Importing libraries

In [1]:
```
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
from sklearn import metrics
%matplotlib inline
```

Importing datset

In [4]:
```
dataset = pd.read_csv('Summary of Weather.csv')
```

```
/opt/anaconda3/envs/FPAD/lib/python3.7/site-packages/IPython/core/in
teractiveshell.py:3063: DtypeWarning: Columns (7,8,18,25) have mixed
types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [5]:
```
dataset.shape
```

Out[5]:
```
(119040, 31)
```

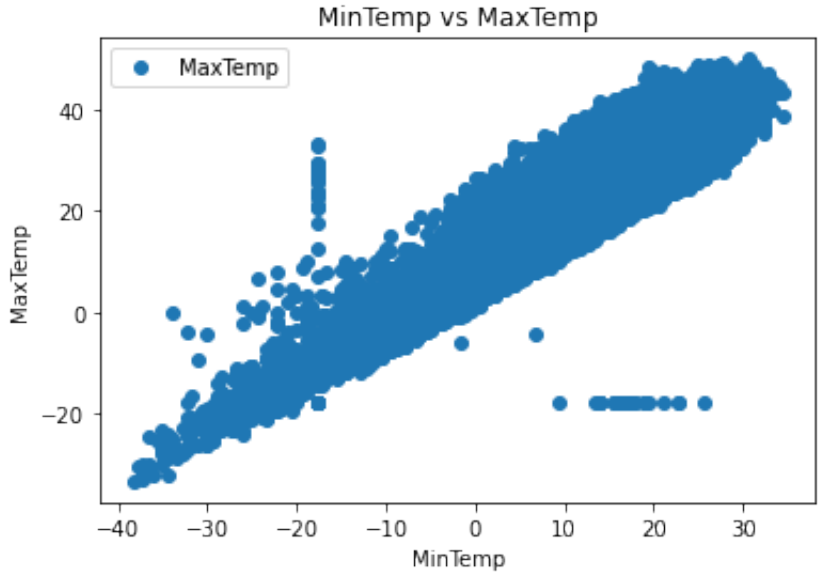
In [6]:
```
dataset.describe()
```

Out[6]:

	STA	WindGustSpd	MaxTemp	MinTemp	MeanTemp	
count	119040.000000	532.000000	119040.000000	119040.000000	119040.000000	119040.00
mean	29659.435795	37.774534	27.045111	17.789511	22.411631	43.80
