CSE315:Introduction to Data Science

WEEK-8

Exploratory Data Analysis

- Exploratory Data Analysis (EDA) Knowing the unseen patterns inside hole data. Through EDA we can extract various information from the data set, which is helpful in making various decisions as well as in selecting the machine learning model.
- EDA is done through various types of statistical methods, data visualization etc.

Primary purpose of exploratory data analysis

- To inquire about datasets and various variables.
- To Inquire about null values and other undesirable values.
- To Know the descriptive statistics of the data set.
- To Know the relationship between different variables.
- To Extract various hidden information.
- To Decide on a machine learning model.

Exploratory Data Analysis

- The data set that we will use for data analysis is the weather data of Dhaka from 1956 to 2018.
- The data set has been collected from Bangladesh Meteorological Department.

Identification of dataset variables:

- YEAR- Observation year
- Month- Month
- MaxTemp- Mean Maximum temperature per month (Unit-Celsius)
- MinTemp- Mean Minimum temperature per month (Unit-Celsius)
- RelativeHumidity- Mean RelativeHumidity per month (Unit-%)
- Rainfall- Total Rainfall per month (Unit-mm)

Step 1: Load Data

- import matplotlib.pyplot as plt
- import matplotlib.mlab as mlab
- import seaborn as sns
- import pandas as pd
- import numpy as np
- url= 'data-dhaka-weather1953-2016.csv'
- df = pd.read_csv(url)
- df.head()

Code Link:
https://drive.google.com/file/d/1fXdXMpjTVWljN798pQiJqROywWAujTuG/view?usp=sharing

Step 2: Checking Null Value

Next we check if there are any null values in our dataset.

By null value checking we know that there are no null values in our dataset. That means we are ready for further analysis.

#Checking NaN value

- df.isnull().values.any()

#output: False

Descriptive Statistics

- The average rainfall is 16.38 mm, in some months there was no rainfall at all and in some months it was up to 858 mm.
- The average humidity is 85.20. The standard deviation of humidity is relatively low i.e. the variance of humidity is low.
- The average minimum air temperature is 21.51C, with a minimum value of 9.5 C and a maximum value of 26.1 C.
- The average maximum air temperature is 33.43 degrees Celsius.
- The standard deviation of the highest temperature is the minimum temperature less than standard deviation. That is, the difference in temperature is less than in cold.

	YEAR	Month	MaxTemp	MinTemp	RelativeHumidity	Rainfall
count	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000
mean	1984.500000	6.500000	33.438529	21.51875	75.207031	168.377214
std	18.484992	3.454302	2.964510	4.97461	8.593081	173.784463
min	1953.000000	1.000000	24.300000	9.50000	49.000000	0.000000
25%	1968.750000	3.750000	31.875000	17.15000	69.000000	13.000000
50%	1984.500000	6.500000	33.900000	23.70000	76.000000	121.000000
75%	2000.250000	9.250000	35.525000	25.80000	83.000000	273.250000
max	2016.000000	12.000000	40.800000	28.10000	90.000000	856.000000

df.describe()

Correlation

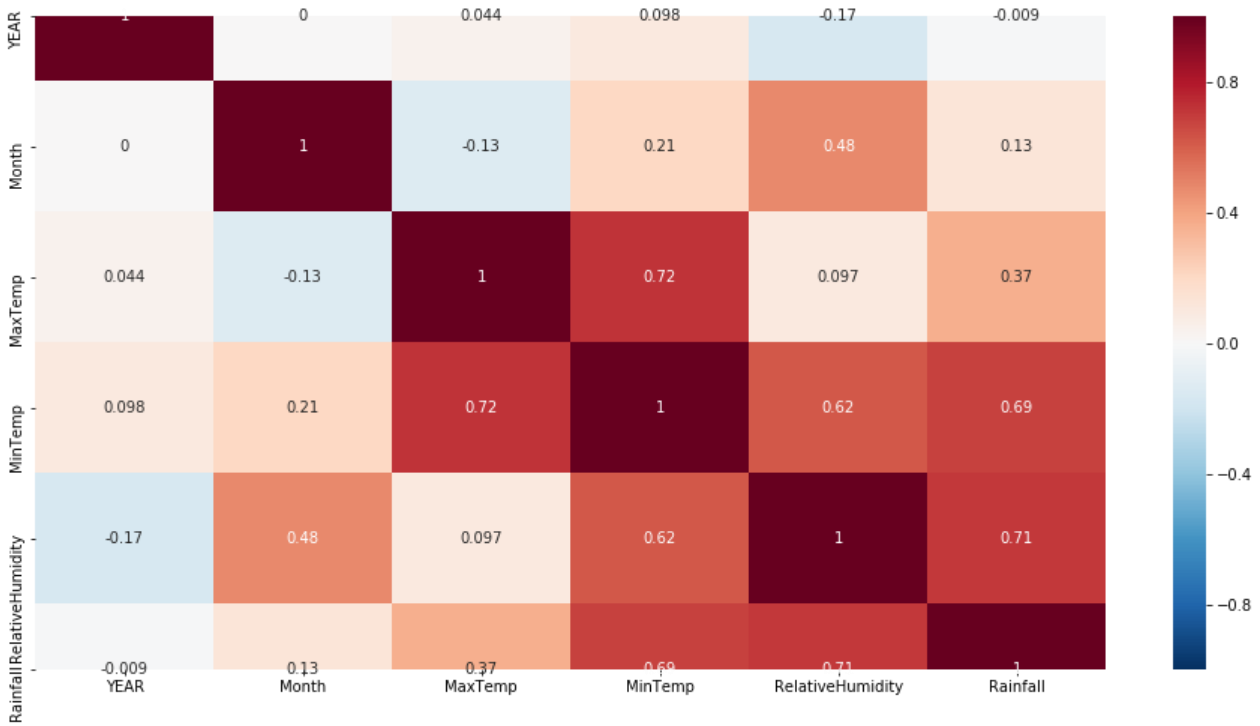
- Through the correlation function, we can easily see the linear correlation between different variables.
- Through this we find the relationship between different variables from a relatively strong correlation.
- We see that humidity has the most significant relationship with precipitation, followed by the lowest temperature in the air.

