```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metrics
```

I am using Linear Regression algorithm for predicative weather analysis

```
df = pd.read_csv("/content/drive/MyDrive/AIML_Codes/weather.csv")
#print(dataset.shape)
```

```
print(df.shape)
```

```
(366, 22)
```

```
print(df.describe())
```

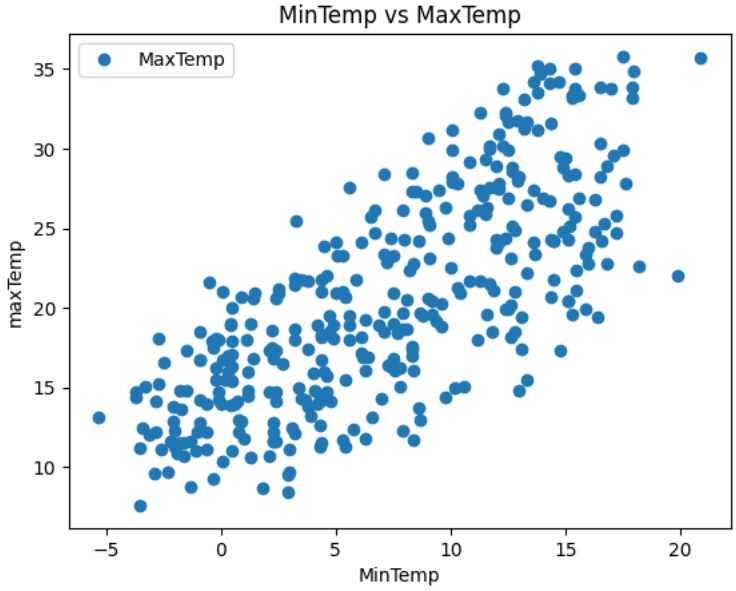
```
               MinTemp     MaxTemp    Rainfall  Evaporation    Sunshine  \
count  366.000000  366.000000  366.000000   366.000000  363.000000
mean     7.265574   20.550273    1.428415     4.521858    7.909366
std      6.025800    6.690516    4.225800     2.669383    3.481517
min     -5.300000    7.600000    0.000000     0.200000    0.000000
25%      2.300000   15.025000    0.000000     2.200000    5.950000
50%      7.450000   19.650000    0.000000     4.200000    8.600000
75%     12.500000   25.500000    0.200000     6.400000   10.500000
max     20.900000   35.800000   39.800000    13.800000   13.600000

       WindGustSpeed  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  \
count     364.000000    359.000000    366.000000   366.000000   366.000000
mean       39.840659      9.651811     17.986339    72.035519    44.519126
std        13.059807      7.951929      8.856997    13.137058    16.850947
min        13.000000      0.000000      0.000000    36.000000    13.000000
25%        31.000000      6.000000     11.000000    64.000000    32.250000
50%        39.000000      7.000000     17.000000    72.000000    43.000000
75%        46.000000     13.000000     24.000000    81.000000    55.000000
max        98.000000     41.000000     52.000000    99.000000    96.000000

       Pressure9am  Pressure3pm    Cloud9am    Cloud3pm     Temp9am  \
count   366.000000   366.000000  366.000000  366.000000  366.000000
mean   1019.709016  1016.810383    3.890710    4.024590   12.358470
std       6.686212     6.469422    2.956131    2.666268    5.630832
min     996.500000   996.800000    0.000000    0.000000    0.100000
25%    1015.350000  1012.800000    1.000000    1.000000    7.625000
50%    1020.150000  1017.400000    3.500000    4.000000   12.550000
75%    1024.475000  1021.475000    7.000000    7.000000   17.000000
max    1035.700000  1033.200000    8.000000    8.000000   24.700000

          Temp3pm     RISK_MM
count  366.000000  366.000000
mean    19.230874    1.428415
std      6.640346    4.225800
min      5.100000    0.000000
25%     14.150000    0.000000
50%     18.550000    0.000000
75%     24.000000    0.200000
max     34.500000   39.800000
```

```
#We are plotting and seeing relationship in a 2D plot
df.columns=df.columns.str.strip()
#The above code is for trimming the string
df.plot(x='MinTemp', y='MaxTemp', style='o')
plt.title('MinTemp vs MaxTemp')
plt.xlabel('MinTemp')
plt.ylabel('maxTemp')
plt.show()
```




```
plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(df['MaxTemp'])
plt.show()
#The only thing i did this is to understand that my average temp is between 15 & 20
```

