```
# coding: utf-8


# In[483]:



#LIST OF PACKAGES REQUIRED TO RUN THIS CODE.KINDLY INSTALL THESE USING PIP BEFORE RUNNING THIS CODE
#Author-Akash Gandhi(University of South FLorida)
import json
import requests
import os
import urllib.request
#import os
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import datetime
from sklearn import linear_model
from sklearn import metrics
from sklearn.cross_validation import train_test_split

#CODE SNIPPET TO MAKE API CALLS TO WUNDERGROUND.SINCE WUNDERGROUND LIMITS THE
#USAGE OF API CALLS EVERY DAY(500/DAY) AND MIN(10/MIN)
#THEREFORE i HAVE MADE ONLY 1 SUCCEFULL ATTEMPT TO GATHER THE DATA
#FROM API CALL AND SAVED THE DATAFRAME WITHIN A LOCAL CSV FILE
#FOR FURTHER ITERATION I HAVE USED THE SAME CSV FILE.TO GENERATE NEW DATA
#UNCOMMENT BELOW CODE AND USE YOUR KEY TO MAKE THE API CALLS
```

'''#Author-Akash Gandhi(University of South FLorida)

#df=pd.DataFrame(columns=['stationid','softwaretype','hour','minute','pretty','YYYYMMDD','tzname','dewpti','dewptm','heatindexi','heatindexm','hum','precip_ratei','precip_ratem','pressurei','pressurem','precip_totali','precip_totalm','tempi','tempm','solarradiation','UV','wdird','wdire','wgusti','wgustm','windchilli','windchillm','wspdi','wspdm'])

summary=pd.DataFrame(columns=['stationid','pretty','YYYYMMDD','tzname','meantempm','meantempi','meandewpti','meandewptm','meanwindspdm','meanwindspdi','meanwdire','meanwdird','humidity','maxtempm','maxtempi','mintempm','mintempi','maxhumidity','minhumidity','maxdewptm','maxdewpti','mindewpti','mindewptm','maxpressurem','maxpressurei','maxwspdi','maxwspdm','minpressurei','minpressurem','precipm','precipi'])

def appendframe(url,mydict,caldate):

    global summary

    stationid=str(url.split("_")[1].split(":")[1].split(".")[0])

    pretty=mydict['history']['dailysummary'][0]['date']['pretty']

    tzname=mydict['history']['dailysummary'][0]['date']['tzname']

    humidity=mydict['history']['dailysummary'][0]['humidity']

    maxdewpti=mydict['history']['dailysummary'][0]['maxdewpti']

    maxdewptm=mydict['history']['dailysummary'][0]['maxdewptm']

    maxhumidity=mydict['history']['dailysummary'][0]['maxhumidity']

    maxpressurei=mydict['history']['dailysummary'][0]['maxpressurei']

    maxpressurem=mydict['history']['dailysummary'][0]['maxpressurem']

    maxtempi=mydict['history']['dailysummary'][0]['maxtempi']

    maxtempm=mydict['history']['dailysummary'][0]['maxtempm']

    maxwspdi=mydict['history']['dailysummary'][0]['maxwspdi']

    maxwspdm=mydict['history']['dailysummary'][0]['maxwspdm']

    meandewpti=mydict['history']['dailysummary'][0]['meandewpti']

    meandewptm=mydict['history']['dailysummary'][0]['meandewptm']

    meantempi=mydict['history']['dailysummary'][0]['meantempi']

