applied-datascience-phase4

October 28, 2023

1 Date - 26/10/2023

2 Team ID - 3872

3 Project Title - Product Demand Prediction using ML

4 Importing Dependencies

[37]:
```
import pandas as pd
import re
import matplotlib.pyplot as plt
import os
import plotly.express as px
import numpy as np
```

5 Loading Dataset

[6]:
```
df = pd.read_csv("C:\\Users\\Dell\\Downloads\\trainnew.csv")
```

6 Data Exploration

[7]:
```
df
```

[7]:

	date	store	item	sales
0	01-01-2013	1	1	13
1	02-01-2013	1	1	11
2	03-01-2013	1	1	14
3	04-01-2013	1	1	13
4	05-01-2013	1	1	10
…	…	…	…	…
912995	27-12-2017	10	50	63
912996	28-12-2017	10	50	59
912997	29-12-2017	10	50	74
912998	30-12-2017	10	50	62
912999	31-12-2017	10	50	82

```
[913000 rows x 4 columns]
```

[8]:
```
df.set_index('date',inplace=True)
```

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

[9]:
```
            store  item  sales
date
01-01-2013      1     1     13
02-01-2013      1     1     11
03-01-2013      1     1     14
04-01-2013      1     1     13
05-01-2013      1     1     10
```

[10]:
```
store_sales=df.groupby(by='store')[['sales']].sum()
store_sales
```

[10]:
```
         sales
store
1      4315603
2      6120128
3      5435144
4      5012639
5      3631016
6      3627670
7      3320009
8      5856169
9      5025976
10     5360158
```

[12]:
```
store=store_sales.index
store
```

[12]:
```
Int64Index([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], dtype='int64', name='store')
```

7 Pre-Processing and Visualisation of Data

[13]:
```
fig = px.bar(store_sales,color=store)
fig.show()
```

[14]:
```
fig = px.histogram(df[df.item==1][['sales']],labels=dict(value="Sales"))
fig.show()
```

[15]:
```
fig = px.line(df[(df.item==1) & (df.store==4)][['sales']],y='sales')
fig.show()
```

[17]:
```
df_1_1=df[(df.item==1) & (df.store==1)][['sales']]
df_1_1
```

[17]:
```
            sales
date
01-01-2013     13
02-01-2013     11
03-01-2013     14
04-01-2013     13
05-01-2013     10
…             …
27-12-2017     14
28-12-2017     19
29-12-2017     15
30-12-2017     27
31-12-2017     23

[1826 rows x 1 columns]
```

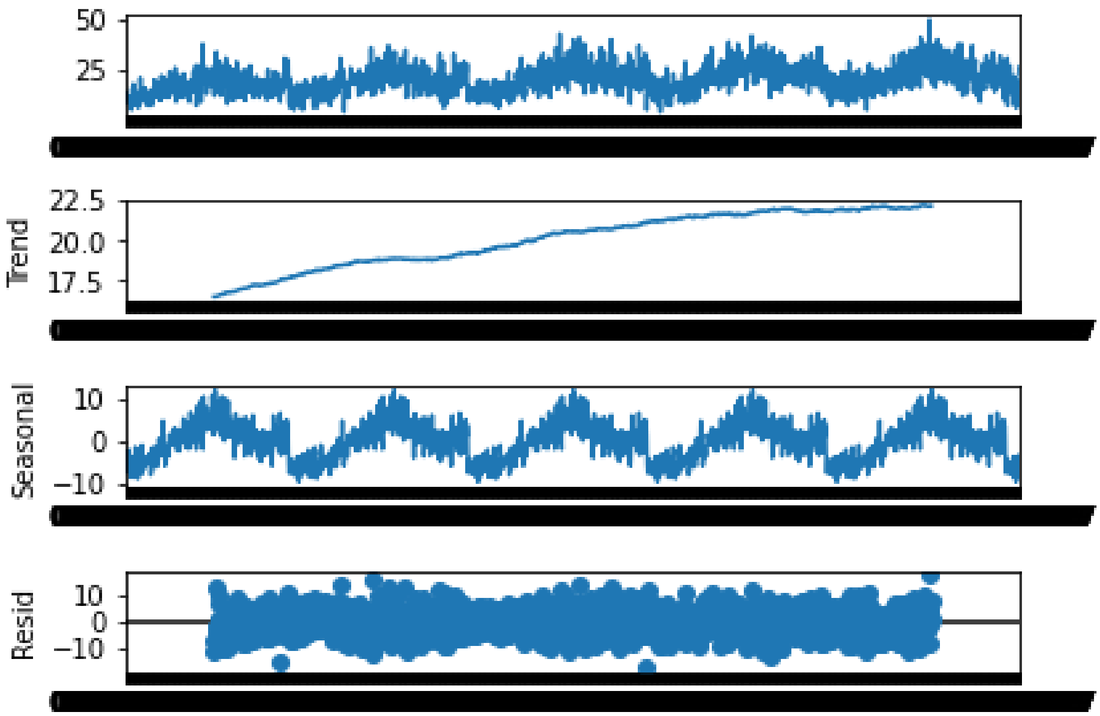
[18]:
```
fig = px.line(df_1_1)
fig.show()
```

