Aula2_1_HoltWinters

July 15, 2020

1 Métodos de Suavização de Holt e Holt-Winters

por **Cibele Russo**

Baseado em

- Moretting, P.A.; Toloi, C.M.C. "Análise de Séries Temporais". Blucher, 2004.
- Ehlers, R.S. (2009) Análise de Séries Temporais, http://www.icmc.usp.br/~ehlers/stemp/stemp.pdf. Acessado em 28/06/2020.

Métodos de Holt e Holt-Winters: - Metcalfe, Andrew V., and Cowpertwait, Paul S.P. Introductory time series with R. Springer-Verlag New York, 2009. - Holt, C. (1957). Forecasting trends and seasonals by exponentially weighted averages. O.W.R. Memorandum no.52, Carregie Institute of Technology. - Winters, P. (1960). Forecasting sales by exponentially weighted moving averages. Management Science, 6:324–342.

Implementações: - Brownlee, Jason. Introduction to time series forecasting with python: how to prepare data and develop models to predict the future. Machine Learning Mastery, 2017. - https://www.statsmodels.org/devel/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html

- https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python/

1.1 MMS e MMEP

Na aula anterior, vimos métodos de suavização médias móveis e médias móveis exponencialmente ponderadas considerando a seguinte decomposição de uma série temporal $\{Z_t\}$

$$Z_t = \mu_t + a_t, \text{ para } t = 1, \ldots, N$$

com $E(a_t) = 0$ e $Var(a_t) = \sigma^2$ e μ_t um parâmetro desconhecido.

Esses métodos não levam em consideração que a série tem uma componente de tendência.

São técnicas descritivas não aconselháveis para fazer previsão.

Queremos estimar a tendência e a sazonalidade, e com isso já podemos fazer previsões!

Considere inicialmente que a série temporal seja composta localmente por soma de nível, tendência e erro aleatório (ruído), com média zero e variância constante σ_a^2, isto é

$$Z_t = \mu_t + T_t + a_t, \text{ com } t = 1, \ldots, N.$$

1.2 Método de Holt

Parecido com a MMEP, porém utiliza uma nova constante para modelar a **tendência**

Os valores do nível e da tendência da série, no instante t, serão estimados por

$\overline{Z}_t = AZ_t + (1-A)(\overline{Z}_{t-1} + \widehat{T}_{t-1})$, $0 < A < 1$ e $t = 2, \ldots, N$

$\widehat{T}_t = C(\overline{Z}_t - \overline{Z}_{t-1}) + (1-C)\widehat{T}_{t-1}$, $0 < C < 1$ e $t = 2, \ldots, N$,

respectivamente, A e C são denominadas constantes de suavização.

1.2.1 Previsão

A previsão para o valor Z_{t+h} com origem em t é dada por

$\widehat{Z}_t(h) = \overline{Z}_t + h\widehat{T}_t$, $\forall h > 0$

ou seja, adiciona-se ao valor básico $\overline{Z}_t$ a tendência multiplicada pelo número de passos à frente que se deseja prever (h).

Assim, para predizer a observação Z_{t+1}, podemos fazer

$\overline{Z}_{t+1} = AZ_{t+1} + (1-A)(\overline{Z}_t + \widehat{T}_t)$,

$\widehat{T}_{t+1} = C(\overline{Z}_{t+1} - \overline{Z}_t) + (1-C)\widehat{T}_t$

e a nova previsão do valor Z_{t+h} será

$$\widehat{Z}_{t+1}(h-1) = \overline{Z}_{t+1} + (h-1)\widehat{T}_{t+1}.$$

Precisamos assumir hipóteses sobre os valores iniciais, por exemplo, assumir que $\widehat{T}_2 = Z_2 - Z_1$ e $\overline{Z}_2 = Z_2$. Se Z_t for gerado por um processo $ARIMA(0,2,2)$, essa previsão será ótima.

As constantes A e C são escolhidas de forma que a soma dos erros quadráticos de previsão seja mínimo.

1.3 Método de Holt-Winters

Para séries com padrão de comportamento mais complexo, existem outras formas de suavização, tais com os métodos de Holt-Winters.

Considere uma série sazonal com período s.

1.3.1 Modelo com sazonalidade multiplicativa e tendência aditiva

$$Z_t = \mu_t F_t + T_t + a_t, \text{ com } t = 1, \ldots, N.$$

As três equações de suavização são dadas por

$\widehat{F}_t = D\left(\frac{Z_t}{\overline{Z}_t}\right) + (1-D)\widehat{F}_{t-s}, 0 < D < 1, t = s+1, \ldots, N$

$\overline{Z}_t = A\left(\frac{Z_t}{\widehat{F}_{t-s}}\right) + (1-A)(\overline{Z}_{t-1} + \widehat{T}_{t-1})$, $0 < A < 1$ e $t = s+1, \ldots, N$

$\widehat{T}_t = C(\overline{Z}_t - \overline{Z}_{t-1}) + (1-C)\widehat{T}_{t-1},\ 0 < C < 1$ e $t = s+1, \ldots, N$,

e representam estimativas do fator sazonal, do nível e da tendência, respectivamente, e A, C e D são constantes de suavização.

1.3.2 Previsão

$\widehat{Z}_t(h) = (\overline{Z}_t + h\widehat{T}_t)\widehat{F}_{t+h-s}, h = 1, 2, \ldots, s$

$\widehat{Z}_t(h) = (\overline{Z}_t + h\widehat{T}_t)\widehat{F}_{t+h-2s}, h = s+1, \ldots, 2s$

$\vdots \quad = \quad \vdots$

onde $\overline{Z}_t$, $\widehat{F}_t$ e $\widehat{T}_t$ são obtidos das equações do método.

Para fazermos atualizações das previsões, quando temos uma nova observação Z_{t+1}, utilizamos os valores

$$\widehat{F}_{t+1} = D\left(\frac{Z_{t+1}}{\overline{Z}_{t+1}}\right) + (1-D)\widehat{F}_{t+1-s},$$

$$\overline{Z}_{t+1} = A\left(\frac{Z_{t+1}}{\widehat{F}_{t+1-s}}\right) + (1-A)(\overline{Z}_t + \widehat{T}_t),$$

$$\widehat{T}_{t+1} = C(\overline{Z}_{t+1} - \overline{Z}_t) + (1-C)\widehat{T}_t,$$

e a nova previsão para a observação Z_{t+h} será

$\widehat{Z}_{t+1}(h-1) = (\overline{Z}_{t+1} + (h-1)\widehat{T}_{t+1})\widehat{F}_{t+1+h-s}, h = 1, 2, \ldots, s+1$

$\widehat{Z}_{t+1}(h-1) = (\overline{Z}_{t+1} + (h-1)\widehat{T}_{t+1})\widehat{F}_{t+1+h-2s}, h = s+2, \ldots, 2s+1$

Também é necessário fazer suposições sobre os valores iniciais para o processo.

1.3.3 Modelo com sazonalidade e tendência aditivas

$$Z_t = \mu_t + T_t + F_t + a_t, \text{ com } t = 1, \ldots, N.$$

As estimativas do fator sazonal, do nível e da tendência da série são dadas por

$\widehat{F}_t = D(Z_t - \overline{Z}_t) + (1-D)\widehat{F}_{t-s}, 0 < D < 1, t = s+1, \ldots, N$

$\overline{Z}_t = A(Z_t - \widehat{F}_{t-s}) + (1-A)(\overline{Z}_{t-1} + \widehat{T}_{t-1}),\ 0 < A < 1$ e $t = s+1, \ldots, N$

$\widehat{T}_t = C(\overline{Z}_t - \overline{Z}_{t-1}) + (1-C)\widehat{T}_{t-1},\ 0 < C < 1$ e $t = s+1, \ldots, N$,

respectivamente, e A, C e D são constantes de suavização e são estimadas de forma que minimizem a soma dos quadrados dos erros dos ajustes.

1.3.4 Previsão

$\widehat{Z}_t(h) = \overline{Z}_t + h\widehat{T}_t + \widehat{F}_{t+h-s}, h = 1, 2, \ldots, s$

$\widehat{Z}_t(h) = \overline{Z}_t + h\widehat{T}_t + \widehat{F}_{t+h-2s}, h = s+1, \ldots, 2s$

$\vdots \quad = \quad \vdots$

onde $\overline{Z}_t$, $\widehat{F}_t$ e $\widehat{T}_t$ são obtidos das equações do método.