std	20953.209402	10.297808	8.717817	8.334572	8.297982	1.13
min	10001.000000	18.520000	-33.333333	-38.333333	-35.555556	40.00
25%	11801.000000	29.632000	25.555556	15.000000	20.555556	43.00
50%	22508.000000	37.040000	29.444444	21.111111	25.555556	44.00
75%	33501.000000	43.059000	31.666667	23.333333	27.222222	45.00
max	82506.000000	75.932000	50.000000	34.444444	40.000000	45.00

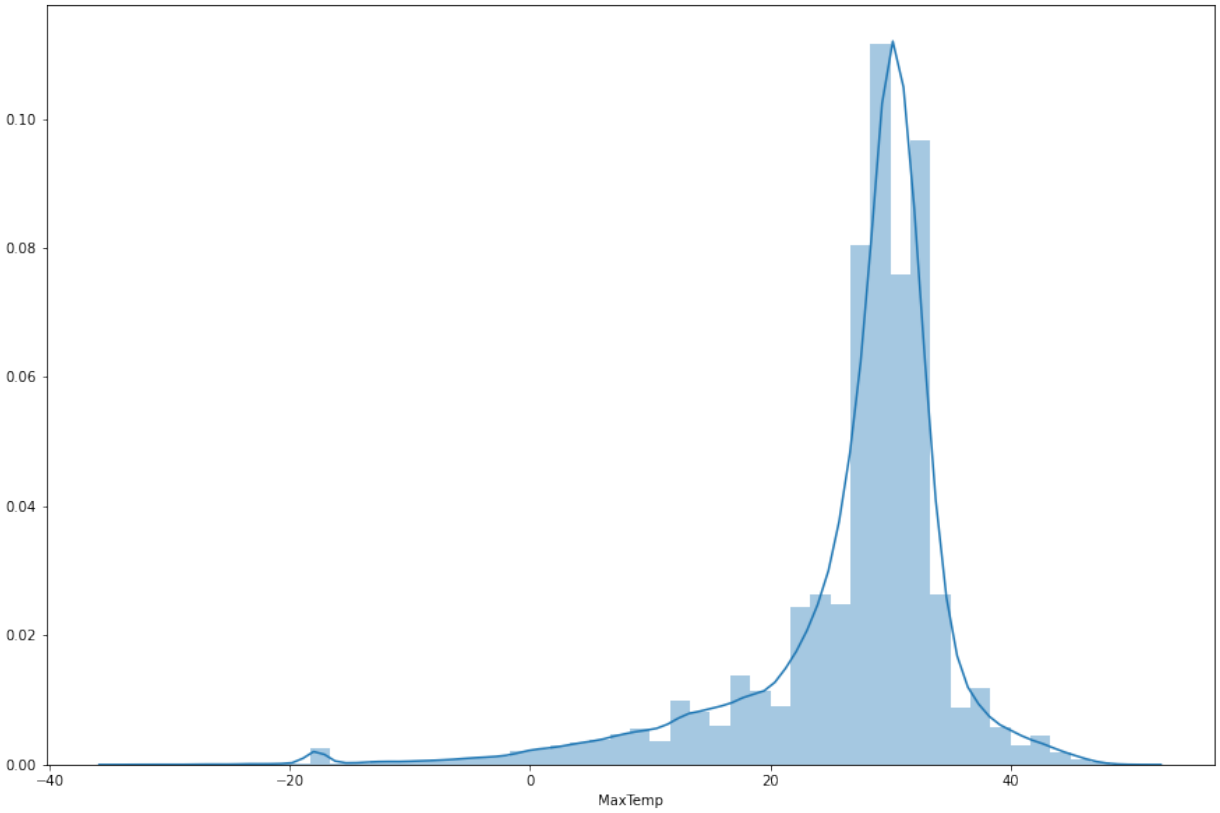
8 rows × 24 columns

In [7]:
```
dataset.plot(x='MinTemp', y='MaxTemp', style='o')
plt.title('MinTemp vs MaxTemp')
plt.xlabel('MinTemp')
plt.ylabel('MaxTemp')
plt.show()
```



In [8]:
```
plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(dataset['MaxTemp'])
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x112039c50>



