Alam_intro_to_stat_analysis_final_proj

December 8, 2020

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```python
weather = pd.read_csv("C:/Users/shams/OneDrive/Desktop/All_project/
↪Intro_to_stat_analysis_proj/weatherHistory.csv")
weather.head()
```

[2]:
```
                  Formatted Date        Summary Precip Type  Temperature (C)  \
0  2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain         9.472222
1  2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain         9.355556
2  2006-04-01 02:00:00.000 +0200  Mostly Cloudy        rain         9.377778
3  2006-04-01 03:00:00.000 +0200  Partly Cloudy        rain         8.288889
4  2006-04-01 04:00:00.000 +0200  Mostly Cloudy        rain         8.755556

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                  7.388889      0.89            14.1197
1                  7.227778      0.86            14.2646
2                  9.377778      0.89             3.9284
3                  5.944444      0.83            14.1036
4                  6.977778      0.83            11.0446

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                   251.0          15.8263         0.0               1015.13
1                   259.0          15.8263         0.0               1015.63
2                   204.0          14.9569         0.0               1015.94
3                   269.0          15.8263         0.0               1016.41
4                   259.0          15.8263         0.0               1016.51

                       Daily Summary
0  Partly cloudy throughout the day.
1  Partly cloudy throughout the day.
2  Partly cloudy throughout the day.
3  Partly cloudy throughout the day.
4  Partly cloudy throughout the day.
```

[ ]:
```
"""The CSV file includes a hourly/daily summary for Szeged, Hungary area,␣
↪between 2006 and 2016.It is a very large data set
with dimension (96453,12).
My analysis was to find is there a relationship between humidity and␣
↪temperature? What about between humidity and apparent
Temperature."""
```

[ ]:
```
"""Data Background"""
"""
4-year air temperature data from June 2014 to May 2018 in  10-min averages were␣
↪used based  on  1-min measurements.
To  compare  temperature  modifying effects of different LCZs in Szeged LCZ␣
↪averages were used. Consequently,
in case of LCZ 2 and 3 it means only one station, however in LCZ 6 the average␣
↪of ten stations  was  used according
to the size  of this LCZ class.
Available from:
https://www.researchgate.net/publication/
327867323_Weather_and_climate_modeling_possibilities_using_local_climate_zone_concept_and_
observation_network_in_Szeged_Hungary [accessed Dec 08 2020].
"""
```

[26]:
```
weather[weather.isnull().any(axis=1)]
```

[26]:
```
                    Formatted Date        Summary Precip Type  \
52672  2012-04-11 16:00:00.000 +0200  Mostly Cloudy         NaN
52674  2012-04-11 18:00:00.000 +0200  Mostly Cloudy         NaN
52675  2012-04-11 19:00:00.000 +0200  Mostly Cloudy         NaN
52677  2012-04-11 21:00:00.000 +0200  Mostly Cloudy         NaN
52678  2012-04-11 22:00:00.000 +0200  Mostly Cloudy         NaN
…                                …              …         …
95584  2016-10-31 19:00:00.000 +0100  Mostly Cloudy         NaN
95585  2016-10-31 20:00:00.000 +0100  Mostly Cloudy         NaN
95586  2016-10-31 21:00:00.000 +0100  Mostly Cloudy         NaN
95587  2016-10-31 22:00:00.000 +0100  Partly Cloudy         NaN
95588  2016-10-31 23:00:00.000 +0100  Mostly Cloudy         NaN

       Temperature (C)  Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
52672        19.016667                 19.016667      0.26            14.8764
52674        17.850000                 17.850000      0.28            13.7977
52675        16.322222                 16.322222      0.32            10.8192
52677        12.566667                 12.566667      0.43             9.0160
52678        12.927778                 12.927778      0.47            17.6295
…                  …                       …       …                  …
95584         8.322222                  7.044444      0.85             8.0339
95585         7.627778                  6.183333      0.87             8.2271
95586         7.111111                  5.511111      0.89             8.5008
```

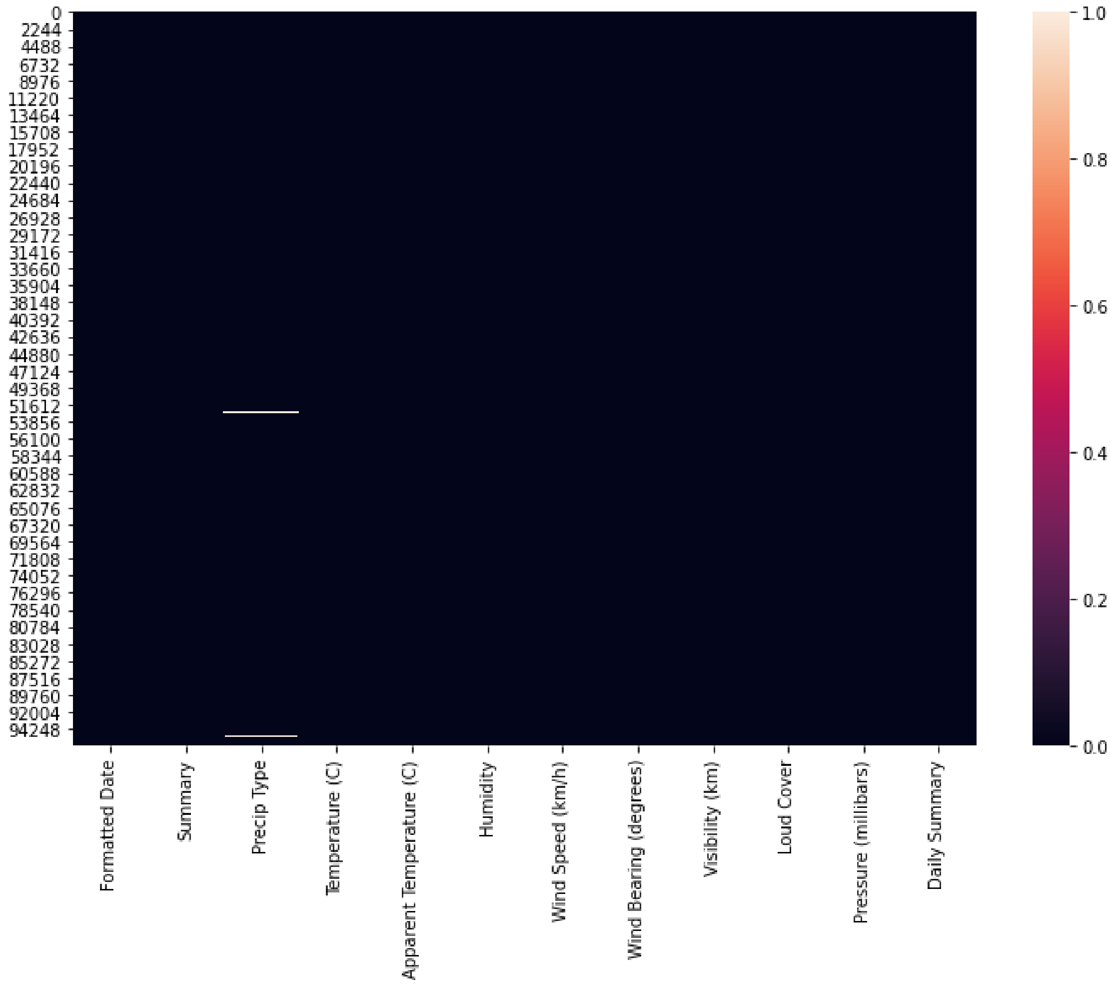
```
95587         6.672222                    4.961111      0.90              8.6457
95588         6.322222                    4.588889      0.91              8.4686

       Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
52672                   163.0            9.982         0.0
52674                   169.0            9.982         0.0
52675                   151.0            9.982         0.0
52677                   159.0            9.982         0.0
52678                   197.0           16.100         0.0
…                       …               …          …
95584                   290.0            0.000         0.0
95585                   293.0            0.000         0.0
95586                   297.0            0.000         0.0
95587                   299.0            0.000         0.0
95588                   299.0            0.000         0.0

       Pressure (millibars)                        Daily Summary
52672               1002.40              Mostly cloudy until night.
52674               1001.79              Mostly cloudy until night.
52675               1001.60              Mostly cloudy until night.
52677               1001.92              Mostly cloudy until night.
52678               1002.20              Mostly cloudy until night.
…                     …                                         …
95584               1021.73  Mostly cloudy starting in the afternoon.
95585               1021.76  Mostly cloudy starting in the afternoon.
95586               1021.81  Mostly cloudy starting in the afternoon.
95587               1021.83  Mostly cloudy starting in the afternoon.
95588               1021.80  Mostly cloudy starting in the afternoon.