	YEAR	Month	MaxTemp	MinTemp	RelativeHumidity	Rainfall
YEAR	1.000000	0.000000	0.044080	0.098500	-0.165042	-0.009006
Month	0.000000	1.000000	-0.134986	0.210387	0.481070	0.125966
MaxTemp	0.044080	-0.134986	1.000000	0.724186	0.097398	0.365135
MinTemp	0.098500	0.210387	0.724186	1.000000	0.624770	0.687317
RelativeHumidity	-0.165042	0.481070	0.097398	0.624770	1.000000	0.708577
Rainfall	-0.009006	0.125966	0.365135	0.687317	0.708577	1.000000

df.corr()

HeatMap

- correlation = df.corr()
- plt.figure(figsize=(16, 8))
- sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1, cmap="RdBu_r")
- plt.show()



Monthly Average Rainfall

- We can find out the average rainfall for each month through groupby and mean functions.
- Through this we understand that the rainfall is generally higher in the middle of the year.
- On the other hand, rainfall is relatively less at the beginning and end of the year.

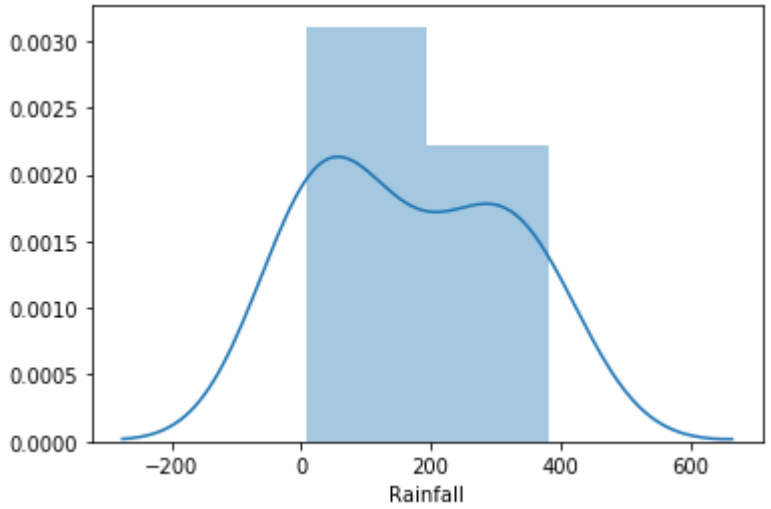
Month	
1	6.781250
2	19.703125
3	54.307812
4	126.431250
5	271.131250
6	360.075000
7	382.076563
8	313.971875
9	284.639063
10	164.120312
11	28.017187
12	9.271875

```
rain=df.groupby('Month')['Rainfall'].mean()
rain
```

Distribution Plot

- The following function allows us to draw a distribution plot of a variable.

sns.distplot(rain)



Find out Most Rainy Months

- By sorting we can easily find out which months have the most rainfall and which months have the least rainfall.

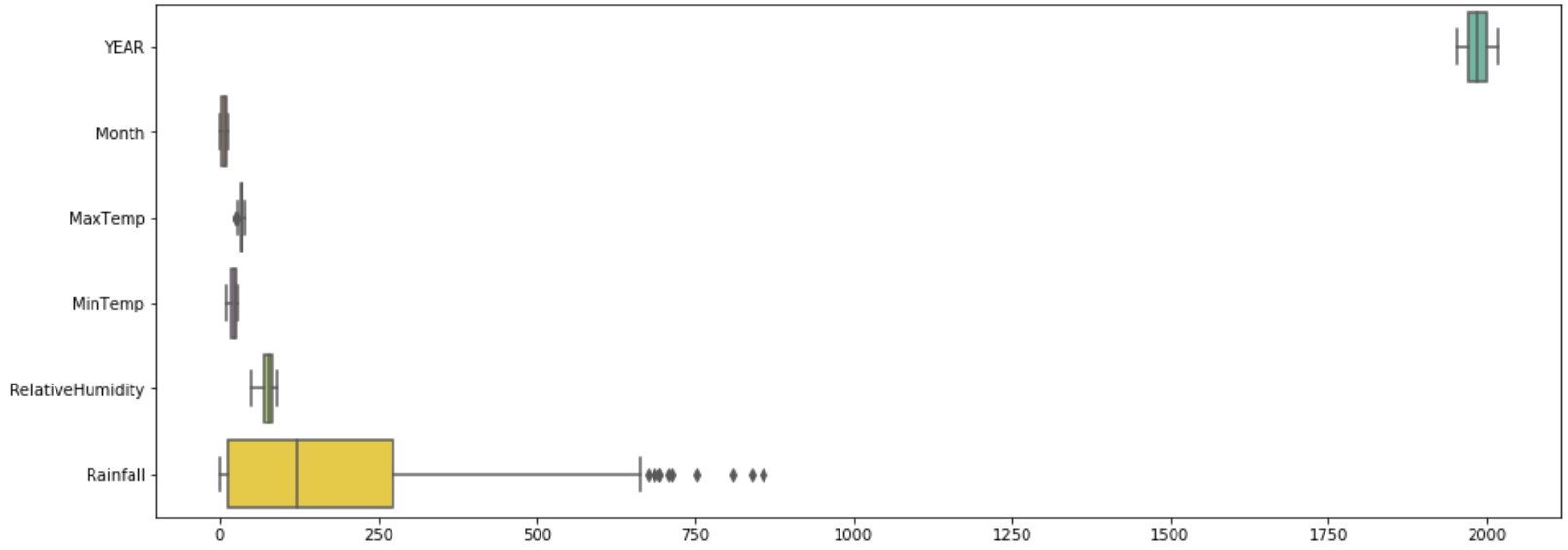
```
df.groupby('Month')['Rainfall'].mean().sort_values(ascending=False)
```

```
Month
7     382.076563
6     360.075000
8     313.971875
9     284.639063
5     271.131250
10    164.120312
4     126.431250
3      54.307812
11     28.017187
2      19.703125
12      9.271875
1       6.781250
```

