```
In [1]: import pandas as pd # It is used to work on structured data
        import matplotlib.pyplot as plt # It is used to visualize the data
        import seaborn as sns # It is used to do visualization of Regression very neatly
```

```
In [2]: df= pd.read_csv('temperatures.csv')
```

```
In [3]: df
        # To see first five lines we can write as --> df.head()
```

Out[3]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN-FEB	MAR-MAY	JUN-SEP	OCT-DEC
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21	30.39	30.47	29.97	27.31	24.49	28.96	23.27	31.46	31.27	27.25
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92	30.73	29.80	29.12	26.31	24.04	29.22	25.75	31.76	31.09	26.49
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34	29.98	29.85	29.04	26.08	23.65	28.47	24.24	30.71	30.92	26.26
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36	30.09	30.04	29.20	26.36	23.63	28.49	23.62	30.95	30.66	26.40
4	1905	22.00	22.83	26.68	30.01	33.32	33.25	31.44	30.68	30.12	30.67	27.52	23.82	28.30	22.25	30.00	31.33	26.57
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	2013	24.56	26.59	30.62	32.66	34.46	32.44	31.07	30.76	31.04	30.27	27.83	25.37	29.81	25.58	32.58	31.33	27.83
113	2014	23.83	25.97	28.95	32.74	33.77	34.15	31.85	31.32	30.68	30.29	28.05	25.08	29.72	24.90	31.82	32.00	27.81
114	2015	24.58	26.89	29.07	31.87	34.09	32.48	31.88	31.52	31.55	31.04	28.10	25.67	29.90	25.74	31.68	31.87	28.27
115	2016	26.94	29.72	32.62	35.38	35.72	34.03	31.64	31.79	31.66	31.98	30.11	28.01	31.63	28.33	34.57	32.28	30.03
116	2017	26.45	29.46	31.60	34.95	35.84	33.82	31.88	31.72	32.22	32.29	29.60	27.18	31.42	27.95	34.13	32.41	29.69

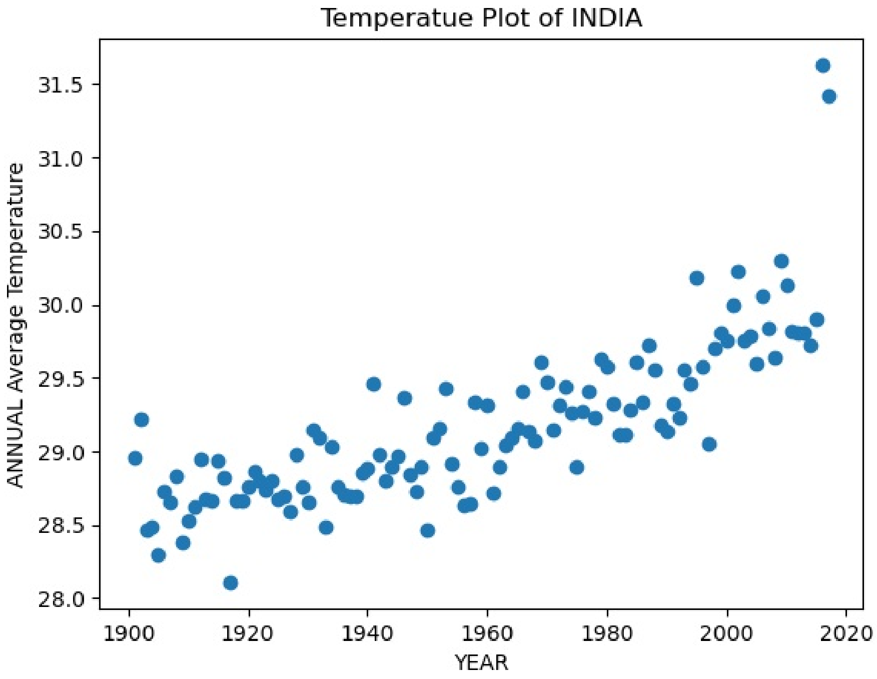
117 rows × 18 columns

```
In [4]: # Input Data
        x=df['YEAR']

        # Output Data
        y= df['ANNUAL']
```

```
In [7]: # plt.figure(figsize=(16,9)) # To give the size to the plot
        plt.title('Temperatue Plot of INDIA') # To give the Title to the Plot
        plt.xlabel('YEAR') # To give the X-axis name
        plt.ylabel('ANNUAL Average Temperature') # To give the Y-axis name
        plt.scatter(x,y) # To give the type of the plot/graph
```

```
Out[7]: <matplotlib.collections.PathCollection at 0x205a2d6aad0>
```



```
In [8]: # In order to make the Regression Model we have to give Two Dimensional Data means rows and columns both
        # But as you see here it has only rows and not columns
        # Note it is the Requirement of Python Library and not of Machine
        x.shape
```

```
Out[8]:  (117,)

In [9]:  x=x.values # To convert the values in an Array

In [10]: x

Out[10]: array([1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911,
                1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,
                1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933,
                1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944,
                1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955,
                1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966,
                1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977,
                1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,
                1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
                2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
                2011, 2012, 2013, 2014, 2015, 2016, 2017])

In [11]: x=x.reshape(117,1) # Here we can give the requires no. of rows and columns using "Reshape"

In [13]: x.shape

Out[13]: (117, 1)

In [14]: from sklearn.linear_model import LinearRegression # To import a class named "LinearRegression" from "scikit lea

In [16]: regressor= LinearRegression() # To create object of the class "LineearRegression"

In [17]: regressor.fit(x,y) # To train our algorithm we use "fit()"

Out[17]: ▾ LinearRegression
         LinearRegression()

In [18]: regressor.coef_ # To get the "Slope" of regression i.e. "m" of "y=mx+c"

Out[18]: array([0.01312158])

In [19]: regressor.intercept_ # To get the value "c" of "y=mx+c"

Out[19]: np.float64(3.4761897126187016)

In [22]: regressor.predict([[2024]]) # To predict the values. Here we are predicting the Average Temperature of the Year

Out[22]: array([30.03427031])

In [23]: regressor.predict([[2035]])

Out[23]: array([30.1786077])

In [24]: regressor.predict([[2070]])

Out[24]: array([30.63786305])

In [25]: predicted= regressor.predict(x) # To get the Predicted Values from the Algorithm

In [26]: predicted
```