[517 rows x 12 columns]
```

[33]:

```
import seaborn as sns
plt.figure(figsize=(12,8))
weather.isna().sum()
sns.heatmap(weather.isna())
```

[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2823da74688>
```



[5]:
```python
"""We can see that the Precip Type variable have some missing values, as the␣
↪dataset is huge we will remove the rows"""
"""Dropping the missing values"""
dat1 = weather.dropna()
dat1.isna().sum()
```

[5]:
```
Formatted Date              0
Summary                     0
Precip Type                 0
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
```

```
Pressure (millibars)        0
Daily Summary               0
dtype: int64
```

[ ]:
```
"""We will perform a time analysis on the dataset, as time and season is an␣
 ↪important componed for weather analysis"""
"""Time Analysis"""
```

[6]:
```
dat1['Formatted Date'] = pd.to_datetime(dat1['Formatted Date'], utc=True)
dat1['Formatted Date']
```

```
C:\Users\shams\AppData\Local\Continuum\anaconda3\lib\site-
packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[6]:
```
0        2006-03-31 22:00:00+00:00
1        2006-03-31 23:00:00+00:00
2        2006-04-01 00:00:00+00:00
3        2006-04-01 01:00:00+00:00
4        2006-04-01 02:00:00+00:00
                   …
96448    2016-09-09 17:00:00+00:00
96449    2016-09-09 18:00:00+00:00
96450    2016-09-09 19:00:00+00:00
96451    2016-09-09 20:00:00+00:00
96452    2016-09-09 21:00:00+00:00
Name: Formatted Date, Length: 95936, dtype: datetime64[ns, UTC]
```

[7]:
```
#Set index as "Date

dat_index = dat1.set_index('Formatted Date')
dat_index.head()
```

[7]:
```
                                 Summary Precip Type  Temperature (C)  \
Formatted Date
2006-03-31 22:00:00+00:00  Partly Cloudy        rain         9.472222
2006-03-31 23:00:00+00:00  Partly Cloudy        rain         9.355556
2006-04-01 00:00:00+00:00  Mostly Cloudy        rain         9.377778
2006-04-01 01:00:00+00:00  Partly Cloudy        rain         8.288889
2006-04-01 02:00:00+00:00  Mostly Cloudy        rain         8.755556

                           Apparent Temperature (C)  Humidity  \
```

```
                           Formatted Date
2006-03-31 22:00:00+00:00                    7.388889      0.89
2006-03-31 23:00:00+00:00                    7.227778      0.86
2006-04-01 00:00:00+00:00                    9.377778      0.89
2006-04-01 01:00:00+00:00                    5.944444      0.83
2006-04-01 02:00:00+00:00                    6.977778      0.83

                           Wind Speed (km/h)  Wind Bearing (degrees)  \
Formatted Date
2006-03-31 22:00:00+00:00            14.1197                   251.0
2006-03-31 23:00:00+00:00            14.2646                   259.0
2006-04-01 00:00:00+00:00             3.9284                   204.0
2006-04-01 01:00:00+00:00            14.1036                   269.0
2006-04-01 02:00:00+00:00            11.0446                   259.0

                           Visibility (km)  Loud Cover  Pressure (millibars)  \
Formatted Date
2006-03-31 22:00:00+00:00          15.8263         0.0               1015.13
2006-03-31 23:00:00+00:00          15.8263         0.0               1015.63
2006-04-01 00:00:00+00:00          14.9569         0.0               1015.94
2006-04-01 01:00:00+00:00          15.8263         0.0               1016.41
2006-04-01 02:00:00+00:00          15.8263         0.0               1016.51

                                                  Daily Summary
Formatted Date
2006-03-31 22:00:00+00:00  Partly cloudy throughout the day.
2006-03-31 23:00:00+00:00  Partly cloudy throughout the day.
2006-04-01 00:00:00+00:00  Partly cloudy throughout the day.
2006-04-01 01:00:00+00:00  Partly cloudy throughout the day.
2006-04-01 02:00:00+00:00  Partly cloudy throughout the day.
```

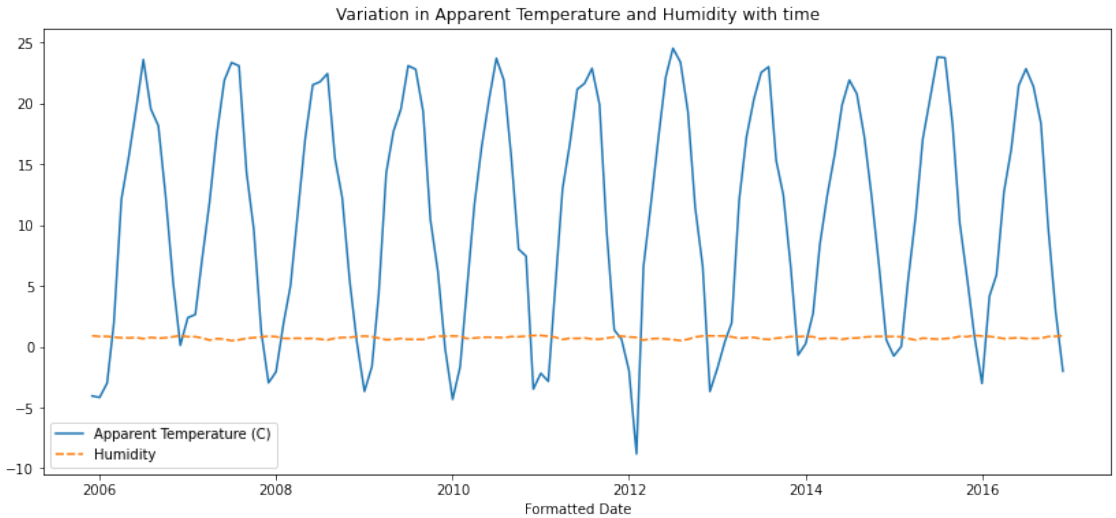
[8]:
```
data_columns = ['Apparent Temperature (C)', 'Humidity']
df_monthly_mean = dat_index[data_columns].resample('MS').mean()
df_monthly_mean.head()
```

[8]:
```
                           Apparent Temperature (C)  Humidity
Formatted Date
2005-12-01 00:00:00+00:00                 -4.050000  0.890000
2006-01-01 00:00:00+00:00                 -4.173708  0.834610
2006-02-01 00:00:00+00:00                 -2.990716  0.843467
2006-03-01 00:00:00+00:00                  1.969780  0.778737
2006-04-01 00:00:00+00:00                 12.098827  0.728625
```

[9]:
```
#Plotting Variation in Apparent Temperature and Humidity with time

import seaborn as sns
import warnings
```

```
warnings.filterwarnings("ignore")

plt.figure(figsize=(14,6))
plt.title("Variation in Apparent Temperature and Humidity with time")
sns.lineplot(data=df_monthly_mean)
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x28238300f88>