8 Stationary

mean,variance and co-variance is constant over periods

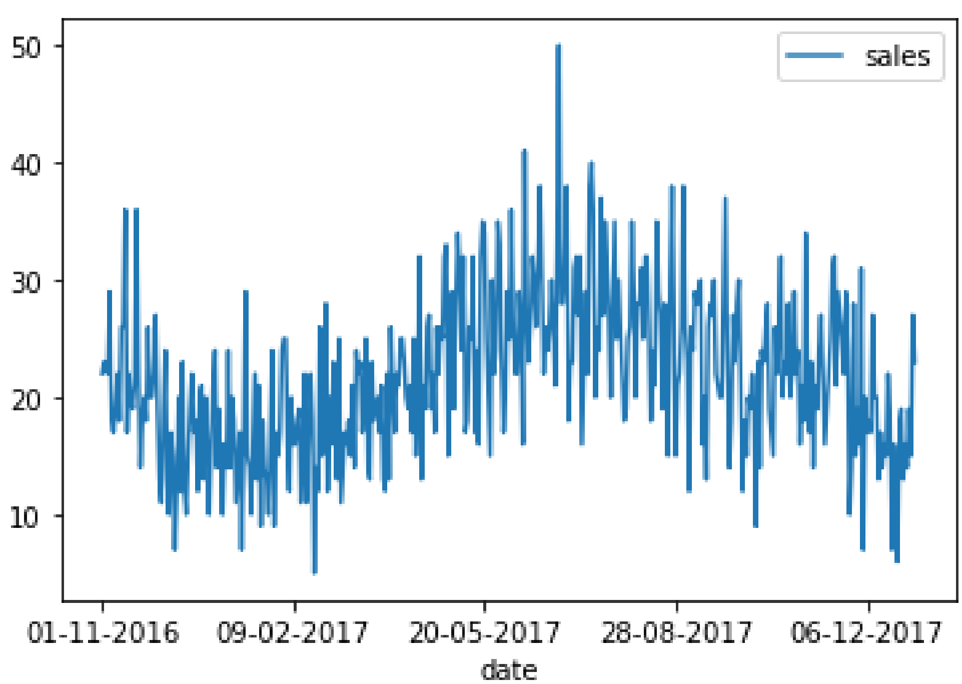
[19]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df_1_1, model='additive', period=365)
plt.figure(figsize=(36, 24))
result.plot()
plt.show()
```