    meantempm=mydict['history']['dailysummary'][0]['meantempm']

```
    meanwdird=mydict['history']['dailysummary'][0]['meanwdird']
    meanwdire=mydict['history']['dailysummary'][0]['meanwdire']
    meanwindspdi=mydict['history']['dailysummary'][0]['meanwindspdi']
    meanwindspdm=mydict['history']['dailysummary'][0]['meanwindspdm']
    mindewpti=mydict['history']['dailysummary'][0]['mindewpti']
    mindewptm=mydict['history']['dailysummary'][0]['mindewptm']
    minhumidity=mydict['history']['dailysummary'][0]['minhumidity']
    minpressurei=mydict['history']['dailysummary'][0]['minpressurei']
    minpressurem=mydict['history']['dailysummary'][0]['minpressurem']
    mintempi=mydict['history']['dailysummary'][0]['mintempi']
    mintempm=mydict['history']['dailysummary'][0]['mintempm']
    precipi=mydict['history']['dailysummary'][0]['precipi']
    precipm=mydict['history']['dailysummary'][0]['precipm']

summary=summary.append({"stationid":stationid,"pretty":pretty,"YYYYMMDD":caldate,"tzname":tzna
me,"maxdewpti":maxdewpti,"maxdewptm":maxdewptm,

"meandewpti":meandewpti,"meandewptm":meandewptm,"maxhumidity":maxhumidity,"humidity":hu
midity,"meantempi":meantempi,

"meantempm":meantempm,"meanwindspdi":meanwindspdi,"meanwindspdm":meanwindspdm,"minde
wpti":mindewpti,"mindewptm":mindewptm,

"minhumidity":minhumidity,"minpressurei":minpressurei,"minpressurem":minpressurem,

"precipi":precipi,"precipm":precipm,"maxpressurei":maxpressurei,"maxpressurem":maxpressurem,"ma
xtempi":maxtempi,

"maxtempm":maxtempm,"meanwdird":meanwdird,"meanwdire":meanwdire,"mintempi":mintempi,"mi
ntempm":mintempm,
                    "maxwspdi":maxwspdi,"maxwspdm":maxwspdm},ignore_index=True)
    return
#newkey-<YOUR API KEY>
```

```
#oldkey-<YOUR API KEY>
base = datetime.datetime.today()
numdays=801 #NUMBER OF DAYS DATA TO BE EXTRACTED
date_list = [base - datetime.timedelta(days=x) for x in range(1, numdays)]
for i in range (0,len(date_list)):
   day=str(date_list[i].date().day)
   month=str(date_list[i].date().month)
   year=str(date_list[i].date().year)
   if len(day)==1:
      day='0'+day
   if len(month)==1:
      month='0'+month
   caldate=year+month+day
   #history="history_"+caldate
   url1="http://api.wunderground.com/api/<YOUR API
KEY>/history_"+caldate+"/q/pws:KFLTAMPA46.json"
   url2="http://api.wunderground.com/api/<YOUR API
KEY>/history_"+caldate+"/q/pws:KFLTAMPA156.json"
   url3="http://api.wunderground.com/api/<YOUR API
KEY>/history_"+caldate+"/q/pws:KFLTAMPA114.json"
   url4="http://api.wunderground.com/api/<YOUR API
KEY>/history_"+caldate+"/q/pws:KFLTAMPA169.json"
   l=[url1,url2,url3,url4]
   for k in range (0,len(l)):
      with urllib.request.urlopen(l[k]) as response:
         a = response.read()
      mydict1=json.loads(a)
      appendframe(l[k],mydict1,caldate)
      mydict1={}
```

```
    #a=k.decode("utf-8")'''


#COMMENT THIS LINE IF YOU ARE MAKING AN API CALL WITH YOUR KEY.TO AVOID

#MAKING REPEATED CALLS TO THE API I HAVE USED CSV FILE THAT WAS CREATED BY MAKING EARLIER API CALLS

summary=pd.read_csv("weatherdata1.csv")


#summary.to_csv("weatherdata.csv")

summary=summary.drop(summary.columns[0],axis=1)

summary2=pd.DataFrame(columns=['pretty','YYYYMMDD','tzname','KFLTAMPA46_meantempm','KFLTAMPA46_meantempi','KFLTAMPA46_meandewpti','KFLTAMPA46_meandewptm','KFLTAMPA46_meanwindspdm','KFLTAMPA46_meanwindspdi','KFLTAMPA46_meanwdire','KFLTAMPA46_humidity','KFLTAMPA46_maxtempm','KFLTAMPA46_maxtempi','KFLTAMPA46_mintempm','KFLTAMPA46_mintempi','KFLTAMPA46_maxhumidity','KFLTAMPA46_minhumidity','KFLTAMPA46_maxdewptm','KFLTAMPA46_maxdewpti','KFLTAMPA46_mindewpti','KFLTAMPA46_mindewptm','KFLTAMPA46_maxpressurem','KFLTAMPA46_maxpressurei','KFLTAMPA46_maxwspdi','KFLTAMPA46_maxwspdm','KFLTAMPA46_minpressurei','KFLTAMPA46_minpressurem','KFLTAMPA46_precipm','KFLTAMPA46_precipi','KFLTAMPA156_meantempm',