[10]:
```
"""We can see a sinusoidal shaped variation of the apparent temperature with␣
↪time which denotes the cycle of season.
From 2006-2016 the Humidity has been pretty constant with no big variation. """

#retrieving the data of a particular month from every year, say April

df1 = df_monthly_mean[df_monthly_mean.index.month==4]
print(df1)

df1.dtypes
```

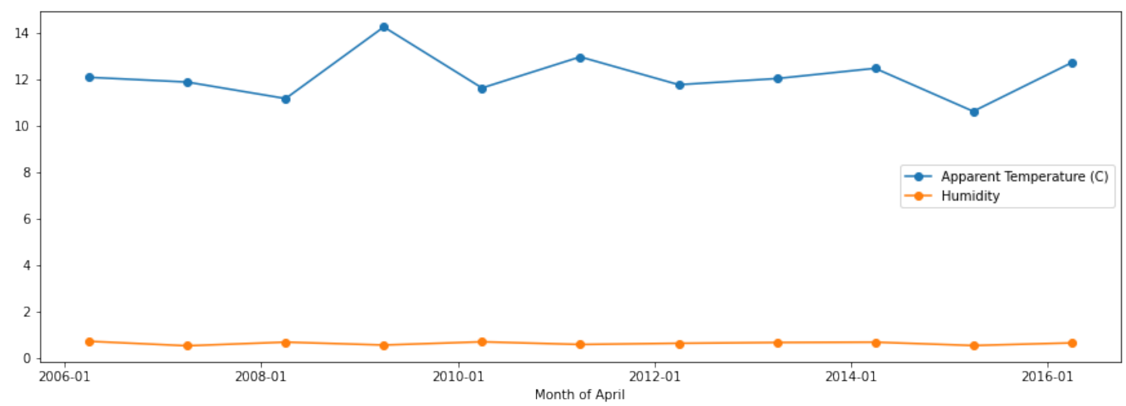
```
                           Apparent Temperature (C)  Humidity
Formatted Date
2006-04-01 00:00:00+00:00                 12.098827  0.728625
2007-04-01 00:00:00+00:00                 11.894421  0.536361
2008-04-01 00:00:00+00:00                 11.183688  0.693194
2009-04-01 00:00:00+00:00                 14.267076  0.567847
2010-04-01 00:00:00+00:00                 11.639406  0.706875
2011-04-01 00:00:00+00:00                 12.978997  0.591625
2012-04-01 00:00:00+00:00                 11.780703  0.643583
2013-04-01 00:00:00+00:00                 12.045563  0.677667
2014-04-01 00:00:00+00:00                 12.486181  0.691403
```

```
2015-04-01 00:00:00+00:00                    10.632801   0.547764
2016-04-01 00:00:00+00:00                    12.731427   0.659972
```

[10]:
```
Apparent Temperature (C)    float64
Humidity                    float64
dtype: object
```

[11]:
```python
#Plotting each years Humidity and Temperature change

import matplotlib.dates as mdates
from datetime import datetime

fig, ax = plt.subplots(figsize=(15,5))
ax.plot(df1.loc['2006-04-01':'2016-04-01', 'Apparent Temperature (C)'],␣
↪marker='o', linestyle='-',label='Apparent Temperature (C)')
ax.plot(df1.loc['2006-04-01':'2016-04-01', 'Humidity'], marker='o',␣
↪linestyle='-',label='Humidity')
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax.legend(loc = 'center right')
ax.set_xlabel('Month of April')
```

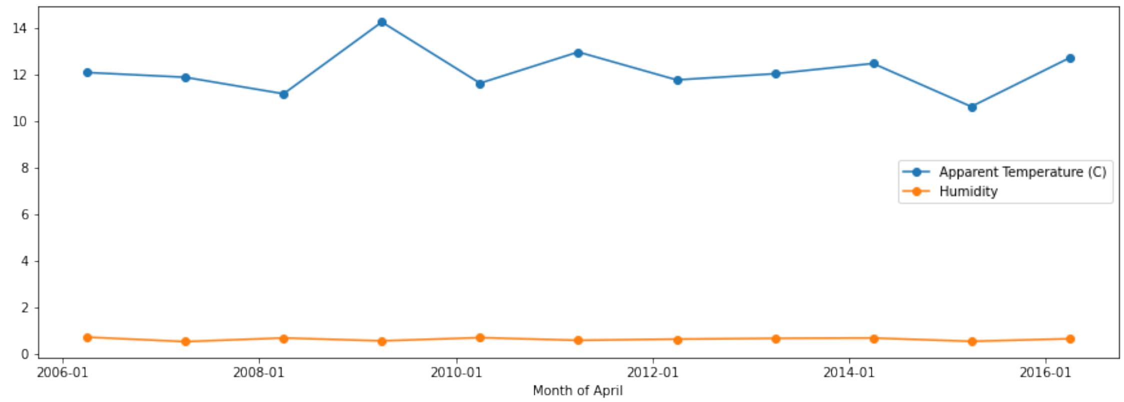
[11]:
```
Text(0.5, 0, 'Month of April')
```



[12]:
```python
"""We performed more closer analysis on the month april. We can see that the␣
↪apparent temperature does not fluctuate much
and changes a little in the range 10-15(C). So from the plot there the␣
↪temperature is moderate without any dominant
changes and the humidity remains the same like the rest of the year"""
#Plotting each years Humidity and Temperature change

import matplotlib.dates as mdates
from datetime import datetime
```

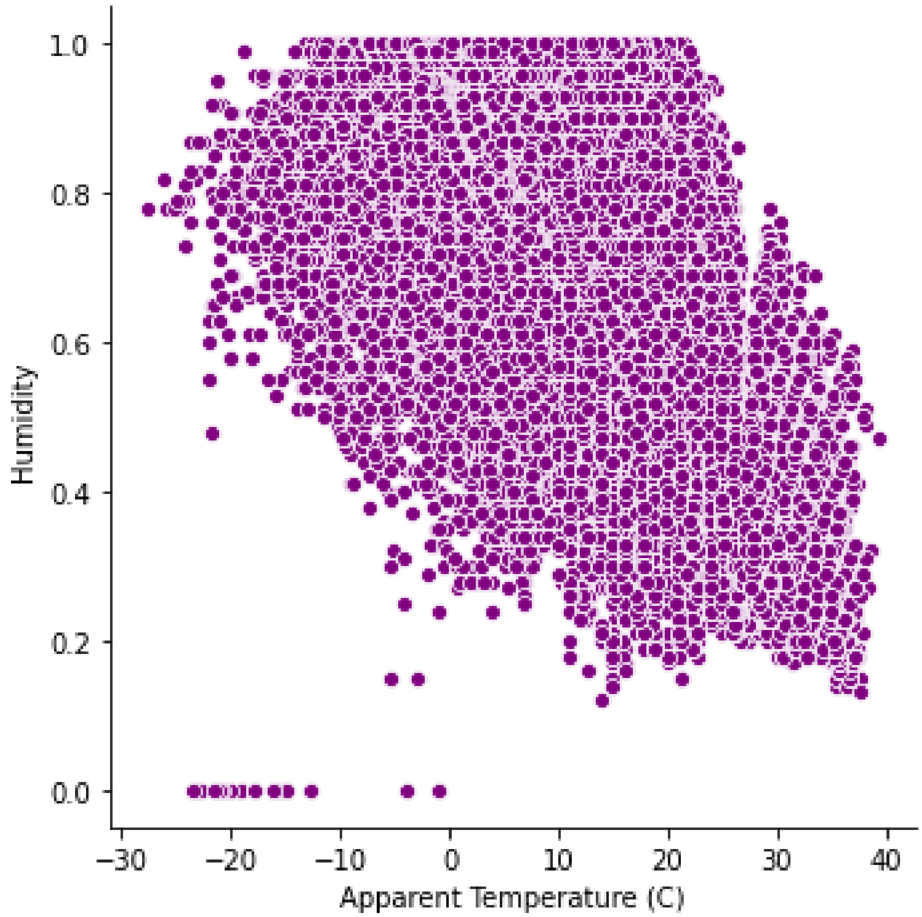
```
fig, ax = plt.subplots(figsize=(15,5))
ax.plot(df1.loc['2006-04-01':'2016-04-01', 'Apparent Temperature (C)'],␣
↪marker='o', linestyle='-',label='Apparent Temperature (C)')
ax.plot(df1.loc['2006-04-01':'2016-04-01', 'Humidity'], marker='o',␣
↪linestyle='-',label='Humidity')
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax.legend(loc = 'center right')
ax.set_xlabel('Month of April')
```