```
<Figure size 2592x1728 with 0 Axes>
```



[20]:
```
df_1_1.iloc[1400:,].plot()
```

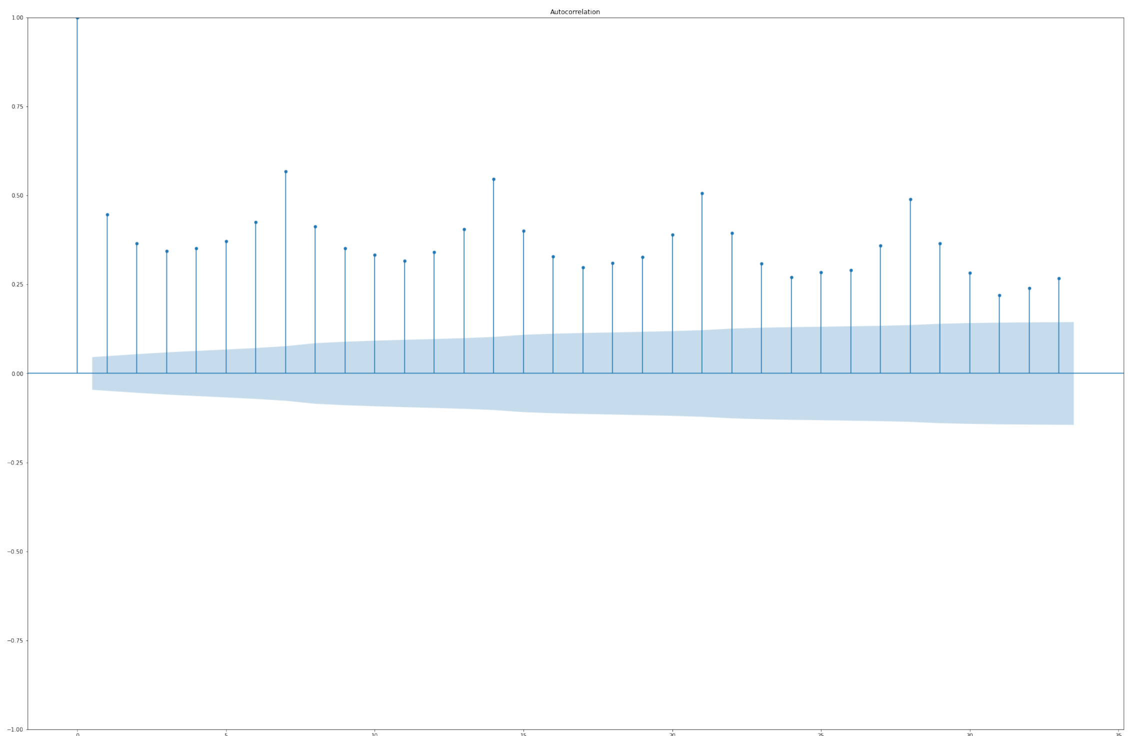
[20]:
```
<AxesSubplot:xlabel='date'>
```



```
[21]: pd.options.plotting.backend = "plotly"
```

```
[22]: import statsmodels.api as sm

      fig, ax = plt.subplots(figsize=(36,24))
      sm.graphics.tsa.plot_acf(df_1_1, ax=ax)
      plt.show()
```



9 Ad Fuller Hypothesis test

Null Hypothesis (H0): If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary

p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.

p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

```
[23]: from statsmodels.tsa.stattools import adfuller
      hypothesis_test=adfuller(df_1_1)
      print('ADF Statistic: %f' % hypothesis_test[0])
      print('p-value: %f' % hypothesis_test[1])
```

```
print('Critical Values:')
for key, value in hypothesis_test[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -3.157671
p-value: 0.022569
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

[24]:
```
fig = px.histogram(df_1_1)
fig.show()
```

[25]:
```
df_1_1.diff(periods=1).fillna(0).head()
```

[25]:

	sales
date	
01-01-2013	0.0
02-01-2013	-2.0
03-01-2013	3.0
04-01-2013	-1.0
05-01-2013	-3.0

[27]:
```
df_diff=df_1_1.diff(periods=1) #Integrated of order 1 denoted by d
df_diff
```

[27]:

	sales
date	
01-01-2013	NaN
02-01-2013	-2.0
03-01-2013	3.0
04-01-2013	-1.0
05-01-2013	-3.0
…	…
27-12-2017	-2.0
28-12-2017	5.0
29-12-2017	-4.0
30-12-2017	12.0
31-12-2017	-4.0

[1826 rows x 1 columns]

[28]:
```
df_diff=df_diff[1:]
df_diff.head()## 1 Lag
```

[28]:

```
            sales
date
02-01-2013   -2.0
03-01-2013    3.0
04-01-2013   -1.0
05-01-2013   -3.0
06-01-2013    2.0
```

