WEATHER PREDICTION

PEDDU.SAI HARIKA

AM.EN.U4AIE19047

Collecting Data From Weather Underground

In [1]:

```
from datetime import datetime, timedelta
import time
from collections import namedtuple
import pandas as pd
import requests
import matplotlib.pyplot as plt
```

In [2]:

```
API_KEY = 'Insert API Key Here'
BASE_URL = "http://api.wunderground.com/api/{}/history_{}/q/26.9124,75.7873.json" #latitude and longitude of jaipur
```

In [3]:

```
target_date = datetime(2018, 3, 12)
features = ["date", "meantempm", "meandewptm", "meanpressurem", "maxhumidity", "minhumidity", "maxtempm",
            "mintempm", "maxdewptm", "mindewptm", "maxpressurem", "minpressurem", "precipm"]
DailySummary = namedtuple("DailySummary", features) # date to be change after the getting the exact 500 dataset to increase further
```

In [4]:

```
def extract_weather_data(url, api_key, target_date, days):
    records = []
    for _ in range(days):
        request = BASE_URL.format(API_KEY, target_date.strftime('%Y%m%d'))
        response = requests.get(request)
        if response.status_code == 200:
            data = response.json()['history']['dailysummary'][0]
            records.append(DailySummary(
                date=target_date,
                meantempm=data['meantempm'],
                meandewptm=data['meandewptm'],
                meanpressurem=data['meanpressurem'],
                maxhumidity=data['maxhumidity'],
                minhumidity=data['minhumidity'],
                maxtempm=data['maxtempm'],
                mintempm=data['mintempm'],
                maxdewptm=data['maxdewptm'],
                mindewptm=data['mindewptm'],
                maxpressurem=data['maxpressurem'],
                minpressurem=data['minpressurem'],
                precipm=data['precipm']))
        time.sleep(6)
        target_date += timedelta(days=1)
    return records
```

In [ ]:

```
records = extract_weather_data(BASE_URL, API_KEY, target_date, 10)
```

In [ ]:

```
records += extract_weather_data(BASE_URL, API_KEY, target_date, 30)
```

In [12]:

```
df = pd.DataFrame(records, columns=features).set_index('date')
```

In [13]:

```
tmp = df[['meantempm', 'meandewptm']].tail(10)
tmp
```

Out[13]:

date	meantempm	meandewptm
2018-03-12	27	3
2018-03-13	28	1
2018-03-14	27	2
2018-03-15	26	5
2018-03-16	26	4
2018-03-17	24	2
2018-03-18	25	1
2018-03-19	26	3
2018-03-20	27	7
2018-03-21	26	2

In [14]:

```
tmp = df[['meantempm', 'meandewptm']].head(10)
tmp
```

Out[14]:

date	meantempm	meandewptm
2018-03-12	27	3
2018-03-13	28	1
2018-03-14	27	2
2018-03-15	26	5
2018-03-16	26	4
2018-03-17	24	2
2018-03-18	25	1
2018-03-19	26	3
2018-03-20	27	7
2018-03-21	26	2

In [ ]:

```
df.to_csv('JaipurRawData3.csv')
```

In [17]:

```
df = pd.read_csv('JaipurRawData3.csv').set_index('date')
```

In [21]:

```
tmp = df[['meantempm', 'meandewptm']].head(10)
tmp
```

Out[21]:

date	meantempm	meandewptm
2016-05-01	34	-1
2016-05-02	36	4
2016-05-03	35	6
2016-05-04	34	7
2016-05-05	31	11
2016-05-06	28	13
2016-05-07	30	10
2016-05-08	34	8
2016-05-09	34	11
2016-05-10	34	16

In [19]:

```
tmp = df[['meantempm', 'meandewptm']].tail(10)
tmp
```

Out[19]:

date	meantempm	meandewptm
2018-03-02	26	6
2018-03-03	26	4
2018-03-04	25	8
2018-03-05	23	7
2018-03-06	22	4
2018-03-07	24	2
2018-03-08	24	1
2018-03-09	26	3
2018-03-10	26	4
2018-03-11	26	3

In [22]:

```
# 1 day prior
N = 1

# target measurement of mean temperature
feature = 'meantempm'

# total number of rows
rows = tmp.shape[0]

# a list representing Nth prior measurements of feature
# notice that the front of the list needs to be padded with N
# None values to maintain the constistent rows length for each N
nth_prior_measurements = [None]*N + [tmp[feature][i-N] for i in range(N, rows)]

# make a new column name of feature_N and add to DataFrame
```

```
col_name = "{}_{}".format(feature, N)
tmp[col_name] = nth_prior_measurements
tmp
```

Out[22]:

date	meantempm	meandewptm	meantempm_1
2016-05-01	34	-1	NaN
2016-05-02	36	4	34.0
2016-05-03	35	6	36.0
2016-05-04	34	7	35.0
2016-05-05	31	11	34.0
2016-05-06	28	13	31.0
2016-05-07	30	10	28.0
2016-05-08	34	8	30.0
2016-05-09	34	11	34.0
2016-05-10	34	16	34.0

In [23]:

```
def derive_nth_day_feature(df, feature, N):
    rows = df.shape[0]
    nth_prior_meassurements = [None]*N + [df[feature][i-N] for i in range(N, rows)]
    col_name = "{}_{}".format(feature, N)
    df[col_name] = nth_prior_meassurements
```

In [24]:

```
for feature in features:
    if feature != 'date':
        for N in range(1, 4):
            derive_nth_day_feature(df, feature, N)
```

In [25]:

```
df.columns
```

Out[25]:

```
Index(['meantempm', 'meandewptm', 'meanpressurem', 'maxhumidity',
       'minhumidity', 'maxtempm', 'mintempm', 'maxdewptm', 'mindewptm',
       'maxpressurem', 'minpressurem', 'precipm', 'meantempm_1', 'meantempm_2',
       'meantempm_3', 'meandewptm_1', 'meandewptm_2', 'meandewptm_3',
       'meanpressurem_1', 'meanpressurem_2', 'meanpressurem_3',
       'maxhumidity_1', 'maxhumidity_2', 'maxhumidity_3', 'minhumidity_1',
       'minhumidity_2', 'minhumidity_3', 'maxtempm_1', 'maxtempm_2',
       'maxtempm_3', 'mintempm_1', 'mintempm_2', 'mintempm_3', 'maxdewptm_1',
       'maxdewptm_2', 'maxdewptm_3', 'mindewptm_1', 'mindewptm_2',
       'mindewptm_3', 'maxpressurem_1', 'maxpressurem_2', 'maxpressurem_3',
       'minpressurem_1', 'minpressurem_2', 'minpressurem_3', 'precipm_1',
       'precipm_2', 'precipm_3'],
      dtype='object')
```

