pda-6

March 22, 2025

Time Series Analysis(TDA)

Time Series Analysis(TDA) is a method of analyzing the data points collected over time to identify patterns,trends and seasonal variations.It is used to forecast future values based on historical data.

Importing Libraries

[5]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
```

Loading and viewing the data

[7]:
```
df=pd.read_csv(r"C:\Users\DELL\Downloads\DailyDelhiClimateTrain.csv")
```

[8]:
```
df.head()
```

[8]:

	date	meantemp	humidity	wind_speed	meanpressure
0	2013-01-01	10.000000	84.500000	0.000000	1015.666667
1	2013-01-02	7.400000	92.000000	2.980000	1017.800000
2	2013-01-03	7.166667	87.000000	4.633333	1018.666667
3	2013-01-04	8.666667	71.333333	1.233333	1017.166667
4	2013-01-05	6.000000	86.833333	3.700000	1016.500000

Info

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1462 entries, 0 to 1461
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   date          1462 non-null   object
 1   meantemp      1462 non-null   float64
 2   humidity      1462 non-null   float64
```

```
 3   wind_speed    1462 non-null   float64
 4   meanpressure  1462 non-null   float64
dtypes: float64(4), object(1)
memory usage: 57.2+ KB
```

setting date as index

[10]:
```
#checking for nulls in date column
print(df[df["date"].isna()])
```

```
Empty DataFrame
Columns: [date, meantemp, humidity, wind_speed, meanpressure]
Index: []
```

[11]:
```
#convert object into date datatype
df["date"]=pd.to_datetime(df["date"],errors="coerce")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1462 entries, 0 to 1461
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   date          1462 non-null   datetime64[ns]
 1   meantemp      1462 non-null   float64
 2   humidity      1462 non-null   float64
 3   wind_speed    1462 non-null   float64
 4   meanpressure  1462 non-null   float64
dtypes: datetime64[ns](1), float64(4)
memory usage: 57.2 KB
```

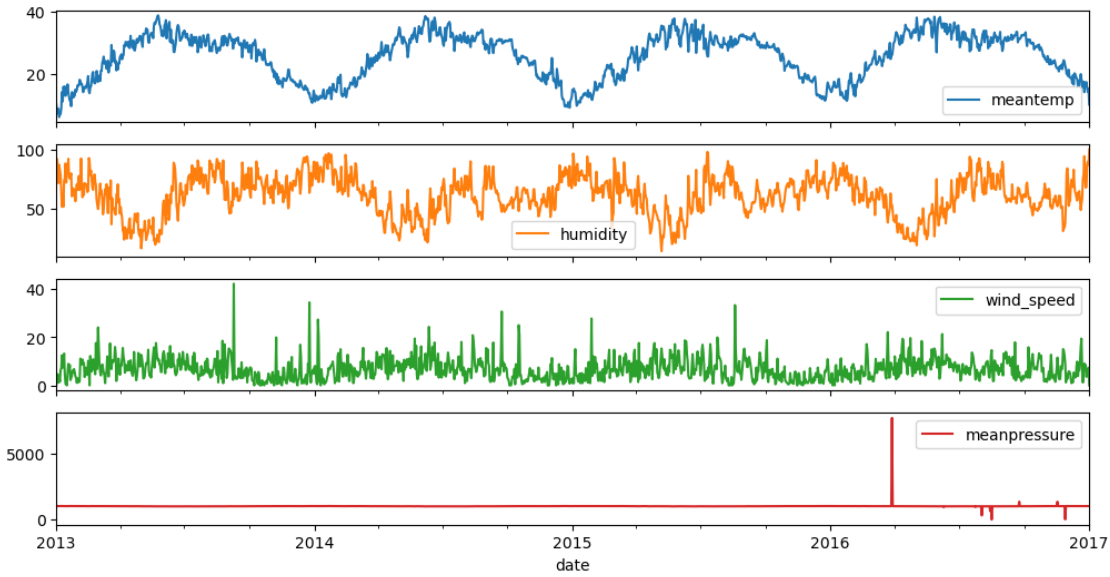
[12]:
```
#setting index
df.set_index("date",inplace=True)
df.head()
```