Para fazermos atualizações das previsões, quando temos uma nova observação Z_{t+1}, utilizamos os valores

$\$\widehat{F}\{t+1\} = D\ (Z\{t+1\} - \overline{Z}$

1.3.5 Média móvel simples

- Simples implementação;
- Aplicável mesmo com um número pequeno de observações;
- Flexibilidade de acordo com o tamanho da janela

Porém

- Método descritivo e não recomendável para previsões
- Não leva em consideração a componente de tendência na série

1.3.6 Média móvel exponencialmente ponderada

- Fácil compreensão e aplicabilidade;
- Necessidade de armazenar somente Z_t, $\overline{Z}_t$ e α

Porém

- Método descritivo e não recomendável para previsões
- Não leva em consideração a componente de tendência na série

1.3.7 Método de Holt

- Pode ser usado para prever séries que tem tendência
- Apresenta um nível maior de dificuldade para encontrar os valores apropriados para as constantes de suavização A e C, em geral que minimizem a soma dos quadrados dos erros dos ajustes

1.3.8 Método de Holt-Winters

- Pode ser usado para prever séries que tem tendência e sazonalidade, seja ela aditiva ou multiplicativa

Porém

- Impossibilidade e dificuldade para estudar propriedades estatísticas como média e variância das previsões, e consequentemente construção de intervalos de confiança para as previsões
- Nível maior de dificuldade para encontrar os valores apropriados para as constantes de suavização A, C e D, em geral que minimizem a soma dos quadrados dos erros dos ajustes

1.3.9 Aplicação: dados de passageiros aéreos

[1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

[2]:
```
pkgdir = '/home/cibele/CibelePython/AprendizadoDinamico/Data'

passageiros = pd.read_csv(f'{pkgdir}/airline_passengers.csv', index_col=0,␣
 ↪parse_dates=True)

passageiros.head()
```

[2]:
```
            Milhares de passageiros
Mês
1949-01-01                      112
1949-02-01                      118
1949-03-01                      132
1949-04-01                      129
1949-05-01                      121
```

[3]:
```
passageiros.dropna(inplace=True)
```

[4]:
```
passageiros.index
```

[4]:
```
DatetimeIndex(['1949-01-01', '1949-02-01', '1949-03-01', '1949-04-01',
               '1949-05-01', '1949-06-01', '1949-07-01', '1949-08-01',
               '1949-09-01', '1949-10-01',
               ...
               '1960-03-01', '1960-04-01', '1960-05-01', '1960-06-01',
               '1960-07-01', '1960-08-01', '1960-09-01', '1960-10-01',
               '1960-11-01', '1960-12-01'],
              dtype='datetime64[ns]', name='Mês', length=144, freq=None)
```

Estabeleça a frequência do DatetimeIndex!

Para construir um modelo de suavização Holt-Winters, o pacote statsmodels precisa saber a frequência dos dados. Para os dados de passageiros aéreos, as observações são coletadas no início do mês, usaremos MS.

Veja aqui uma lista de possibilidades.

[5]:
```
passageiros.index.freq = 'MS'
passageiros.index
```

[5]:
```
DatetimeIndex(['1949-01-01', '1949-02-01', '1949-03-01', '1949-04-01',
               '1949-05-01', '1949-06-01', '1949-07-01', '1949-08-01',
               '1949-09-01', '1949-10-01',
```

```
       ...
       '1960-03-01', '1960-04-01', '1960-05-01', '1960-06-01',
       '1960-07-01', '1960-08-01', '1960-09-01', '1960-10-01',
       '1960-11-01', '1960-12-01'],
      dtype='datetime64[ns]', name='Mês', length=144, freq='MS')
```

1.4 Suavização exponencial simples

Uma variação da função Holt-Winters de statsmodels apresenta a Suavização exponencial simples, que fornece os mesmos cálculos da média móvel exponencialmente ponderada do método .ewm do pandas que vimos na última aula.

[6]:
```
# É possível ajustar a MMEP usando o pacote statsmodels.tsa.holtwinters e a␣
↪função SimpleExpSmoothing

from statsmodels.tsa.holtwinters import SimpleExpSmoothing

span = 12
alpha = 2/(span+1)

passageiros['MMEP12'] = passageiros['Milhares de passageiros'].
↪ewm(alpha=alpha,adjust=False).mean()
passageiros['SES12'] = SimpleExpSmoothing(passageiros['Milhares de␣
↪passageiros']).fit(smoothing_level=alpha,optimized=False).fittedvalues.
↪shift(-1)
passageiros.head()
```

[6]:

Mês	Milhares de passageiros	MMEP12	SES12
1949-01-01	112	112.000000	112.000000
1949-02-01	118	112.923077	112.923077
1949-03-01	132	115.857988	115.857988
1949-04-01	129	117.879836	117.879836
1949-05-01	121	118.359861	118.359861

1.5 Método de Holt

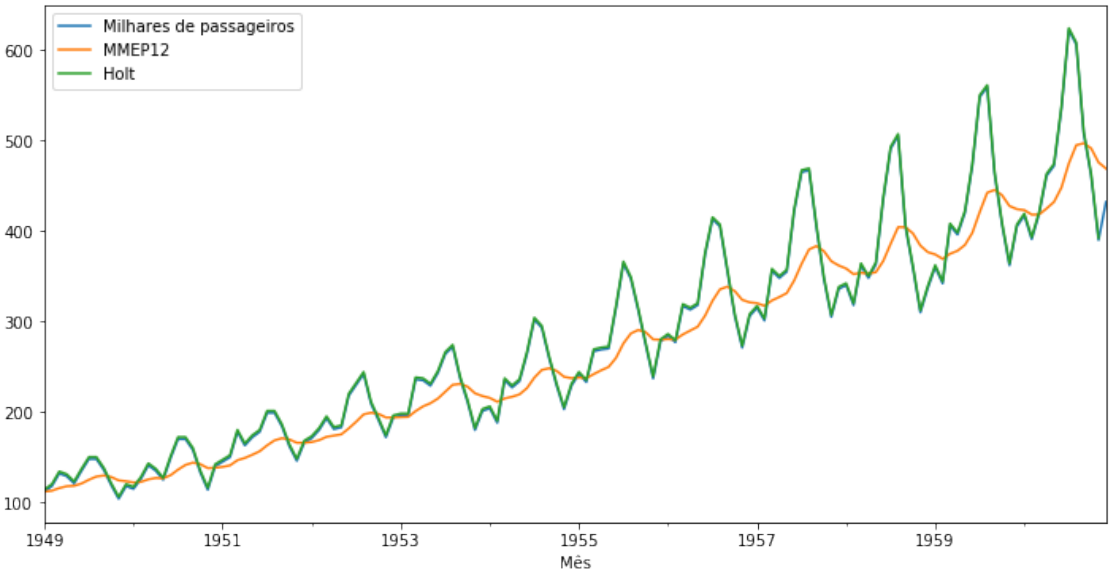
[7]:
```
# Método de Holt

from statsmodels.tsa.api import ExponentialSmoothing


modelo =  ExponentialSmoothing(passageiros['Milhares de passageiros'],␣
↪trend='add');
```

```
ajustado = modelo.fit();

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

passageiros.head()
```

[7]:

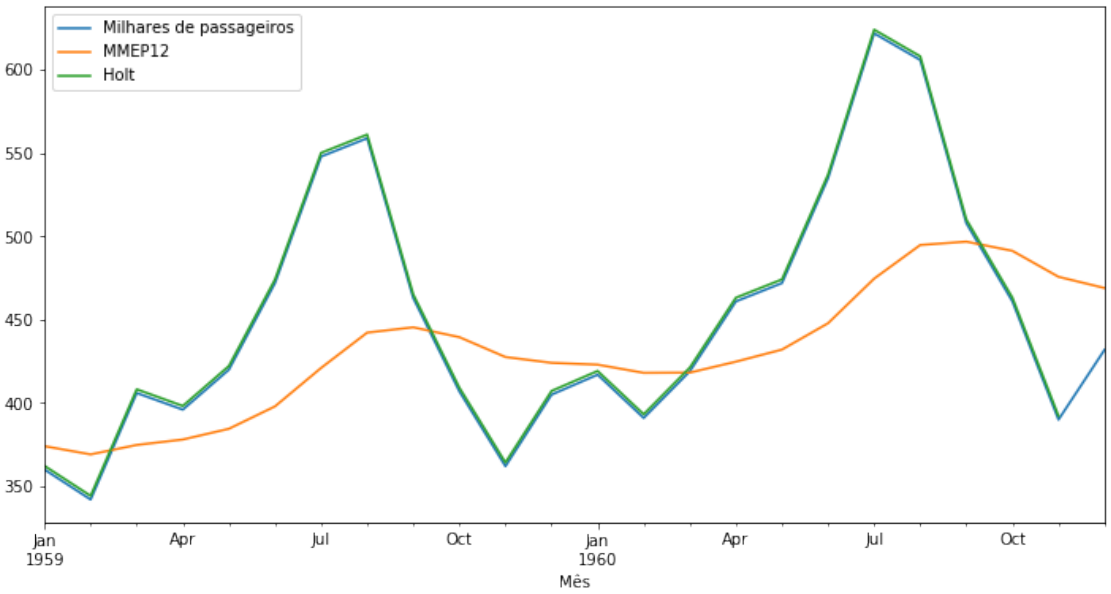
Mês	Milhares de passageiros	MMEP12	SES12	Holt
1949-01-01	112	112.000000	112.000000	114.237764
1949-02-01	118	112.923077	112.923077	120.237764
1949-03-01	132	115.857988	115.857988	134.237764
1949-04-01	129	117.879836	117.879836	131.237764
1949-05-01	121	118.359861	118.359861	123.237764

[8]:
```
passageiros[['Milhares de passageiros','MMEP12','Holt']].plot(figsize=(12,6)).
↪autoscale(axis='x',tight=True);
```



Podemos notar que o método de Holt já apresenta um ajuste bem melhor do que a MMEP!

[9]:
```
passageiros[['Milhares de passageiros','MMEP12','Holt']].iloc[-24:].
↪plot(figsize=(12,6)).autoscale(axis='x',tight=True);
```



1.6 Método de Holt-Winters - sazonalidade aditiva ou multiplicativa

[10]:
```
# Ajuste do modelo pelo Método de Holt-Winters com sazonalidade aditiva

modelo =  ExponentialSmoothing(passageiros['Milhares de␣
 ↪passageiros'],trend='add',seasonal='add',seasonal_periods=12);

ajustado = modelo.fit();

passageiros['Holt-Winters-adit-12'] = ajustado.fittedvalues;

passageiros.head()
```

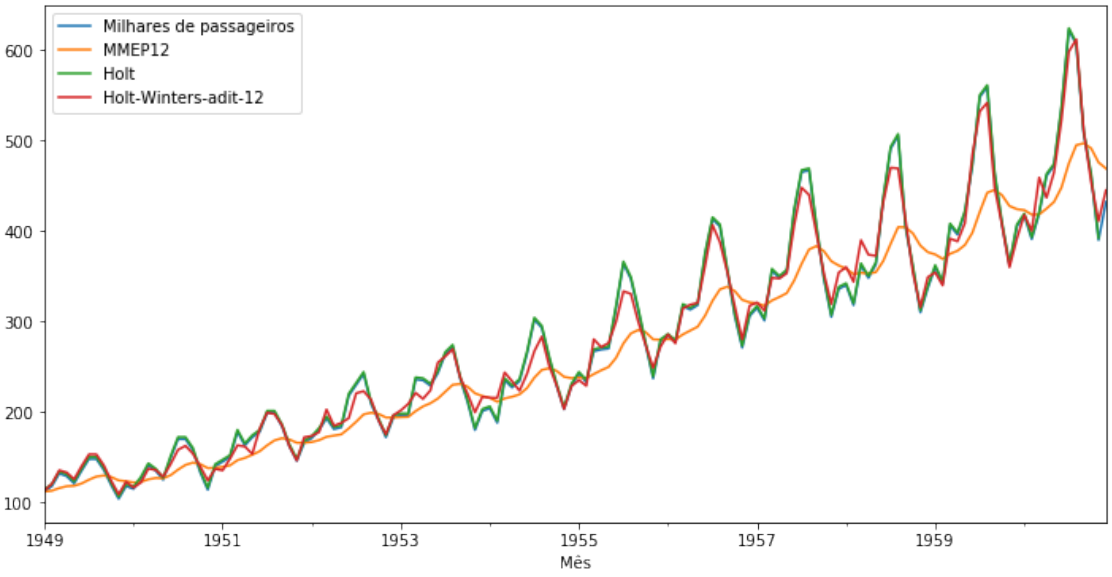
```
/home/cibele/anaconda3/lib/python3.7/site-
packages/statsmodels/tsa/holtwinters.py:744: ConvergenceWarning: Optimization
failed to converge. Check mle_retvals.
  ConvergenceWarning)
```

[10]:

Mês	Milhares de passageiros	MMEP12	SES12	Holt	Holt-Winters-adit-12
1949-01-01	112	112.000000	112.000000	114.237764	
1949-02-01	118	112.923077	112.923077	120.237764	
1949-03-01	132	115.857988	115.857988	134.237764	
1949-04-01	129	117.879836	117.879836	131.237764	
1949-05-01	121	118.359861	118.359861	123.237764	

```
Mês
1949-01-01            113.081288
1949-02-01            120.550747
1949-03-01            135.527329
1949-04-01            133.155064
1949-05-01            125.656114
```

[11]:
```
passageiros[['Milhares de passageiros','MMEP12','Holt', 'Holt-Winters-adit-12']].
 ↪plot(figsize=(12,6)).autoscale(axis='x',tight=True);
```

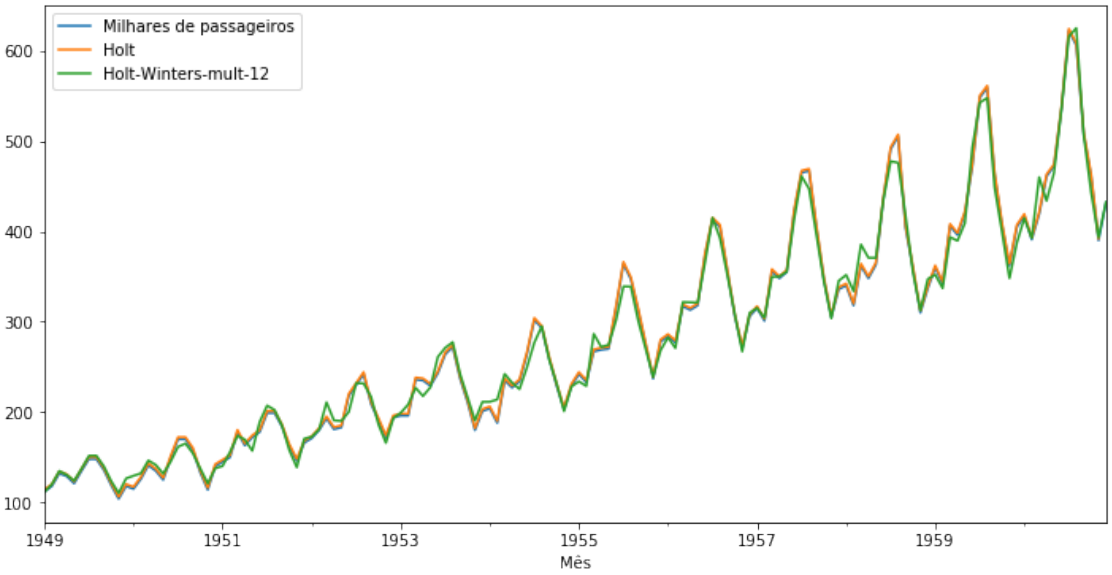


[12]:
```
# Ajuste do modelo pelo Método de Holt-Winters com sazonalidade multiplicativa

modelo =  ExponentialSmoothing(passageiros['Milhares de␣
 ↪passageiros'],trend='add',seasonal='mul',seasonal_periods=12)

ajustado = modelo.fit()

passageiros['Holt-Winters-mult-12'] = ajustado.fittedvalues

passageiros.head()
```

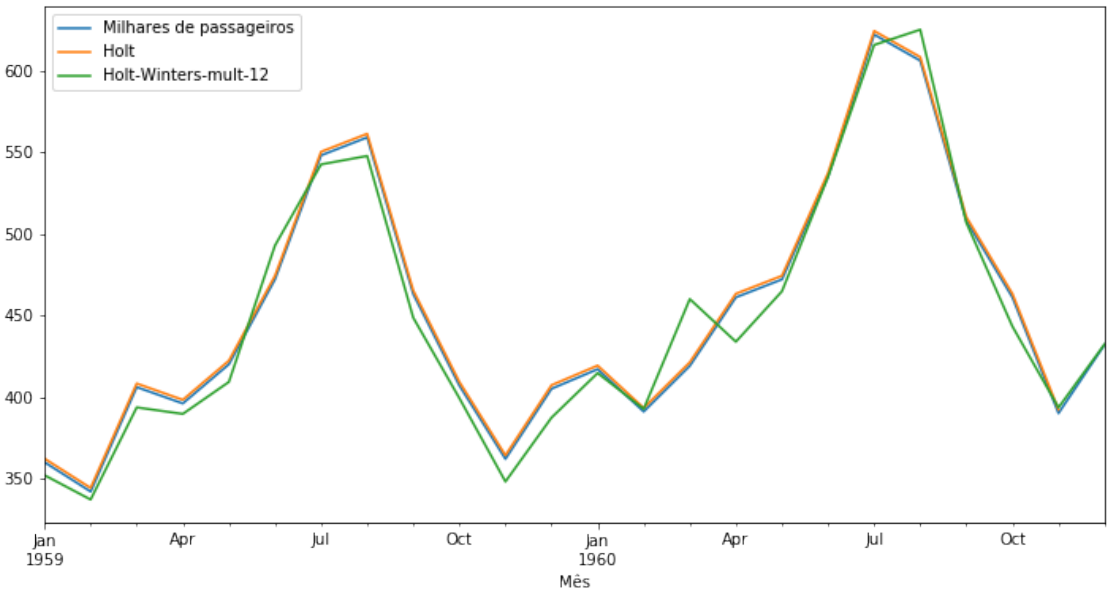
[12]:
```
            Milhares de passageiros      MMEP12       SES12        Holt  \
Mês
1949-01-01                      112  112.000000  112.000000  114.237764
1949-02-01                      118  112.923077  112.923077  120.237764
1949-03-01                      132  115.857988  115.857988  134.237764
1949-04-01                      129  117.879836  117.879836  131.237764
1949-05-01                      121  118.359861  118.359861  123.237764
```

Mês	Holt-Winters-adit-12	Holt-Winters-mult-12
1949-01-01	113.081288	111.419153
1949-02-01	120.550747	120.062406
1949-03-01	135.527329	134.827449
1949-04-01	133.155064	130.592737
1949-05-01	125.656114	124.085854

[13]:
```
passageiros[['Milhares de passageiros','Holt','Holt-Winters-mult-12']].
↪plot(figsize=(12,6)).autoscale(axis='x',tight=True);
```



[14]:
```
## Olhando somente para os últimos dois anos

passageiros[['Milhares de passageiros','Holt','Holt-Winters-mult-12']].iloc[-24:
↪].plot(figsize=(12,6)).autoscale(axis='x',tight=True);
```



Aparentemente, o método de Holt está performando melhor para esses dados.

Vamos ver como se comportam na predição de observações futuras?

[15]:
```
len(passageiros)
```

[15]:
```
144
```

1.7 Previsão com o método Holt-Winters

Neste exemplo, usaremos o mesmo conjunto de dados airlines_passengers e os dividiremos em 108 registros de treinamento e 36 registros de teste. Em seguida, avaliaremos o desempenho do modelo.

[16]:
```
import pandas as pd
import numpy as np
%matplotlib inline

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


passageiros = pd.read_csv(f'{pkgdir}/airline_passengers.
 ↪csv',index_col='Mês',parse_dates=True)
passageiros.index.freq = 'MS'
passageiros.head()
```

[16]:
```
            Milhares de passageiros
Mês
```

```
1949-01-01                     112
1949-02-01                     118
1949-03-01                     132
1949-04-01                     129
1949-05-01                     121
```

[17]:
```
passageiros.tail()
```

[17]:
```
            Milhares de passageiros
Mês
1960-08-01                     606
1960-09-01                     508
1960-10-01                     461
1960-11-01                     390
1960-12-01                     432
```

[18]:
```
passageiros.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 144 entries, 1949-01-01 to 1960-12-01
Freq: MS
Data columns (total 1 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Milhares de passageiros  144 non-null    int64
dtypes: int64(1)
memory usage: 2.2 KB
```

1.8 Divisão da base em treino e teste

[20]:
```
dados_treino = passageiros.iloc[:108] # Dados de treinamento até observação 108,␣
 ↪sem incluí-la
dados_teste = passageiros.iloc[108:] # Dados de teste a partir da observação 108
```

1.9 Ajuste do modelo

[21]:
```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

ajustado_HW = ExponentialSmoothing(dados_treino['Milhares de␣
 ↪passageiros'],trend='add',seasonal='mul',seasonal_periods=12).fit()
```

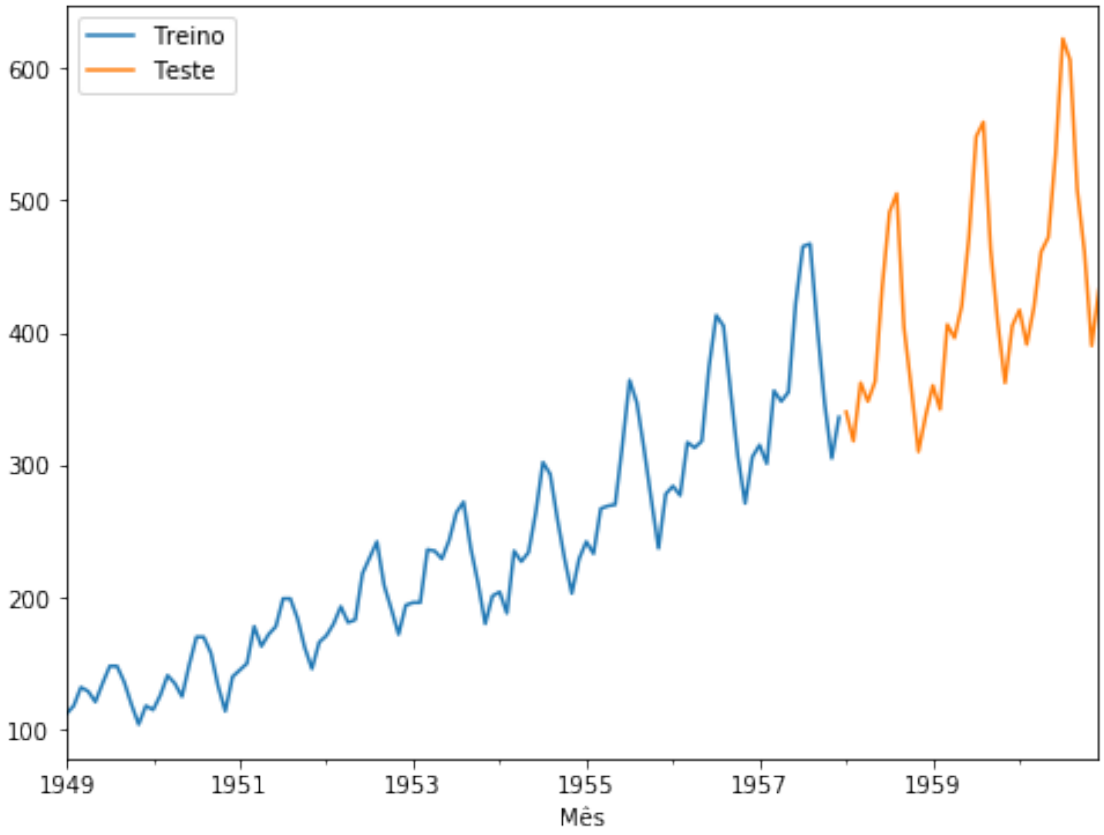
1.10 Avaliação do modelo com predições para dados de teste

[22]:
```
predito_HW = ajustado_HW.forecast(36).rename('Previsão Holt-Winters')
```