10 ARIMA Model

[29]:

```python
from statsmodels.tsa.arima.model import ARIMA
import itertools
p=d=q=range(0,5)
pdq =list(itertools.product(p,d,q))
X = df_1_1.values
size = int(len(X) * 0.66)
predictions = []
X = df_1_1.values
size = int(len(X) * 0.88)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
train_date=df_1_1.index[0:size]
test_date=df_1_1.index[size:len(X)]
```

[30]:

```python
import warnings
warnings.filterwarnings("ignore")
AIC={}
for i in pdq:
    try:
        model_arima=ARIMA(train,order=(i))
        model_fit=model_arima.fit()
        print(model_fit.aic," ",i)
        AIC[model_fit.aic]=i
    except:
        continue
```

```
10589.258825757217   (0, 0, 0)
10367.211558053592   (0, 0, 1)
10292.248736752164   (0, 0, 2)
10235.806148907097   (0, 0, 3)
10225.534338201956   (0, 0, 4)
10791.119185694395   (0, 1, 0)
9910.661190803017   (0, 1, 1)
9906.846994978841   (0, 1, 2)
9903.313527540227   (0, 1, 3)
9887.288179269826   (0, 1, 4)
```

12476.917019602452   (0, 2, 0)
10794.809935356132   (0, 2, 1)
9921.970189919604   (0, 2, 2)
9916.711356681992   (0, 2, 3)
9914.84604318356   (0, 2, 4)
14367.53299034889   (0, 3, 0)
12479.530565552162   (0, 3, 1)
10834.485738150088   (0, 3, 2)
9953.66790734285   (0, 3, 3)
9950.869265251953   (0, 3, 4)
16347.36217794071   (0, 4, 0)
14369.095456628256   (0, 4, 1)
12488.828263383144   (0, 4, 2)
10838.663618973975   (0, 4, 3)
10071.001282578654   (0, 4, 4)
10262.952804275377   (1, 0, 0)
9916.577889615008   (1, 0, 1)
9912.96129577811   (1, 0, 2)
9909.156376131332   (1, 0, 3)
9892.166744955215   (1, 0, 4)
10454.955245295758   (1, 1, 0)
9907.543771792818   (1, 1, 1)
9903.903615887375   (1, 1, 2)
9885.426266661272   (1, 1, 3)
9881.724509931599   (1, 1, 4)
11650.873403993923   (1, 2, 0)
10459.542829317172   (1, 2, 1)
9917.480272946566   (1, 2, 2)
9918.933060756273   (1, 2, 3)
9917.292260110651   (1, 2, 4)
13142.511008426038   (1, 3, 0)
11654.974822945816   (1, 3, 1)
10526.702429799823   (1, 3, 2)
9943.214979938348   (1, 3, 3)
9942.938064959064   (1, 3, 4)
14803.393185868223   (1, 4, 0)
13145.781936880012   (1, 4, 1)
11666.083617024   (1, 4, 2)
10542.34367819974   (1, 4, 3)
10850.817052533253   (1, 4, 4)
10192.108054857934   (2, 0, 0)
9913.646873568243   (2, 0, 1)
9909.91514019587   (2, 0, 2)
9911.675290010067   (2, 0, 3)
9911.511106769318   (2, 0, 4)
10312.527638268386   (2, 1, 0)
9902.914260425598   (2, 1, 1)
9905.65318892507   (2, 1, 2)

```
9692.107514061445   (2, 1, 3)
9887.63985849464   (2, 1, 4)
11252.829638375366   (2, 2, 0)
10317.800246073679   (2, 2, 1)
9914.086667596106   (2, 2, 2)
9916.469541707018   (2, 2, 3)
9860.877022964894   (2, 2, 4)
12470.320627575566   (2, 3, 0)
11257.954058368356   (2, 3, 1)
10346.0962203885   (2, 3, 2)
10480.187621017532   (2, 3, 3)
9939.576945406781   (2, 3, 4)
13892.129959259542   (2, 4, 0)
12474.928756359677   (2, 4, 1)
11270.532905928456   (2, 4, 2)
10354.717929403585   (2, 4, 3)
10496.986964206539   (2, 4, 4)
10151.265296728536   (3, 0, 0)
9908.84836490576   (3, 0, 1)
9911.70625785237   (3, 0, 2)
9885.749692520694   (3, 0, 3)
9857.167657643346   (3, 0, 4)
10230.239182131572   (3, 1, 0)
9891.149376016629   (3, 1, 1)
9873.28040159759   (3, 1, 2)
9709.991113487413   (3, 1, 3)
9697.084060748633   (3, 1, 4)
11042.885219536221   (3, 2, 0)
10235.88557127723   (3, 2, 1)
10321.702578635279   (3, 2, 2)
10321.796940904173   (3, 2, 3)
9917.591448502491   (3, 2, 4)
12072.667002657818   (3, 3, 0)
11048.740238628694   (3, 3, 1)
10338.13868628243   (3, 3, 3)
10486.73147621081   (3, 3, 4)
13337.292125253318   (3, 4, 0)
12078.294134713986   (3, 4, 1)
11062.263242636967   (3, 4, 2)
11274.012442140145   (3, 4, 3)
10367.36835918028   (3, 4, 4)
10121.97235496898   (4, 0, 0)
9896.778058844127   (4, 0, 1)
9911.2777404579   (4, 0, 2)
9860.908026374787   (4, 0, 3)
9898.718366654326   (4, 0, 4)
10144.478539940836   (4, 1, 0)
9869.833703642613   (4, 1, 1)
```

```
9876.680692066726   (4, 1, 2)
9864.348718290654   (4, 1, 3)
9696.446827925429   (4, 1, 4)
10895.908496291271   (4, 2, 0)
10150.588172826467   (4, 2, 1)
9878.290413387931   (4, 2, 2)
9901.193051361708   (4, 2, 3)
9829.318174199076   (4, 2, 4)
11742.780886425277   (4, 3, 0)
10902.378067206168   (4, 3, 1)
10167.387074769971   (4, 3, 2)
10250.663062035026   (4, 3, 3)
10334.289004090419   (4, 3, 4)
12722.90555233343   (4, 4, 0)
11749.333275279172   (4, 4, 1)
10916.340003755337   (4, 4, 2)
11060.753363559079   (4, 4, 3)
11186.980981014593   (4, 4, 4)
```