Attributes are the independent variables while labels are dependent variables whose values are to be predicted. In our dataset, we only have two columns. We want to predict the MaxTemp depending upon the MinTemp recorded. Therefore our attribute set will consist of the "MinTemp" column which is stored in the X variable, and the label will be the "MaxTemp" column which is stored in y variable.

In [9]:
```
X = dataset['MinTemp'].values.reshape(-1,1)
y = dataset['MaxTemp'].values.reshape(-1,1)
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [11]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train) #training the algorithm
```

Out[11]:
```
LinearRegression()
```

In [12]:
```
#To retrieve the intercept:
print(regressor.intercept_)
#For retrieving the slope:
print(regressor.coef_)
```

```
[10.66185201]
[[0.92033997]]
```

In [13]:
```
y_pred = regressor.predict(X_test)
```

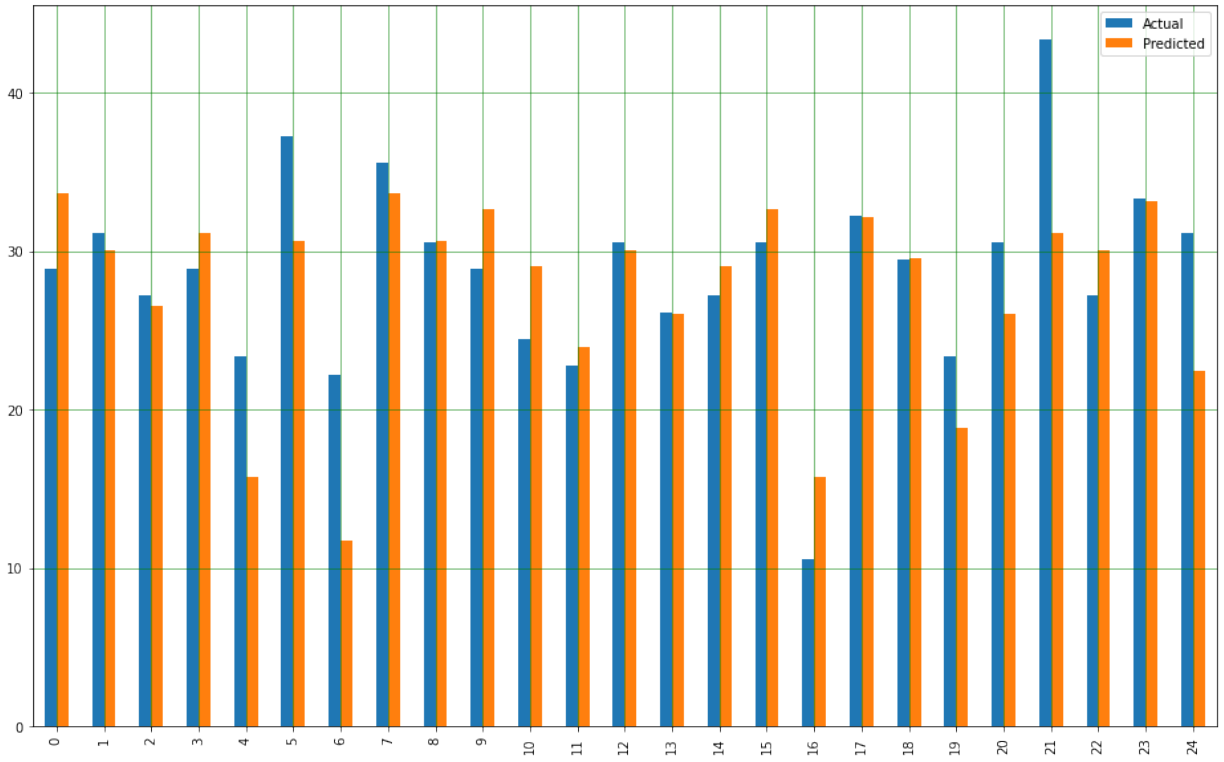
In [14]:
```
df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.fla
df
```

Out[14]:

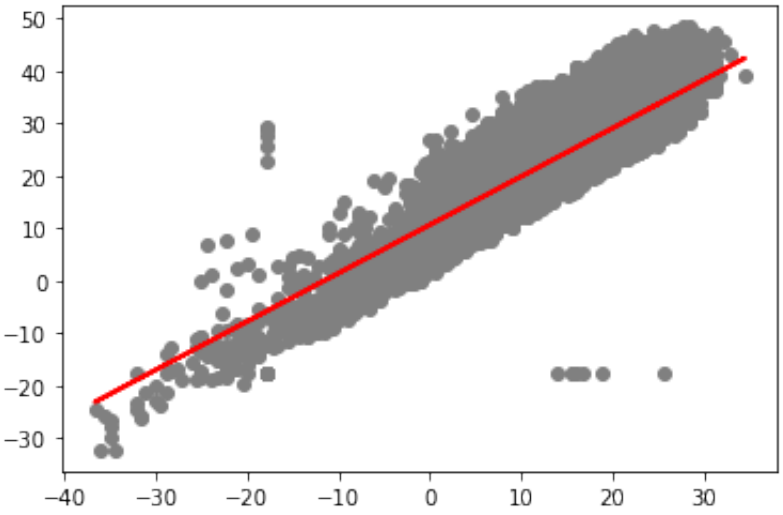
	Actual	Predicted
0	28.888889	33.670351
1	31.111111	30.091251
2	27.222222	26.512151
3	28.888889	31.113851
4	23.333333	15.774852
...	...	...
23803	32.777778	32.136451
23804	32.222222	29.068651
23805	31.111111	32.647751
23806	31.111111	30.602551
23807	36.666667	31.625151

23808 rows × 2 columns

In [15]:
```
df1 = df.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()
```



In [16]:
```
plt.scatter(X_test, y_test,  color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



In [17]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pr
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y
```

```
Mean Absolute Error: 3.19932917837853
Mean Squared Error: 17.631568097568447
Root Mean Squared Error: 4.198996082109204
```

In [ ]:
```

```