In [26]:

```
# make list of original features without meantempm, mintempm, and maxtempm
to_remove = [feature
             for feature in features
             if feature not in ['meantempm', 'mintempm', 'maxtempm']]

# make a list of columns to keep
to_keep = [col for col in df.columns if col not in to_remove]
```

```
# select only the columns in to_keep and assign to df
df = df[to_keep]
df.columns
```

Out[26]:

```
Index(['meantempm', 'maxtempm', 'mintempm', 'meantempm_1', 'meantempm_2',
       'meantempm_3', 'meandewptm_1', 'meandewptm_2', 'meandewptm_3',
       'meanpressurem_1', 'meanpressurem_2', 'meanpressurem_3',
       'maxhumidity_1', 'maxhumidity_2', 'maxhumidity_3', 'minhumidity_1',
       'minhumidity_2', 'minhumidity_3', 'maxtempm_1', 'maxtempm_2',
       'maxtempm_3', 'mintempm_1', 'mintempm_2', 'mintempm_3', 'maxdewptm_1',
       'maxdewptm_2', 'maxdewptm_3', 'mindewptm_1', 'mindewptm_2',
       'mindewptm_3', 'maxpressurem_1', 'maxpressurem_2', 'maxpressurem_3',
       'minpressurem_1', 'minpressurem_2', 'minpressurem_3', 'precipm_1',
       'precipm_2', 'precipm_3'],
      dtype='object')
```

In [27]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 679 entries, 2016-05-01 to 2018-03-11
Data columns (total 39 columns):
meantempm          679 non-null int64
maxtempm           679 non-null int64
mintempm           679 non-null int64
meantempm_1        678 non-null float64
meantempm_2        677 non-null float64
meantempm_3        676 non-null float64
meandewptm_1       678 non-null float64
meandewptm_2       677 non-null float64
meandewptm_3       676 non-null float64
meanpressurem_1    678 non-null float64
meanpressurem_2    677 non-null float64
meanpressurem_3    676 non-null float64
maxhumidity_1      678 non-null float64
maxhumidity_2      677 non-null float64
maxhumidity_3      676 non-null float64
minhumidity_1      678 non-null float64
minhumidity_2      677 non-null float64
minhumidity_3      676 non-null float64
maxtempm_1         678 non-null float64
maxtempm_2         677 non-null float64
maxtempm_3         676 non-null float64
mintempm_1         678 non-null float64
mintempm_2         677 non-null float64
mintempm_3         676 non-null float64
maxdewptm_1        678 non-null float64
maxdewptm_2        677 non-null float64
maxdewptm_3        676 non-null float64
mindewptm_1        678 non-null float64
mindewptm_2        677 non-null float64
mindewptm_3        676 non-null float64
maxpressurem_1     678 non-null float64
maxpressurem_2     677 non-null float64
maxpressurem_3     676 non-null float64
minpressurem_1     678 non-null float64
minpressurem_2     677 non-null float64
minpressurem_3     676 non-null float64
precipm_1          678 non-null float64
precipm_2          677 non-null float64
precipm_3          676 non-null float64
dtypes: float64(36), int64(3)
memory usage: 232.2+ KB
```

In [28]:

```
df = df.apply(pd.to_numeric, errors='coerce')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 679 entries, 2016-05-01 to 2018-03-11
Data columns (total 39 columns):
meantempm           679 non-null int64
maxtempm            679 non-null int64
mintempm            679 non-null int64
meantempm_1         678 non-null float64
meantempm_2         677 non-null float64
meantempm_3         676 non-null float64
meandewptm_1        678 non-null float64
meandewptm_2        677 non-null float64
meandewptm_3        676 non-null float64
meanpressurem_1     678 non-null float64
meanpressurem_2     677 non-null float64
meanpressurem_3     676 non-null float64
maxhumidity_1       678 non-null float64
maxhumidity_2       677 non-null float64
maxhumidity_3       676 non-null float64
minhumidity_1       678 non-null float64
minhumidity_2       677 non-null float64
minhumidity_3       676 non-null float64
maxtempm_1          678 non-null float64
maxtempm_2          677 non-null float64
maxtempm_3          676 non-null float64
mintempm_1          678 non-null float64
mintempm_2          677 non-null float64
mintempm_3          676 non-null float64
maxdewptm_1         678 non-null float64
maxdewptm_2         677 non-null float64
maxdewptm_3         676 non-null float64
mindewptm_1         678 non-null float64
mindewptm_2         677 non-null float64
mindewptm_3         676 non-null float64
maxpressurem_1      678 non-null float64
maxpressurem_2      677 non-null float64
maxpressurem_3      676 non-null float64
minpressurem_1      678 non-null float64
minpressurem_2      677 non-null float64
minpressurem_3      676 non-null float64
precipm_1           678 non-null float64
precipm_2           677 non-null float64
precipm_3           676 non-null float64
dtypes: float64(36), int64(3)
memory usage: 232.2+ KB
```

In [29]:

```
# Call describe on df and transpose it due to the large number of columns
spread = df.describe().T

# precalculate interquartile range for ease of use in next calculation
IQR = spread['75%'] - spread['25%']

# create an outliers column which is either 3 IQRs below the first quartile or
# 3 IQRs above the third quartile
spread['outliers'] = (spread['min']<(spread['25%']-(3*IQR)))|(spread['max'] > (spread['7
5%']+3*IQR))