[31]:
```
AIC[min(AIC.keys())]
```

[31]:
```
(2, 1, 3)
```

11 Selecting the parameter (p,d,q) -> (4,3,2) as it has minimum AIC

[32]:
```
model_arima=ARIMA(train,order=(2,1,3))
model_fit=model_arima.fit()
```

[33]:
```
model_fit.summary()
```

[33]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1606
Model:                 ARIMA(2, 1, 3)   Log Likelihood               -4840.054
Date:                Sat, 28 Oct 2023   AIC                           9692.108
Time:                        12:24:54   BIC                           9724.393
Sample:                             0   HQIC                          9704.094
                               - 1606
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.2461      0.002    771.732      0.000       1.243       1.249
```

```
ar.L2         -0.9983      0.002   -613.892      0.000      -1.002      -0.995
ma.L1         -2.1397      0.013   -164.035      0.000      -2.165      -2.114
ma.L2          2.0949      0.019    111.773      0.000       2.058       2.132
ma.L3         -0.8825      0.013    -68.298      0.000      -0.908      -0.857
sigma2        24.5610      0.825     29.782      0.000      22.945      26.177
================================================================================
===
Ljung-Box (L1) (Q):                   4.88   Jarque-Bera (JB):
9.24
Prob(Q):                              0.03   Prob(JB):
0.01
Heteroskedasticity (H):               1.29   Skew:
0.11
Prob(H) (two-sided):                  0.00   Kurtosis:
3.30
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

[34]:
```
residuals=pd.DataFrame(model_fit.resid)
residuals.plot()
print(residuals.describe())
```

```
                 0
count  1606.000000
mean      0.092104
std       4.941581
min     -17.190146
25%      -3.216191
50%      -0.036393
75%       3.204789
max      19.337018
```