Boxplot

- Boxplot of whole dataset – We can create boxplot of whole dataset at once with below function. Through box plot we can easily understand if there is any extreme value in the data

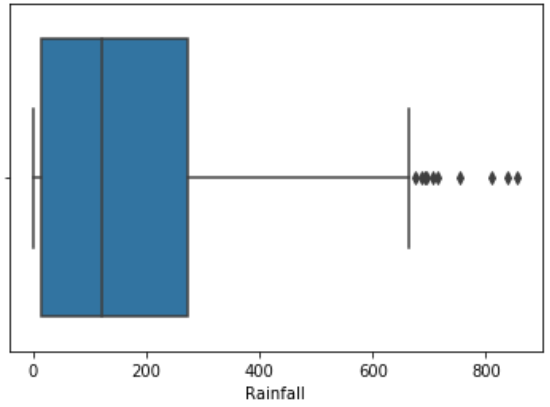
```
plt.figure(figsize=(16, 6))
ax = sns.boxplot(data=df, orient="h", palette="Set2")
```



Individual boxplots

- sometimes it can be difficult to understand the meaning of the boxplot of the entire dataset, because if there are many variables together and the difference in their value ranges is high, sometimes the matter becomes very complicated.
- In this case we can create separate box plots if we want. From the rainfall boxplot we see that the minimum value of rainfall is zero and the maximum value is 700 or so. Values after this are shown as extreme values or outliers. That is, they are different. Maybe these are values during heavy rains. On the other hand, most rainfall ranges between zero and 300.

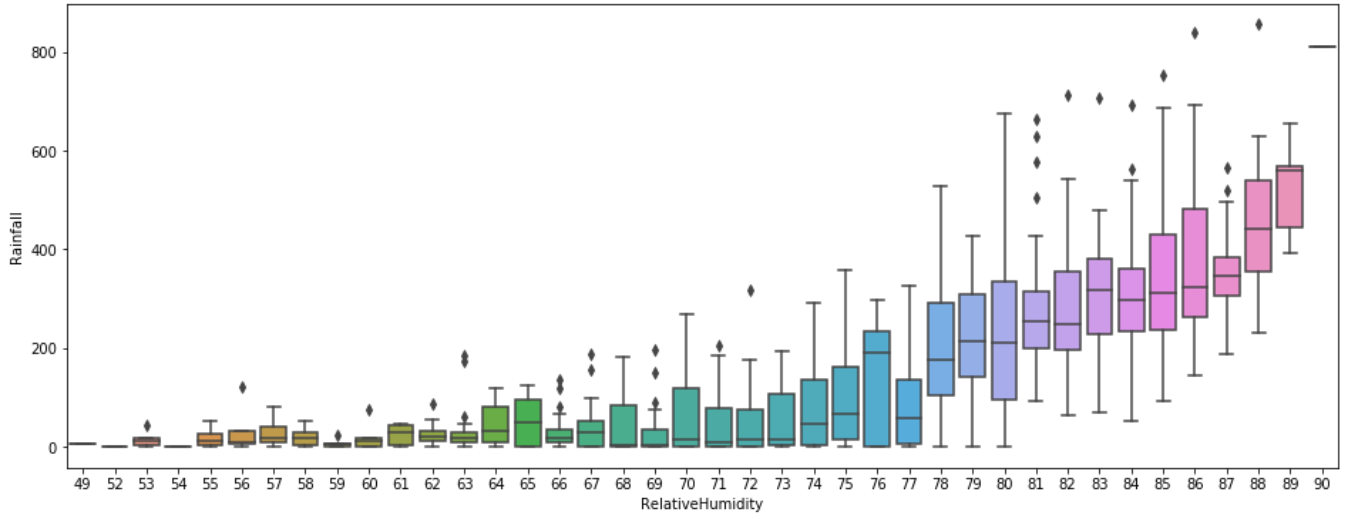
```
ax = sns.boxplot(x=df["Rainfall"])
```



Boxplot of two variables together

- We can boxplot of two variables together if we want. Rainfall and Humidity
- From the boxplot we can see that with the increase in humidity, the rainfall has also increased.

