Vaibhav Kumar

Roll No : 19

Linear Regression , Weather Dataset , Weather Prediction Model

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split
```

In [2]:

```
dfw=pd.read_csv('D:\\vk\\TRIM 3\\ML\\DATASET\\Weather.csv')
```

```
C:\Users\studentadmin\AppData\Local\Temp\ipykernel_2960\2889413408.py:1: DtypeWarnin
g: Columns (7,8,18,25) have mixed types. Specify dtype option on import or set low_me
mory=False.
  dfw=pd.read_csv('D:\\vk\\TRIM 3\\ML\\DATASET\\Weather.csv')
```

In [3]:

```
dfw
```

Out[3]:

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeath
0	10001	1942-7-1	1.016	NaN	25.555556	22.222222	23.888889	0.0	N
1	10001	1942-7-2	0	NaN	28.888889	21.666667	25.555556	0.0	N
2	10001	1942-7-3	2.54	NaN	26.111111	22.222222	24.444444	0.0	N
3	10001	1942-7-4	2.54	NaN	26.666667	22.222222	24.444444	0.0	N
4	10001	1942-7-5	0	NaN	26.666667	21.666667	24.444444	0.0	N
...	...	...	...	...	...	...	...	...	...
119035	82506	1945-12-27	0	NaN	28.333333	18.333333	23.333333	0.0	N
119036	82506	1945-12-28	9.906	NaN	29.444444	18.333333	23.888889	0.0	
119037	82506	1945-12-29	0	NaN	28.333333	18.333333	23.333333	0.0	
119038	82506	1945-12-30	0	NaN	28.333333	18.333333	23.333333	0.0	N
119039	82506	1945-12-31	0	NaN	29.444444	17.222222	23.333333	0.0	N

119040 rows × 31 columns

In [4]:

```
dfw.head()
```

Out[4]:

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather	Y
0	10001	1942-7-1	1.016	NaN	25.555556	22.222222	23.888889	0.0	NaN	4
1	10001	1942-7-2	0	NaN	28.888889	21.666667	25.555556	0.0	NaN	4
2	10001	1942-7-3	2.54	NaN	26.111111	22.222222	24.444444	0.0	NaN	4
3	10001	1942-7-4	2.54	NaN	26.666667	22.222222	24.444444	0.0	NaN	4
4	10001	1942-7-5	0	NaN	26.666667	21.666667	24.444444	0.0	NaN	4

5 rows × 31 columns

In [5]:

```
dfw.describe()
```

Out[5]:

	STA	WindGustSpd	MaxTemp	MinTemp	MeanTemp	YR	
count	119040.000000	532.000000	119040.000000	119040.000000	119040.000000	119040.000000	11
mean	29659.435795	37.774534	27.045111	17.789511	22.411631	43.805284	
std	20953.209402	10.297808	8.717817	8.334572	8.297982	1.136718	
min	10001.000000	18.520000	-33.333333	-38.333333	-35.555556	40.000000	
25%	11801.000000	29.632000	25.555556	15.000000	20.555556	43.000000	
50%	22508.000000	37.040000	29.444444	21.111111	25.555556	44.000000	
75%	33501.000000	43.059000	31.666667	23.333333	27.222222	45.000000	
max	82506.000000	75.932000	50.000000	34.444444	40.000000	45.000000	

8 rows × 24 columns

In [6]:

```
dfw.shape
```

Out[6]:

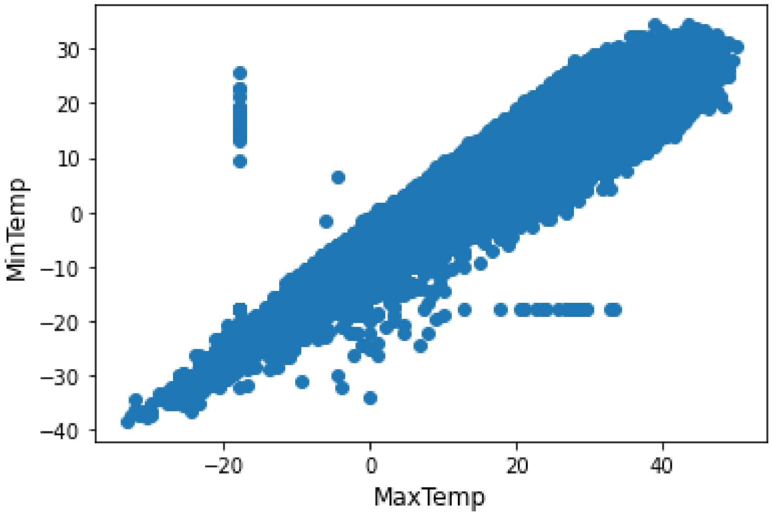
```
(119040, 31)
```

In [7]:

```
x=dfw['MaxTemp']
y=dfw['MinTemp']
```

In [8]:

```
plt.scatter(x,y)
plt.xlabel('MaxTemp',fontsize='12')
plt.ylabel('MinTemp',fontsize='12')
plt.show()
```



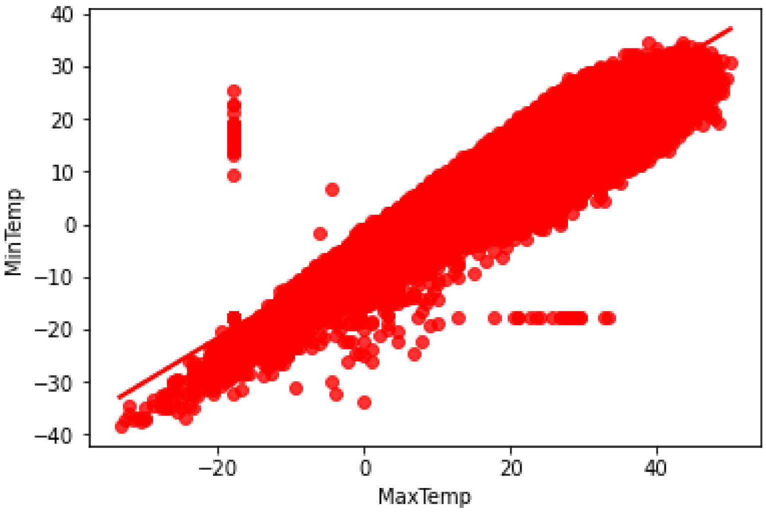
In [9]:

```
sns.regplot(x,y,color='red')
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the foll
owing variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:

```
<AxesSubplot:xlabel='MaxTemp', ylabel='MinTemp'>
```



In [10]:
```
x.head()
```

Out[10]:
```
0    25.555556
1    28.888889
2    26.111111
3    26.666667
4    26.666667
Name: MaxTemp, dtype: float64
```

In [11]:
```
y.head()
```

Out[11]:
```
0    22.222222
1    21.666667
2    22.222222
3    22.222222
4    21.666667
Name: MinTemp, dtype: float64
```

In [12]:
```
x.shape
```

Out[12]:
```
(119040,)
```

In [13]:
```
X_=x.values.reshape(-1,1)
```

In [14]:
```
X_.shape
```

Out[14]:
```
(119040, 1)
```

In [15]:
```
x
```

Out[15]:
```
0         25.555556
1         28.888889
2         26.111111
3         26.666667
4         26.666667
            ...
119035    28.333333
119036    29.444444
119037    28.333333
119038    28.333333
119039    29.444444
Name: MaxTemp, Length: 119040, dtype: float64
```

In [16]:
```
X_
```

Out[16]:
```
array([[25.55555556],
       [28.88888889],
       [26.11111111],
       ...,
       [28.33333333],
       [28.33333333],
       [29.44444444]])
```

Model

In [17]:
```
X_train,X_test,y_train,y_test=train_test_split(X_,y,test_size=0.2,random_state=30)
```

In [18]:
```
X_train.shape
```

Out[18]:
```
(95232, 1)
```

In [19]:
```
X_test.shape
```

Out[19]:
```
(23808, 1)
```

In [20]:
```
LR=LinearRegression()
LR.fit(X_train,y_train)
```

Out[20]:
```
LinearRegression()
```

In [21]:
```
y_pred=LR.predict(X_test)
```

In [22]:
```
y_test
```

Out[22]:
```
39071     12.222222
5109      22.222222
1113      22.222222
117003     7.777778
106549    26.111111
            ...
83309     12.222222
75290      6.111111
62785      8.888889
80737      6.111111
25569     22.777778
Name: MinTemp, Length: 23808, dtype: float64
```

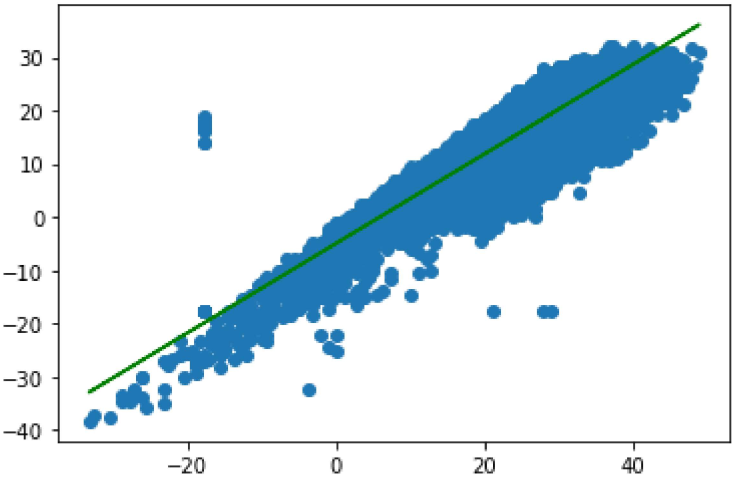
In [23]:
```
y_pred
```

Out[23]:
```
array([14.66774639, 22.13310546, 18.86701087, ..., 15.13433134,
        9.53531203, 20.2667657 ])
```