[35]:
```
predictions=[]
# walk-forward validation
for t in range(len(test)):
        model = ARIMA(history, order=(2,1,3))
        model_fit = model.fit()
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
```

```python
        print('predicted=%f, expected=%f' % (yhat, obs))
# evaluate forecasts
```

```
predicted=27.014689, expected=24.000000
predicted=26.981263, expected=35.000000
predicted=26.085309, expected=33.000000
predicted=24.286540, expected=23.000000
predicted=23.824463, expected=17.000000
predicted=24.115892, expected=20.000000
predicted=26.265065, expected=29.000000
predicted=27.834004, expected=25.000000
predicted=27.793873, expected=36.000000
predicted=26.835758, expected=27.000000
predicted=22.741266, expected=22.000000
predicted=22.778420, expected=22.000000
predicted=23.662570, expected=29.000000
predicted=26.439024, expected=26.000000
predicted=28.461578, expected=16.000000
predicted=27.514132, expected=41.000000
predicted=27.294858, expected=28.000000
predicted=24.742585, expected=23.000000
predicted=23.295488, expected=26.000000
predicted=24.454934, expected=32.000000
predicted=27.589481, expected=30.000000
predicted=30.009703, expected=26.000000
predicted=29.054758, expected=31.000000
predicted=27.797055, expected=38.000000
predicted=26.765646, expected=30.000000
predicted=25.623780, expected=22.000000
predicted=25.782154, expected=26.000000
predicted=28.593973, expected=24.000000
predicted=29.855815, expected=26.000000
predicted=30.216172, expected=30.000000
predicted=28.386865, expected=26.000000
predicted=25.966049, expected=21.000000
predicted=24.016095, expected=32.000000
predicted=25.354538, expected=50.000000
predicted=29.682084, expected=28.000000
predicted=31.222880, expected=28.000000
predicted=30.983385, expected=31.000000
predicted=30.100690, expected=38.000000
predicted=29.078938, expected=18.000000
predicted=26.740388, expected=23.000000
predicted=27.061104, expected=23.000000
predicted=28.273150, expected=31.000000
predicted=30.216498, expected=32.000000
predicted=30.760861, expected=27.000000
```

```
predicted=29.092167, expected=32.000000
predicted=27.376850, expected=16.000000
predicted=25.096592, expected=23.000000
predicted=26.135289, expected=29.000000
predicted=27.923094, expected=22.000000
predicted=29.075939, expected=38.000000
predicted=30.042912, expected=40.000000
predicted=29.660204, expected=36.000000
predicted=27.907223, expected=20.000000
predicted=26.068171, expected=26.000000
predicted=26.683383, expected=24.000000
predicted=28.538663, expected=37.000000
predicted=31.907527, expected=27.000000
predicted=30.966636, expected=35.000000
predicted=29.180499, expected=32.000000
predicted=27.038005, expected=27.000000
predicted=27.114331, expected=20.000000
predicted=27.309966, expected=28.000000
predicted=29.924245, expected=35.000000
predicted=31.616307, expected=25.000000
predicted=30.581541, expected=30.000000
predicted=28.729308, expected=26.000000
predicted=26.517540, expected=22.000000
predicted=24.998188, expected=18.000000
predicted=25.455944, expected=19.000000
predicted=27.298759, expected=25.000000
predicted=28.595551, expected=26.000000
predicted=28.303139, expected=35.000000
predicted=27.303675, expected=29.000000
predicted=24.672558, expected=20.000000
predicted=23.418478, expected=28.000000
predicted=25.390942, expected=28.000000
predicted=27.785753, expected=31.000000
predicted=29.666261, expected=25.000000
predicted=29.268383, expected=32.000000
predicted=27.840662, expected=32.000000
predicted=25.505138, expected=26.000000
predicted=24.758030, expected=18.000000
predicted=25.222340, expected=24.000000
predicted=27.519882, expected=21.000000
predicted=28.362037, expected=35.000000
predicted=29.730694, expected=29.000000
predicted=27.410630, expected=27.000000
predicted=24.598266, expected=19.000000
predicted=23.173292, expected=28.000000
predicted=25.682544, expected=26.000000
predicted=27.547293, expected=15.000000
predicted=28.214188, expected=30.000000
```

```
predicted=28.511802, expected=38.000000
predicted=26.877328, expected=26.000000
predicted=24.569911, expected=15.000000
predicted=22.002948, expected=21.000000
predicted=24.205396, expected=22.000000
predicted=26.047624, expected=26.000000
predicted=28.291282, expected=38.000000
predicted=28.721508, expected=26.000000
predicted=26.867292, expected=23.000000
predicted=23.518074, expected=12.000000
predicted=20.872683, expected=26.000000
predicted=23.527834, expected=24.000000
predicted=26.566789, expected=29.000000
predicted=28.328275, expected=28.000000
predicted=27.400075, expected=28.000000
predicted=25.275223, expected=30.000000
predicted=23.239090, expected=16.000000
predicted=21.647957, expected=20.000000
predicted=23.554000, expected=13.000000
predicted=25.272237, expected=26.000000
predicted=27.703868, expected=28.000000
predicted=26.770066, expected=27.000000
predicted=23.728456, expected=30.000000
predicted=21.887240, expected=22.000000
predicted=20.933478, expected=21.000000
predicted=22.687021, expected=20.000000
predicted=25.809871, expected=20.000000
predicted=27.556994, expected=28.000000
predicted=27.413917, expected=37.000000
predicted=25.448219, expected=24.000000
predicted=22.183907, expected=14.000000
predicted=19.937734, expected=18.000000
predicted=21.707481, expected=27.000000
predicted=25.913473, expected=23.000000
predicted=27.836438, expected=28.000000
predicted=27.472351, expected=30.000000
predicted=24.915137, expected=21.000000
predicted=20.474163, expected=12.000000
predicted=18.955107, expected=18.000000
predicted=20.761533, expected=15.000000
predicted=24.110770, expected=20.000000
predicted=25.482658, expected=19.000000
predicted=25.070614, expected=22.000000
predicted=21.019599, expected=19.000000
predicted=18.131126, expected=9.000000
predicted=16.172270, expected=23.000000
predicted=18.840923, expected=14.000000
predicted=22.046509, expected=24.000000
```