[12]:

date	meantemp	humidity	wind_speed	meanpressure
2013-01-01	10.000000	84.500000	0.000000	1015.666667
2013-01-02	7.400000	92.000000	2.980000	1017.800000
2013-01-03	7.166667	87.000000	4.633333	1018.666667
2013-01-04	8.666667	71.333333	1.233333	1017.166667
2013-01-05	6.000000	86.833333	3.700000	1016.500000

visualize

[13]:
```
df.plot(figsize=(12,6),subplots=True)
plt.show()
```



conclusions: 1.temperature in starting of the year has been increased than that of ending of the year. 2.wind speed: from the year 2013 to 2014 there are two spikes.and 2014 to 2015 their are 2 spikes,in 2015 to 2016 their is 1 spike,in the year 2016 to 2017 there is no spikes are found. 3.the mean pressure remains constant until the year 2016 then their is sudden increase in the years between 2016 and 2017.

stationarity: A time series is stationarity if its statistical properties(mean,variance,autocorrelation) remain constant over time.

Hypothesis of ADF test: - Null Hypothesis:The time series has a unit root(i.e, it is non-stationary) - Alternative Hypothesis: The time series does not have a unit root(i.e,it is stationery).

Interpreting ADF test results: - if the p-value is less than 0.05,reject null hypothesis->the series is stationary. - if the p-value is more then 0.05,fall to reject hypothesis->the series is not stationary.

[14]:
```
adfuller_result=adfuller(df["meantemp"])
```

[15]:
```
print(adfuller_result)
```

```
(-2.0210690559206728, 0.27741213723016056, 10, 1451, {'1%': -3.4348647527922824,
'5%': -2.863533960720434, '10%': -2.567831568508802}, 5423.895746470953)
```

[16]:
```
if adfuller_result[1]<0.05:
    print("Stationary")
else:
    print("not stationary")
```

```
not stationary
```

1 Differncing to remove trend:if the series is non stationary,apply differencing.

Differencing is a technique used to make a non stationary time series stationary by removing trends or seasonality.it involves subtracting the previous observation from the current observation.

temperature=[20,21,22,24,25,27,28,27] Difference=[1,1,2,1,2,1,-1] The new series fluctuates around zero -2 to 2

Differencing:

[17]:
```
df["meantemp_diff"]=df["meantemp"].diff()
df.head()
```

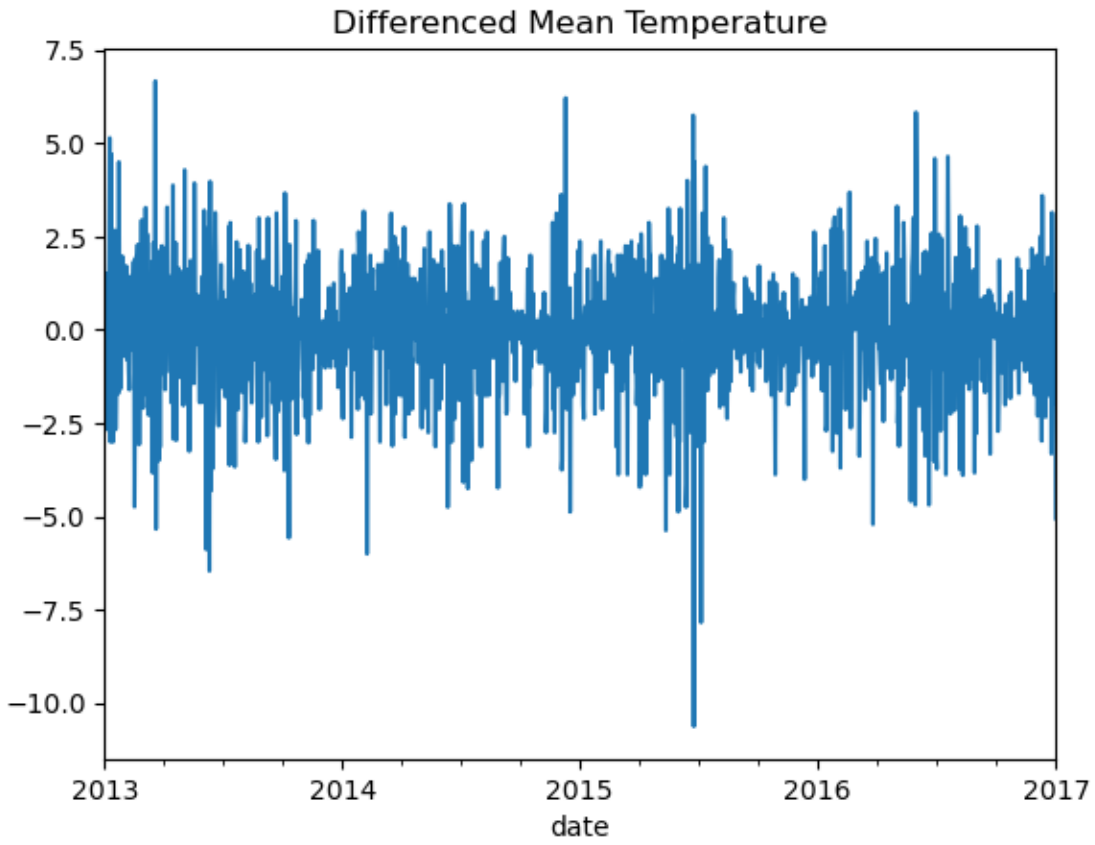
[17]:

date	meantemp	humidity	wind_speed	meanpressure	meantemp_diff
2013-01-01	10.000000	84.500000	0.000000	1015.666667	NaN
2013-01-02	7.400000	92.000000	2.980000	1017.800000	-2.600000
2013-01-03	7.166667	87.000000	4.633333	1018.666667	-0.233333
2013-01-04	8.666667	71.333333	1.233333	1017.166667	1.500000
2013-01-05	6.000000	86.833333	3.700000	1016.500000	-2.666667

[18]:
```
adfuller_result_afterdiff=adfuller(df["meantemp_diff"].dropna())
if adfuller_result_afterdiff[1]>0.05:
    print("non-stationary")
else:
    print("stationary")
```

```
stationary
```

[20]:
```
df["meantemp_diff"].plot(title="Differenced Mean Temperature")
plt.show()
```



Conclusions: 1.It has outliers in years 2014 and 2015,Their is drastic change in the temperature. 2.The average is around 0 where we found the dark colour. 3.Extreme low temperature is found between the year 2015 and 2016. 4.The data is stationary.

2 Use Seasonal decompositon to analyze the trend seasonality and residuals

- trend-the long term pattern(increase or decrease overtime).
- Seasonality-the repeating patterns at fixed intervals(e.g, monthly sales spikes).
- residual(noise)-the random variations that are not explained by trend or seasonality.