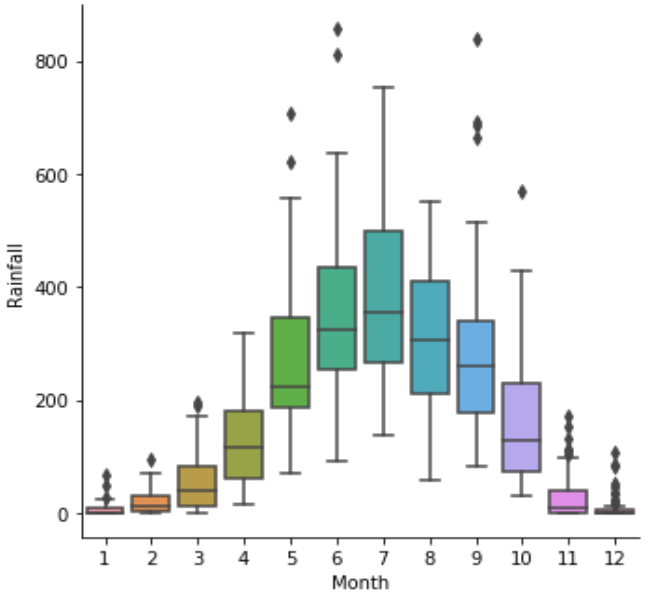
```
plt.figure(figsize=(16, 6))
ax = sns.boxplot(x="RelativeHumidity", y="Rainfall", data=df)
```



Category Boxplot

- With Category Boxplot we can boxplot any other variable along with the category variable.
- Boxplotting the rainfall category by month, we see that the higher the amount of rainfall in the middle months of the year, the greater the breadth of data in these months.
- Again, the data spread of the months in which the rainfall is less is less.

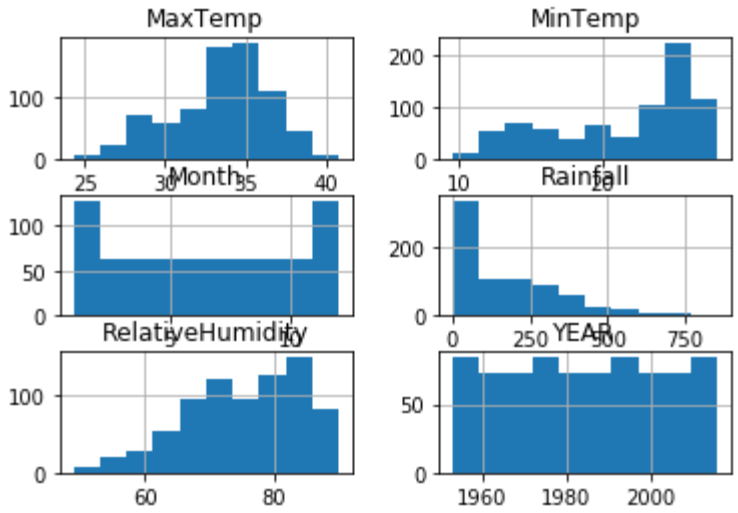
```
plt.figure(figsize=(16, 6))
ax=sns.catplot(x="Month", y="Rainfall", kind="box", data=df);
```



Histogram

- We can draw the histogram of the entire dataset together with the following function.

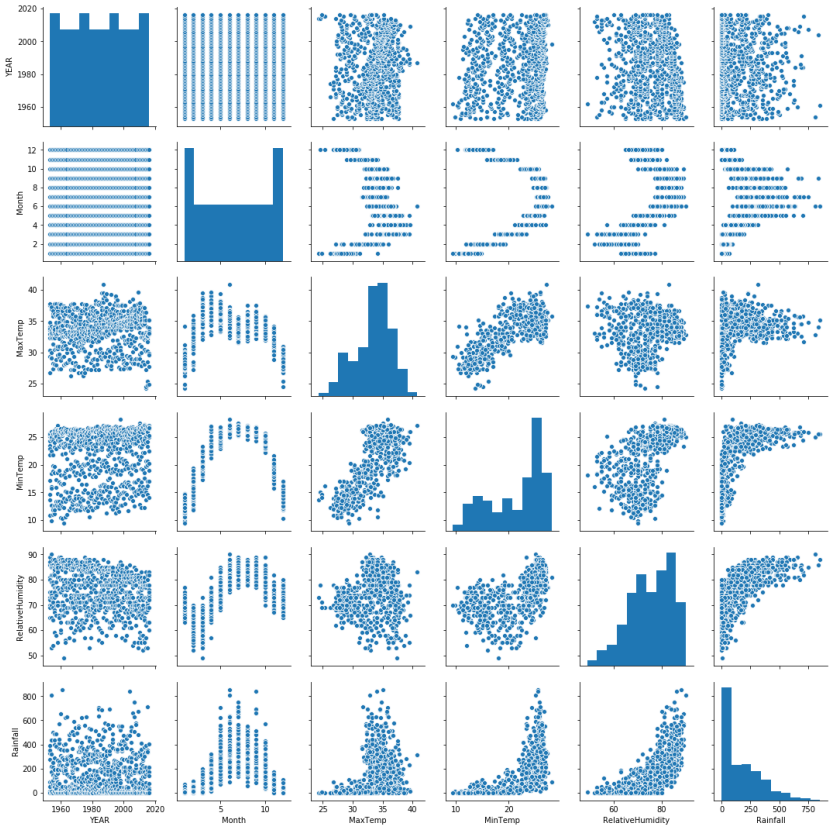
df.hist()



Pair plot

- A pair plot is a method of drawing a scatter plot of each variable and a distribution plot of each variable together with each variable in the dataset.

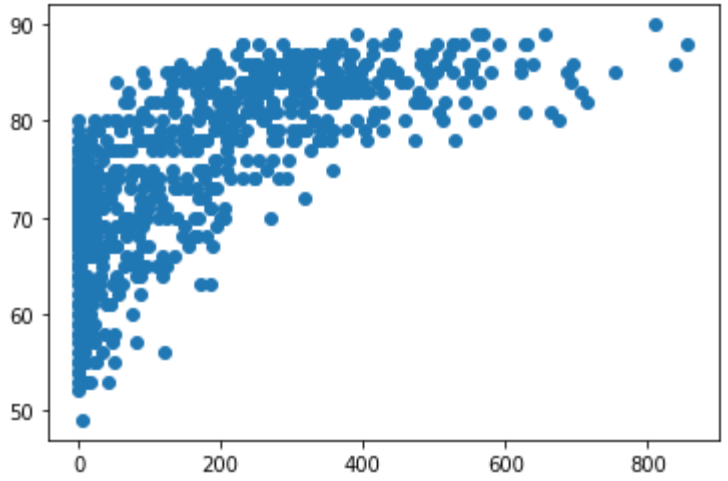
```
sns.pairplot(df);
```



Scatter Plot

- We can draw the relationship between any two variables separately in a scatter plot.
- From the inside scatter plot of rainfall and humidity we can clearly understand that rainfall increases as humidity increases. The greater the correlation between the variables, the closer the dots will be to a linear line on the scatter plot.

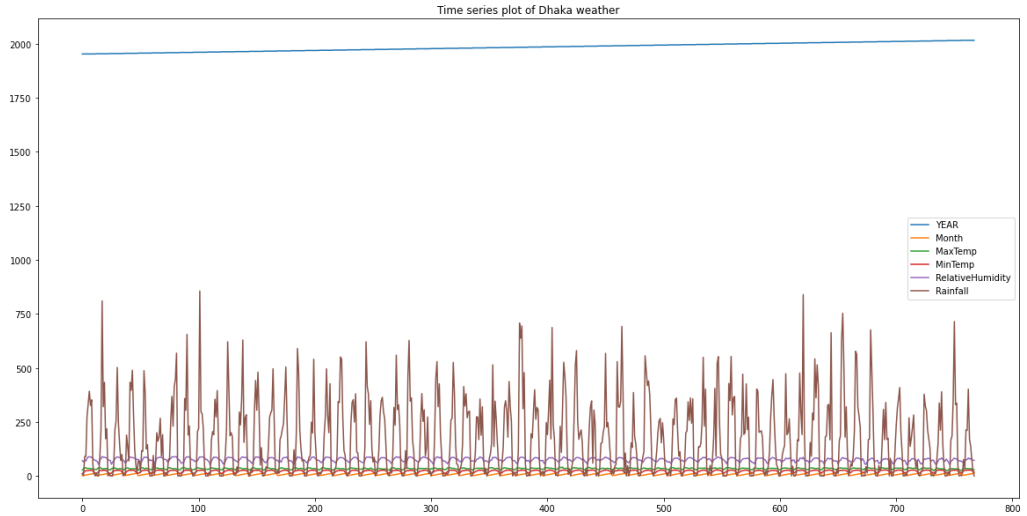
```
plt.scatter(df.Rainfall,df.RelativeHumidity)
```



Time Series Plot

- We can draw a time series plot of the entire data set over time by drawing a line plot.
- Through this we can see how the values of various variables have changed (upward or downward) over time.

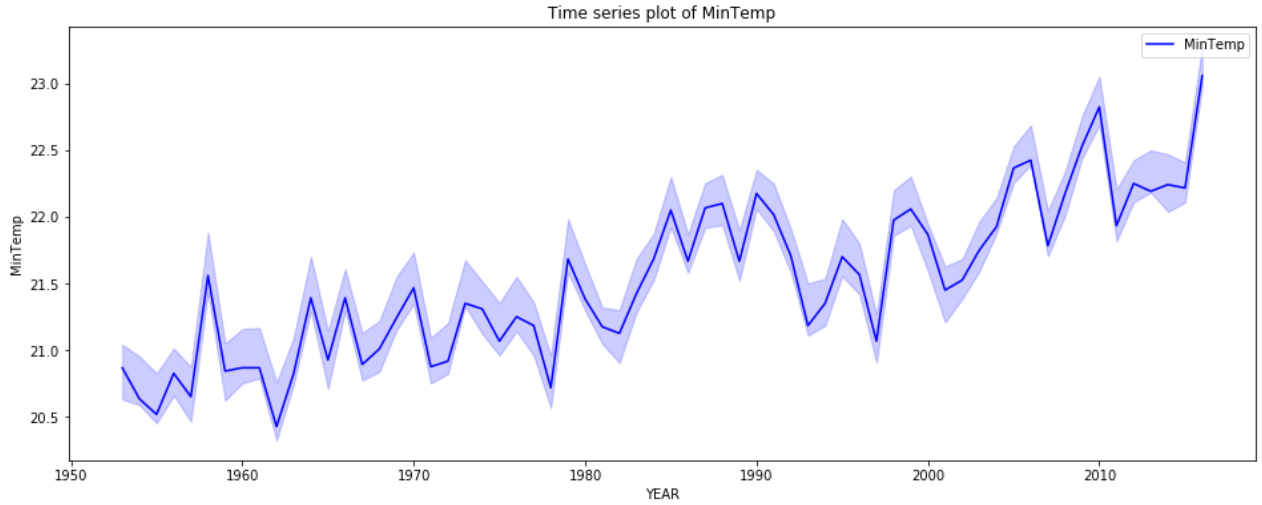
```
import matplotlib.pyplot as plt

df.plot(figsize=(20,10))
plt.title("Time series plot of Dhaka weather")
plt.show()
```



Time series of low temperature changes

- Through the short code below, we can see that from 1953 to 2017, the low temperature of the air gradually increased.

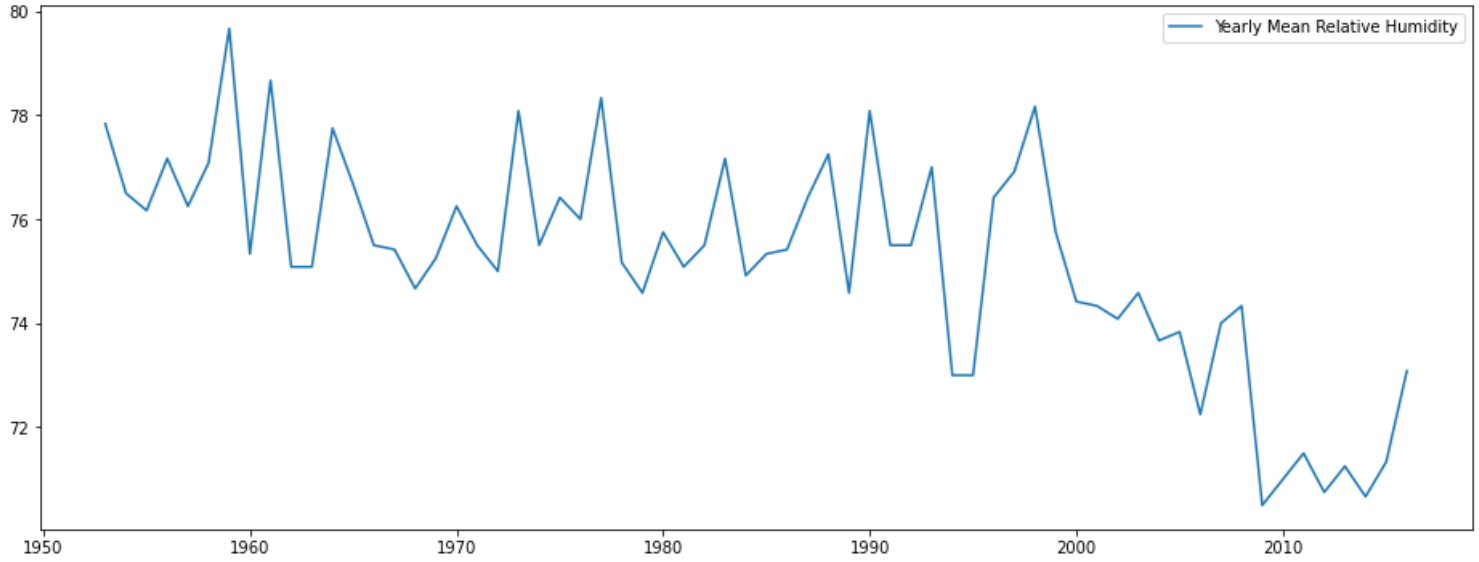
```
plt.figure(figsize=(16, 6))
plt.title('Time series plot of MinTemp')
ax = sns.lineplot(x="YEAR", y="MinTemp",ci=10,color="blue",
label="MinTemp",data=df)
```



Time series of average humidity

- From 1953 to 2018, if we do time series with the average annual humidity, we can see that the humidity of the air has gradually decreased.