# just display the features containing extreme outliers
spread.ix[spread.outliers,]
```

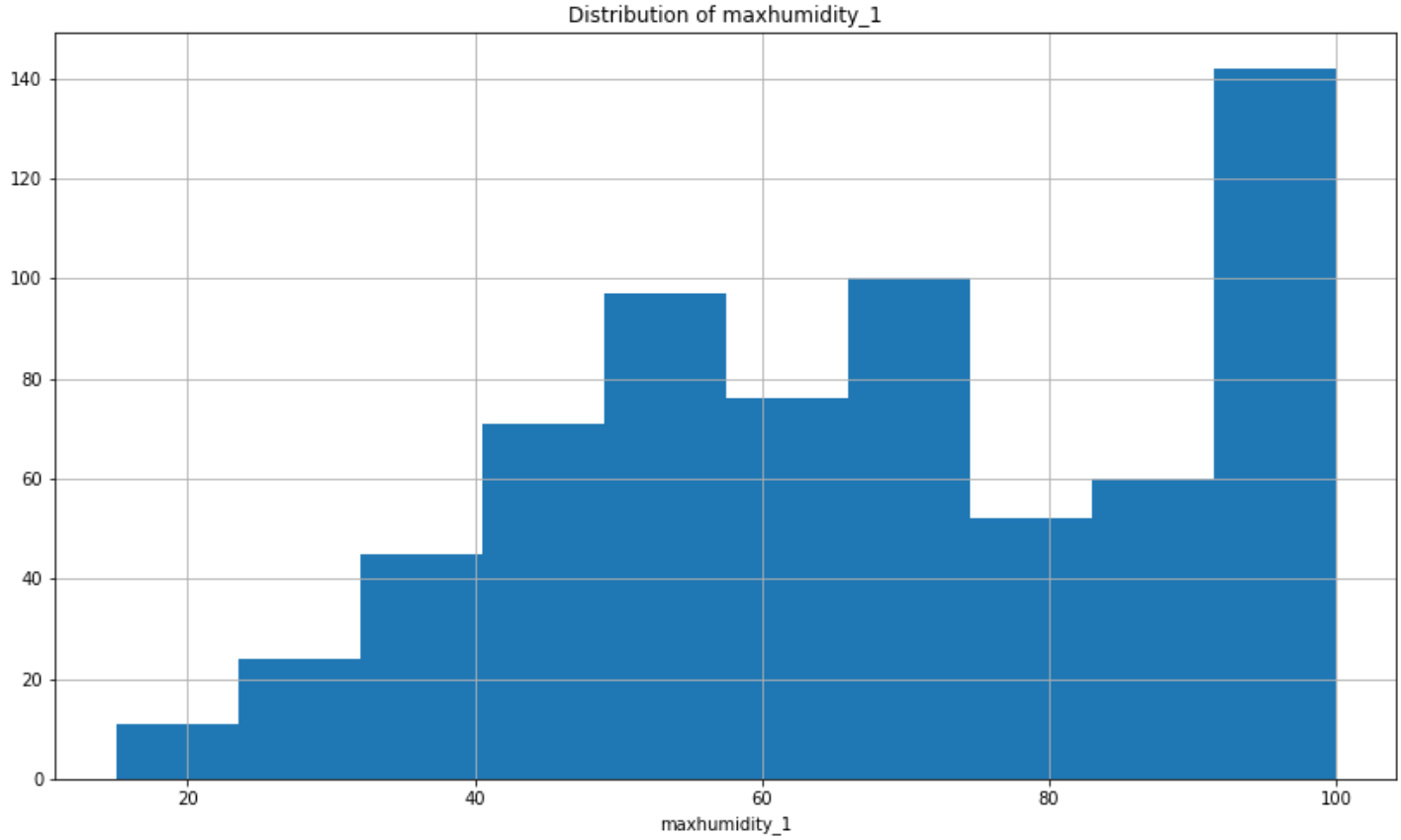
```
C:\python\envs\tensorflow\lib\site-packages\ipykernel_launcher.py:12: DeprecationWarning:

.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  if sys.path[0] == '':
```

Out[29]:

	count	mean	std	min	25%	50%	75%	max	outliers
mindewptm_1	678.0	7.454277	11.696310	-94.0	0.0	6.0	18.0	25.0	True
mindewptm_2	677.0	7.465288	11.701441	-94.0	0.0	6.0	18.0	25.0	True
mindewptm_3	676.0	7.477811	11.705565	-94.0	0.0	6.0	18.0	25.0	True
precipm_1	678.0	1.246903	5.428048	0.0	0.0	0.0	0.0	57.0	True
precipm_2	677.0	1.248744	5.431849	0.0	0.0	0.0	0.0	57.0	True
precipm_3	676.0	1.250592	5.435659	0.0	0.0	0.0	0.0	57.0	True

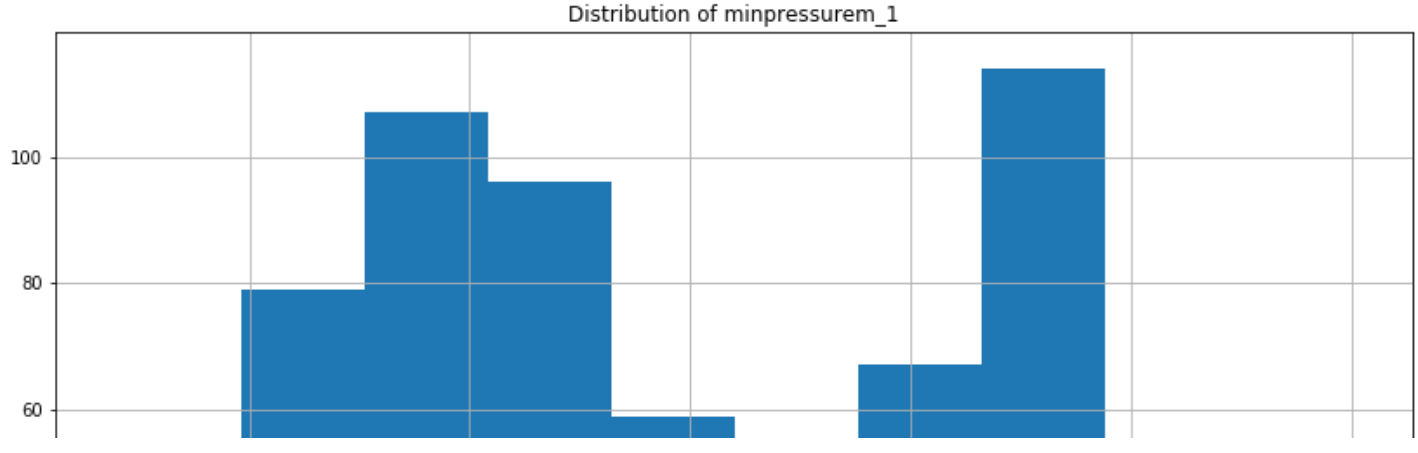
In [30]:

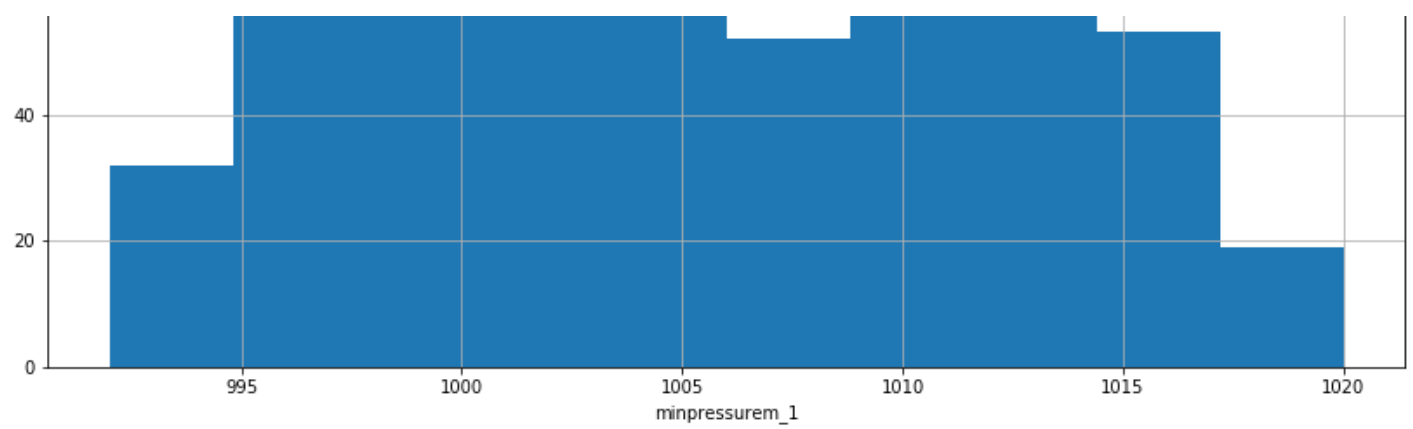
```
%matplotlib inline
plt.rcParams['figure.figsize'] = [14, 8]
df.maxhumidity_1.hist()
plt.title('Distribution of maxhumidity_1')
plt.xlabel('maxhumidity_1')
plt.show()
```



In [31]:

```
df.minpressurem_1.hist()
plt.title('Distribution of minpressurem_1')
plt.xlabel('minpressurem_1')
plt.show()
```





In [32]:

```
# iterate over the precip columns
for precip_col in ['precipm_1', 'precipm_2', 'precipm_3']:
    # create a boolean array of values representing nans
    missing_vals = pd.isnull(df[precip_col])
    df[precip_col][missing_vals] = 0
```

```
C:\python\envs\tensorflow\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexin
g.html#indexing-view-versus-copy
  """
```

In [33]:

```
df = df.dropna()
```

In [34]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 676 entries, 2016-05-04 to 2018-03-11
Data columns (total 39 columns):
meantempm           676 non-null int64
maxtempm            676 non-null int64
mintempm            676 non-null int64
meantempm_1         676 non-null float64
meantempm_2         676 non-null float64
meantempm_3         676 non-null float64
meandewptm_1        676 non-null float64
meandewptm_2        676 non-null float64
meandewptm_3        676 non-null float64
meanpressurem_1     676 non-null float64
meanpressurem_2     676 non-null float64
meanpressurem_3     676 non-null float64
maxhumidity_1       676 non-null float64
maxhumidity_2       676 non-null float64
maxhumidity_3       676 non-null float64
minhumidity_1       676 non-null float64
minhumidity_2       676 non-null float64
minhumidity_3       676 non-null float64
maxtempm_1          676 non-null float64
maxtempm_2          676 non-null float64
maxtempm_3          676 non-null float64
mintempm_1          676 non-null float64
mintempm_2          676 non-null float64
mintempm_3          676 non-null float64
maxdewptm_1         676 non-null float64
maxdewptm_2         676 non-null float64
maxdewptm_3         676 non-null float64
mindewptm_1         676 non-null float64
mindewptm_2         676 non-null float64
mindewptm_3         676 non-null float64
```

```
maxpressurem_1      676 non-null float64
maxpressurem_2      676 non-null float64
maxpressurem_3      676 non-null float64
minpressurem_1      676 non-null float64
minpressurem_2      676 non-null float64
minpressurem_3      676 non-null float64
precipm_1           676 non-null float64
precipm_2           676 non-null float64
precipm_3           676 non-null float64
dtypes: float64(36), int64(3)
memory usage: 211.2+ KB
```

In [35]:

```
df.to_csv('Weather.csv')
```

LINEAR REGRESSION MODEL

In [1]:

```
import pandas as pd
df = pd.read_csv(r'C:\Users\New\Desktop\Python Project\Weather.csv').set_index('date')
```

In [2]:

```
df.corr()[['meantempm']].sort_values('meantempm')
```

Out[2]:

	meantempm
minpressurem_1	-0.830438
minpressurem_2	-0.809432
minpressurem_3	-0.795442
meanpressurem_1	-0.794987
meanpressurem_2	-0.776774
maxpressurem_1	-0.764999
meanpressurem_3	-0.764715
maxpressurem_2	-0.745532
maxpressurem_3	-0.732154
maxhumidity_1	-0.254148
maxhumidity_2	-0.232007
maxhumidity_3	-0.214156
precipm_1	0.032026
precipm_2	0.048588
precipm_3	0.077058
minhumidity_1	0.085576
minhumidity_2	0.092412
minhumidity_3	0.094188
mindewptm_3	0.382905
mindewptm_2	0.400483
mindewptm_1	0.412313
meandewptm_3	0.469458
meandewptm_2	0.479336
meandewptm_1	0.492208
maxdewptm_2	0.530555

	meantempm
maxdewptm_3	0.572555
maxdewptm_2	0.575732
maxdewptm_1	0.588225
mintempm_3	0.887204
maxtempm_3	0.892415
mintempm_2	0.903810
maxtempm_2	0.912651
meantempm_3	0.921327
mintempm_1	0.930626
meantempm_2	0.939371
maxtempm_1	0.940667
maxtempm	0.959088
meantempm_1	0.966986
mintempm	0.968887
meantempm	1.000000

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 676 entries, 2016-05-04 to 2018-03-11
Data columns (total 39 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   meantempm         676 non-null    int64  
 1   maxtempm          676 non-null    int64  
 2   mintempm          676 non-null    int64  
 3   meantempm_1       676 non-null    float64
 4   meantempm_2       676 non-null    float64
 5   meantempm_3       676 non-null    float64
 6   meandewptm_1      676 non-null    float64
 7   meandewptm_2      676 non-null    float64
 8   meandewptm_3      676 non-null    float64
 9   meanpressurem_1   676 non-null    float64
 10  meanpressurem_2   676 non-null    float64
 11  meanpressurem_3   676 non-null    float64
 12  maxhumidity_1     676 non-null    float64
 13  maxhumidity_2     676 non-null    float64
 14  maxhumidity_3     676 non-null    float64
 15  minhumidity_1     676 non-null    float64
 16  minhumidity_2     676 non-null    float64
 17  minhumidity_3     676 non-null    float64
 18  maxtempm_1        676 non-null    float64
 19  maxtempm_2        676 non-null    float64
 20  maxtempm_3        676 non-null    float64
 21  mintempm_1        676 non-null    float64
 22  mintempm_2        676 non-null    float64
 23  mintempm_3        676 non-null    float64
 24  maxdewptm_1       676 non-null    float64
 25  maxdewptm_2       676 non-null    float64
 26  maxdewptm_3       676 non-null    float64
 27  mindewptm_1       676 non-null    float64
 28  mindewptm_2       676 non-null    float64
 29  mindewptm_3       676 non-null    float64
 30  maxpressurem_1    676 non-null    float64
 31  maxpressurem_2    676 non-null    float64
 32  maxpressurem_3    676 non-null    float64
 33  minpressurem_1    676 non-null    float64
 34  minpressurem_2    676 non-null    float64
 35  minpressurem_3    676 non-null    float64
 36  precipm_1         676 non-null    float64
 37  precipm_2         676 non-null    float64
 38  precipm_3         676 non-null    float64
```