[12]: Text(0.5, 0, 'Month of April')



[13]:
```
sns.relplot(data = dat1, x = "Apparent Temperature (C)", y = "Humidity", color␣
↪= 'purple')
```

[13]: <seaborn.axisgrid.FacetGrid at 0x28239bad648>



[14]:
```
"""From the rel plot we can see the intensity of humidity with Apparent␣
↪temperature. We can see some instances of zero
level humidity at very cold temperatures"""

dat2 = dat1.drop(['Formatted Date','Loud Cover', 'Daily Summary'], axis =1)
dat2.head()
```

[14]:
```
         Summary Precip Type  Temperature (C)  Apparent Temperature (C)  \
0  Partly Cloudy        rain         9.472222                  7.388889
1  Partly Cloudy        rain         9.355556                  7.227778
2  Mostly Cloudy        rain         9.377778                  9.377778
3  Partly Cloudy        rain         8.288889                  5.944444
4  Mostly Cloudy        rain         8.755556                  6.977778

   Humidity  Wind Speed (km/h)  Wind Bearing (degrees)  Visibility (km)  \
0      0.89            14.1197                   251.0          15.8263
1      0.86            14.2646                   259.0          15.8263
2      0.89             3.9284                   204.0          14.9569
```

```
3      0.83            14.1036                     269.0           15.8263
4      0.83            11.0446                     259.0           15.8263

   Pressure (millibars)
0               1015.13
1               1015.63
2               1015.94
3               1016.41
4               1016.51
```

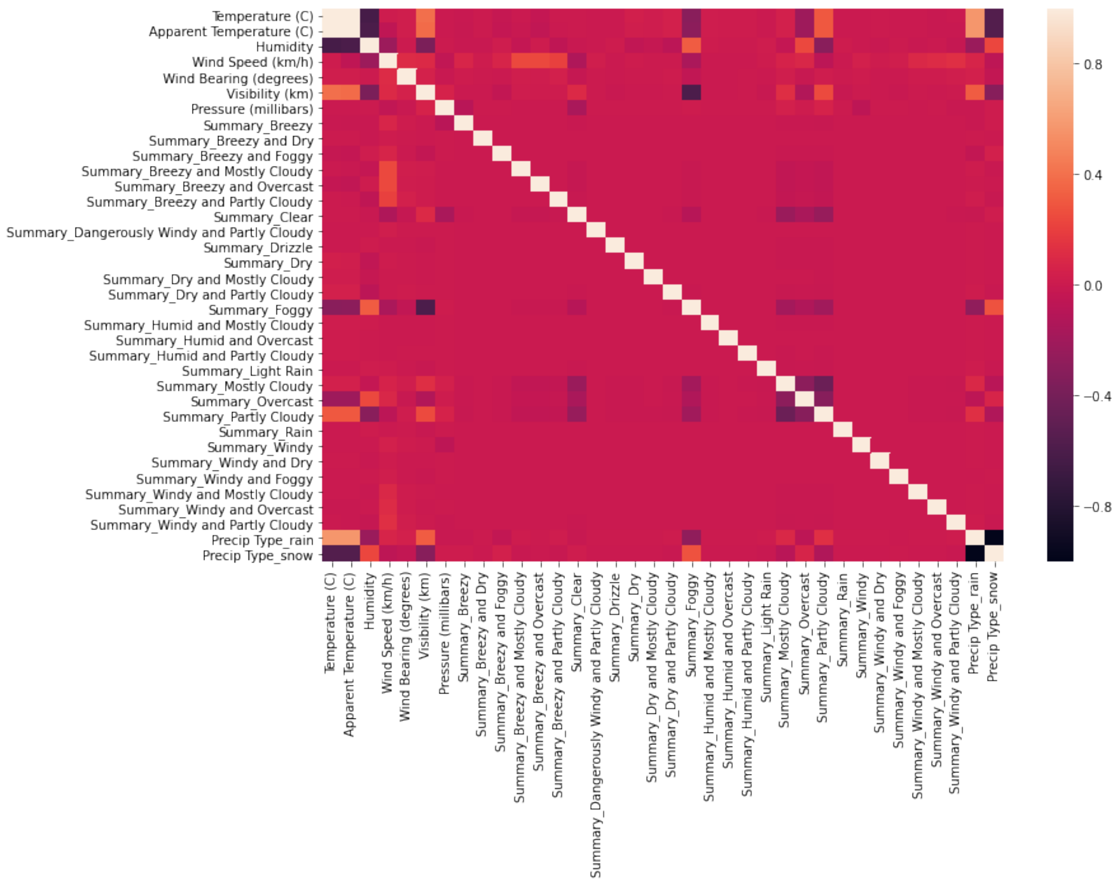
[15]:
```python
#in order to convert the categorical to dummy variables.

dat2_dummy = pd.get_dummies(dat2)
dat2_dummy.head()
```

[15]:
```
   Temperature (C)  Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0         9.472222                  7.388889      0.89            14.1197
1         9.355556                  7.227778      0.86            14.2646
2         9.377778                  9.377778      0.89             3.9284
3         8.288889                  5.944444      0.83            14.1036
4         8.755556                  6.977778      0.83            11.0446

   Wind Bearing (degrees)  Visibility (km)  Pressure (millibars)  \
0                   251.0          15.8263               1015.13
1                   259.0          15.8263               1015.63
2                   204.0          14.9569               1015.94
3                   269.0          15.8263               1016.41
4                   259.0          15.8263               1016.51

   Summary_Breezy  Summary_Breezy and Dry  Summary_Breezy and Foggy  …  \
0               0                       0                         0  …
1               0                       0                         0  …
2               0                       0                         0  …
3               0                       0                         0  …
4               0                       0                         0  …

   Summary_Partly Cloudy  Summary_Rain  Summary_Windy  Summary_Windy and Dry  \
0                      1             0              0                      0
1                      1             0              0                      0
2                      0             0              0                      0
3                      1             0              0                      0
4                      0             0              0                      0

   Summary_Windy and Foggy  Summary_Windy and Mostly Cloudy  \
0                        0                                0
1                        0                                0
2                        0                                0
```

```
3                        0                                 0
4                        0                                 0

   Summary_Windy and Overcast  Summary_Windy and Partly Cloudy  \
0                           0                                0
1                           0                                0
2                           0                                0
3                           0                                0
4                           0                                0

   Precip Type_rain  Precip Type_snow
0                 1                 0
1                 1                 0
2                 1                 0
3                 1                 0
4                 1                 0

[5 rows x 36 columns]
```

[61]:
```python
"""Lets plot the correlation plot to determine the correlation between the␣
 ↪variables"""
plt.figure(figsize=(12,8))
sns.heatmap(dat2_dummy.corr())
```

[61]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2820e706208>
```



[ ]:
```
"""From the heatmap of the correlation plot we can see that Apparent␣
↪Temperature is very positive correlated with the
Apparent Temperature denoting that they must have linear correlation between␣
↪them, whereasHumidity is
very negatively correlated with the Apparent Temperature. Also we can see that␣
↪visibility is negatively correlated
with Humidity.
```

[37]:
```
#Now for the model

