
Now plot the data,



It is clear that the value SO₂ was at a peak in 1997 and now from 2005–2015 is being maintained at the lower levels. But be the measures would have taken to reduce it so. In addition to that, the value of NO₂ is also only slightly greater than the minimum value.

While doing this, I thought of plotting rspm and spm values too.





This gave an alarming signal that the value **spm** in **Andhra Pradesh** is hiking. It's 220 $\mu\text{g}/\text{m}^3$ for the past 6 years (2010–2015). It's really an alerting one.

I found a **news article** regarding this hike and the state government's action to reduce it.

The officials have taken this decision after identifying the pollution levels which already crossed danger levels. The increasing vehicular movement and construction activity are also some of the reasons for the increasing pollution levels, a study by officials revealed.

According to the study, the Suspended Particulate Matter (SPM) is above the normal levels in the city air. As per the norms of Pollution Control Board (PCB), the SPM level should not cross 100 micron grams per cubic metre. But presently the SPM crossed 115 mg on average in the city limits.

The SPM levels have crossed normal parameters following developments after state bifurcation, including shifting of employees from Hyderabad to Vijayawada city. After the shifting of administrative setup to Vijayawada city, the vehicle movement has registered 10 per cent growth.

Like this, you can dig data into any deeper levels to find surprising facts. Data analysis is all about unraveling the hidden information.

This article is just to pave a path for you to start analyzing the data and understand the importance of it.

Transform data into insights! 💡✓


Updated ~ 4th March

I have added repository link below which contains overall coding part done in this tutorial along with reader notes in .ipynb file. :-)

BalaKowsalya/indian-air-quality

This repository contains Python code for analyzing Indian Air Quality dataset taken from Kaggle ...

github.com

If you find this tutorial useful, don't forget to click/tap on the  button as long as you can. :-) Follow to get more interesting tutorials.

Data Preprocessing: A Practical Guide

Understand it thoroughly by preprocessing — Titanic Dataset

medium.com

#100daysofMLcoding

End of Day #8. Happy Learning!

[Data Science](#)[Exploratory Data Analysis](#)[Python](#)[Data Analysis](#)[Data](#)[About](#) [Help](#) [Legal](#)

Get the Medium app