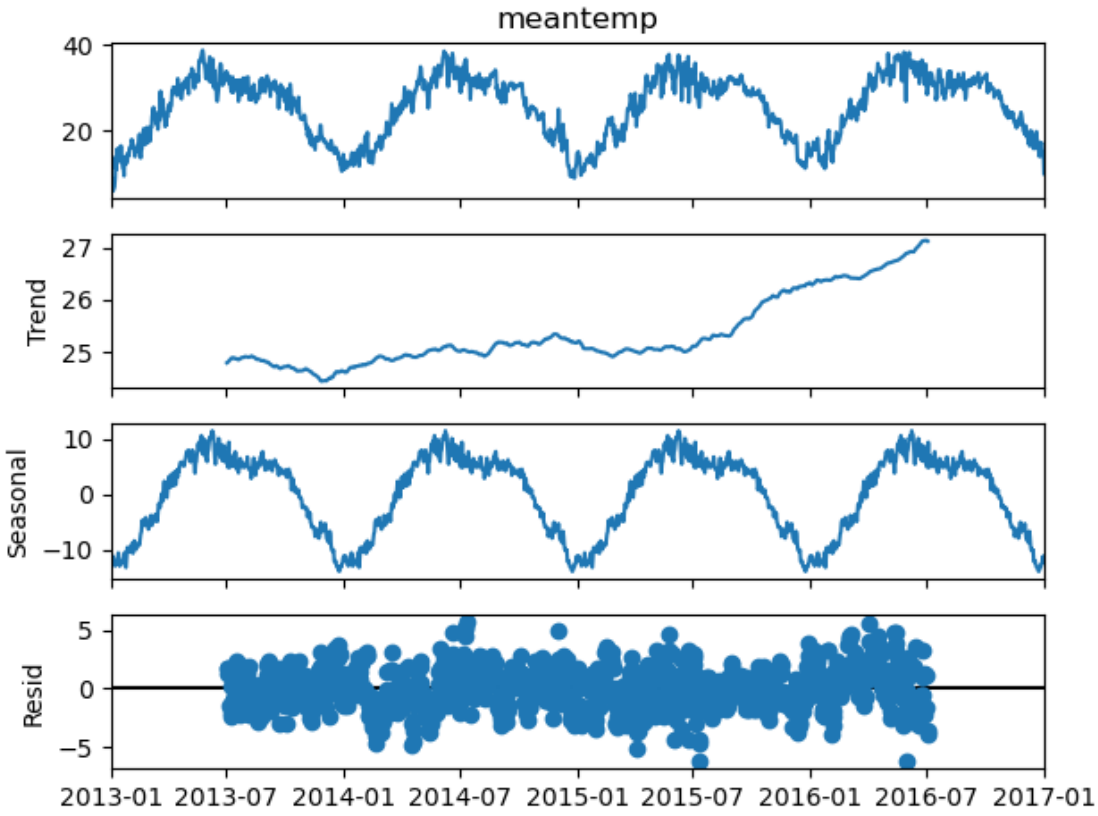
Interpreting the output:

- original series:The raw time series data.
- trend component:the general direction of the data overtime.
- seasonal component:The repeating patterns(e.g higher sales in december)
- residual component:The remaining part after removing trend and seasonality

Decomposing:

[23]:
```
decompose=seasonal_decompose(df["meantemp"],model="additive",period=365)
```

[24]:
```
decompose.plot()
plt.show()
```



Conclusions: 1.Trend: The temperature decreased in the year 2014 and their is constant rapid increase around 2016. 2.Seasonal: The data is non stationary.It keeps on increasing and decreasing in the temperature. 3.Residual: The temperature started at 2013 it remains constant until the year 2016.

ARIMA

[25]:
```
len(df)
```

[25]: 1462

[26]:
```
print(len(df)*0.8)
```

1169.6000000000001

[31]:
```
train=df.iloc[0:1169]
test=df.iloc[1169:]
len(test)
```

[31]: 293

[32]:
```
mymodel=ARIMA(train["meantemp"],order=(1,1,1))
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
```

[33]:
```
mymodel=mymodel.fit()
```

[34]:
```
forecast=mymodel.forecast(steps=len(test))
print(forecast)
```

```
2016-03-15    22.826205
2016-03-16    23.085687
2016-03-17    23.234913
2016-03-18    23.320731
2016-03-19    23.370084
                 …
2016-12-28    23.436880
2016-12-29    23.436880
2016-12-30    23.436880
2016-12-31    23.436880
2017-01-01    23.436880
Freq: D, Name: predicted_mean, Length: 293, dtype: float64
```