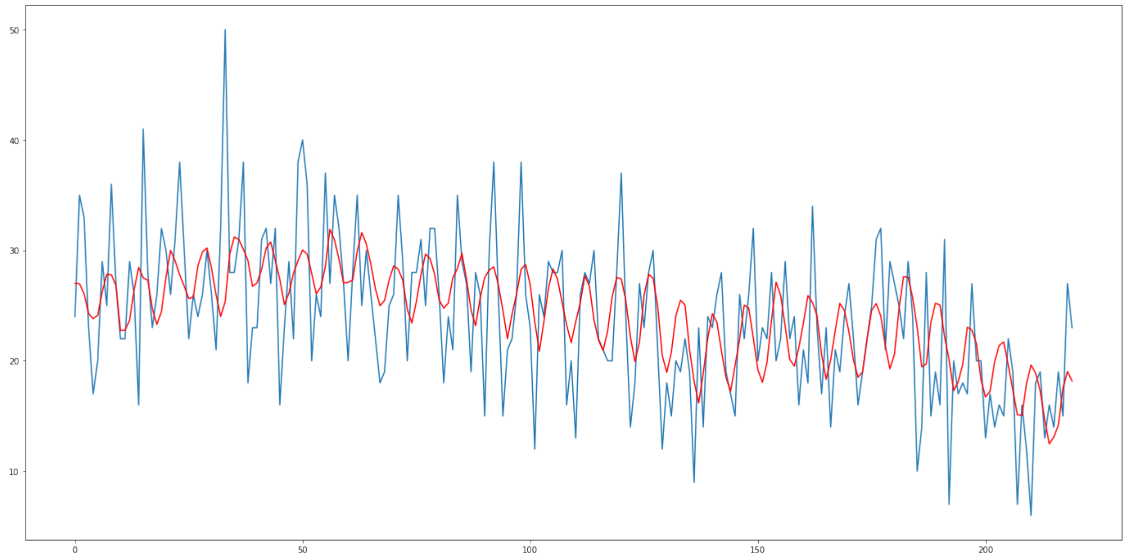
```
predicted=24.269391, expected=23.000000
predicted=23.497440, expected=26.000000
predicted=20.892565, expected=28.000000
predicted=18.501675, expected=19.000000
predicted=17.228505, expected=17.000000
predicted=19.593699, expected=15.000000
predicted=21.971182, expected=26.000000
predicted=25.053825, expected=22.000000
predicted=24.771190, expected=26.000000
predicted=22.041907, expected=32.000000
predicted=19.195112, expected=20.000000
predicted=18.051549, expected=23.000000
predicted=19.837799, expected=22.000000
predicted=23.674724, expected=28.000000
predicted=27.126687, expected=20.000000
predicted=25.925557, expected=22.000000
predicted=23.123457, expected=29.000000
predicted=20.131362, expected=22.000000
predicted=19.515732, expected=24.000000
predicted=21.185886, expected=16.000000
predicted=23.437882, expected=21.000000
predicted=25.903704, expected=18.000000
predicted=25.279438, expected=34.000000
predicted=24.076905, expected=23.000000
predicted=20.554705, expected=17.000000
predicted=18.322410, expected=23.000000
predicted=20.122828, expected=14.000000
predicted=22.744935, expected=21.000000
predicted=25.201545, expected=19.000000
predicted=24.564640, expected=24.000000
predicted=22.624039, expected=27.000000
predicted=20.060128, expected=22.000000
predicted=18.510852, expected=16.000000
predicted=19.006004, expected=19.000000
predicted=22.026822, expected=22.000000
predicted=24.583372, expected=25.000000
predicted=25.182608, expected=31.000000
predicted=24.052685, expected=32.000000
predicted=21.333531, expected=21.000000
predicted=19.274976, expected=29.000000
predicted=20.594117, expected=27.000000
predicted=24.722266, expected=25.000000
predicted=27.622496, expected=22.000000
predicted=27.578649, expected=29.000000
predicted=25.711318, expected=24.000000
predicted=22.989431, expected=10.000000
predicted=19.449577, expected=14.000000
predicted=19.726659, expected=28.000000
```

```
predicted=23.474602, expected=15.000000