```
dtypes: float64(36), int64(3)
memory usage: 211.2+ KB
```

In [4]:

```
df.shape
```

Out[4]:

```
(676, 39)
```

In [5]:

```
df.size
```

Out[5]:

```
26364
```

In [6]:

```
list(df.columns)
```

Out[6]:

```
['meantempm',
 'maxtempm',
 'mintempm',
 'meantempm_1',
 'meantempm_2',
 'meantempm_3',
 'meandewptm_1',
 'meandewptm_2',
 'meandewptm_3',
 'meanpressurem_1',
 'meanpressurem_2',
 'meanpressurem_3',
 'maxhumidity_1',
 'maxhumidity_2',
 'maxhumidity_3',
 'minhumidity_1',
 'minhumidity_2',
 'minhumidity_3',
 'maxtempm_1',
 'maxtempm_2',
 'maxtempm_3',
 'mintempm_1',
 'mintempm_2',
 'mintempm_3',
 'maxdewptm_1',
 'maxdewptm_2',
 'maxdewptm_3',
 'mindewptm_1',
 'mindewptm_2',
 'mindewptm_3',
 'maxpressurem_1',
 'maxpressurem_2',
 'maxpressurem_3',
 'minpressurem_1',
 'minpressurem_2',
 'minpressurem_3',
 'precipm_1',
 'precipm_2',
 'precipm_3']
```

In [7]:

```
df.describe(include='all')
```

Out[7]:

	meantempm	maxtempm	mintempm	meantempm_1	meantempm_2	meantempm_3	meandewptm_1	meandewptm_2

count	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
mean	26.053254	32.523669	19.630178	26.066568	26.081361	26.093195	11.995562	11.995562
std	6.208964	5.987966	6.825827	6.218491	6.230216	6.237655	8.731127	8.731127
min	10.000000	18.000000	3.000000	10.000000	10.000000	10.000000	-10.000000	-10.000000
25%	20.000000	28.000000	13.000000	20.000000	20.000000	20.000000	5.000000	5.000000
50%	28.000000	33.000000	22.000000	28.000000	28.000000	28.000000	10.000000	10.000000
75%	30.000000	37.000000	25.000000	30.000000	30.000000	30.000000	21.000000	21.000000
max	38.000000	46.000000	32.000000	38.000000	38.000000	38.000000	26.000000	26.000000

8 rows × 39 columns

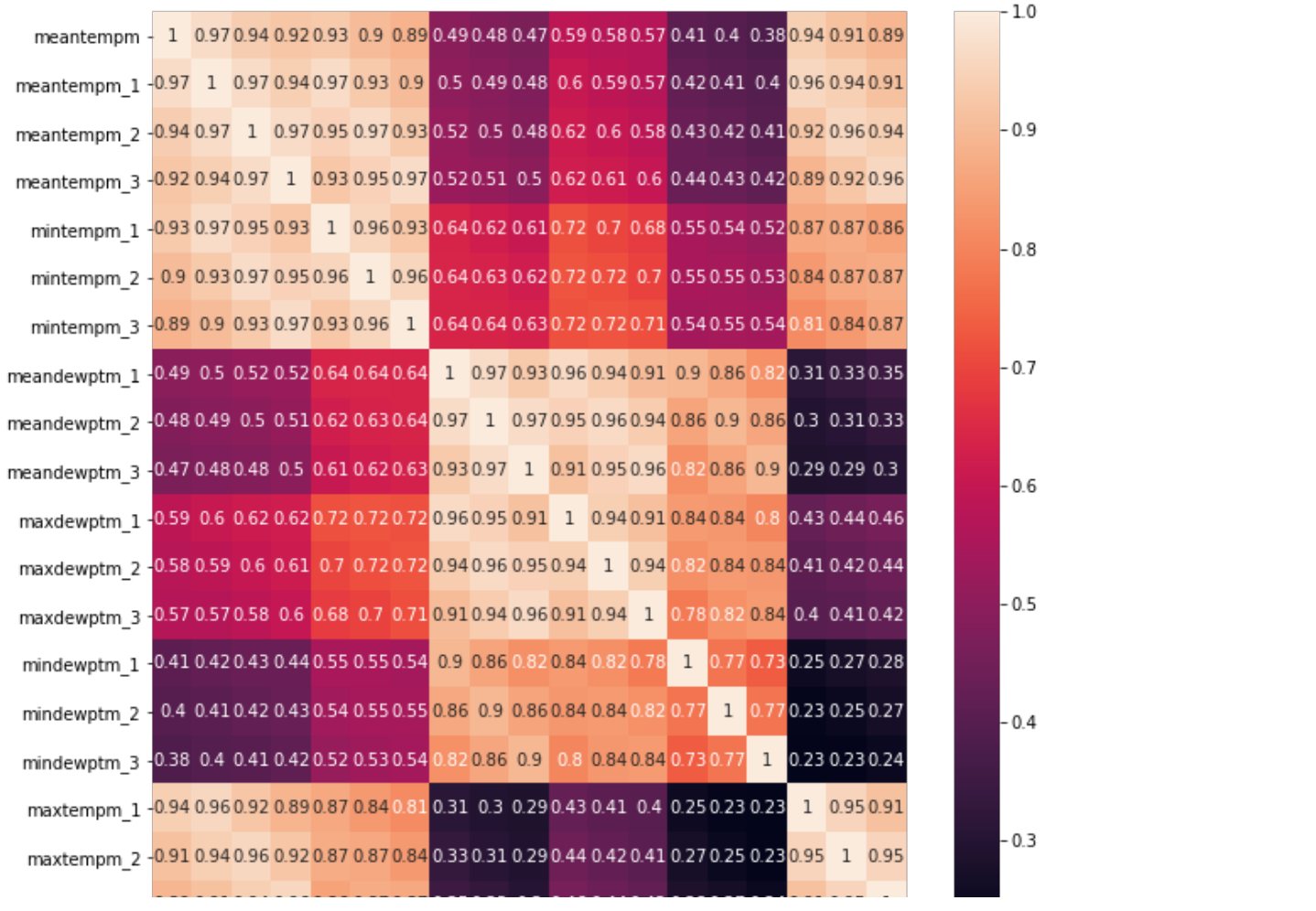
In [8]:

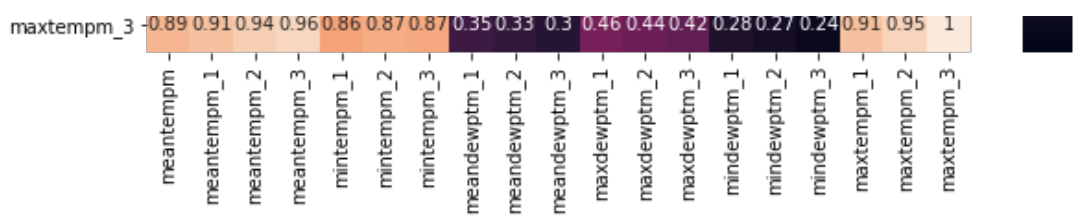
```
predictors = ['meantempm_1',   'meantempm_2',   'meantempm_3',
              'mintempm_1',    'mintempm_2',    'mintempm_3',
              'meandewptm_1', 'meandewptm_2', 'meandewptm_3',
              'maxdewptm_1',   'maxdewptm_2',   'maxdewptm_3',
              'mindewptm_1',   'mindewptm_2',   'mindewptm_3',
              'maxtempm_1',    'maxtempm_2',    'maxtempm_3']
df2 = df[['meantempm'] + predictors]
```

Visualizing the Relationships

In [9]:

```
#plotting the correlation
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize  =(10,10))
sns.heatmap(df2.corr(),annot=True)
plt.show()
```



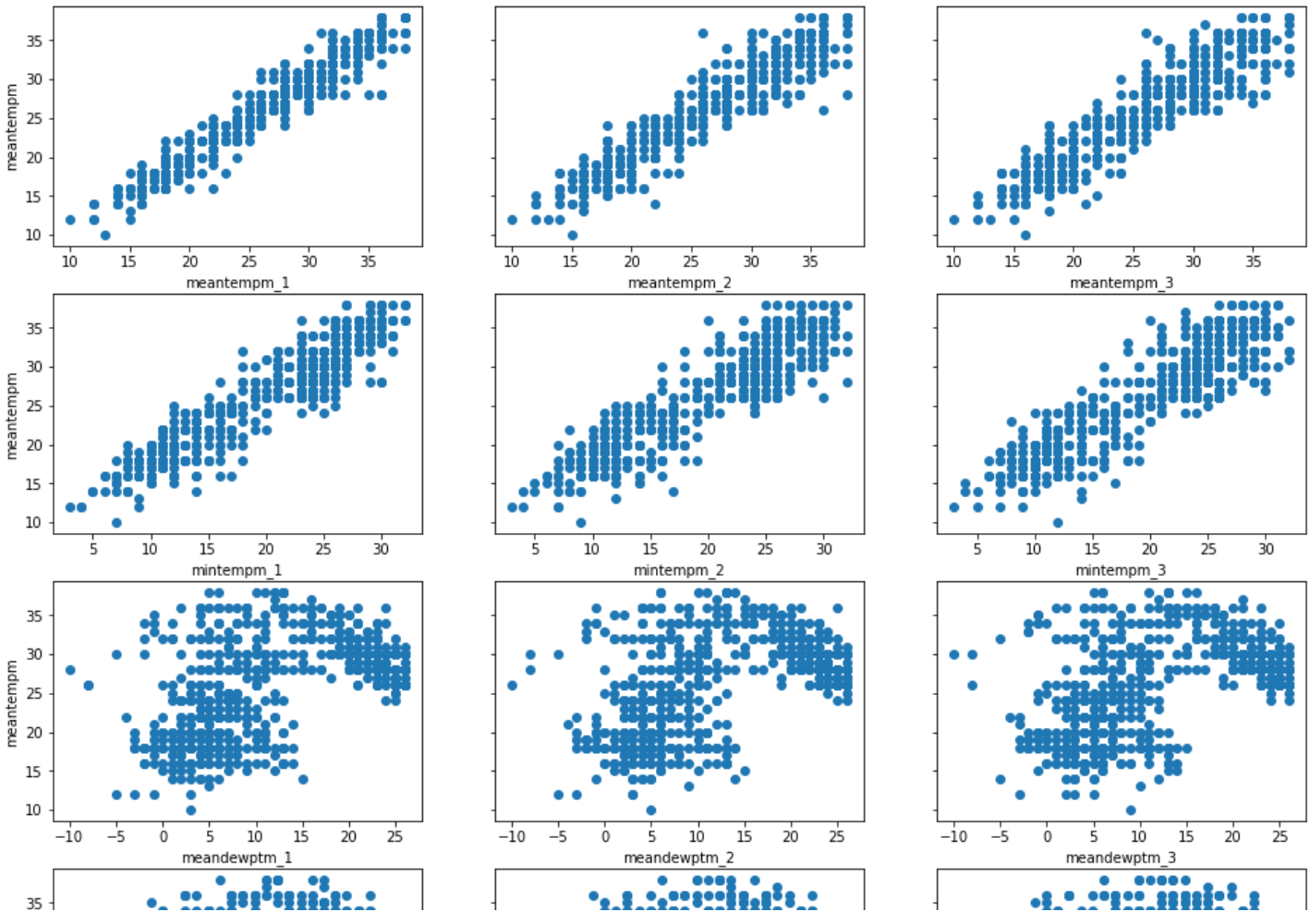


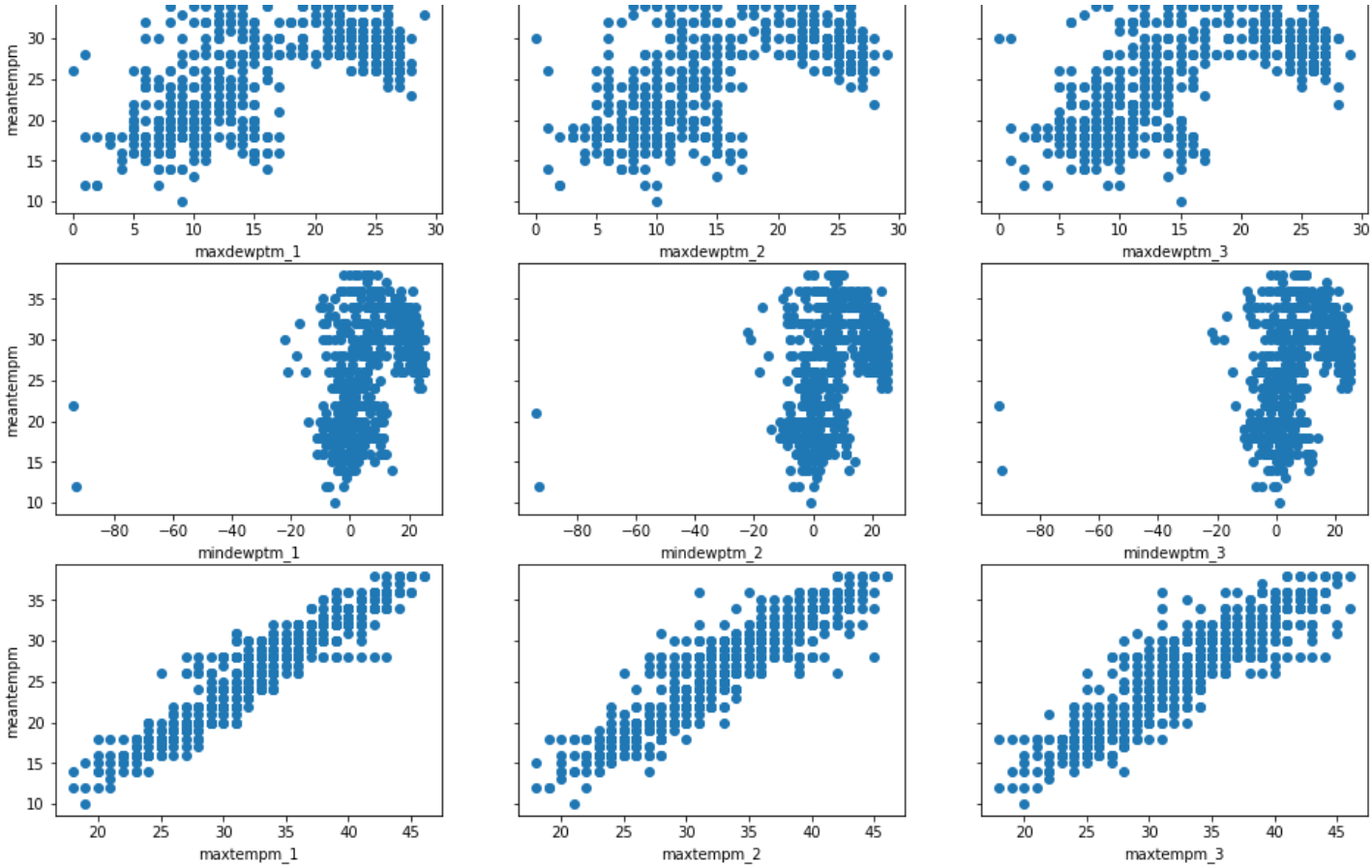
In [10]:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

In [11]:

```
%matplotlib inline

# manually set the parameters of the figure to and appropriate size
plt.rcParams['figure.figsize'] = [16, 22]

# call subplots specifying the grid structure we desire and that
# the y axes should be shared
fig, axes = plt.subplots(nrows=6, ncols=3, sharey=True)

# Since it would be nice to loop through the features in to build this plot
# let us rearrange our data into a 2D array of 6 rows and 3 columns
arr = np.array(predictors).reshape(6, 3)

# using enumerate to loop over the arr 2D array of rows and columns
# and create scatter plots of each meantempm vs each feature
for row, col_arr in enumerate(arr):
    for col, feature in enumerate(col_arr):
        axes[row, col].scatter(df2[feature], df2['meantempm'])
        if col == 0:
            axes[row, col].set(xlabel=feature, ylabel='meantempm')
        else:
            axes[row, col].set(xlabel=feature)
plt.show()
```





In [12]:

```
# import the relevant module
import statsmodels.api as sm

# separating our my predictor variables (X) from my outcome variable y
X = df2[predictors]
y = df2['meantempm']

# Adding a constant to the predictor variable set to represent the Bo intercept
X = sm.add_constant(X)
```

In [13]:

```
# (1) selecting a significance value
alpha = 0.05

# (2) Fitting the model
model = sm.OLS(y, X).fit()

# (3) evaluating the coefficients' p-values
model.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	meantempm	**R-squared:**	0.945
Model:	OLS	**Adj. R-squared:**	0.943
Method:	Least Squares	**F-statistic:**	626.4
Date:	Fri, 04 Dec 2020	**Prob (F-statistic):**	0.00
Time:	22:38:05	**Log-Likelihood:**	-1213.1
No. Observations:	676	**AIC:**	2464.
Df Residuals:	657	**BIC:**	2550.
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.2438	0.553	-2.247	0.025	-2.331	-0.157
meantempm_1	0.0882	0.163	0.542	0.588	-0.231	0.408
meantempm_2	0.2695	0.163	1.656	0.098	-0.050	0.589
meantempm_3	0.1839	0.163	1.125	0.261	-0.137	0.505
mintempm_1	0.2278	0.089	2.573	0.010	0.054	0.402
mintempm_2	-0.1240	0.090	-1.372	0.170	-0.301	0.053
mintempm_3	-0.0101	0.088	-0.116	0.908	-0.182	0.162
meandewptm_1	0.0769	0.040	1.931	0.054	-0.001	0.155
meandewptm_2	-0.0397	0.050	-0.793	0.428	-0.138	0.059
meandewptm_3	-0.0190	0.044	-0.434	0.665	-0.105	0.067
maxdewptm_1	-0.0470	0.037	-1.256	0.210	-0.121	0.027
maxdewptm_2	0.0018	0.038	0.047	0.962	-0.073	0.077
maxdewptm_3	0.1017	0.037	2.761	0.006	0.029	0.174
mindewptm_1	-0.0055	0.011	-0.488	0.626	-0.028	0.017
mindewptm_2	0.0034	0.011	0.298	0.766	-0.019	0.025
mindewptm_3	-0.0148	0.011	-1.316	0.189	-0.037	0.007
maxtempm_1	0.6232	0.089	7.016	0.000	0.449	0.798
maxtempm_2	-0.2539	0.093	-2.729	0.007	-0.437	-0.071
maxtempm_3	-0.0518	0.090	-0.574	0.566	-0.229	0.125

Omnibus:	89.268	Durbin-Watson:	2.040
Prob(Omnibus):	0.000	Jarque-Bera (JB):	267.497
Skew:	-0.639	Prob(JB):	8.20e-59
Kurtosis:	5.805	Cond. No.	890.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [14]:

```
# (4) - Using pandas drop function to remove this column from X
X = X.drop('meandewptm_3', axis=1)

# (5) Fitting the model
model = sm.OLS(y, X).fit()

model.summary()
```

Out[14]:

OLS Regression Results

Dep. Variable:	meantempm	R-squared:	0.945
Model:	OLS	Adj. R-squared:	0.944
Method:	Least Squares	F-statistic:	664.1
Date:	Fri, 04 Dec 2020	Prob (F-statistic):	0.00
Time:	22:38:22	Log-Likelihood:	-1213.2
No. Observations:	676	AIC:	2462.
Df Residuals:	658	BIC:	2544.
Df Model:	17		

Covariance Type:	nonrobust

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.2578	0.552	-2.278	0.023	-2.342	-0.174
meantempm_1	0.0874	0.163	0.537	0.591	-0.232	0.407
meantempm_2	0.2706	0.163	1.664	0.097	-0.049	0.590
meantempm_3	0.1820	0.163	1.114	0.266	-0.139	0.503
mintempm_1	0.2291	0.088	2.591	0.010	0.055	0.403
mintempm_2	-0.1261	0.090	-1.399	0.162	-0.303	0.051
mintempm_3	-0.0095	0.087	-0.109	0.913	-0.181	0.162
meandewptm_1	0.0777	0.040	1.954	0.051	-0.000	0.156
meandewptm_2	-0.0481	0.046	-1.043	0.297	-0.139	0.042
maxdewptm_1	-0.0469	0.037	-1.253	0.211	-0.120	0.027
maxdewptm_2	0.0019	0.038	0.051	0.959	-0.073	0.077
maxdewptm_3	0.0907	0.027	3.405	0.001	0.038	0.143
mindewptm_1	-0.0056	0.011	-0.494	0.621	-0.028	0.017
mindewptm_2	0.0035	0.011	0.309	0.758	-0.019	0.026
mindewptm_3	-0.0170	0.010	-1.696	0.090	-0.037	0.003
maxtempm_1	0.6232	0.089	7.021	0.000	0.449	0.798
maxtempm_2	-0.2549	0.093	-2.742	0.006	-0.437	-0.072
maxtempm_3	-0.0472	0.090	-0.527	0.598	-0.223	0.129

Omnibus:	90.365	**Durbin-Watson:**	2.042
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	274.440
Skew:	-0.642	**Prob(JB):**	2.55e-60
Kurtosis:	5.845	**Cond. No.**	879.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [15]:

```
model = sm.OLS(y, X).fit()
model.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	meantempm	**R-squared:**	0.945
Model:	OLS	**Adj. R-squared:**	0.944
Method:	Least Squares	**F-statistic:**	664.1
Date:	Fri, 04 Dec 2020	**Prob (F-statistic):**	0.00
Time:	22:38:36	**Log-Likelihood:**	-1213.2
No. Observations:	676	**AIC:**	2462.
Df Residuals:	658	**BIC:**	2544.
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.2578	0.552	-2.278	0.023	-2.342	-0.174

meantempm_1	0.0874	0.163	0.537	0.591	-0.232	0.407
meantempm_2	0.2706	0.163	1.664	0.097	-0.049	0.590
meantempm_3	0.1820	0.163	1.114	0.266	-0.139	0.503
mintempm_1	0.2291	0.088	2.591	0.010	0.055	0.403
mintempm_2	-0.1261	0.090	-1.399	0.162	-0.303	0.051
mintempm_3	-0.0095	0.087	-0.109	0.913	-0.181	0.162
meandewptm_1	0.0777	0.040	1.954	0.051	-0.000	0.156
meandewptm_2	-0.0481	0.046	-1.043	0.297	-0.139	0.042
maxdewptm_1	-0.0469	0.037	-1.253	0.211	-0.120	0.027
maxdewptm_2	0.0019	0.038	0.051	0.959	-0.073	0.077
maxdewptm_3	0.0907	0.027	3.405	0.001	0.038	0.143
mindewptm_1	-0.0056	0.011	-0.494	0.621	-0.028	0.017
mindewptm_2	0.0035	0.011	0.309	0.758	-0.019	0.026
mindewptm_3	-0.0170	0.010	-1.696	0.090	-0.037	0.003
maxtempm_1	0.6232	0.089	7.021	0.000	0.449	0.798
maxtempm_2	-0.2549	0.093	-2.742	0.006	-0.437	-0.072
maxtempm_3	-0.0472	0.090	-0.527	0.598	-0.223	0.129

Omnibus:	90.365	**Durbin-Watson:**	2.042
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	274.440
Skew:	-0.642	**Prob(JB):**	2.55e-60
Kurtosis:	5.845	**Cond. No.**	879.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Using SciKit-Learn's LinearRegression Module to Predict the Weather

We are splitting the data into two parts i.e. 80% of the data is gonna be in the training model and the 20% of the data is going to be in the testing model and will predict the Apparent Temperature.

In [16]:

```
from sklearn.model_selection import train_test_split
```

In [17]:

```
# firstly removing the const column because unlike statsmodels, SciKit-Learn will add tha
t in for us
X = X.drop('const', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12
)
```

Showing the Actual Apparent Temperature and the predicted Apparent Temperature

In [19]:

```
from sklearn.linear_model import LinearRegression
```

In [20]:

```
# instantiating the regressor class
regressor = LinearRegression()

# fit the build the model by fitting the regressor to the training data
regressor.fit(X_train, y_train)

# making a prediction set using the test set
prediction = regressor.predict(X_test)

dt = pd.DataFrame({'Actual':y_test, 'Predicted': prediction})
dt
```

Out[20]:

date	Actual	Predicted
2016-05-15	36	34.597629
2017-06-29	28	28.671393
2016-08-13	28	26.986263
2017-01-14	12	13.160377
2017-02-27	22	22.175811
...	...	...
2017-05-30	31	28.871915
2018-02-13	19	17.985989
2016-07-26	30	30.269302
2017-02-09	17	17.739589
2017-02-05	21	20.416324

136 rows × 2 columns

Calculating the error in prediction

In [21]:

```
# Evaluating the prediction accuracy of the model
from sklearn.metrics import mean_absolute_error, median_absolute_error
print("The Explained Variance: %.2f" % regressor.score(X_test, y_test))
print("The Mean Absolute Error: %.2f degrees celsius" % mean_absolute_error(y_test, predi
ction))
print("The Median Absolute Error: %.2f degrees celsius" % median_absolute_error(y_test, p
rediction))
```

```
The Explained Variance: 0.95
The Mean Absolute Error: 1.11 degrees celsius
The Median Absolute Error: 0.91 degrees celsius
```