[38]:
```
test["forecast"]=forecast
test.head()
```

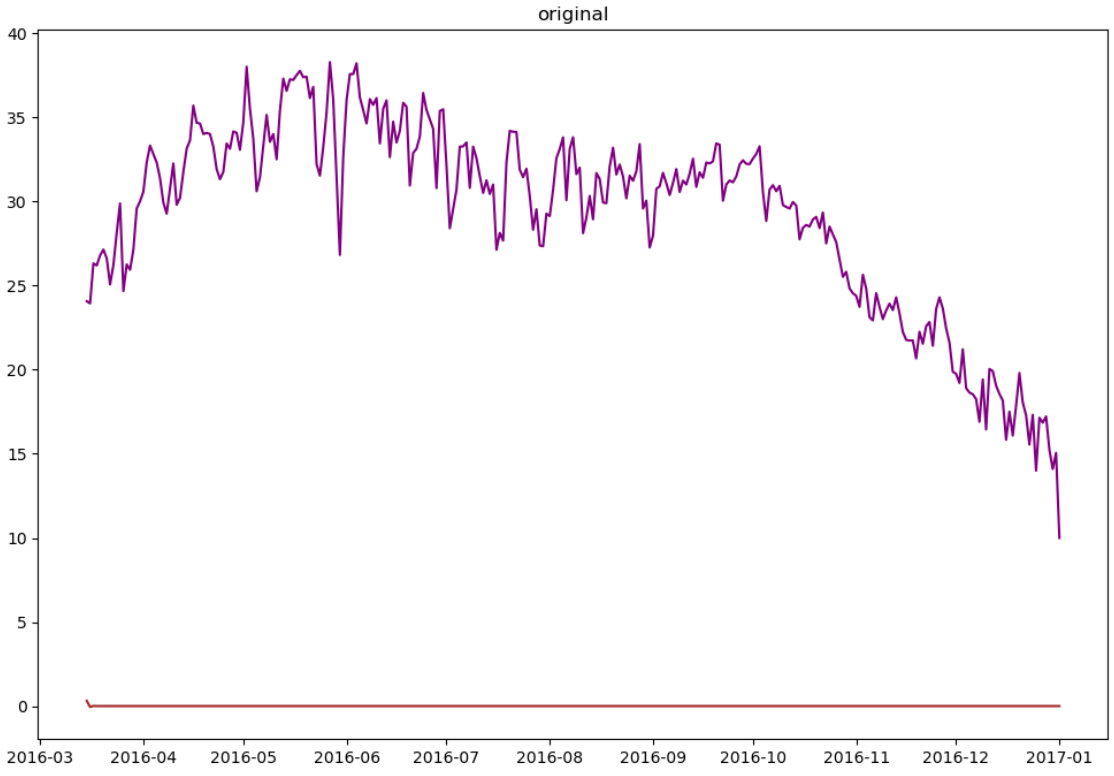
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_5844\793870190.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test["forecast"]=forecast
```

[38]:

date	meantemp	humidity	wind_speed	meanpressure	meantemp_diff	forecast
2016-03-15	24.066667	58.933333	8.646667	1014.866667	1.691667	22.826205
2016-03-16	23.937500	53.750000	10.881250	1012.812500	-0.129167	23.085687
2016-03-17	26.312500	50.312500	6.843750	1010.437500	2.375000	23.234913
2016-03-18	26.187500	61.250000	6.712500	1009.812500	-0.125000	23.320731
2016-03-19	26.785714	61.857143	3.578571	1009.214286	0.598214	23.370084

[58]:
```
plt.figure(figsize=(12,8))
plt.plot(test.index,test["meantemp"],color="purple",label="original")
plt.plot(test.index,test["forecast"],color="brown",label="original")
plt.title("original")
plt.show()
```



[43]:
```
len(df)
```

[43]: 1462

[44]:
```
print(len(df)*0.8)
```

```
1169.6000000000001
```

[45]:
```
train=df.iloc[0:1169]
test=df.iloc[1169:]
len(test)
```

[45]: 293

[47]:
```
mymodel1=ARIMA(train["meantemp_diff"],order=(1,1,1))
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
```

[48]:
```
mymodel1=mymodel1.fit()
```

[49]:
```
forecast=mymodel1.forecast(steps=len(test))
print(forecast)
```