Out[26]:
```
array([28.4203158 , 28.43343739, 28.44655897, 28.45968055, 28.47280213,
       28.48592371, 28.49904529, 28.51216687, 28.52528846, 28.53841004,
       28.55153162, 28.5646532 , 28.57777478, 28.59089636, 28.60401794,
       28.61713952, 28.63026111, 28.64338269, 28.65650427, 28.66962585,
       28.68274743, 28.69586901, 28.70899059, 28.72211218, 28.73523376,
       28.74835534, 28.76147692, 28.7745985 , 28.78772008, 28.80084166,
       28.81396324, 28.82708483, 28.84020641, 28.85332799, 28.86644957,
       28.87957115, 28.89269273, 28.90581431, 28.91893589, 28.93205748,
       28.94517906, 28.95830064, 28.97142222, 28.9845438 , 28.99766538,
       29.01078696, 29.02390855, 29.03703013, 29.05015171, 29.06327329,
       29.07639487, 29.08951645, 29.10263803, 29.11575961, 29.1288812 ,
       29.14200278, 29.15512436, 29.16824594, 29.18136752, 29.1944891 ,
       29.20761068, 29.22073227, 29.23385385, 29.24697543, 29.26009701,
       29.27321859, 29.28634017, 29.29946175, 29.31258333, 29.32570492,
       29.3388265 , 29.35194808, 29.36506966, 29.37819124, 29.39131282,
       29.4044344 , 29.41755599, 29.43067757, 29.44379915, 29.45692073,
       29.47004231, 29.48316389, 29.49628547, 29.50940705, 29.52252864,
       29.53565022, 29.5487718 , 29.56189338, 29.57501496, 29.58813654,
       29.60125812, 29.6143797 , 29.62750129, 29.64062287, 29.65374445,
       29.66686603, 29.67998761, 29.69310919, 29.70623077, 29.71935236,
       29.73247394, 29.74559552, 29.7587171 , 29.77183868, 29.78496026,
       29.79808184, 29.81120342, 29.82432501, 29.83744659, 29.85056817,
       29.86368975, 29.87681133, 29.88993291, 29.90305449, 29.91617608,
       29.92929766, 29.94241924])
```

In [27]:
```
y # It is the actual value
```

Out[27]:
```
0      28.96
1      29.22
2      28.47
3      28.49
4      28.30
       ...
112    29.81
113    29.72
114    29.90
115    31.63
116    31.42
Name: ANNUAL, Length: 117, dtype: float64
```

In [31]:
```
# Here we use numpy to perform Mathematical Operations
import numpy as np
```

In [32]:
```
# To caluculate Mean Absolute Error
np.mean(abs(y - predicted))
```

Out[32]:
```
np.float64(0.22535284978630413)
```

In [33]:
```
# To find the MAE, we have Prebuilt Function as
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y,predicted) # Here we have to pass the Actual and Predicted values
```

Out[33]:
```
0.22535284978630413
```

In [34]:
```
# To find Mean Squared Error(MSE)
np.mean((y-predicted)**2) # Here we took Square of the answer
```

Out[34]:
```
np.float64(0.10960795229110352)
```

In [35]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y,predicted) # Here we are using the function for finding the MSE
```

Out[35]:
```
0.10960795229110352
```

In [37]:
```
# To find the value of R-Square Metrics i.e. R-Squared Error
from sklearn.metrics import r2_score # r2_score is used to find out the linearity in the data
r2_score(y,predicted)
```

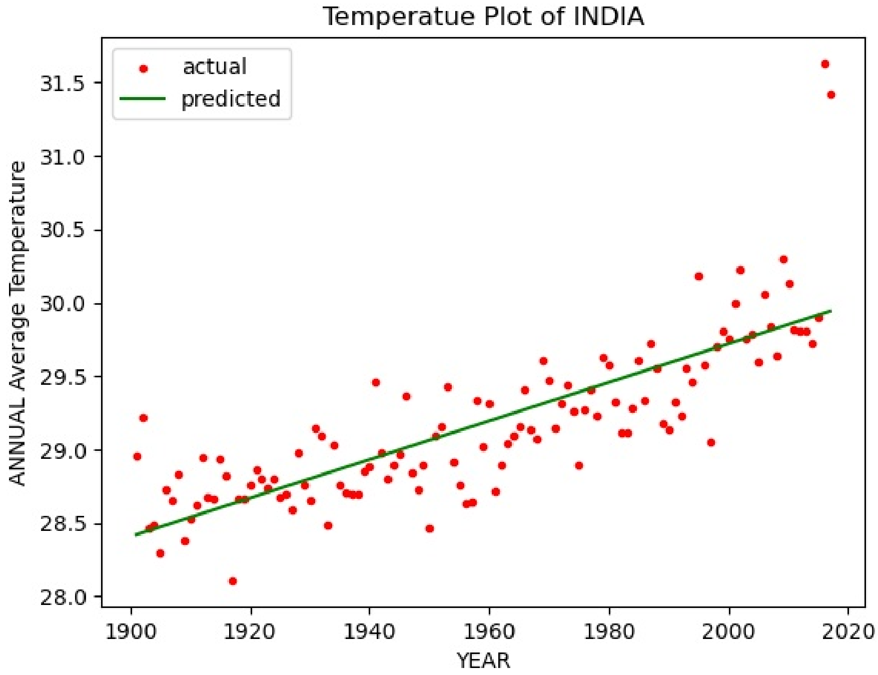
Out[37]:
```
0.6418078912783682
```

In [38]:
```
# We can also use following to get the value of R-Squared Error
regressor.score(x,y)
```

Out[38]:
```
0.6418078912783682
```

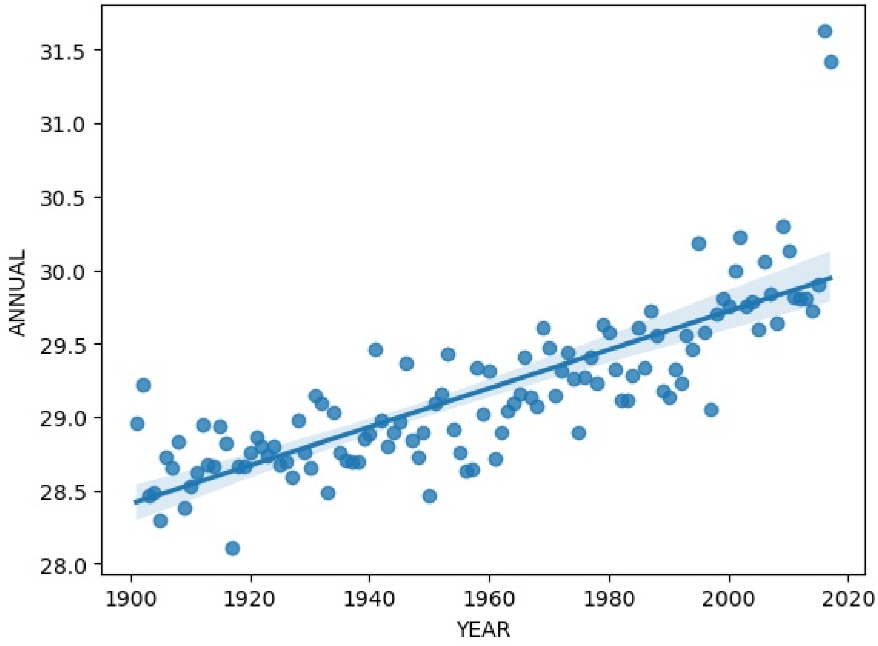
In [42]:
```
# plt.figure(figsize=(16,9)) # To give the size to the plot
plt.title('Temperatue Plot of INDIA') # To give the Title to the Plot
plt.xlabel('YEAR') # To give the X-axis name
plt.ylabel('ANNUAL Average Temperature') # To give the Y-axis name
plt.scatter(x,y, label='actual', color='r', marker='.') # To give the type of the plot/graph. Here we used "Mar
plt.plot(x,predicted, label='predicted', color='g')
plt.legend() # This is used to get the values with their colors in the form of a "Box"
```

```
Out[42]: <matplotlib.legend.Legend at 0x205a54851d0>
```



```
In [43]: # To get the Graphical representation of the Data, we can simply use "seaborn", in which we have Prebuilt funct
         sns.regplot(x='YEAR',y='ANNUAL',data=df)
```

```
Out[43]: <Axes: xlabel='YEAR', ylabel='ANNUAL'>
```



```
In [ ]:
```