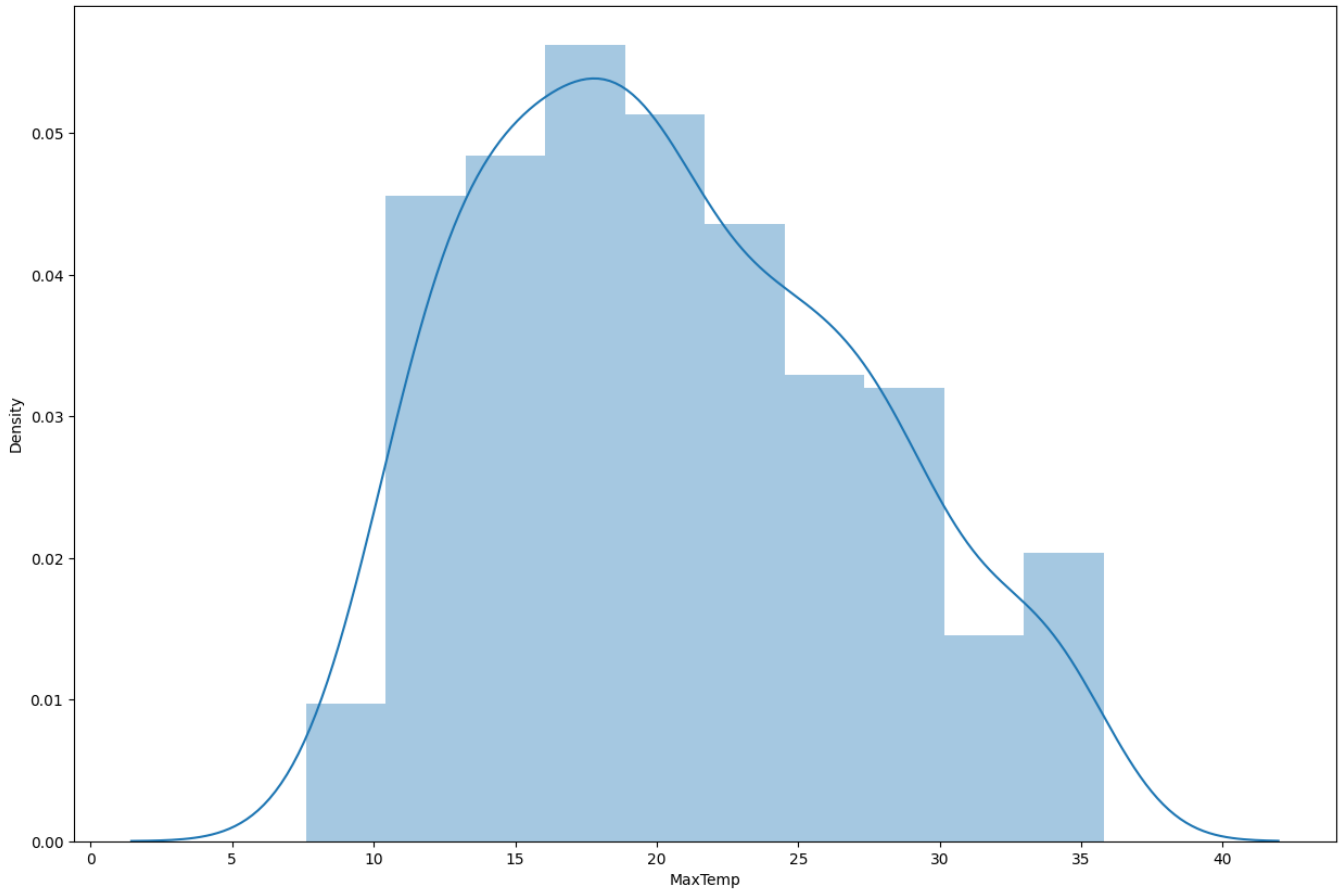
```
<ipython-input-33-aaabbe8e414a>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  seabornInstance.distplot(df['MaxTemp'])
```




```
#we gonna do something ike data splicing
#it basically is splitting data in training and testing data
x = df['MinTemp'].values.reshape(-1,1)
y = df['MaxTemp'].values.reshape(-1,1)
```

after splitting data into trainign and testing data imma just gonna deal with MinTemp and MaxTemp

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 0)
```

```
regressor = LinearRegression()
#we import LR class and create instance
regressor.fit(x_train, y_train) #training algorithm
```

```
▾ LinearRegression
LinearRegression()
```

```
#To retrive the intercept
print('Intercept:',regressor.intercept_)
print('Coefficient',regressor.coef_)
#I don't understand wtf does the below values mean but it looks cool
```

```
Intercept: [14.56202411]
Coefficient [[0.81953755]]
```

Edit : So basically we use concept of maths of straight lines in JEE i.e, intercept is just where we hit upon the axis and coefficient also makes sense with jee concepts[link text]

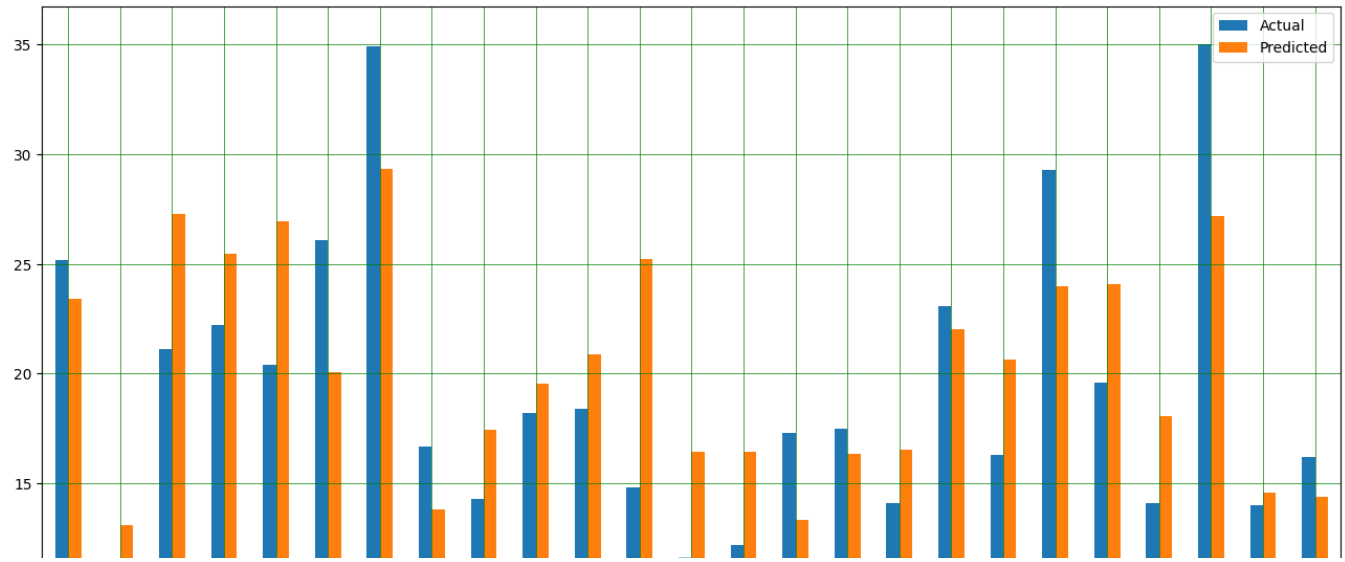
Coefficient is nothing but Beta values

```
#now we gotta test
y_pred = regressor.predict(x_test)
#I kinda messed up inn names lol do naming it data_frame
data_frame = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted':y_pred.flatten()})
print(data_frame)
```

```
         Actual  Predicted
     0     25.2  23.413030
     1     11.5  13.086857
     2     21.1  27.264856
     3     22.2  25.461874
     4     20.4  26.937041
     ..     ...        ...
     69    18.9  20.216833
     70    22.8  27.674625
     71    16.1  21.446140
     72    25.1  24.970151
     73    12.2  14.070302

     [74 rows x 2 columns]
```

```
#now let's use a plot
#since i am freaked up in naming i am going to use df1 for simplicity
df1 = data_frame.head(25)
df1.plot(kind = 'bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```




```
#the straight line shows the linear relationship
plt.scatter(x_test, y_test, color='gray')
plt.plot(x_test, y_pred, color='red', linewidth=2)
plt.show()
```