In [24]:
```
weights = LR.coef_
intercept = LR.intercept_
print(weights,intercept)
```

```
[0.8398529] -4.928821159366091
```

In [25]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test,y_pred, color='green')
plt.show()
```



In [26]:

```
df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
df
```

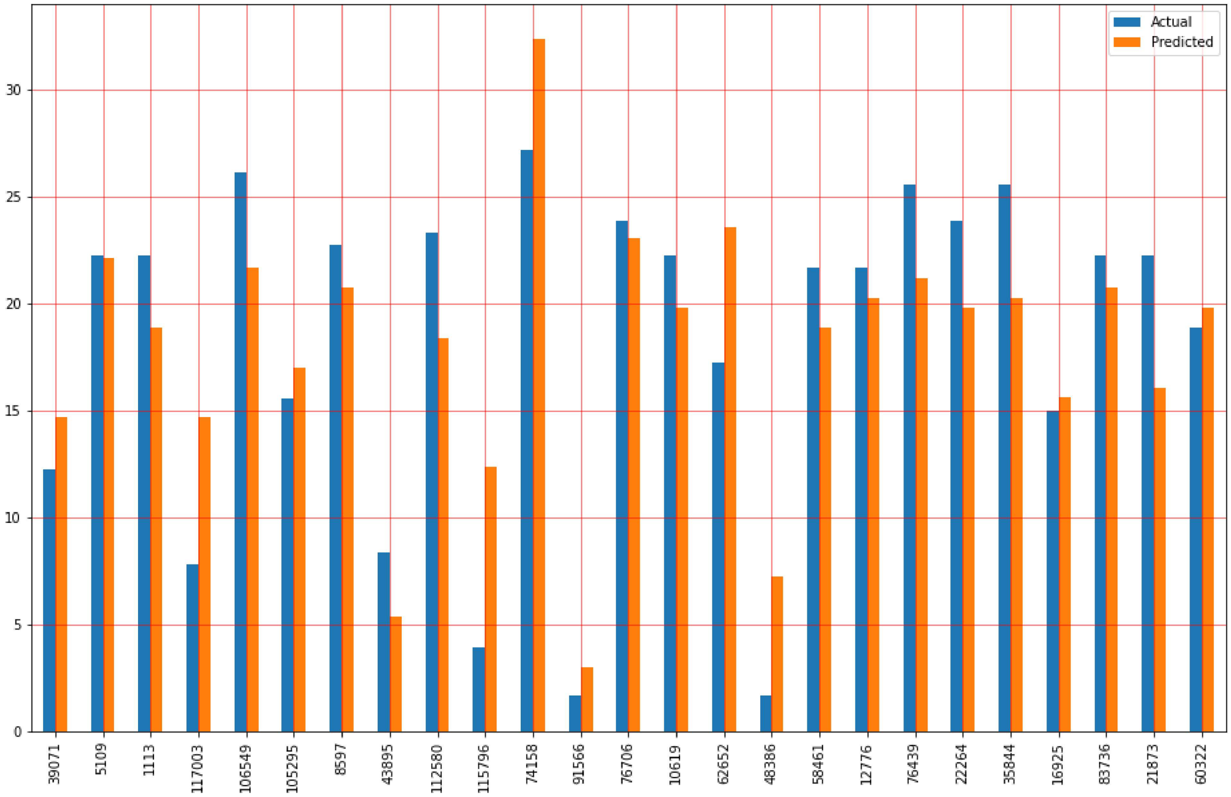
Out[26]:

	Actual	Predicted
39071	12.222222	14.667746
5109	22.222222	22.133105
1113	22.222222	18.867011
117003	7.777778	14.667746
106549	26.111111	21.666521
...	...	...
83309	12.222222	11.868237
75290	6.111111	11.401652
62785	8.888889	15.134331
80737	6.111111	9.535312
25569	22.777778	20.266766

23808 rows × 2 columns

In [27]:

```
df1=df.head(25)
df1.plot(kind='bar',figsize=(16,10))
plt.grid(which='major',linestyle='-',linewidth='0.5',color='red')
plt.grid(which='minor',linestyle=':',linewidth='0.5',color='green')
plt.show()
```



In [28]:

```
print('Mean Absolute Error',mean_absolute_error(y_test,y_pred))
print('Mean Squared Error',mean_squared_error(y_test,y_pred))
print('Root Mean Sqaured Error',np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error 3.086071147707306
Mean Squared Error 15.642509497194942
Root Mean Sqaured Error 3.955061250751364
```

80:20 ::: Mean Absolute Error 3.086071147707306 Mean Squared Error 15.642509497194942 Root Mean Sqaured Error 3.955061250751364

90:10 :: Mean Absolute Error 3.1047256786542192 Mean Squared Error 15.83161054325677 Root Mean Sqaured Error 3.97889564367511

70:30 Mean Absolute Error 3.1013916713826886 Mean Squared Error 15.768060731561533 Root Mean Sqaured Error 3.970901752947501

80:20 split has most least error compared to others

In [ ]: