# Prática_6_respostas

August 19, 2020

## 1 Prática 6

***Aprendizado Dinâmico***

por **Cibele Russo** (ICMC/USP - São Carlos SP)

**MBA em Ciências de Dados**

Nesta prática vamos comparar o Método Theta simples de Fiorucci et al. (2016) com o Método de Holt, visto na Aula 2, para previsões dos 30 próximos dias para os dados PETR4. Repita com janelas menores como 20 ou 10 dias.

**1. Defina a função sThetaF como vista em aula. É ela que usaremos neste exercício.**

[50]:
```
## Implementação

# Fonte: https://github.com/MinhDg00/theta/tree/master/src
# Baseado em Fiorucci et. al (2016) https://github.com/cran/forecTheta

# Implement standard Theta method based on Fiorucci et al. (2016)
# Reference: https://github.com/cran/forecTheta

import sys
import numpy as np
import pandas as pd
import statsmodels as sm
import warnings
from scipy.stats import norm
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from sklearn.linear_model import LinearRegression


def sThetaF(y, s_period = 1, h = 10, s = None):
        """
        @param y : array-like time series data
        @param s_period : the no. of observations before seasonal pattern repeats
        @param h : number of period for forcasting
```

```
        @s : additive or multiplicative
        """
        fcast = {} # store result
        n = y.index.size
        x = y.copy()
        m = s_period
        time_y = np.array(np.arange(n))/m + 1
        time_fc = time_y[n-1] + np.array(np.arange(1,h+1))/m

        s_type = 'multiplicative'
        if s is not None:
                if s == 'additive':
                        s = True
                        s_type = 'additive'

        # Seasonality Test & Decomposition
        if s is not None and m >= 4:
                r = (acf(x, nlags = m+1))[1:]
                clim = 1.64/sqrt(n) * np.sqrt(np.cumsum([1, 2 * np.square(r)]))
                s = abs(r[m-1]) > clim[m-1]
        else:
                if not s:
                        s = False


        if s:
                decomp = seasonal_decompose(x, model = s_type)
                if s_type == 'additive' or (s_type -- 'multiplicative' and␣
↪any(decomp < 0.01)):
                        s_type = 'additive'
                        decomp = seasonal_decompose(x, model = 'additive').
↪seasonal
                        x = x - decomp
                else:
                        x = x/decomp


        ## Find Theta Line
        model = LinearRegression().fit(time_y.reshape(-1,1), x)
        fcast['mean'] = model.intercept_ + model.coef_ * time_fc


        return fcast
```

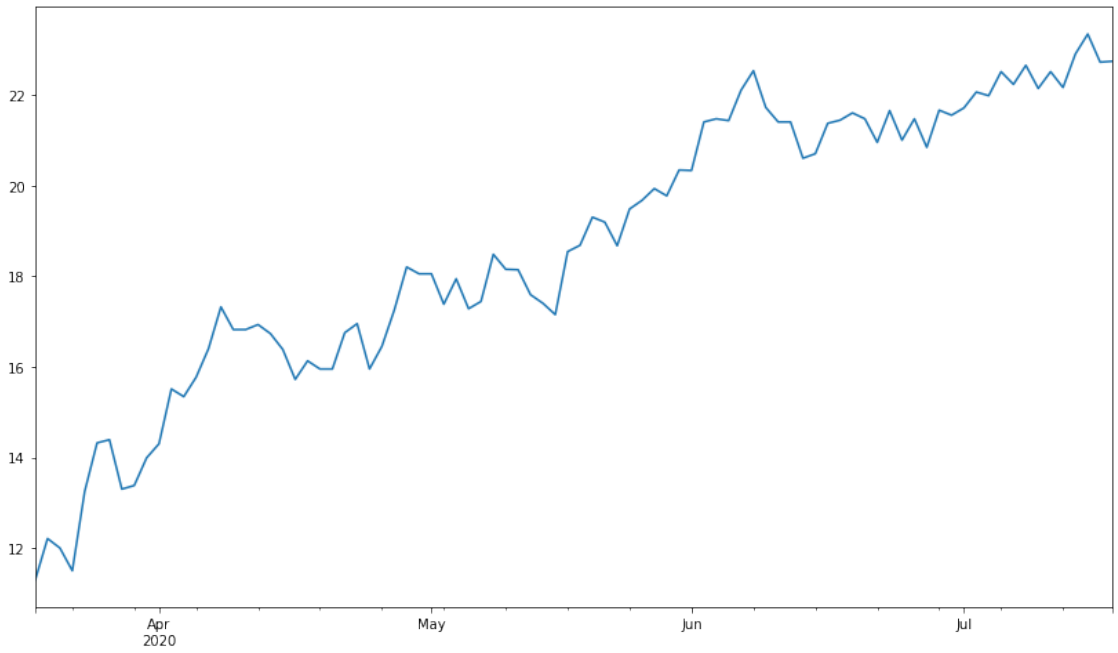
**2. Carregue as bibliotecas e faça a leitura dos dados PETR4.**

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

pkgdir = '/home/cibele/CibelePython/AprendizadoDinamico/Data'

# PETR4 - Leitura dos dados
df = pd.read_csv(f'{pkgdir}/PETR4.csv', index_col='Date', parse_dates=True)


idx = pd.date_range(start=df.index.min(), end=df.index.max(), freq='B')
df = df.reindex(idx)
df.fillna(method='ffill', inplace=True)
```

**3. Faça um gráfico da série.**

[2]:
```
plt.rcParams['figure.figsize'] = [14,8]


df['Close'].plot()
```

[2]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd1d1eeec90>




**4. Defina bases de treino e teste, deixando 30 observações para a base de teste.**

[6]:
```
treino = df.iloc[:-30]
teste = df.iloc[-30:]
```

**5. Prepare os dados para utilizar a função SThetaF.**

[7]:
```
data = pd.Series(treino['Close'], df.index)
```

**6. Obtenha as previsões pelo Método Theta de Fiorucci (2016) com a função sThetaF.**

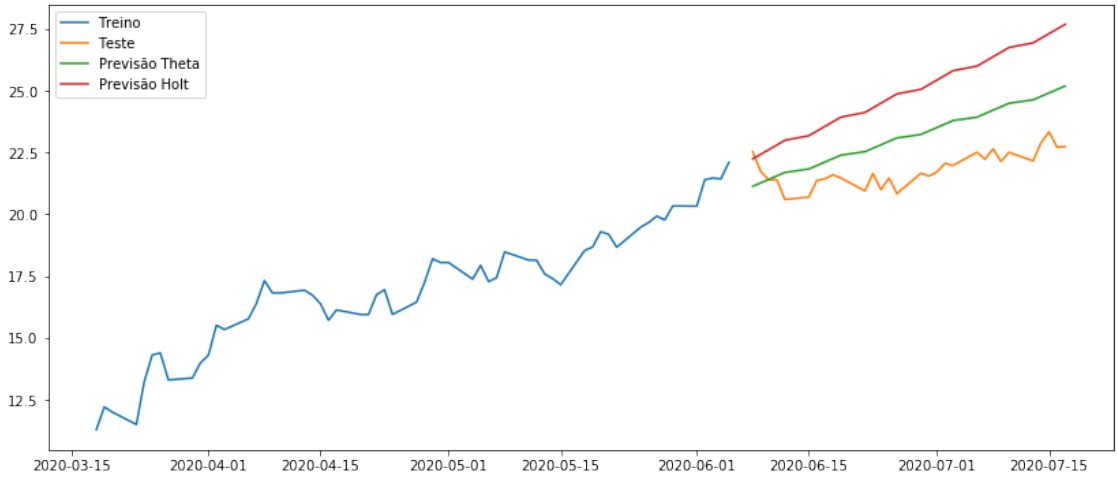
[63]:
```
from datetime import timedelta

data = pd.Series(treino['Close'], treino.index)

modelo = pd.DataFrame()

modelo['mean'] = sThetaF(data, h = 30)['mean']

modelo.index = pd.date_range(start=treino.index.
 ↪max()+timedelta(days=1),periods=30,freq='B')
```

**7. Aplique o Método de Holt.**

[64]:
```
# Método de Holt

from statsmodels.tsa.api import ExponentialSmoothing

modelo_H =  ExponentialSmoothing(treino['Close'], trend='add');

ajustado = modelo_H.fit();

df['Holt'] = ajustado.fittedvalues.shift(-1);

predito_H = ajustado.forecast(30).rename('Previsão Holt')
```

**8. Represente graficamente os resultados, com as bases de treino e teste.**

[62]:
```
mean = modelo['mean']

plt.figure(figsize = (14,6))

plt.plot(treino['Close'], label='Treino')
plt.plot(teste['Close'], label='Teste')
plt.plot(mean, label='Previsão Theta')
plt.plot(predito_H, label='Previsão Holt')

plt.legend()
plt.show()
```




**9. Utilize o EQM para comparar as previsões.**

[14]:
```
from sklearn.metrics import mean_squared_error

error = mean_squared_error(teste['Close'], modelo['mean'])
print(f'EQM Theta: {error:11.10}')


error = mean_squared_error(teste['Close'], predito_H)
print(f'EQM Holt: {error:11.10}')
```

```
EQM Theta: 2.531271192
EQM Holt: 11.34486653
```

**10. Mude o tamanho das bases de treino e teste, para por exemplo 10 ou 20 observações na amostra de teste. Os resultados são similares?**

**11. Repita os procedimentos para obter previsões para dados futuros, ou seja, utilizando todos os dados disponíveis e fazendo previsões para os próximos 10 dias.**

[16]:
```
data = pd.Series(df['Close'], df.index)
```

[18]:
```
from datetime import timedelta

data = pd.Series(df['Close'], df.index)

modelo = pd.DataFrame()

modelo['mean'] = sThetaF(data,  h = 10)['mean']

modelo.index = pd.date_range(start=df.index.
 ↪max()+timedelta(days=1),periods=10,freq='B')
```

[19]:
```
# Método de Holt

from statsmodels.tsa.api import ExponentialSmoothing

modelo_H =  ExponentialSmoothing(df['Close'], trend='add');

ajustado = modelo_H.fit();

df['Holt'] = ajustado.fittedvalues.shift(-1);

predito_H = ajustado.forecast(10).rename('Previsão Holt')
```

[20]:
```
mean = modelo['mean']

plt.figure(figsize = (14,6))

plt.plot( df['Close'], label='Dados observados')
plt.plot( modelo['mean'], label='Previsão Theta')
plt.plot(predito_H, label='Previsão Holt')

plt.legend()
plt.show()
```