'KFLTAMPA156_meantempi','KFLTAMPA156_meandewpti','KFLTAMPA156_meandewptm','KFLTAMPA156_meanwindspdm','KFLTAMPA156_meanwindspdi','KFLTAMPA156_meanwdire','KFLTAMPA156_humidity','KFLTAMPA156_maxtempm','KFLTAMPA156_maxtempi','KFLTAMPA156_mintempm','KFLTAMPA156_mintempi','KFLTAMPA156_maxhumidity','KFLTAMPA156_minhumidity','KFLTAMPA156_maxdewptm','KFLTAMPA156_maxdewpti','KFLTAMPA156_mindewpti','KFLTAMPA156_mindewptm','KFLTAMPA156_maxpressurem','KFLTAMPA156_maxpressurei','KFLTAMPA156_maxwspdi','KFLTAMPA156_maxwspdm','KFLTAMPA156_minpressurei','KFLTAMPA156_minpressurem','KFLTAMPA156_precipm','KFLTAMPA156_precipi',


'KFLTAMPA114_meantempm','KFLTAMPA114_meantempi','KFLTAMPA114_meandewpti','KFLTAMPA114_meandewptm','KFLTAMPA114_meanwindspdm','KFLTAMPA114_meanwindspdi','KFLTAMPA114_meanwdire','KFLTAMPA114_humidity','KFLTAMPA114_maxtempm','KFLTAMPA114_maxtempi','KFLTAMPA114_mintempm','KFLTAMPA114_mintempi','KFLTAMPA114_maxhumidity','KFLTAMPA114_minhumidity','KFLTAMPA114_maxdewptm','KFLTAMPA114_maxdewpti','KFLTAMPA114_mindewpti','KFLTAMPA114_mindewptm','KFLTAMPA114_maxpressurem','KFLTAMPA114_maxpressurei','KFLTAMPA114_maxwspdi','KFLTAMPA114_maxwspdm','KFLTAMPA114_minpressurei','KFLTAMPA114_minpressurem','KFLTAMPA114_precipm','KFLTAMPA114_precipi'


,'KFLTAMPA169_meantempm','KFLTAMPA169_meantempi','KFLTAMPA169_meandewpti','KFLTAMPA169_meandewptm','KFLTAMPA169_meanwindspdm','KFLTAMPA169_meanwindspdi','KFLTAMPA169_mea
```

```
nwdire','KFLTAMPA169_humidity','KFLTAMPA169_maxtempm','KFLTAMPA169_maxtempi','KFLTAMPA1
69_mintempm','KFLTAMPA169_mintempi','KFLTAMPA169_maxhumidity','KFLTAMPA169_minhumidity','
KFLTAMPA169_maxdewptm','KFLTAMPA169_maxdewpti','KFLTAMPA169_mindewpti','KFLTAMPA169_
mindewptm','KFLTAMPA169_maxpressurem','KFLTAMPA169_maxpressurei','KFLTAMPA169_maxwspdi',
'KFLTAMPA169_maxwspdm','KFLTAMPA169_minpressurei','KFLTAMPA169_minpressurem','KFLTAMPA1
69_precipm','KFLTAMPA169_precipi'])



#summary.columns


for i in range (0,len(summary)):
    if i+3>=len(summary):
        break;
    else:
        if(summary['stationid'].loc[i]=='KFLTAMPA46' and summary['stationid'].loc[i+1]=='KFLTAMPA156'
and summary['stationid'].loc[i+2]=='KFLTAMPA114' and summary['stationid'].loc[i+3]=='KFLTAMPA169'):


summary2=summary2.append({"pretty":summary['pretty'].loc[i],"YYYYMMDD":summary['YYYYMMDD'].
loc[i],"tzname":summary['tzname'].loc[i],"KFLTAMPA46_meantempm":summary['meantempm'].loc[i],"
KFLTAMPA46_meantempi":summary['meantempi'].loc[i],"KFLTAMPA46_meandewpti":summary['mean
dewpti'].loc[i],"KFLTAMPA46_meandewptm":summary['meandewptm'].loc[i],"KFLTAMPA46_meanwind
spdm":summary['meanwindspdm'].loc[i],"KFLTAMPA46_meanwindspdi":summary['meanwindspdi'].loc[
i],"KFLTAMPA46_meanwdire":summary['meanwdire'].loc[i],"KFLTAMPA46_humidity":summary['humidit
y'].loc[i],"KFLTAMPA46_maxtempm":summary['maxtempm'].loc[i],"KFLTAMPA46_maxtempi":summary[
'maxtempi'].loc[i],"KFLTAMPA46_mintempm":summary['mintempm'].loc[i],"KFLTAMPA46_mintempi":s
ummary['mintempi'].loc[i],"KFLTAMPA46_maxhumidity":summary['maxhumidity'].loc[i],"KFLTAMPA46_
minhumidity":summary['minhumidity'].loc[i],"KFLTAMPA46_maxdewptm":summary['maxdewptm'].loc[i
],"KFLTAMPA46_maxdewpti":summary['maxdewpti'].loc[i],"KFLTAMPA46_mindewpti":summary['minde
wpti'].loc[i],"KFLTAMPA46_mindewptm":summary['mindewptm'].loc[i],"KFLTAMPA46_maxpressurem":
summary['maxpressurem'].loc[i],"KFLTAMPA46_maxpressurei":summary['maxpressurei'].loc[i],"KFLTAM
PA46_maxwspdi":summary['maxwspdi'].loc[i],"KFLTAMPA46_maxwspdm":summary['maxwspdm'].loc[i]
,"KFLTAMPA46_minpressurei":summary['minpressurei'].loc[i],"KFLTAMPA46_minpressurem":summary['
minpressurem'].loc[i],"KFLTAMPA46_precipm":summary['precipm'].loc[i],"KFLTAMPA46_precipi":summ
ary['precipi'].loc[i],


"KFLTAMPA156_meantempm":summary['meantempm'].loc[i+1],"KFLTAMPA156_meantempi":summary
['meantempi'].loc[i+1],"KFLTAMPA156_meandewpti":summary['meandewpti'].loc[i+1],"KFLTAMPA156_
meandewptm":summary['meandewptm'].loc[i+1],"KFLTAMPA156_meanwindspdm":summary['meanwi
```