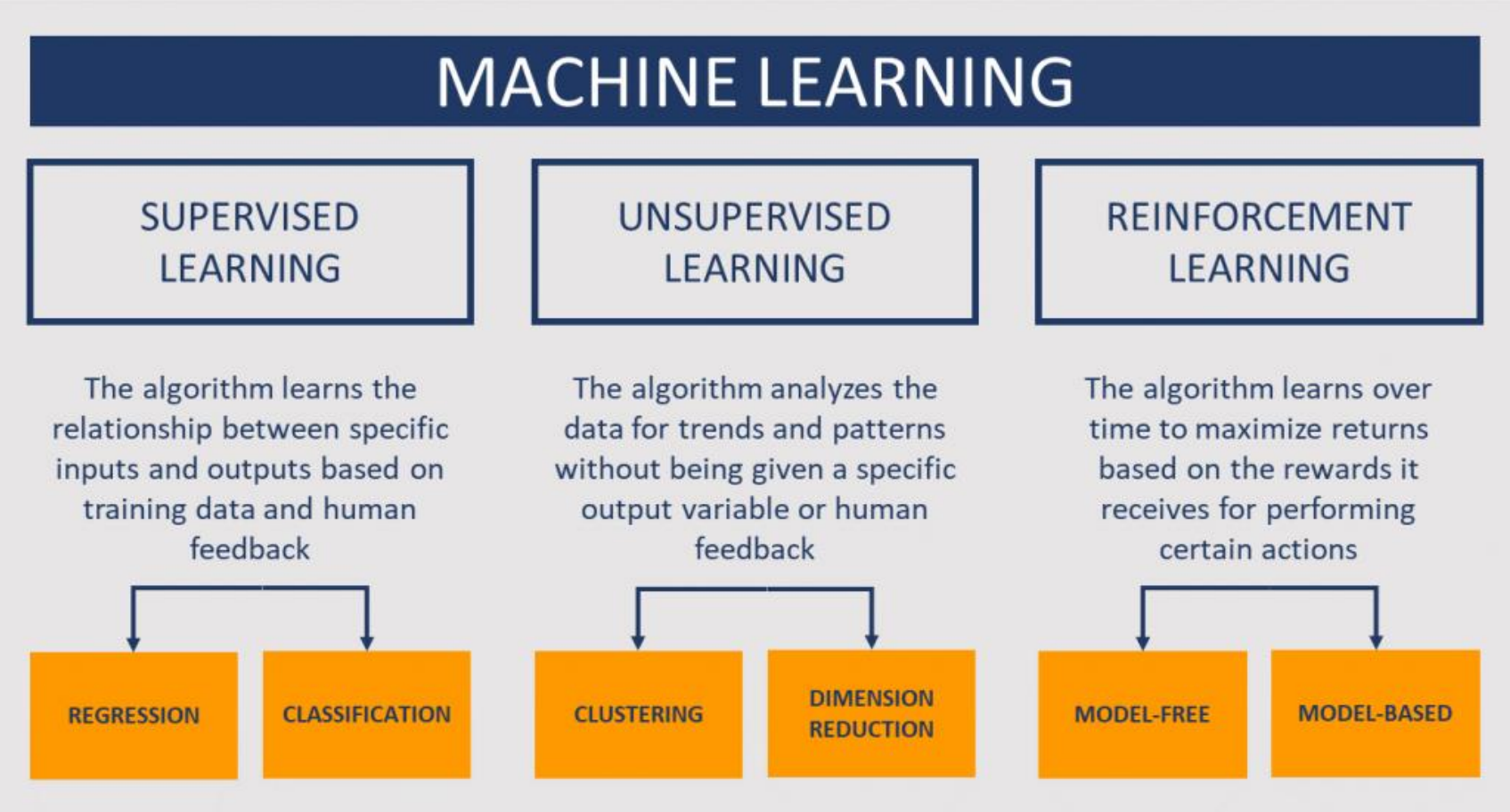
```
rh=df.groupby('YEAR')['RelativeHumidity'].mean()
plt.figure(figsize=(16, 6))
ax = sns.lineplot( label="Yearly Mean Relative Humidity",data=rh)
```