Y=dat2_dummy[['Apparent Temperature (C)']]
X=dat2_dummy.drop('Apparent Temperature (C)',axis=1)
```

[38]:
```
Y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 95936 entries, 0 to 96452
Data columns (total 1 columns):
Apparent Temperature (C)    95936 non-null float64
dtypes: float64(1)
```

```
memory usage: 1.5 MB
```

[ ]:
```
sns.pairplot(dat2_dummy)
```

[39]:
```
#Significance test
import statsmodels.api as sm
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[40]:
```
# define our intput
X1 = sm.add_constant(X)
# create a OLS model
model = sm.OLS(Y, X1)
# fit the data
est = model.fit()
print(est.summary())
```

```
                              OLS Regression Results
================================================================================
====
Dep. Variable:     Apparent Temperature (C)   R-squared:
0.990
Model:                                  OLS   Adj. R-squared:
0.990
Method:                       Least Squares   F-statistic:
2.928e+05
Date:                      Tue, 08 Dec 2020   Prob (F-statistic):
0.00
Time:                              09:58:31   Log-Likelihood:
-1.4195e+05
No. Observations:                     95936   AIC:
2.840e+05
Df Residuals:                         95902   BIC:
2.843e+05
Df Model:                                33
Covariance Type:                  nonrobust
================================================================================
===============================
                                                  coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
-------------------------------
const                                          -1.2965      0.058    -22.345
0.000      -1.410      -1.183
Temperature (C)                                 1.1161      0.001   1933.943
0.000       1.115       1.117
Humidity                                        0.7866      0.026     30.812
0.000       0.737       0.837
```

Wind Speed (km/h)	-0.1045	0.001	-172.107	0.000	-0.106	-0.103
Wind Bearing (degrees)	0.0005	3.23e-05	15.890	0.000	0.000	0.001
Visibility (km)	0.0042	0.001	3.848	0.000	0.002	0.006
Pressure (millibars)	0.0001	3.01e-05	4.961	0.000	9.02e-05	0.000
Summary_Breezy	-1.0207	0.159	-6.403	0.000	-1.333	-0.708
Summary_Breezy and Dry	2.2819	1.026	2.223	0.026	0.270	4.294
Summary_Breezy and Foggy	-2.2499	0.190	-11.859	0.000	-2.622	-1.878
Summary_Breezy and Mostly Cloudy	0.7164	0.089	8.041	0.000	0.542	0.891
Summary_Breezy and Overcast	-0.2753	0.089	-3.097	0.002	-0.450	-0.101
Summary_Breezy and Partly Cloudy	0.5212	0.093	5.613	0.000	0.339	0.703
Summary_Clear	-0.4970	0.078	-6.383	0.000	-0.650	-0.344
Summary_Dangerously Windy and Partly Cloudy	1.1514	1.027	1.121	0.262	-0.861	3.164
Summary_Drizzle	-0.2496	0.181	-1.378	0.168	-0.605	0.105
Summary_Dry	-1.4202	0.192	-7.402	0.000	-1.796	-1.044
Summary_Dry and Mostly Cloudy	-1.3546	0.284	-4.764	0.000	-1.912	-0.797
Summary_Dry and Partly Cloudy	-1.2062	0.135	-8.941	0.000	-1.471	-0.942
Summary_Foggy	-0.1764	0.079	-2.244	0.025	-0.330	-0.022
Summary_Humid and Mostly Cloudy	-0.6094	0.179	-3.397	0.001	-0.961	-0.258
Summary_Humid and Overcast	-0.7111	0.395	-1.802	0.071	-1.484	0.062
Summary_Humid and Partly Cloudy	-0.7095	0.260	-2.729	0.006	-1.219	-0.200
Summary_Light Rain	-0.3391	0.150	-2.258	0.024	-0.633	-0.045
Summary_Mostly Cloudy	-0.3960	0.077	-5.126	0.000	-0.547	-0.245
Summary_Overcast	-0.4943	0.077	-6.387	0.000	-0.646	-0.343
Summary_Partly Cloudy	-0.4978	0.077	-6.437	0.000	-0.649	-0.346

```
Summary_Rain                                     -0.3451      0.333     -1.037
0.300      -0.998       0.307
Summary_Windy                                    -0.1372      0.371     -0.370
0.711      -0.864       0.590
Summary_Windy and Dry                             1.4297      1.027      1.393
0.164      -0.582       3.442
Summary_Windy and Foggy                           2.0167      0.518      3.895
0.000       1.002       3.032
Summary_Windy and Mostly Cloudy                   1.9733      0.190     10.410
0.000       1.602       2.345
Summary_Windy and Overcast                        0.2519      0.171      1.472
0.141      -0.084       0.587
Summary_Windy and Partly Cloudy                   1.0507      0.147      7.140
0.000       0.762       1.339
Precip Type_rain                                 -0.4263      0.030    -14.136
0.000      -0.485      -0.367
Precip Type_snow                                 -0.8701      0.029    -29.515
0.000      -0.928      -0.812
==============================================================================
Omnibus:                     2475.274   Durbin-Watson:                   0.453
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2713.959
Skew:                           0.384   Prob(JB):                         0.00
Kurtosis:                       3.296   Cond. No.                     2.49e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 1.64e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[41]:
```python
import statsmodels.api as sm
cols = list(X.columns)
pmax = 1
while (len(cols)>0):
    p= []
    X_1 = X[cols]
    X_1 = sm.add_constant(X_1)
    model = sm.OLS(Y,X_1).fit()
    p = pd.Series(model.pvalues.values[1:],index = cols)
    pmax = max(p)
    feature_with_p_max = p.idxmax()
    if(pmax>0.05):
            cols.remove(feature_with_p_max)
    else:
        break
selected_features_BE = cols
```

```
print(selected_features_BE)
```

```
['Temperature (C)', 'Humidity', 'Wind Speed (km/h)', 'Wind Bearing (degrees)',
'Visibility (km)', 'Pressure (millibars)', 'Summary_Breezy', 'Summary_Breezy and
Dry', 'Summary_Breezy and Foggy', 'Summary_Breezy and Mostly Cloudy',
'Summary_Breezy and Partly Cloudy', 'Summary_Clear', 'Summary_Dry', 'Summary_Dry
and Mostly Cloudy', 'Summary_Dry and Partly Cloudy', 'Summary_Humid and Mostly
Cloudy', 'Summary_Humid and Partly Cloudy', 'Summary_Mostly Cloudy',
'Summary_Overcast', 'Summary_Partly Cloudy', 'Summary_Windy and Foggy',
'Summary_Windy and Mostly Cloudy', 'Summary_Windy and Overcast', 'Summary_Windy
and Partly Cloudy', 'Precip Type_rain', 'Precip Type_snow']
```

[42]:

```
"""As we can see that the Summary_Windy contains the maximum p-value and is␣
 ↪greater then the significance factor 0.05,
this variable is insignificant. So we need to remove summary_windy count and␣
 ↪keep checking the p-values by removing
insignificant variables until p-value is closer to zero"""
X1_opt = X1.drop('Summary_Windy', axis=1)
X1_opt=X1_opt.drop('Summary_Foggy',axis=1)
X1_opt=X1_opt.drop('Summary_Rain',axis=1)
X1_opt=X1_opt.drop('Summary_Drizzle',axis=1)
X1_opt=X1_opt.drop('Summary_Dangerously Windy and Partly Cloudy',axis=1)
X1_opt=X1_opt.drop('Summary_Light Rain',axis=1)
X1_opt=X1_opt.drop('Summary_Windy and Dry',axis=1)
X1_opt=X1_opt.drop('Summary_Humid and Overcast',axis=1)