ndspdm'].loc[i+1],"KFLTAMPA156_meanwindspdi":summary['meanwindspdi'].loc[i+1],"KFLTAMPA156_meanwdire":summary['meanwdire'].loc[i+1],"KFLTAMPA156_humidity":summary['humidity'].loc[i+1],"KFLTAMPA156_maxtempm":summary['maxtempm'].loc[i],"KFLTAMPA156_maxtempi":summary['maxtempi'].loc[i],"KFLTAMPA156_mintempm":summary['mintempm'].loc[i+1],"KFLTAMPA156_mintempi":summary['mintempi'].loc[i+1],"KFLTAMPA156_maxhumidity":summary['maxhumidity'].loc[i+1],"KFLTAMPA156_minhumidity":summary['minhumidity'].loc[i+1],"KFLTAMPA156_maxdewptm":summary['maxdewptm'].loc[i+1],"KFLTAMPA156_maxdewpti":summary['maxdewpti'].loc[i+1],"KFLTAMPA156_mindewpti":summary['mindewpti'].loc[i+1],"KFLTAMPA156_mindewptm":summary['mindewptm'].loc[i+1],"KFLTAMPA156_maxpressurem":summary['maxpressurem'].loc[i+1],"KFLTAMPA156_maxpressurei":summary['maxpressurei'].loc[i+1],"KFLTAMPA156_maxwspdi":summary['maxwspdi'].loc[i+1],"KFLTAMPA156_maxwspdm":summary['maxwspdm'].loc[i+1],"KFLTAMPA156_minpressurei":summary['minpressurei'].loc[i+1],"KFLTAMPA156_minpressurem":summary['minpressurem'].loc[i+1],"KFLTAMPA156_precipm":summary['precipm'].loc[i+1],"KFLTAMPA156_precipi":summary['precipi'].loc[i+1],

"KFLTAMPA114_meantempm":summary['meantempm'].loc[i+2],"KFLTAMPA114_meantempi":summary['meantempi'].loc[i+2],"KFLTAMPA114_meandewpti":summary['meandewpti'].loc[i+2],"KFLTAMPA114_meandewptm":summary['meandewptm'].loc[i+2],"KFLTAMPA114_meanwindspdm":summary['meanwindspdm'].loc[i+2],"KFLTAMPA114_meanwindspdi":summary['meanwindspdi'].loc[i+2],"KFLTAMPA114_meanwdire":summary['meanwdire'].loc[i+2],"KFLTAMPA114_humidity":summary['humidity'].loc[i+2],"KFLTAMPA114_maxtempm":summary['maxtempm'].loc[i+2],"KFLTAMPA114_maxtempi":summary['maxtempi'].loc[i+2],"KFLTAMPA114_mintempm":summary['mintempm'].loc[i+2],"KFLTAMPA114_mintempi":summary['mintempi'].loc[i+2],"KFLTAMPA114_maxhumidity":summary['maxhumidity'].loc[i+2],"KFLTAMPA114_minhumidity":summary['minhumidity'].loc[i+2],"KFLTAMPA114_maxdewptm":summary['maxdewptm'].loc[i+2],"KFLTAMPA114_maxdewpti":summary['maxdewpti'].loc[i+2],"KFLTAMPA114_mindewpti":summary['mindewpti'].loc[i+2],"KFLTAMPA114_mindewptm":summary['mindewptm'].loc[i+2],"KFLTAMPA114_maxpressurem":summary['maxpressurem'].loc[i+2],"KFLTAMPA114_maxpressurei":summary['maxpressurei'].loc[i+2],"KFLTAMPA114_maxwspdi":summary['maxwspdi'].loc[i+2],"KFLTAMPA114_maxwspdm":summary['maxwspdm'].loc[i+2],"KFLTAMPA114_minpressurei":summary['minpressurei'].loc[i+2],"KFLTAMPA114_minpressurem":summary['minpressurem'].loc[i+2],"KFLTAMPA114_precipm":summary['precipm'].loc[i+2],"KFLTAMPA114_precipi":summary['precipi'].loc[i+2],

"KFLTAMPA169_meantempm":summary['meantempm'].loc[i+3],"KFLTAMPA169_meantempi":summary['meantempi'].loc[i+3],"KFLTAMPA169_meandewpti":summary['meandewpti'].loc[i+3],"KFLTAMPA169_meandewptm":summary['meandewptm'].loc[i+3],"KFLTAMPA169_meanwindspdm":summary['meanwindspdm'].loc[i+3],"KFLTAMPA169_meanwindspdi":summary['meanwindspdi'].loc[i+3],"KFLTAMPA169_meanwdire":summary['meanwdire'].loc[i+3],"KFLTAMPA169_humidity":summary['humidity'].loc[i+3],"KFLTAMPA169_maxtempm":summary['maxtempm'].loc[i+3],"KFLTAMPA169_maxtempi":summary['maxtempi'].loc[i+3],"KFLTAMPA169_mintempm":summary['mintempm'].loc[i+3],"KFLTAMPA169_mintempi":summary['mintempi'].loc[i+3],"KFLTAMPA169_maxhumidity":summary['maxhumidity'].loc[i+3],"KFLTAMPA169_minhumidity":summary['minhumidity'].loc[i+3],"KFLTAMPA169_maxdewptm":summary['maxdewptm'].loc[i+3],"KFLTAMPA169_maxdewpti":summary['maxdewpti'].loc[i+3],"KFLTAMPA169_mindewpti":summary['mindewpti'].loc[i+3],"KFLTAMPA169_mindewptm":summary['mindewptm'].loc[i+3],"KFLTAMPA169_maxpressurem":summary['maxpressurem'].loc[i+3],"KFLTAMPA169_maxpressurei":summary

```
['maxpressurei'].loc[i+3],"KFLTAMPA169_maxwspdi":summary['maxwspdi'].loc[i+3],"KFLTAMPA169_ma
xwspdm":summary['maxwspdm'].loc[i+3],"KFLTAMPA169_minpressurei":summary['minpressurei'].loc[i+
3],"KFLTAMPA169_minpressurem":summary['minpressurem'].loc[i+3],"KFLTAMPA169_precipm":summa
ry['precipm'].loc[i+3],"KFLTAMPA169_precipi":summary['precipi'].loc[i+3]},ignore_index=True)




history=pd.read_csv("history.csv")


l=list(history['YYYYMMDD'].unique())

df3=pd.DataFrame(columns=['condition_mode','YYYYMMDD'])



for i in range(0,len(l)):
    k=list(history[history['YYYYMMDD']==l[i]]['condition'].mode().values)
    df3=df3.append({"condition_mode":k,"YYYYMMDD":l[i]},ignore_index=True)


df4=pd.merge(df3, summary2, on='YYYYMMDD')


#df4.to_csv("finalmodelinput.csv")


null_columns=df4.columns[df4.isnull().any()]

df5=df4.dropna(how='any')


#df5.to_csv("cleandata.csv")


# Dataset Path

DATASET_PATH = "cleandata.csv"

#COMMENT THIS LINE IF ONE IS USING AN API TO MAKE CALLS

data = pd.read_csv(DATASET_PATH)
```

```
#CLEANING THE DATAFRAME BY REMOVING COLUMN WITH NAME "UNAMED"
#AND ALSO REMOVING ROWS FROM THE DATAFRAME WHICH HAS CONDITIONS AS "UNKNOWN"
data=data.loc[:, ~data.columns.str.contains('^Unnamed')]
data=data[data['condition_mode'] !="['Unknown']"]
for i in range(0,len(data)):
   if(data['condition_mode'].iloc[i]=="['Clear', 'Overcast']"):
      data['condition_mode'].iloc[i]="['Partly Cloudy']"
   elif(data['condition_mode'].iloc[i]=="['Mostly Cloudy', 'Overcast']"):
      data['condition_mode'].iloc[i]="['Overcast']"
   elif(data['condition_mode'].iloc[i]=="['Clear', 'Scattered Clouds']"):
      data['condition_mode'].iloc[i]="['Scattered Clouds']"
   elif(data['condition_mode'].iloc[i]=="['Overcast', 'Scattered Clouds']"):
      data['condition_mode'].iloc[i]="['Overcast']"
   elif(data['condition_mode'].iloc[i]=="['Light Rain', 'Overcast']"):
      data['condition_mode'].iloc[i]="['Drizzle']"
```

```
#DROPPING COLUMNS SINCE SOME VALUES ARE GIVEN IN MULTIPLE UNITS LIKE DEGREE,KELVIN ETC.
#TRAIN AND TEST SPLITTING OF THE DATASET

train=data.iloc[:,1:].drop(['YYYYMMDD','pretty','tzname','KFLTAMPA46_meanwdire','KFLTAMPA156_me
anwdire','KFLTAMPA114_meanwdire','KFLTAMPA169_meanwdire','KFLTAMPA46_meantempm','KFLTA