[23]:
```
predito_HW.head()
```

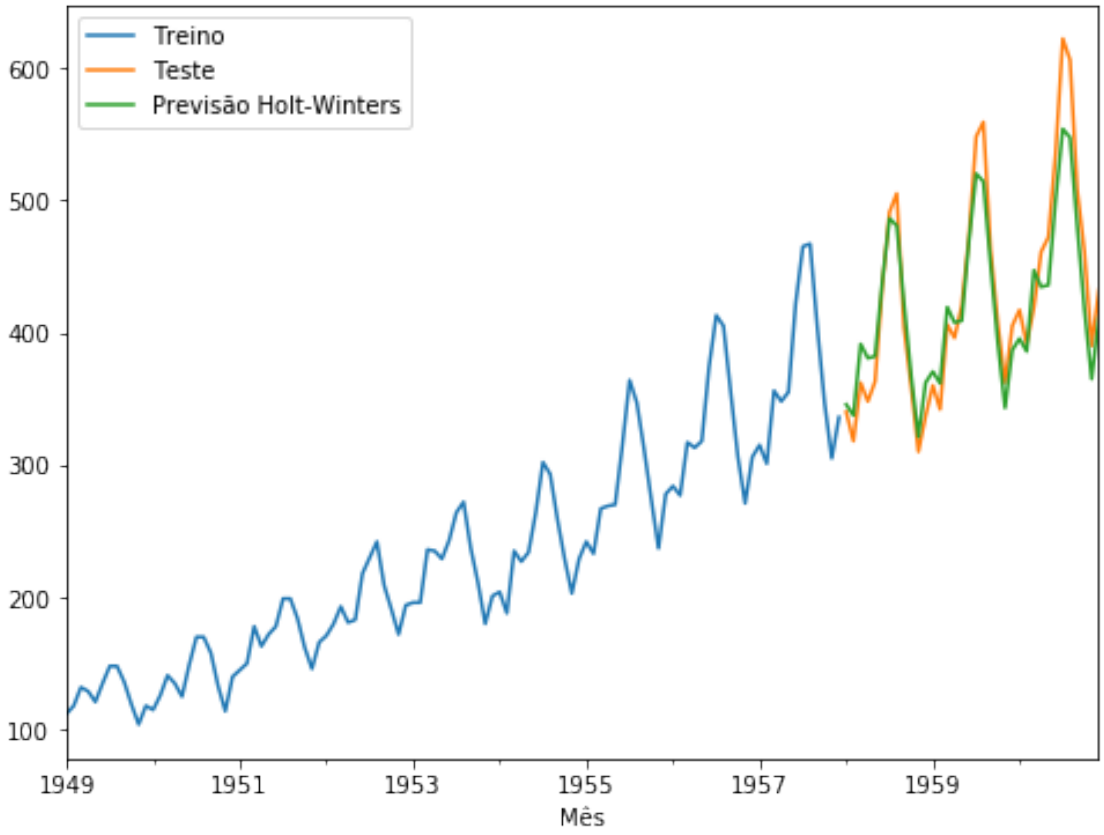
[23]:
```
1958-01-01    345.597476
1958-02-01    337.457193
1958-03-01    391.275555
1958-04-01    380.635131
1958-05-01    382.070390
Freq: MS, Name: Previsão Holt-Winters, dtype: float64
```

[25]:
```
dados_treino['Milhares de passageiros'].plot(legend=True,label='Treino')
dados_teste['Milhares de passageiros'].
 ↪plot(legend=True,label='Teste',figsize=(8,6));
```



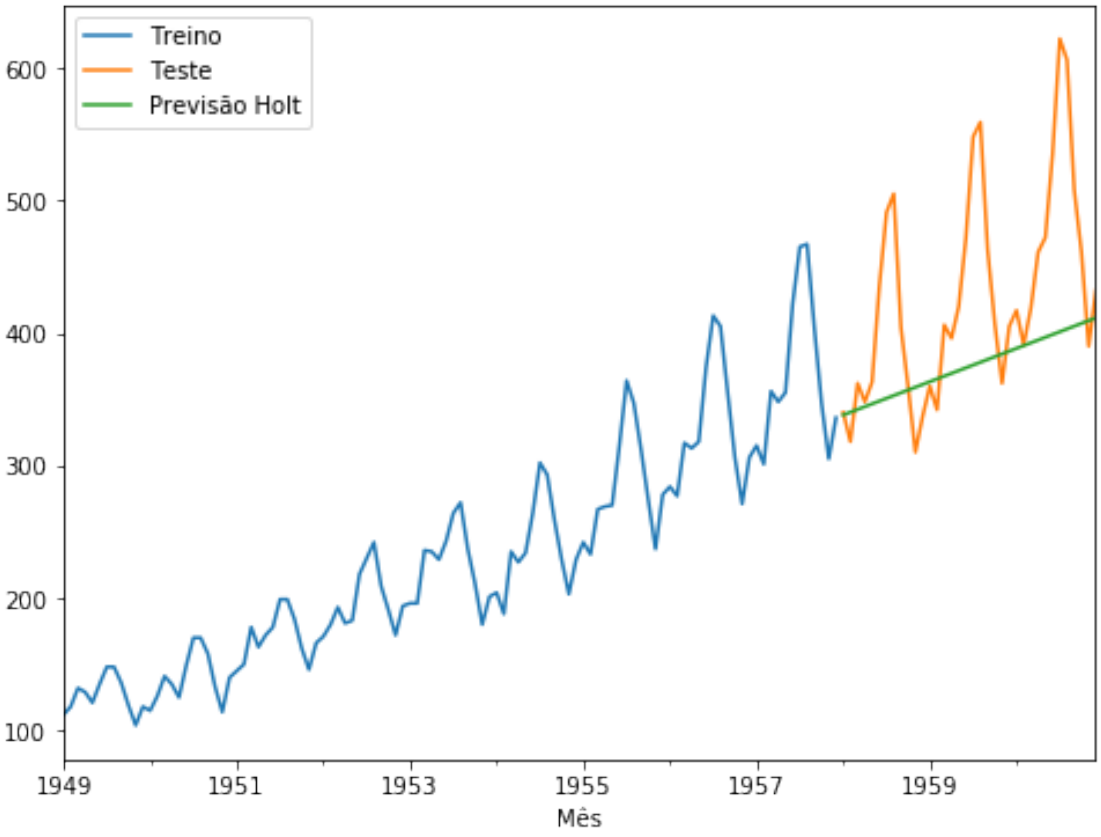
[26]:
```
dados_treino['Milhares de passageiros'].plot(legend=True,label='Treino')
dados_teste['Milhares de passageiros'].
 ↪plot(legend=True,label='Teste',figsize=(8,6))
```

```
predito_HW.plot(legend=True,label='Previsão Holt-Winters');
```



Exercício: Obtenha a previsão pelo Método de Holt

[27]:
```
ajustado_H = ExponentialSmoothing(dados_treino['Milhares de␣
 ↪passageiros'],trend='add').fit()
predito_H = ajustado_H.forecast(36).rename('Previsão Holt')


dados_treino['Milhares de passageiros'].plot(legend=True,label='Treino')
dados_teste['Milhares de passageiros'].
 ↪plot(legend=True,label='Teste',figsize=(8,6))
predito_H.plot(legend=True,label='Previsão Holt');
```



1.11 Métricas de avaliação dos ajustes

Para avaliar a distância da predição para o valor ajustado, usamos o erro quadrático médio e o erro absoluto médio

1.11.1 Erro quadrático médio

$$EQM = MSE = \sum_{i=t+1}^{k} \frac{(Z_i - \widehat{Z}_i)^2}{k-t-1}$$

1.11.2 Erro absoluto médio

$$EAM = MAE = \sum_{i=t+1}^{k} \frac{|Z_t - \widehat{Z}_t|}{k-t-1}$$

[28]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

[29]:
```
mean_absolute_error(dados_teste,predito_HW)
```

[29]: 22.313867163101683

[30]:
```
mean_absolute_error(dados_teste,predito_H)
```

[30]:
```
62.84209977156752
```

[31]:
```
mean_squared_error(dados_teste,predito_HW)
```

[31]:
```
711.1938543442199
```

[32]:
```
mean_squared_error(dados_teste,predito_H)
```

[32]:
```
7695.686676182441
```

[33]:
```
np.sqrt(mean_squared_error(dados_teste,predito_HW))
```

[33]:
```
26.668218057159724
```

[34]:
```
np.sqrt(mean_squared_error(dados_teste,predito_H))
```

[34]:
```
87.7250629876231
```

[35]:
```
dados_teste.describe()
```

[35]:
```
       Milhares de passageiros
count                36.000000
mean                428.500000
std                  79.329152
min                 310.000000
25%                 362.000000
50%                 412.000000
75%                 472.000000
max                 622.000000
```

[36]:
```
predito_HW.describe()
```

[36]:
```
count     36.000000
mean     420.377378
std       60.204436
min      321.238177
25%      378.101116
50%      410.058229
75%      455.878072
max      553.761167
Name: Previsão Holt-Winters, dtype: float64
```

[37]:
```
predito_H.describe()
```

[37]:
```
count     36.000000
mean     374.729062
std       22.055999
min      338.093463
25%      356.411262
50%      374.729062
75%      393.046861
max      411.364661
Name: Previsão Holt, dtype: float64
```

1.12 Previsão para os dados da COVID-19

[38]:
```
# Dados de COVID-19 no estado de SP

import pandas as pd
import numpy as np
%matplotlib inline

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

covidSP = pd.read_csv(f'{pkgdir}/covidSP.csv', index_col=0, parse_dates=True)

covidSP.index = pd.to_datetime(covidSP.index)
```

[42]:
```
covidSP.head()
```

[42]:

date	confirmed	deaths
2020-02-26	0	0
2020-02-27	0	0
2020-02-28	1	0
2020-02-29	0	0
2020-03-01	0	0

[43]:
```
covidSP.index.min()
```

[43]:
```
Timestamp('2020-02-26 00:00:00')
```

[44]:
```
covidSP.index.max()
```

[44]:
```
Timestamp('2020-07-11 00:00:00')
```

[45]:
```
idx = pd.date_range(start='2020-02-28', end='2020-07-11', freq='D')
idx
```

[45]:
```
DatetimeIndex(['2020-02-28', '2020-02-29', '2020-03-01', '2020-03-02',
               '2020-03-03', '2020-03-04', '2020-03-05', '2020-03-06',
               '2020-03-07', '2020-03-08',
               ...
               '2020-07-02', '2020-07-03', '2020-07-04', '2020-07-05',
               '2020-07-06', '2020-07-07', '2020-07-08', '2020-07-09',
               '2020-07-10', '2020-07-11'],
              dtype='datetime64[ns]', length=135, freq='D')
```

[49]:
```
covidSP = covidSP.reindex(idx)
covidSP.head(30)
```

[49]:

	confirmed	deaths
2020-02-28	1.0	0.0
2020-02-29	0.0	0.0
2020-03-01	0.0	0.0
2020-03-02	0.0	0.0
2020-03-03	0.0	0.0
2020-03-04	1.0	0.0
2020-03-05	3.0	0.0
2020-03-06	4.0	0.0
2020-03-07	3.0	0.0
2020-03-08	3.0	0.0
2020-03-09	0.0	0.0
2020-03-10	3.0	0.0
2020-03-11	11.0	0.0
2020-03-12	16.0	0.0
2020-03-13	NaN	NaN
2020-03-14	19.0	0.0
2020-03-15	NaN	NaN
2020-03-16	87.0	0.0
2020-03-17	12.0	1.0
2020-03-18	76.0	2.0
2020-03-19	46.0	2.0
2020-03-20	110.0	4.0
2020-03-21	0.0	6.0
2020-03-22	235.0	7.0
2020-03-23	114.0	8.0
2020-03-24	65.0	10.0
2020-03-25	52.0	8.0
2020-03-26	190.0	10.0
2020-03-27	171.0	10.0
2020-03-28	183.0	16.0

[51]:
```
covidSP.fillna(0,inplace=True)
covidSP.head(30)
```

[51]:

	confirmed	deaths
2020-02-28	1.0	0.0
2020-02-29	0.0	0.0
2020-03-01	0.0	0.0
2020-03-02	0.0	0.0
2020-03-03	0.0	0.0
2020-03-04	1.0	0.0
2020-03-05	3.0	0.0
2020-03-06	4.0	0.0
2020-03-07	3.0	0.0
2020-03-08	3.0	0.0
2020-03-09	0.0	0.0
2020-03-10	3.0	0.0
2020-03-11	11.0	0.0
2020-03-12	16.0	0.0
2020-03-13	0.0	0.0
2020-03-14	19.0	0.0
2020-03-15	0.0	0.0
2020-03-16	87.0	0.0
2020-03-17	12.0	1.0
2020-03-18	76.0	2.0
2020-03-19	46.0	2.0
2020-03-20	110.0	4.0
2020-03-21	0.0	6.0
2020-03-22	235.0	7.0
2020-03-23	114.0	8.0
2020-03-24	65.0	10.0
2020-03-25	52.0	8.0
2020-03-26	190.0	10.0
2020-03-27	171.0	10.0
2020-03-28	183.0	16.0

[52]:
```
len(covidSP)
```

[52]:
```
135
```

[53]:
```
covidSP.index
```

[53]:
```
DatetimeIndex(['2020-02-28', '2020-02-29', '2020-03-01', '2020-03-02',
               '2020-03-03', '2020-03-04', '2020-03-05', '2020-03-06',
               '2020-03-07', '2020-03-08',
               ...
               '2020-07-02', '2020-07-03', '2020-07-04', '2020-07-05',
               '2020-07-06', '2020-07-07', '2020-07-08', '2020-07-09',
               '2020-07-10', '2020-07-11'],
              dtype='datetime64[ns]', length=135, freq='D')
```

1.13 Divisão da base em treino e teste

[54]:
```
dados_treino = covidSP.iloc[:125] # Dados de treinamento até observação 125, sem␣
↪incluí-la
dados_teste = covidSP.iloc[125:] # Dados de teste a partir da observação 125
```

[55]:
```
dados_treino
```

[55]:
```
            confirmed  deaths
2020-02-28        1.0     0.0
2020-02-29        0.0     0.0
2020-03-01        0.0     0.0
2020-03-02        0.0     0.0
2020-03-03        0.0     0.0
...               ...     ...
2020-06-27     7073.0   297.0
2020-06-28     6156.0    75.0
2020-06-29     3408.0    60.0
2020-06-30     6235.0   365.0
2020-07-01     8555.0   267.0

[125 rows x 2 columns]
```

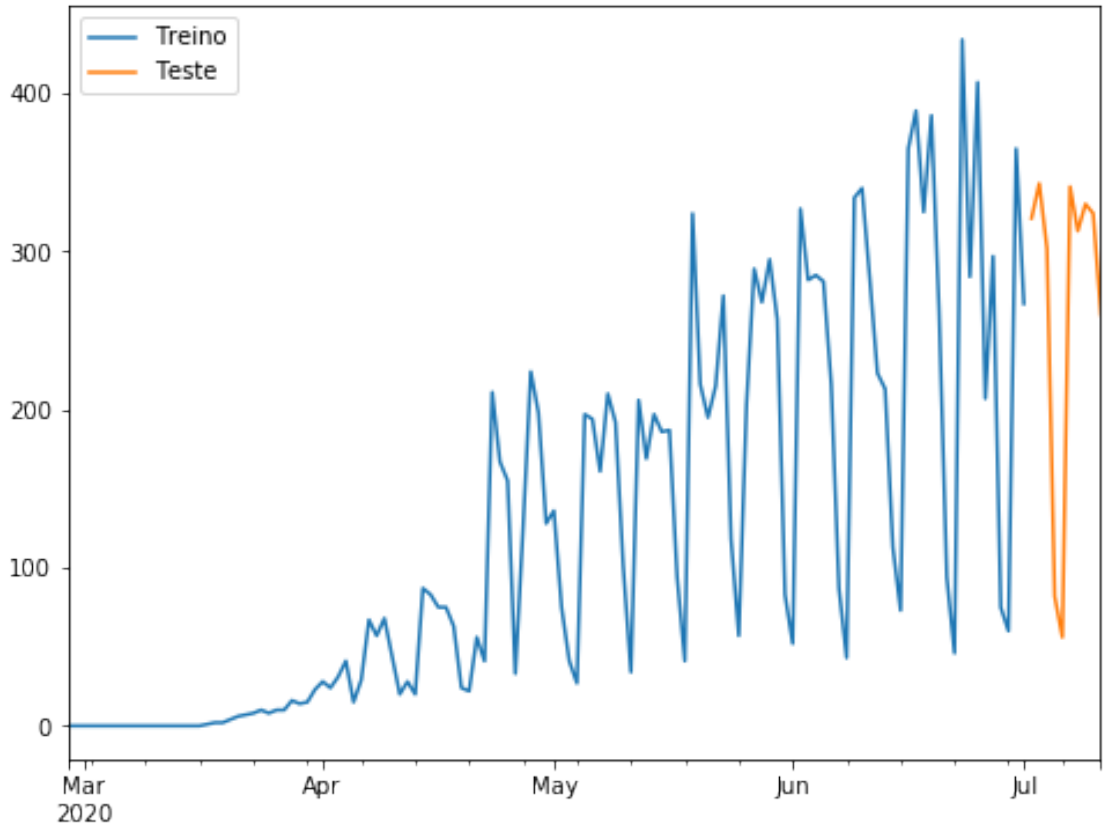
[56]:
```
## Ajuste do modelo
from statsmodels.tsa.holtwinters import ExponentialSmoothing


ajustado =␣
↪ExponentialSmoothing(dados_treino['deaths'],trend='add',seasonal='add',seasonal_periods=7).
↪fit()
```

[57]:
```
# Previsão

predito = ajustado.forecast(10).rename('Previsão Holt-Winters')
```

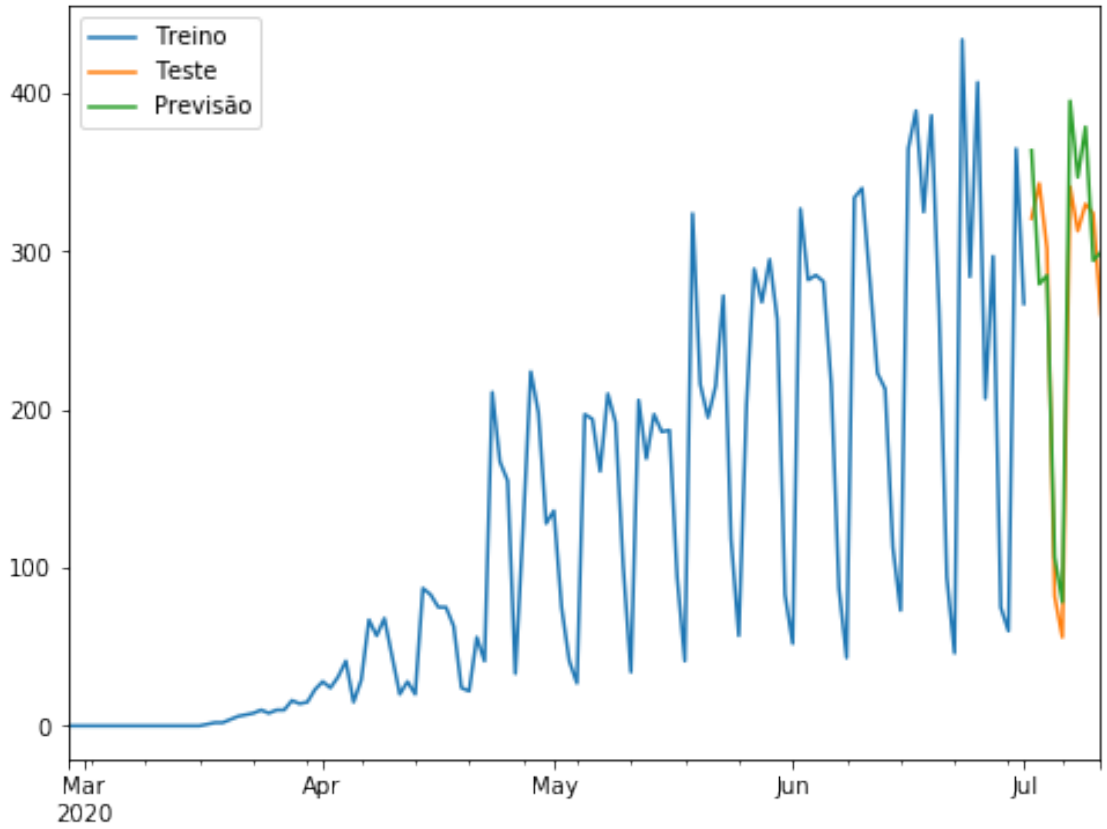
[58]:
```
predito.index = covidSP.index[125:]
predito.index
```

[58]:
```
DatetimeIndex(['2020-07-02', '2020-07-03', '2020-07-04', '2020-07-05',
               '2020-07-06', '2020-07-07', '2020-07-08', '2020-07-09',
               '2020-07-10', '2020-07-11'],
              dtype='datetime64[ns]', freq='D')
```

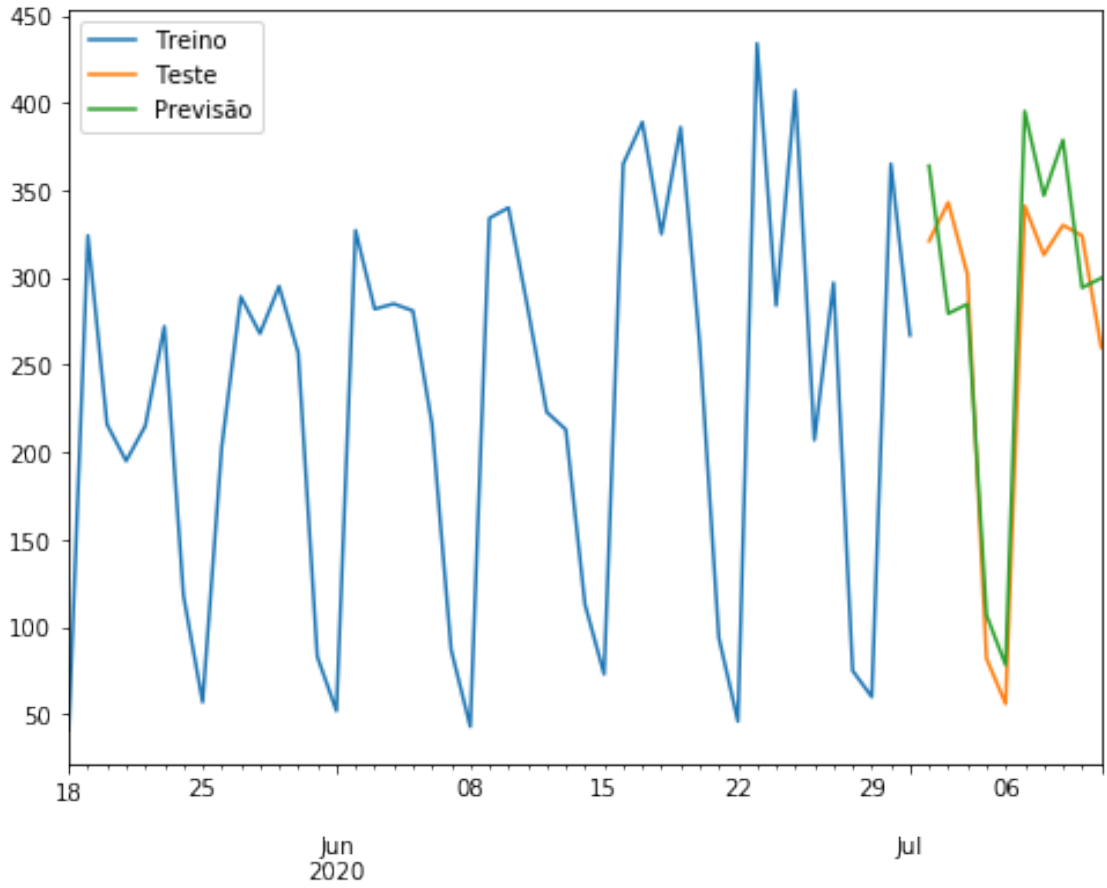
[59]:
```
dados_treino['deaths'].plot(legend=True,label='Treino')
dados_teste['deaths'].plot(legend=True,label='Teste',figsize=(8,6));
```



[60]:
```
dados_treino['deaths'].plot(legend=True,label='Treino')
dados_teste['deaths'].plot(legend=True,label='Teste',figsize=(8,6))
predito.plot(legend=True,label='Previsão');
```



[62]:
```
dados_treino['deaths'].iloc[80:].plot(legend=True,label='Treino')
dados_teste['deaths'].plot(legend=True,label='Teste',figsize=(8,6))
predito.plot(legend=True,label='Previsão');
```