predicted=25.220264, expected=19.000000
predicted=25.054728, expected=16.000000
predicted=22.206468, expected=31.000000
predicted=20.101903, expected=7.000000
predicted=17.293538, expected=20.000000
predicted=18.107455, expected=17.000000
predicted=19.692333, expected=18.000000
predicted=23.046986, expected=17.000000
predicted=22.776453, expected=27.000000
predicted=21.572036, expected=20.000000
predicted=18.305221, expected=20.000000
predicted=16.720953, expected=13.000000
predicted=17.250942, expected=17.000000
predicted=19.922252, expected=14.000000
predicted=21.390554, expected=16.000000
predicted=21.706402, expected=15.000000
predicted=19.606801, expected=22.000000
predicted=17.371238, expected=19.000000
predicted=15.112190, expected=7.000000
predicted=15.051775, expected=16.000000
predicted=17.922928, expected=12.000000
predicted=19.619478, expected=6.000000
predicted=18.883915, expected=18.000000
predicted=17.302115, expected=19.000000
predicted=14.650648, expected=13.000000
predicted=12.483057, expected=16.000000
predicted=13.091333, expected=14.000000
predicted=14.176811, expected=19.000000
predicted=17.492211, expected=15.000000
predicted=19.028952, expected=27.000000
predicted=18.181879, expected=23.000000
```

[38]:

```python
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
plt.figure(figsize=(24,12))
plt.plot(test)
plt.plot(predictions, color='red')
plt.show()
```

```
Test RMSE: 5.558
```



[39]:
```
from sklearn.metrics import mean_absolute_error
print('Mean Absolute Error:',mean_absolute_error(test.reshape(-1),predictions))
```

```
Mean Absolute Error: 4.300249909904669
```

[40]:
```
df_pred=pd.DataFrame({'Predictions':predictions},index=test_date)
```

[41]:
```
df_pred
```

[41]:

date	Predictions
26-05-2017	27.014689
27-05-2017	26.981263
28-05-2017	26.085309
29-05-2017	24.286540
30-05-2017	23.824463
…	…
27-12-2017	13.091333
28-12-2017	14.176811
29-12-2017	17.492211
30-12-2017	19.028952
31-12-2017	18.181879

[220 rows x 1 columns]