MPA156_meantempm','KFLTAMPA114_meantempm','KFLTAMPA169_meantempm'

,'KFLTAMPA46_meanwindspdm','KFLTAMPA156_meanwindspdm','KFLTAMPA169_meanwindspdm','KFL
TAMPA114_meanwindspdm','KFLTAMPA46_meandewptm','KFLTAMPA46_maxtempm','KFLTAMPA46_
mintempm','KFLTAMPA46_maxdewptm','KFLTAMPA46_mindewptm','KFLTAMPA46_maxpressurem','KF
LTAMPA46_maxwspdm','KFLTAMPA46_minpressurem'
```

```
,'KFLTAMPA46_precipm','KFLTAMPA156_meantempm','KFLTAMPA156_meandewptm','KFLTAMPA156_
maxtempm','KFLTAMPA156_mintempm','KFLTAMPA156_mindewptm','KFLTAMPA156_maxpressurem',

                'KFLTAMPA156_maxwspdm','KFLTAMPA156_minpressurem','KFLTAMPA114_mintempm'

,'KFLTAMPA114_maxdewptm','KFLTAMPA114_mindewptm','KFLTAMPA114_maxpressurem','KFLTAMPA
114_maxwspdm','KFLTAMPA114_minpressurem',

'KFLTAMPA114_precipm','KFLTAMPA114_meandewptm','KFLTAMPA169_meandewptm','KFLTAMPA114
_maxtempm','KFLTAMPA169_maxtempm','KFLTAMPA169_mintempm',

'KFLTAMPA169_maxdewptm','KFLTAMPA169_mindewptm','KFLTAMPA169_maxpressurem','KFLTAMPA
169_maxwspdm'

                ,'KFLTAMPA169_minpressurem','KFLTAMPA169_precipm'],axis=1)

test=data.iloc[:,0]


#TRAIN AND TEST SPLIT FOR X & y RESPECTIVELY

train_x, test_x, train_y, test_y = train_test_split(train,test, train_size=0.8)


#EXECUTING SIMPLE LINEAR LOGISTIC REGRESSION

lr = linear_model.LogisticRegression()

lr.fit(train_x, train_y)


#USING MULTINOMINAL LOGISTIC REGRESSION.

#RUN ONLY ONE OF THE TWO AS BOTH USE DIFFERENT KIND OF SOLVERS TO FIT THE REGRESSION
MODEL

#mul_lr = linear_model.LogisticRegression(multi_class='multinomial', solver='newton-
cg',max_iter=500).fit(train_x, train_y)

mul_lr = linear_model.LogisticRegression(multi_class='multinomial',
solver='lbfgs',max_iter=100).fit(train_x, train_y)


print("Logistic regression Train Accuracy :: ", metrics.accuracy_score(train_y, lr.predict(train_x)))

print("Logistic regression Test Accuracy :: ", metrics.accuracy_score(test_y, lr.predict(test_x)))
```

```
print("Multinomial Logistic regression Train Accuracy :: ", metrics.accuracy_score(train_y, mul_lr.predict(train_x)))

print("Multinomial Logistic regression Test Accuracy :: ", metrics.accuracy_score(test_y, mul_lr.predict(test_x)))


#This Graph shows the  temperature data from weather station 169 is not quite accurate as it does not follow a linear trend

k=data['KFLTAMPA169_meandewpti']

l=data['KFLTAMPA156_meandewpti']

import matplotlib.pyplot as plt

#import plotly.plotly as py

plt.scatter(k,l)

plt.xlabel('KFLTAMPA169 mean dew point')

plt.ylabel('KFLTAMPA156 mean dew point')

plt.show()


##This Graph shows the  dewpoint data from weather station KFLTAMPA169 is not quite accurate as it does not follow a linear trend


k=data['KFLTAMPA46_meandewpti']

l=data['KFLTAMPA169_meandewpti']

import matplotlib.pyplot as plt

#import plotly.plotly as py

plt.scatter(k,l)

plt.xlabel('114')

plt.ylabel('169')

plt.show()


# Randomly sample 7 elements from your dataframe
```

```
#df_random = test_x.sample(n=7)


file=open("prediction.txt",'w')

file.write("Predictions made by model as
on:"+str(datetime.datetime.now().date())+","+str(datetime.datetime.now().time())+"\n"+"\n")

file.write("Actual value"+"\t"+"\t"+"Predicted Value"+"\n")

prediction_list=list(mul_lr.predict(test_x))

for datapoint in range(0,len(test_y)):

    file.write(test_y.iloc[datapoint]+"\t"+"\t"+prediction_list[datapoint]+"\n")

file.close()
```