# create a OLS model
model = sm.OLS(Y, X1_opt)
# fit the data
est = model.fit()
print(est.summary())
```

```
                            OLS Regression Results
================================================================================
====
Dep. Variable:     Apparent Temperature (C)   R-squared:
0.990
Model:                                  OLS   Adj. R-squared:
0.990
Method:                       Least Squares   F-statistic:
3.716e+05
Date:                      Tue, 08 Dec 2020   Prob (F-statistic):
0.00
Time:                              09:58:43   Log-Likelihood:
-1.4195e+05
No. Observations:                     95936   AIC:
```

```
2.840e+05
Df Residuals:                   95909   BIC:
2.842e+05
Df Model:                          26
Covariance Type:            nonrobust
================================================================================
====================
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
--------------------
const                          -1.4147      0.028    -50.154      0.000
-1.470      -1.359
Temperature (C)                 1.1160      0.001   1934.439      0.000
1.115       1.117
Humidity                        0.7856      0.026     30.778      0.000
0.736       0.836
Wind Speed (km/h)              -0.1044      0.001   -172.424      0.000
-0.106      -0.103
Wind Bearing (degrees)          0.0005   3.23e-05     15.897      0.000
0.000       0.001
Visibility (km)                 0.0042      0.001      3.800      0.000
0.002       0.006
Pressure (millibars)            0.0001      3e-05      4.968      0.000
9.02e-05       0.000
Summary_Breezy                 -0.8422      0.147     -5.743      0.000
-1.130      -0.555
Summary_Breezy and Dry          2.4605      1.063      2.315      0.021
0.377       4.544
Summary_Breezy and Foggy       -2.0718      0.181    -11.457      0.000
-2.426      -1.717
Summary_Breezy and Mostly Cloudy     0.8952      0.052     17.344      0.000
0.794       0.996
Summary_Breezy and Overcast    -0.0964      0.051     -1.891      0.059
-0.196       0.003
Summary_Breezy and Partly Cloudy     0.7000      0.058     11.984      0.000
0.585       0.814
Summary_Clear                  -0.3178      0.020    -16.164      0.000
-0.356      -0.279
Summary_Dry                    -1.2413      0.183     -6.768      0.000
-1.601      -0.882
Summary_Dry and Mostly Cloudy  -1.1758      0.285     -4.129      0.000
-1.734      -0.618
Summary_Dry and Partly Cloudy  -1.0273      0.116     -8.825      0.000
-1.255      -0.799
Summary_Humid and Mostly Cloudy   -0.4299      0.169     -2.546      0.011
-0.761      -0.099
Summary_Humid and Partly Cloudy   -0.5300      0.258     -2.052      0.040
```

```
-1.036      -0.024
Summary_Mostly Cloudy                -0.2169      0.018    -12.302      0.000
-0.251      -0.182
Summary_Overcast                     -0.3153      0.017    -18.289      0.000
-0.349      -0.281
Summary_Partly Cloudy                -0.3186      0.018    -17.534      0.000
-0.354      -0.283
Summary_Windy and Foggy               2.1951      0.532      4.126      0.000
1.152       3.238
Summary_Windy and Mostly Cloudy       2.1520      0.181     11.859      0.000
1.796       2.508
Summary_Windy and Overcast            0.4305      0.160      2.683      0.007
0.116       0.745
Summary_Windy and Partly Cloudy       1.2293      0.133      9.274      0.000
0.969       1.489
Precip Type_rain                     -0.4856      0.016    -29.517      0.000
-0.518      -0.453
Precip Type_snow                     -0.9291      0.015    -62.269      0.000
-0.958      -0.900
==============================================================================
Omnibus:                     2473.706   Durbin-Watson:                   0.453
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2711.911
Skew:                           0.384   Prob(JB):                         0.00
Kurtosis:                       3.295   Cond. No.                     9.33e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 1.16e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[43]:

```
"""Removing the constant term"""
X1_opt2 = X1_opt.drop('const', axis=1)
```

[44]:

```
"""Linear Regression model"""
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

X_train, X_test, Y_train, Y_test = train_test_split(X1_opt2, Y, test_size = 0.
 ↪5, random_state = 1)
```

[45]:

```
print(X_train.shape)
print(Y_train.shape)
```

```
(47968, 27)
(47968, 1)
```

[46]:
```
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

[46]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[47]:
```
Y_pred = lm.predict(X_test)
```

[48]:
```
Y_pred_dataframe = pd.DataFrame(Y_pred, columns = ['Predicted_App Temp'])
Y_pred_dataframe.head()
```

[48]:

	Predicted_App Temp
0	6.969721
1	5.258607
2	15.935838
3	23.027109
4	2.267791

[49]:
```
Y_test
```

[49]:

	Apparent Temperature (C)
37747	6.405556
50630	6.161111
91632	15.444444
30623	22.394444
71677	1.838889
…	…
15415	-1.816667
38415	-8.488889
28768	-5.794444
45247	19.933333
27644	18.088889

[47968 rows x 1 columns]

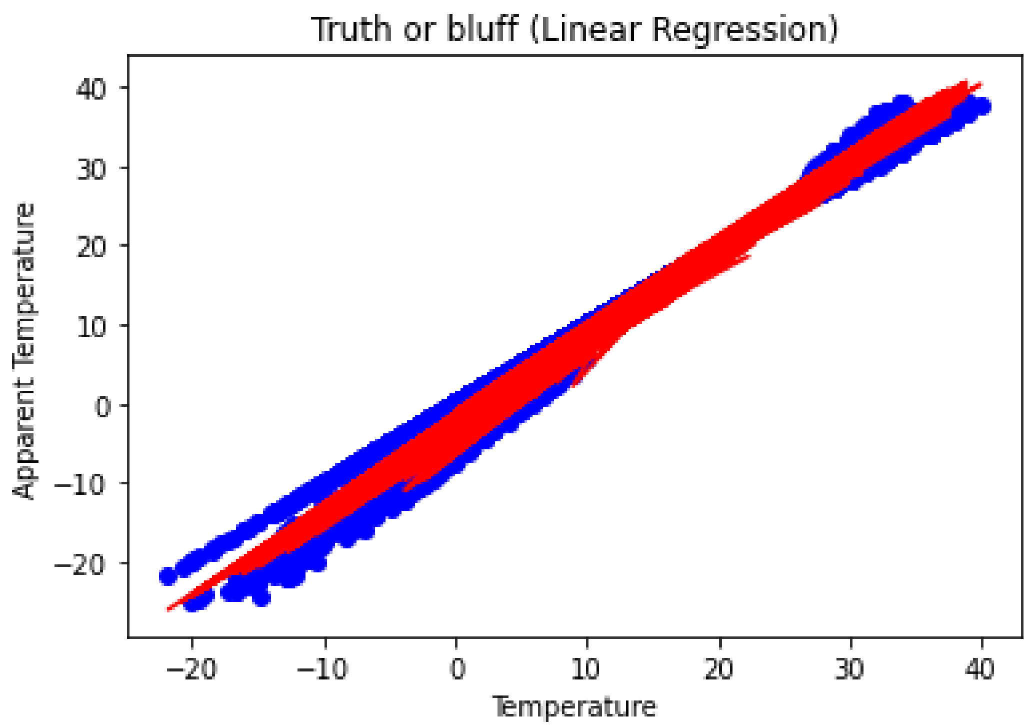
[50]:
```
#lets grab the coefficients and intercept
Y_coeff = lm.coef_
Y_intercept = lm.intercept_
print(Y_coeff)
```

```
[[ 1.11573740e+00  7.35990645e-01 -1.04344707e-01  4.99374599e-04
   4.31574116e-03  1.69615367e-04 -1.11342986e+00  2.43973205e+00
  -1.83824067e+00  9.43096359e-01 -5.62890487e-02  6.78884920e-01
  -3.00063866e-01 -1.41017203e+00 -1.07339950e+00 -1.15543282e+00
  -5.63812735e-01 -4.80990781e-01 -2.12534186e-01 -2.99775469e-01
  -3.15938611e-01  1.81432460e+00  2.26193720e+00  6.65760108e-01
   1.02846618e+00  2.16353837e-01 -2.16353837e-01]]
```

[51]:
```
from sklearn.metrics import mean_squared_error,r2_score
print(Y_intercept)
r2 = r2_score(Y_test,Y_pred)
print(lm.score(X_test,Y_test))
print("R^2 score: ",r2)
print('RMSE:', np.sqrt(mean_squared_error(Y_test, lm.predict(X_test))))
```

```
[-2.09910272]
0.9901798864537803
R^2 score:  0.9901798864537803
RMSE: 1.0614980989251768
```

[52]:
```
#DataFrame.as_matrix(columns=None)
Y_test.head()
plt.scatter(X_test['Temperature (C)'].values,Y_test.values, color = "blue" )
plt.plot(X_test['Temperature (C)'].values,Y_pred, color = "red")
plt.title("Truth or bluff (Linear Regression)")
plt.xlabel('Temperature')
plt.ylabel('Apparent Temperature')
plt.show()
```



[53]:
```
Y_pred
```

[53]:
```
array([[ 6.96972075],
       [ 5.25860737],
       [15.93583807],
       …,
       [-4.49590698],
       [19.67576139],
       [17.97798901]])
```

[54]:
```
"""First we will create a matrix of different containing nonlinear features"""
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 2)
"""x1 as 1st feature, x^2 as second feature"""
X_poly = poly_reg.fit_transform(X_train)#b0+b1x1+b2x2^2
lm_poly = LinearRegression()
lm_poly.fit(X_poly,Y_train)
```

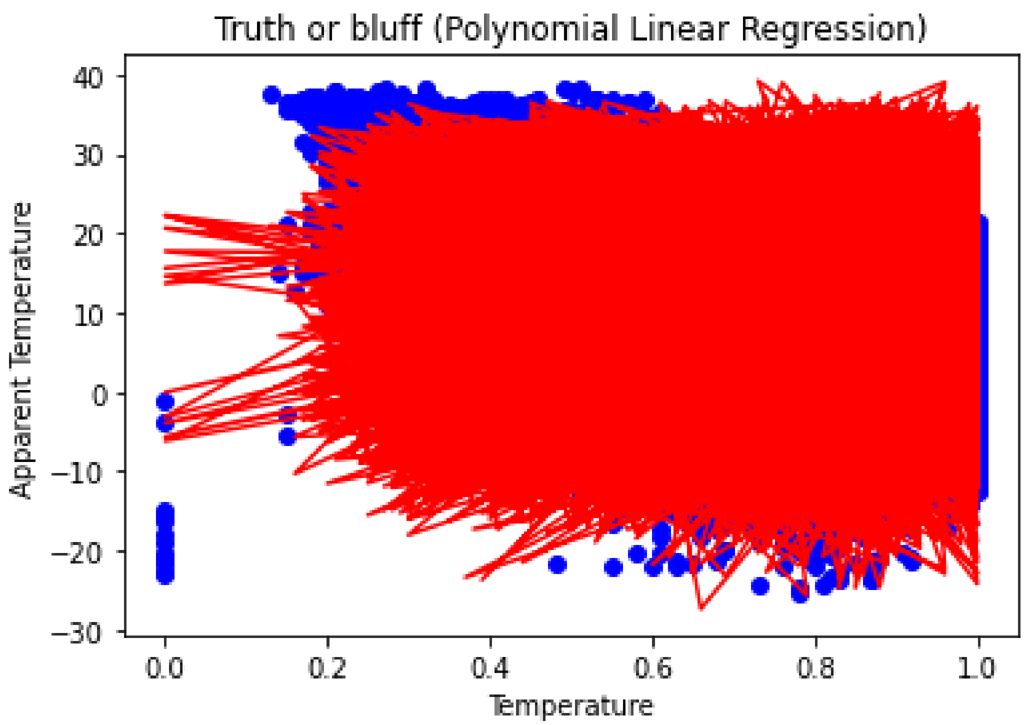
[54]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[55]:
```
X_train['Humidity']
```

[55]:
```
71989    0.65
39793    0.93
12157    0.67
82488    0.72
26755    0.33
         …
21440    0.67
73492    0.92
50057    0.65
5192     0.89
77851    0.96
Name: Humidity, Length: 47968, dtype: float64
```

[56]:
```
Y_poly_pred = lm_poly.predict(X_poly)
```

[57]:
```
plt.scatter(X_test['Humidity'],Y_test, color = "blue" )
plt.plot(X_test['Humidity'],Y_poly_pred, color = "red")
plt.title("Truth or bluff (Polynomial Linear Regression)")
plt.xlabel('Temperature')
plt.ylabel('Apparent Temperature')
plt.show()
```



[69]:
```
from sklearn.metrics import mean_squared_error,r2_score
r2 = -r2_score(Y_test,Y_poly_pred)
#print(lm_poly.score(X_test,Y_test))
print('R^2 score:',r2)
print('RMSE:', np.sqrt(mean_squared_error(Y_test, Y_poly_pred)))
```

```
R^2 score: 0.9911999745738618
RMSE: 15.11535686350149
```

[ ]:
```
"""We can see that the polynomial regression performs a little bit better as␣
 ↪the R^2 score is 0.99017 and 0.9911 for
multiple regression and polynomial regression respectively"""
```

[ ]:
```
"""Now we will apply ridge regression and lasso model to the dataset"""
```

[70]:
```
from sklearn.linear_model import Ridge
from sklearn import model_selection
from sklearn.linear_model import RidgeCV
```

[71]:
```
lm_ridge = Ridge()
lm_ridge.fit(X_train, Y_train)
```

[71]: Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
       normalize=False, random_state=None, solver='auto', tol=0.001)

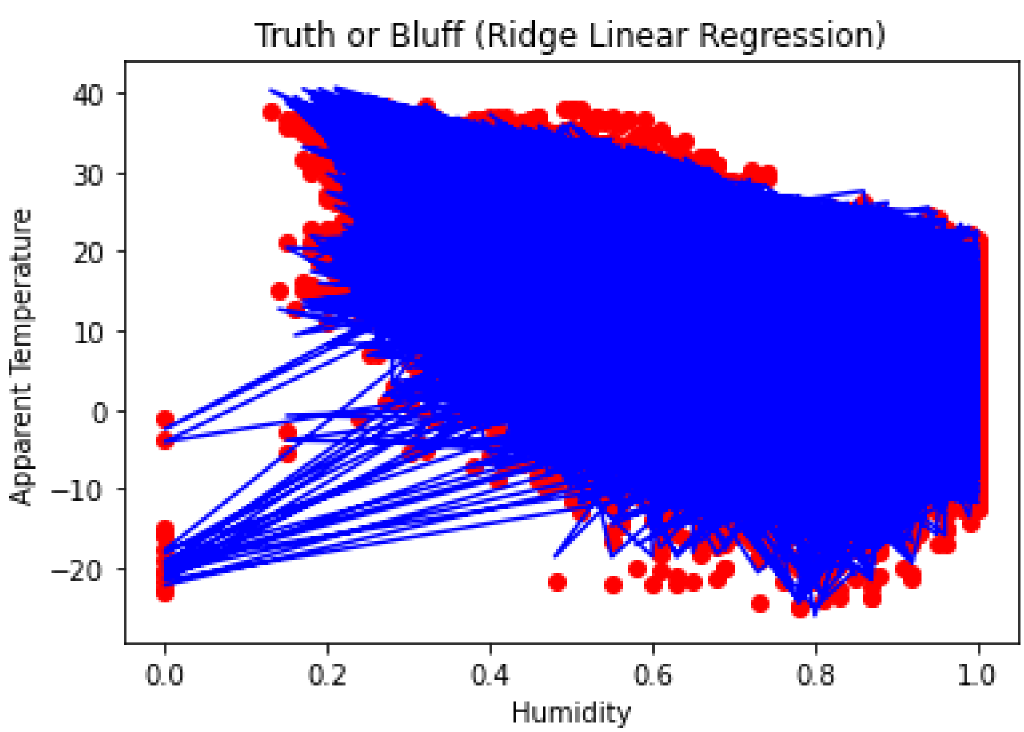
[72]:
```
lm_ridge.coef_
```

[72]:
```
array([[ 1.11572046e+00,  7.36007021e-01, -1.04253523e-01,
         4.99560931e-04,  4.31536851e-03,  1.70605935e-04,
        -1.07803452e+00,  1.21881135e+00, -1.73216556e+00,
         9.37343303e-01, -5.82897501e-02,  6.73203411e-01,
        -2.99812778e-01, -1.34325586e+00, -9.20302546e-01,
        -1.13089373e+00, -5.33961094e-01, -4.43956363e-01,
        -2.12692300e-01, -3.00016258e-01, -3.15922070e-01,
         1.35833272e+00,  2.07064198e+00,  6.37045639e-01,
         9.89884640e-01,  2.16528251e-01, -2.16528251e-01]])
```

[73]:
```
lm_ridge.intercept_
```

[73]: array([-2.10095029])

[74]:
```
plt.scatter(X_test['Humidity'], Y_test, color = 'red')
plt.plot(X_test['Humidity'], lm_ridge.predict(X_test), color = 'blue')
plt.title('Truth or Bluff (Ridge Linear Regression)')
plt.xlabel('Humidity')
plt.ylabel('Apparent Temperature')
plt.show()
```



[77]:
```
"""We can see that most of the predicted values well converge with the test␣
↪values"""
"""Lets compute the R^2 score for validity"""