[63]:
```
## Ajuste do modelo com sazonalidade multiplicativa
from statsmodels.tsa.holtwinters import ExponentialSmoothing

covidSP = covidSP[covidSP['deaths']>0]
```

[64]:
```
covidSP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 117 entries, 2020-03-17 to 2020-07-11
Freq: D
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   confirmed  117 non-null    float64
 1   deaths     117 non-null    float64
dtypes: float64(2)
memory usage: 2.7 KB
```

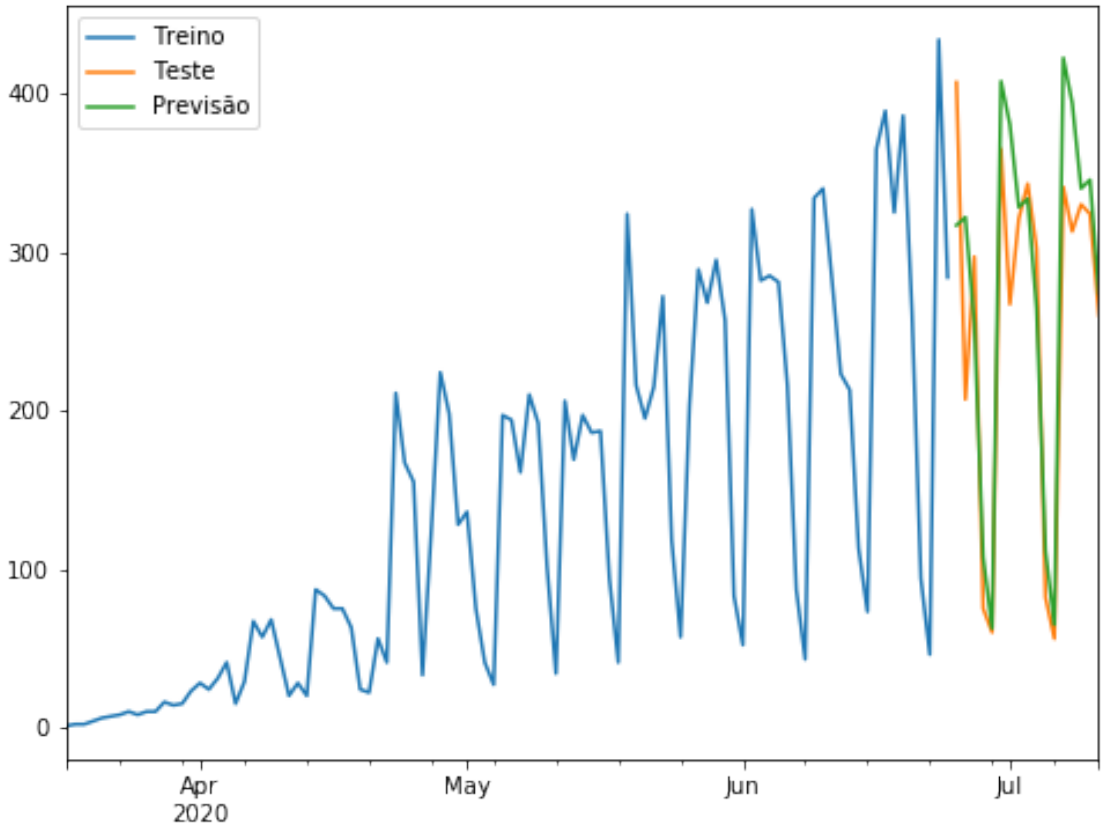
[65]:
```
dados_treino = covidSP.iloc[:100] # Dados de treinamento até observação 100, sem␣
↪incluí-la
dados_teste = covidSP.iloc[100:] # Dados de teste a partir da observação 100
```

[68]:
```
ajustado =␣
↪ExponentialSmoothing(dados_treino['deaths'],trend='mul',seasonal='mul',seasonal_periods=7).
↪fit()
```

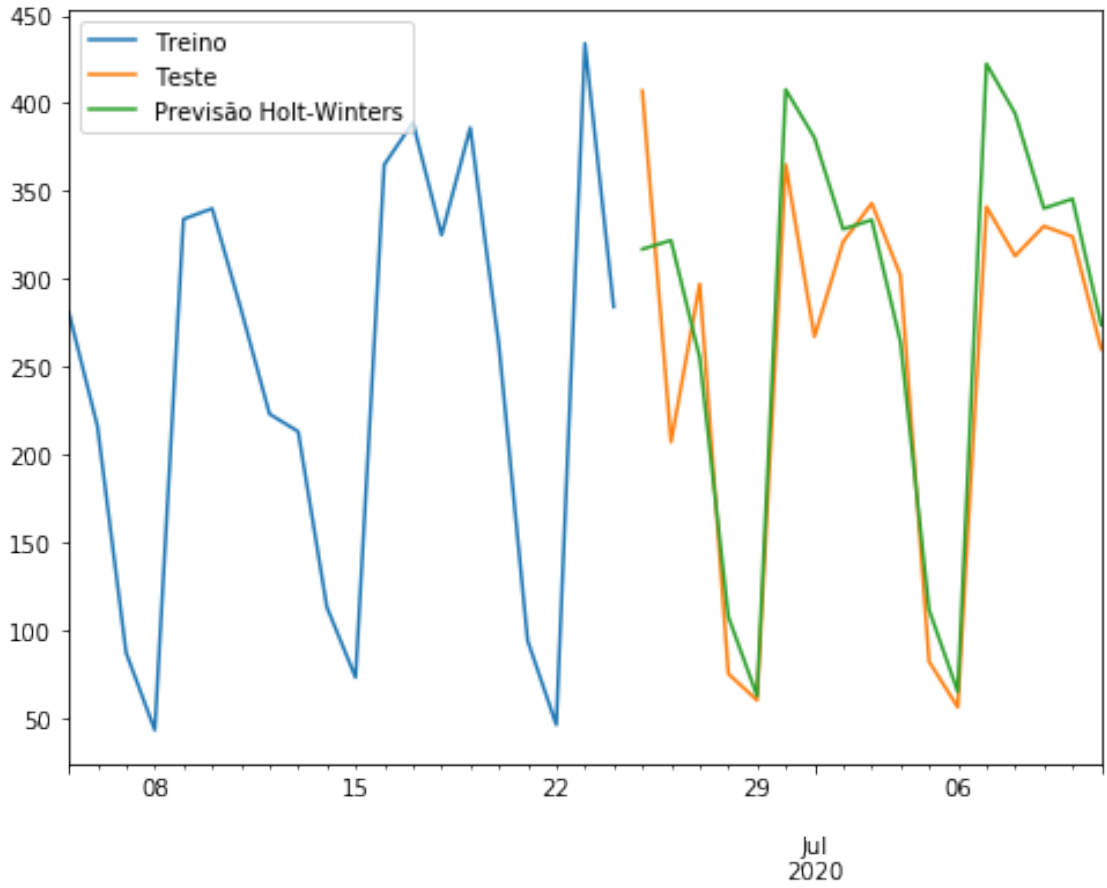
[69]:
```
predito = ajustado.forecast(17).rename('Previsão Holt-Winters')
predito
```

[69]:
```
2020-06-25    316.854942
2020-06-26    321.883512
2020-06-27    255.043314
2020-06-28    107.497517
2020-06-29     62.557095
2020-06-30    407.737360
2020-07-01    380.428034
2020-07-02    328.274698
2020-07-03    333.484503
2020-07-04    264.235320
2020-07-05    111.371831
2020-07-06     64.811712
2020-07-07    422.432606
2020-07-08    394.139026
2020-07-09    340.106033
2020-07-10    345.503605
2020-07-11    273.758615
Freq: D, Name: Previsão Holt-Winters, dtype: float64
```

[70]:
```
dados_treino['deaths'].plot(legend=True,label='Treino')
dados_teste['deaths'].plot(legend=True,label='Teste',figsize=(8,6))
predito.plot(legend=True,label='Previsão');
```



[71]:
```
dados_treino['deaths'].iloc[80:].plot(legend=True,label='Treino')
dados_teste['deaths'].plot(legend=True,label='Teste',figsize=(8,6))
predito.plot(legend=True,label='Previsão Holt-Winters');
```


Treino
Teste
Previsão Holt-Winters
450
400
350
300
250
200
150
100
50
08
15
22
29
06
Jul
2020

[ ]: