In [26]:

```
import pandas as pd
df = pd.read_csv('nyc_weather.csv')
df
```

Out[26]:

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH	Precip
0	1/1/2016	38	23	52	30.03	10	8.0	
1	1/2/2016	36	18	46	30.02	10	7.0	
2	1/3/2016	40	21	47	29.86	10	8.0	
3	1/4/2016	25	9	44	30.05	10	9.0	
4	1/5/2016	20	-3	41	30.57	10	5.0	
5	1/6/2016	33	4	35	30.50	10	4.0	
6	1/7/2016	39	11	33	30.28	10	2.0	
7	1/8/2016	39	29	64	30.20	10	4.0	
8	1/9/2016	44	38	77	30.16	9	8.0	
9	1/10/2016	50	46	71	29.59	4	NaN	
10	1/11/2016	33	8	37	29.92	10	NaN	
11	1/12/2016	35	15	53	29.85	10	6.0	
12	1/13/2016	26	4	42	29.94	10	10.0	
13	1/14/2016	30	12	47	29.95	10	5.0	
14	1/15/2016	43	31	62	29.82	9	5.0	
15	1/16/2016	47	37	70	29.52	8	7.0	
16	1/17/2016	36	23	66	29.78	8	6.0	
17	1/18/2016	25	6	53	29.83	9	12.0	
18	1/19/2016	22	3	42	30.03	10	11.0	
19	1/20/2016	32	15	49	30.13	10	6.0	
20	1/21/2016	31	11	45	30.15	10	6.0	
21	1/22/2016	26	6	41	30.21	9	NaN	
22	1/23/2016	26	21	78	29.77	1	16.0	
23	1/24/2016	28	11	53	29.92	8	6.0	
24	1/25/2016	34	18	54	30.25	10	3.0	
25	1/26/2016	43	29	56	30.03	10	7.0	
26	1/27/2016	41	22	45	30.03	10	7.0	
27	1/28/2016	37	20	51	29.90	10	5.0	
28	1/29/2016	36	21	50	29.58	10	8.0	
29	1/30/2016	34	16	46	30.01	10	7.0	
30	1/31/2016	46	28	52	29.90	10	5.0	

In [27]:

```
df.isna().sum()
```

Out[27]:

```
EST                     0
Temperature             0
DewPoint                0
Humidity                0
Sea Level PressureIn    0
VisibilityMiles         0
WindSpeedMPH            3
PrecipitationIn         0
CloudCover              0
Events                 22
WindDirDegrees          0
dtype: int64
```

In [29]:

```
df = df.drop(['Events', 'PrecipitationIn'], axis=1)
```

In [31]:

```
speed_mode = df['WindSpeedMPH'].mode()[0]
speed_mode
```

Out[31]:

```
5.0
```

In [32]:

```
df['WindSpeedMPH'].fillna(speed_mode, inplace=True)
```

In [33]:

```
df.isna().sum()
```

Out[33]:

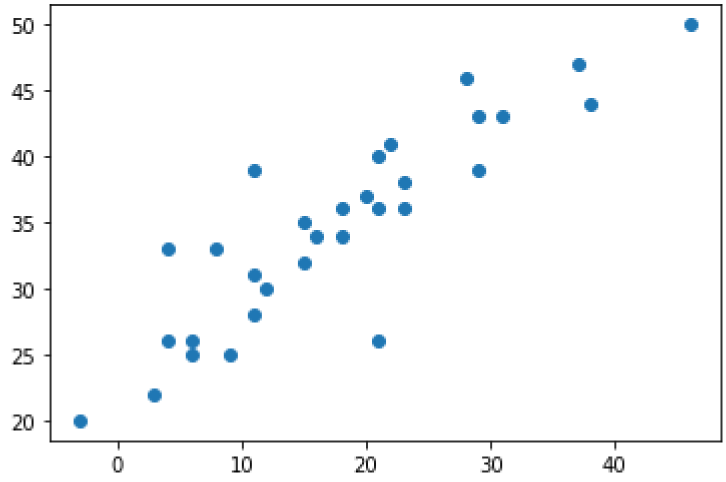
```
EST                     0
Temperature             0
DewPoint                0
Humidity                0
Sea Level PressureIn    0
VisibilityMiles         0
WindSpeedMPH            0
CloudCover              0
WindDirDegrees          0
dtype: int64
```

In [34]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(df['DewPoint'], df['Temperature'])
```

Out[34]:

```
<matplotlib.collections.PathCollection at 0x1ca5a7facd0>
```



In [36]:

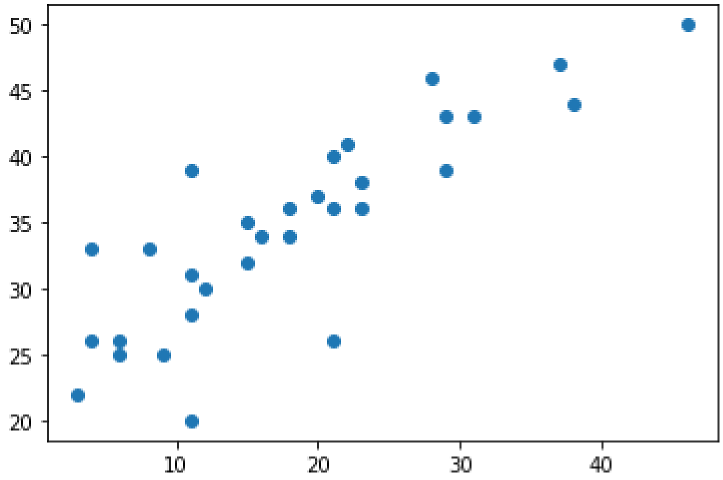
```
dew_mode = df['DewPoint'].mode()[0]
for x in range(0,31):
    if df['DewPoint'][x] < 0:
        df['DewPoint'][x] = dew_mode
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_9168\3913442765.py:4: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  df['DewPoint'][x] = dew_mode
```

In [37]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(df['DewPoint'], df['Temperature'])
```

Out[37]:
```
<matplotlib.collections.PathCollection at 0x1ca5b1dc850>
```



In [48]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['EST'] = le.fit_transform(df['EST'])
```

In [49]:
```
x = df.drop(['WindDirDegrees'], axis=1)
y = df['WindDirDegrees']
```

In [50]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_stat
```

In [51]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [52]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)

x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)
```

In [53]:

```
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression()
model1.fit(x_train,y_train)
```

Out[53]:

```
LogisticRegression()
```

In [54]:

```
y_pred1 = model1.predict(x_test)
y_pred1
```

Out[54]:

```
array([241, 101, 345, 345, 101, 101, 293], dtype=int64)
```

In [55]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred1)
```

Out[55]:

```
0.0
```

In [56]:

```
from sklearn.svm import SVC
model2 = SVC()
model2.fit(x_train, y_train)
```

Out[56]:

```
SVC()
```

In [57]:

```
y_pred2 = model2.predict(x_test)
y_pred2
```

Out[57]:

```
array([293, 345, 345, 293, 345, 293, 293], dtype=int64)
```

In [58]:

```
accuracy_score(y_test, y_pred2)
```

Out[58]:

```
0.14285714285714285
```

In [59]:

```
from sklearn.neighbors import KNeighborsClassifier
model3 = KNeighborsClassifier(n_neighbors = 2)
model3.fit(x_train, y_train)
```

Out[59]:

```
KNeighborsClassifier(n_neighbors=2)
```

In [60]:

```
y_pred3 = model3.predict(x_test)
y_pred3
```

Out[60]:

```
array([277, 101, 235, 284,  79, 101, 286], dtype=int64)
```

In [61]:
```
accuracy_score(y_test, y_pred3)
```

Out[61]:
```
0.0
```

In [62]:
```
from sklearn.linear_model import BayesianRidge
model4 = BayesianRidge()
model4.fit(x_train, y_train)
```

Out[62]:
```
BayesianRidge()
```

In [63]:
```
y_pred4 = model4.predict(x_test)
y_pred4
```

Out[63]:
```
array([253.56941267, 189.16334221, 245.99713142, 240.93405423,
       206.54658623, 172.82248747, 263.03070118])
```

In [64]:
```
accuracy_score(y_test, y_pred4.round())
```

Out[64]:
```
0.0
```

In [65]:
```
from sklearn.linear_model import LinearRegression
model5 = LinearRegression()
model5.fit(x_train,y_train)
```

Out[65]:
```
LinearRegression()
```

In [66]:
```
y_pred5 = model5.predict(x_test)
y_pred5
```

Out[66]:
```
array([255.04337288, 158.86791275, 243.28562892, 235.37558102,
       184.97056893, 134.47511744, 269.03320308])
```

In [67]:
```
accuracy_score(y_test, y_pred5.round())
```

Out[67]:
```
0.0
```

In [ ]:
```

```