MSE_Ridge = metrics.mean_squared_error(Y_test, lm_ridge.predict(X_test))
print('MSE:', MSE_Ridge)
print('RMSE:', np.sqrt(metrics.mean_squared_error(Y_test, lm_ridge.
predict(X_test))))
r2 = r2_score(Y_test,lm_ridge.predict(X_test))
#print(lm_poly.score(X_test,Y_test))
print('R^2 score:',r2)
```

```
MSE: 1.126815266682861
RMSE: 1.061515551785682
R^2 score: 0.9901795635319004
```

[80]:
```
"""We can see that R^2 score is 0.99018 which is denotes that ridge model␣
↪performs very well"""
"""We'll use cross validation to determine the optimal alpha value. By default,␣
↪the ridge regression
cross validation class uses the Leave One Out strategy (k-fold).
We can compare the performance of our model with different alpha
values by taking a look at the mean square error."""
from sklearn.linear_model import RidgeCV
alphas = 10**np.linspace(10,-2,100)*0.5
#alphas=[0, 1, 1000, 1000000]
lm_ridge_opt = RidgeCV(alphas=alphas,scoring = "neg_mean_squared_error", cv␣
↪=10, normalize = True)
lm_ridge_opt.fit(X_train, Y_train)
print("optimum shrinkage parameter using Ridge CV",lm_ridge_opt.alpha_)
print("MSE for Ridge CV",metrics.mean_squared_error(Y_test, lm_ridge_opt.
↪predict(X_test)))
print("RMSE for RidgeCV",np.sqrt(metrics.mean_squared_error(Y_test,␣
↪lm_ridge_opt.predict(X_test))))
```

```
optimum shrinkage parameter using Ridge CV 0.005
MSE for Ridge CV 1.1345524643911613
RMSE for RidgeCV 1.0651537280557963
```

[81]:
```
r2 = r2_score(Y_test,lm_ridge_opt.predict(X_test))
#print(lm_poly.score(X_test,Y_test))
print('R^2 score:',r2)
```

```
R^2 score: 0.9901121321961862
```

[82]:
```
"""So we can see the shrinkage parameter using ridge CV is about 0.05, with␣
↪this optimal value we get good the RMSE value. Hence we choose alpha = 0.005␣
↪the
optimal shrinkage parameter"""

"""Now we perform the Lasso"""
from sklearn.linear_model import Lasso, LassoCV
lassocv = LassoCV(alphas = alphas, cv = 10, max_iter = 100000, normalize = True)
lassocv.fit(X_train, Y_train)
#lasso.set_params(alpha=lassocv.alpha_)
#lasso.fit(X_opt, Y_opt)
print("MSE",metrics.mean_squared_error(Y_test, lassocv.predict(X_test)))
print("RMSE",np.sqrt(metrics.mean_squared_error(Y_test, lassocv.
↪predict(X_test))))
print("optimal shrinkage parameter for Lasso Model:", lassocv.alpha_)
```

```
MSE 2.8963302859859095
RMSE 1.7018608303812357
optimal shrinkage parameter for Lasso Model: 0.005
```

[83]:
```
r2 = r2_score(Y_test,lassocv.predict(X_test))
#print(lm_poly.score(X_test,Y_test))
print('R^2 score:',r2)
```

```
R^2 score: 0.9747578610219854
```

[ ]:
```
"""For the lasso cv model we can see the optimal shrinkage parameter is same␣
↪and it is 0.005"""
"""From the lasso model we can see that the R^2 score is 0.974, hence the␣
↪performance decline for this model. And multiple
regression , polynomial regression and ridge regression performs better then␣
↪this model"""
```

[86]:
```
"""Now we will apply another model Random forest with maximum depth 10"""
from sklearn.ensemble import RandomForestRegressor
lm_rf = RandomForestRegressor(max_depth= 10, random_state = 0, n_estimators =␣
↪10)
lm_rf.fit(X_train, Y_train)
```

[86]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=10,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
```

[87]:
```
r2 = r2_score(Y_test,lm_rf.predict(X_test))
#print(lm_poly.score(X_test,Y_test))
print('R^2 score:',r2)
```

```
R^2 score: 0.9998736673437375
```

[89]:
```
"""Random Forest with max depth 50"""
from sklearn.ensemble import RandomForestRegressor
lm_rf_50 = RandomForestRegressor(max_depth= 50, random_state = 0, n_estimators␣
↪= 10)
lm_rf_50.fit(X_train, Y_train)
```

[89]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=50,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
```

[90]:
```
r2 = r2_score(Y_test,lm_rf_50.predict(X_test))
#print(lm_poly.score(X_test,Y_test))
print('R^2 score:',r2)
```

```
R^2 score: 0.9999459787231981
```

[93]:
```
"""Conclusion"""
from astropy.table import QTable, Table, Column
data_rows = [('Multiple Linear Regression', r2_score(Y_test,Y_pred)),(␣
↪'Polynomial Regression', -r2_score(Y_test,Y_poly_pred)),
             ('Ridge Regression', r2_score(Y_test,lm_ridge_opt.
↪predict(X_test))),
             ('Lasso Model', r2_score(Y_test,lassocv.predict(X_test))),
             ('Random Forest model_ with depth 10', r2_score(Y_test,lm_rf.
↪predict(X_test))),
             ('Random Forest model_ depth 50',r2_score(Y_test,lm_rf_50.
↪predict(X_test))) ]
t = Table(rows=data_rows, names=('Model', 'R^2_score'))
print(t)
```

```
              Model                   R^2_score
---------------------------------- ------------------
        Multiple Linear Regression 0.9901798864537803
             Polynomial Regression 0.9911999745738618
                  Ridge Regression 0.9901121321961862
                       Lasso Model 0.9747578610219854
Random Forest model_ with depth 10 0.9998736673437375
```

```
Random Forest model_ depth 50 0.9999459787231981
```

[ ]:
```
"""This is the summary of the models used and R^2 score. From the summary we␣
↪can see that the Polynomial Linear
regression performs well with comparison to Multiple linear and ridge␣
↪regression for this dataset. But the Random
Forest model with depth 50 perfroms the best with r^2 score of 0.9999"""
```