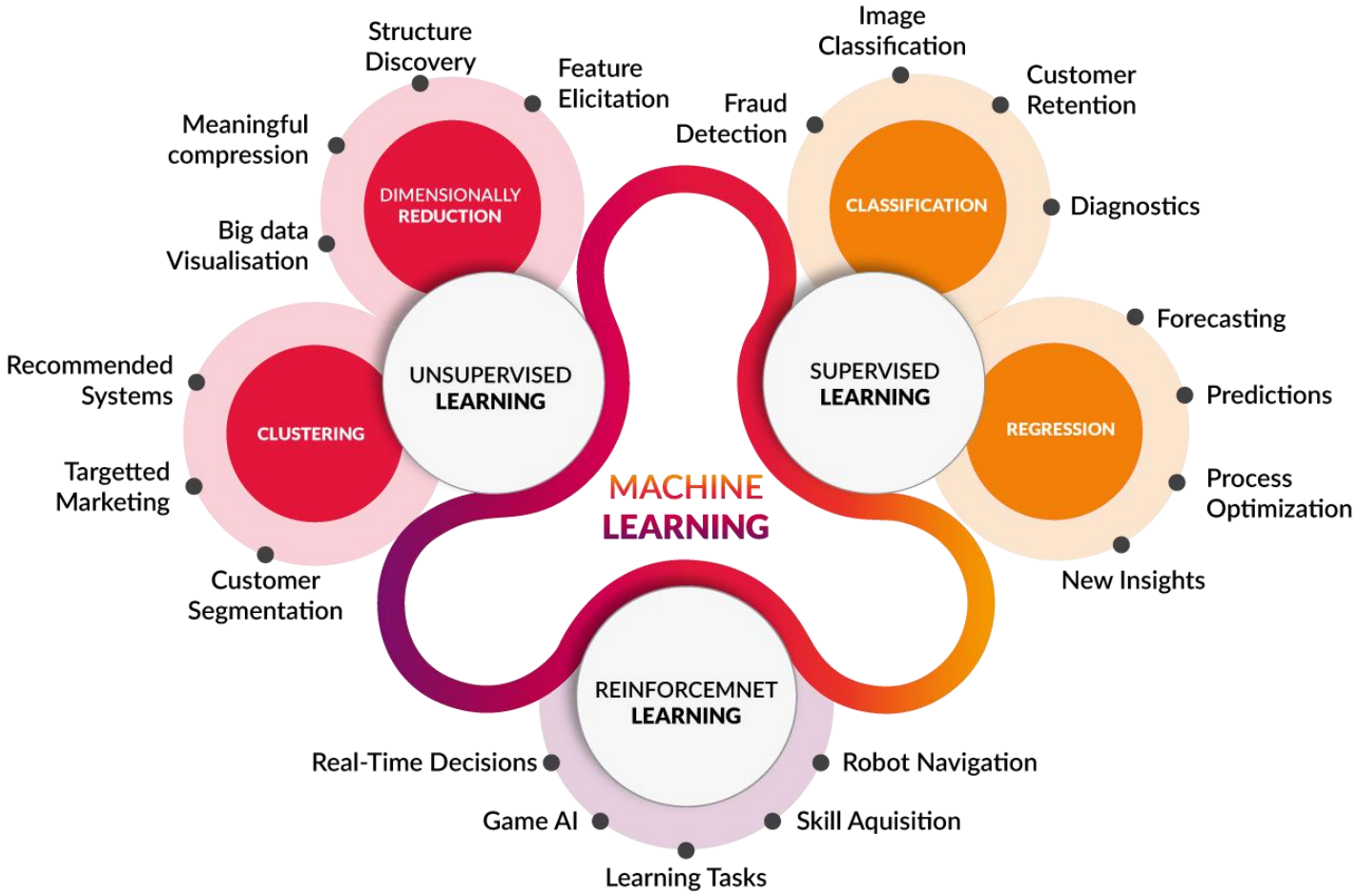
Results of exploratory data analysis

- There are no null or garbage values in the dataset.
- There are many extreme values in the rainfall data.
- The months of June, July and August receive the most rainfall.
- June, July and August are also the most humid months.
- Minimum air temperature is lowest in January, February and December.
- Air temperature is highest in April, May, June and July.
- Humidity has the highest correlation with rainfall.
- Second highest air temperature with precipitation.
- Rainfall has seasonality i.e. there is more rainfall in certain months of the year.
- The lower air temperature increased with time.
- Air humidity decreases with time.

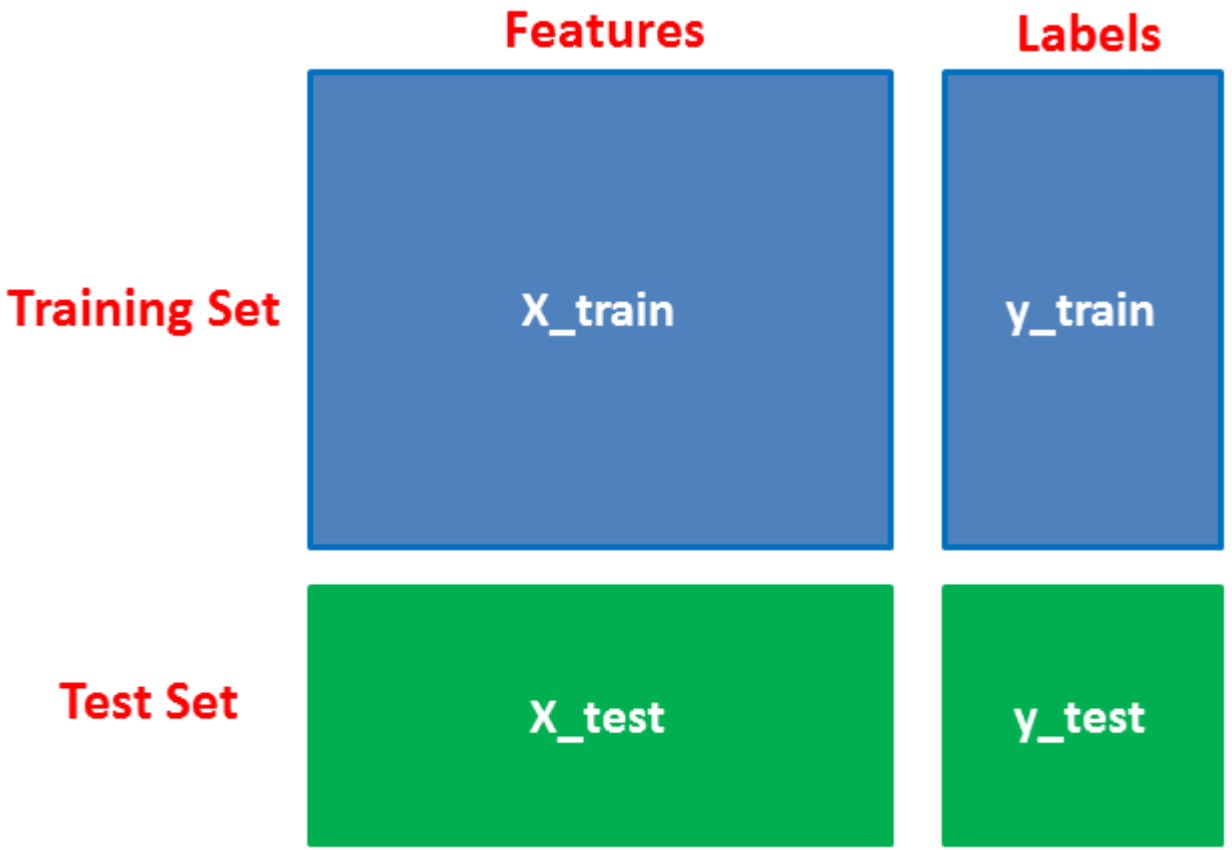
Machine Learning



Machine Learning Applications



Train test Split



Overfitting and underfitting

- **Overfitting** is a lot like leaking questions before the exam. In this case, the question paper is already known, so you can get a very good number in the test, but if something ever goes beyond that question, you have to hit the box!
- In the case of overfitting, the model performs well in the training part but in the real world its performance is absolutely poor.
- Overfitting can occur if the model becomes too complex, such as if the order is too high in polynomial regression.
- If there is too much noise in the dataset, such problems may occur.
- This problem can occur even if the training data is low.
- **Underfitting** is the opposite of overfitting. In this case, the model learns less from the dataset at once.
- This problem can occur if the model is too simple. Feature engineering, applying relatively complex models or using more features can get rid of this problem.