```
2016-03-15    0.322914
2016-03-16   -0.040400
2016-03-17    0.019656
2016-03-18    0.009729
2016-03-19    0.011370
                 …
2016-12-28    0.011137
2016-12-29    0.011137
2016-12-30    0.011137
2016-12-31    0.011137
2017-01-01    0.011137
Freq: D, Name: predicted_mean, Length: 293, dtype: float64
```

[50]:
```
test["forecast"]=forecast
test.head()
```

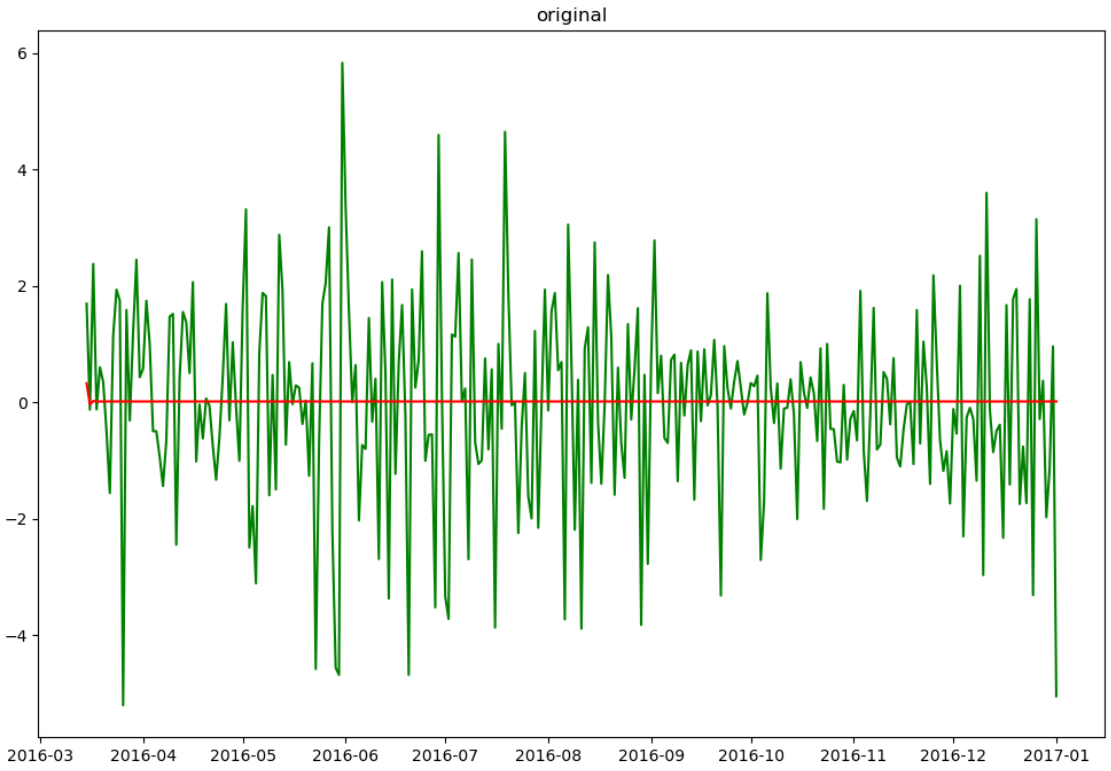
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_5844\793870190.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test["forecast"]=forecast
```

[50]:

date	meantemp	humidity	wind_speed	meanpressure	meantemp_diff	forecast
2016-03-15	24.066667	58.933333	8.646667	1014.866667	1.691667	0.322914
2016-03-16	23.937500	53.750000	10.881250	1012.812500	-0.129167	-0.040400
2016-03-17	26.312500	50.312500	6.843750	1010.437500	2.375000	0.019656
2016-03-18	26.187500	61.250000	6.712500	1009.812500	-0.125000	0.009729
2016-03-19	26.785714	61.857143	3.578571	1009.214286	0.598214	0.011370

[56]:
```
plt.figure(figsize=(12,8))
plt.plot(test.index,test["meantemp_diff"],color="green",label="original")
plt.plot(test.index,test["forecast"],color="red",label="original")
plt.title("original")
plt.show()
```



Conclusion: 15th march,original value=24.066667,model says their is -0.040438 change on next day. 24.066667-0.040438=24.02(predicted)-23.937500(original).