@Odemir Depieri Jr

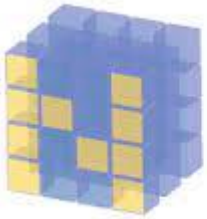
Guia de Serie Temporal - Suavização exponencial

Python

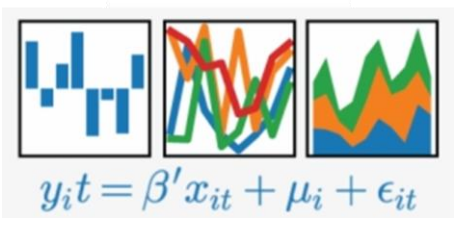
Case com os dados de Companhia Aérea

statsmodels

matplotlib



NumPy



@Odemir Depieri Jr

Resumo sobre **Serie Temporal**

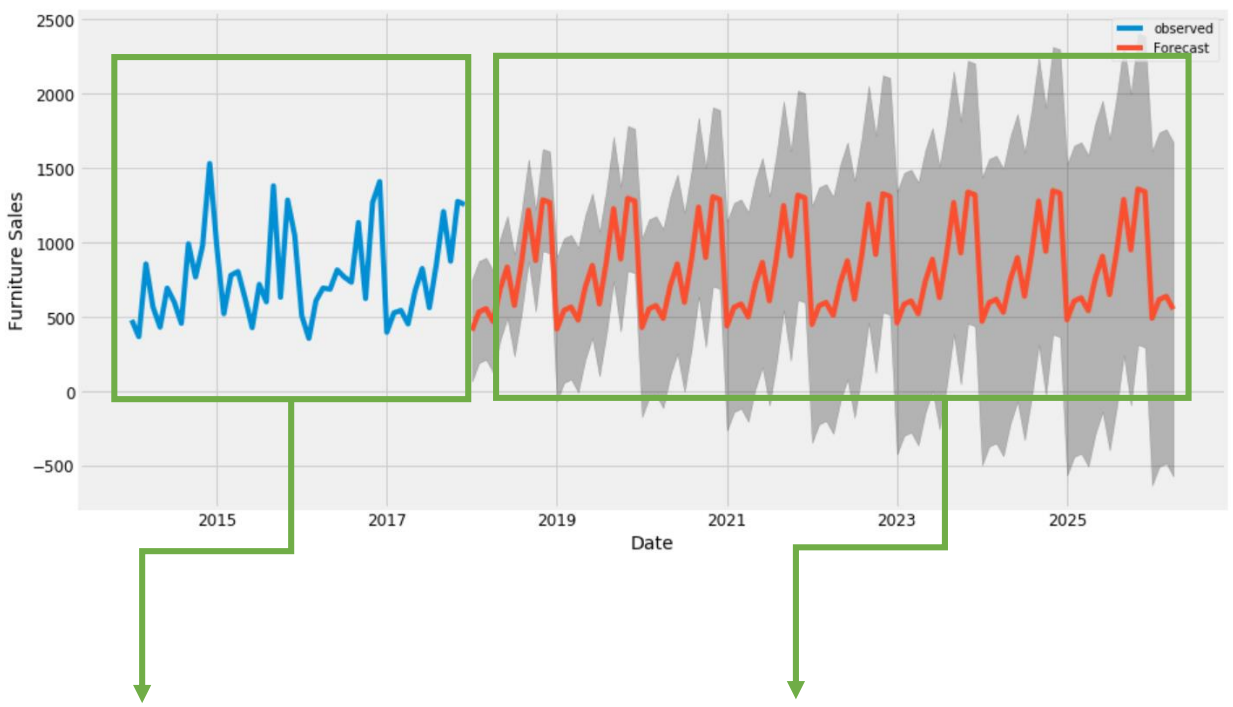
Uma **série temporal** é uma sequência de observações em intervalos de tempo regularmente espaçados.

Exemplo:

- Taxas de desemprego **mensais** para os últimos **cinco anos**
- Produção **diária** em uma fábrica durante um mês
- População em cada **década** de um século

Uma serie temporal **procura padrões** em sequência de intervalos, assim podemos usar a serie temporal para fazer **previsões**.

Exemplo de uma serie gráfica:



Dados Realizados
(Observação)

Previsão
(Forecast)

Mão na Massa !!

Vamos utilizar uma base dados disponível na Kaggle

https://www.kaggle.com/rakannimer/air-passengers

```
[139] # Biblioteca para modelagem de dados
      import pandas as pd

      # Biblioteca para recursos matemáticos
      import numpy as np

      # Bibliotecas de plotagem de dados
      import seaborn as sns
      import matplotlib.pyplot as plt
```

```
[140] # Lendo o Arquivo CSV
      Base_Dados = pd.read_csv('Passageiros_Aereos.csv')
      Base_Dados.head()
```

	Month	#Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

```
[141] # Ajustando a coluna MES para o formato Data
      Base_Dados['Month'] = pd.to_datetime(
          Base_Dados['Month'],
          infer_datetime_format=True ) # Converter string para Data

      # Incluindo o Mes como index
      Base_Dados = Base_Dados.set_index(['Month'])
```

```
[142] # Verificando os 1º Registros
      Base_Dados.head()
```

	#Passengers
Month	
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

[143]
```
# Verificando as colunas
for Coluna in Base_Dados.columns:
  print( Coluna )
```

```
#Passengers
```

[144]
```
# Verificando a dimensão da base de dados
Base_Dados.shape
```

```
(144, 1)
```

[145]
```
# Verificando o formato dos campos
Base_Dados.info()
```

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

```
# Gerando algumas estatística para entender um pouco os dados
# Dicionário para entender as estatísticas abaixo:
# count --> Total de registros
# mean --> Média
# std --> Desvio Padrão
# min --> Valor mínimo
# 25% --> 1º Quartil
# 50% --> Mediana
# 75% --> 3º Quartil
# max --> Valor Maior
# Comando para gerar estatísticas sobre os dados
Base_Dados.describe()
```

	#Passengers
count	144.000000
mean	280.298611
std	119.966317
min	104.000000
25%	180.000000
50%	265.500000
75%	360.500000
max	622.000000

```
[148] # Separando o Eixo do Gráfico
      Eixo_1 = Base_Dados['#Passengers'].values

      # Criando o gráfico para enteder a curva de crescimento
      plt.figure( figsize=(12,5) )
      plt.title('Quantidade de Passageiros')
      plt.xlabel('Periodo')
      plt.ylabel('Quantidade')
      plt.plot( Eixo_1, color='red' )
```


```
[149] # Treinar o modelo da Serie Temporal

      # Importando a Função da Serie
      from statsmodels.tsa.api import ExponentialSmoothing

      # Definindo os parametros
      Funcao_Serie_Temporal = ExponentialSmoothing(
          Base_Dados,
          seasonal_periods=12,
          trend='additive',
          seasonal='additive').fit(use_boxcox=True)
```

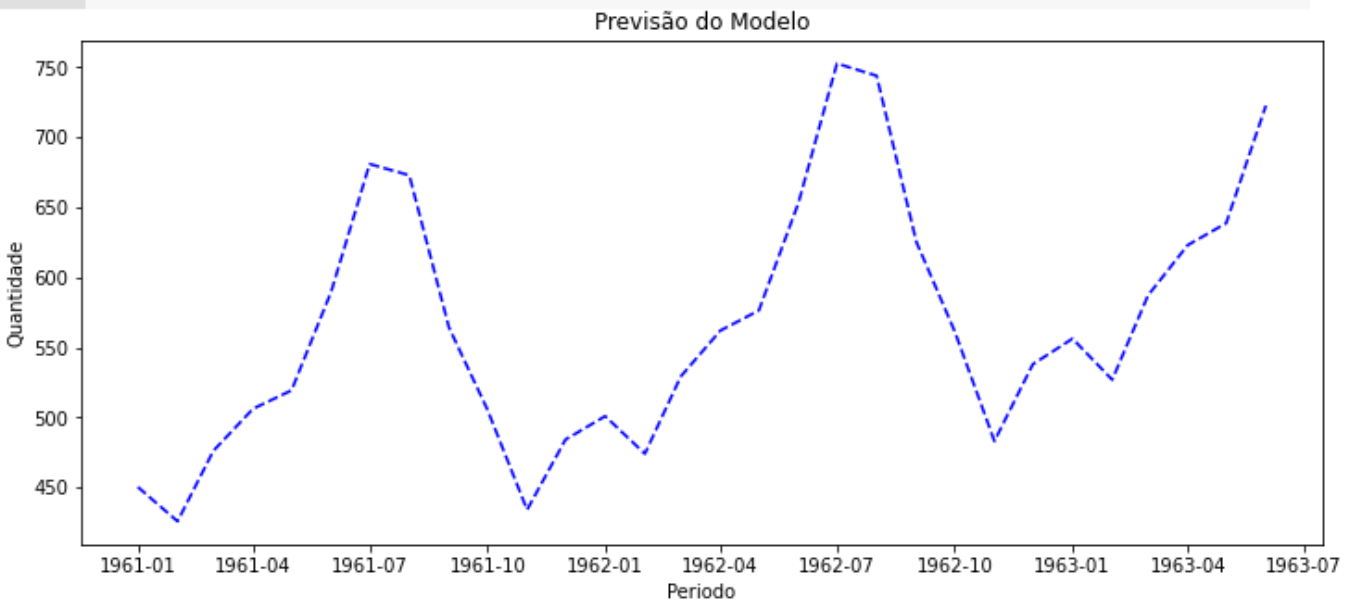
Caso não tenha conhecimento sobre essa tema, vou deixar um artigo bem legal de como funciona uma serie temporal

https://towardsdatascience.com/time-series-in-python-exponential-smoothing-and-arima-processes-2c67f2a52788

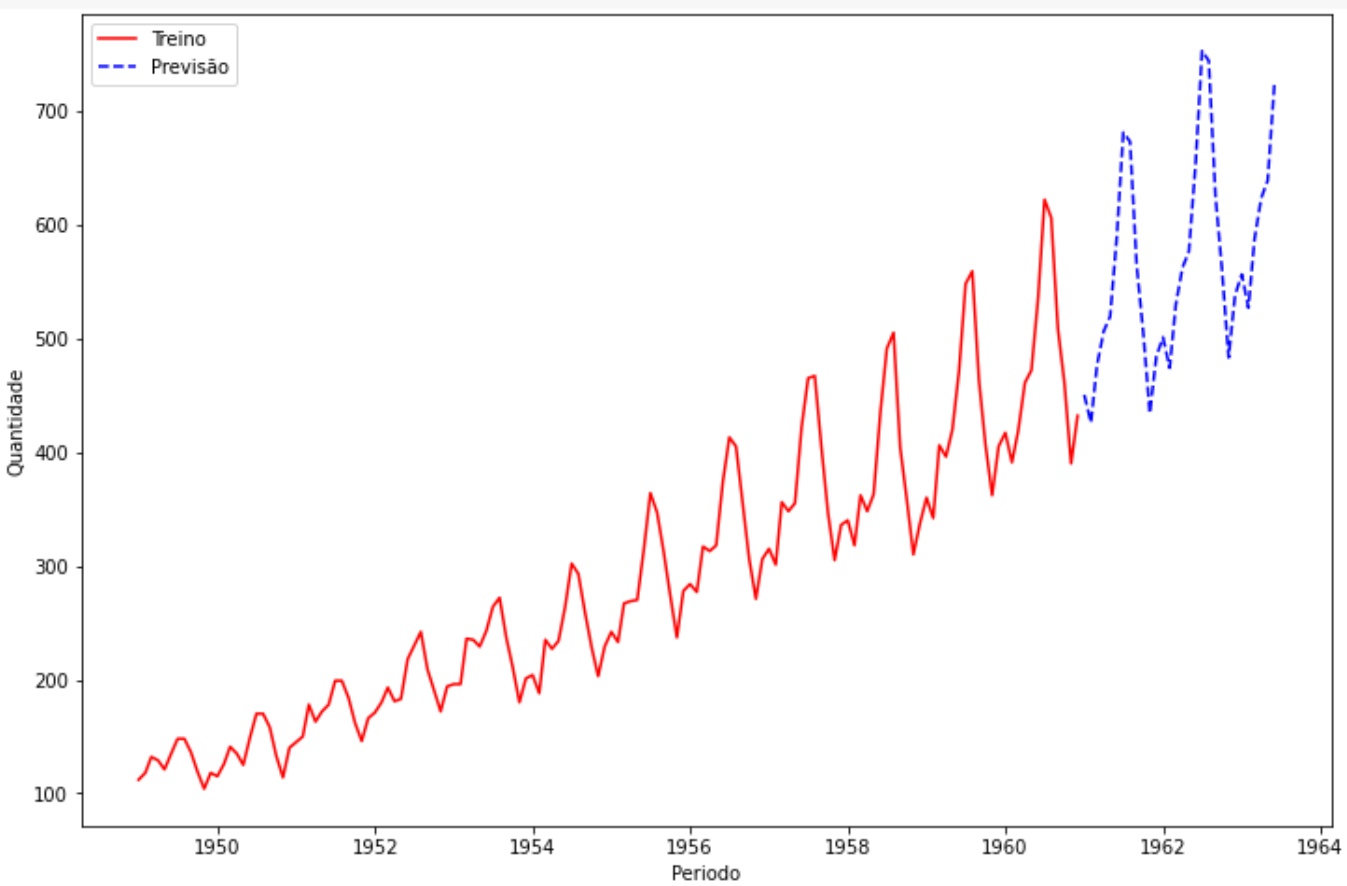
```
# Defiindo os dias para serem previstos
Quantos_Dias_Quer_Prever = 30

# Fazendo a previsao usando o metodo 'FORECAST'
Previsao = Funcao_Serie_Temporal.forecast( Quantos_Dias_Quer_Prever )

# Criando o gráfico com a previsão
plt.figure( figsize=(12,5) )
plt.title('Previsão do Modelo')
plt.xlabel('Periodo')
plt.ylabel('Quantidade')
plt.plot( Previsao, color='blue', linestyle='--' );
```



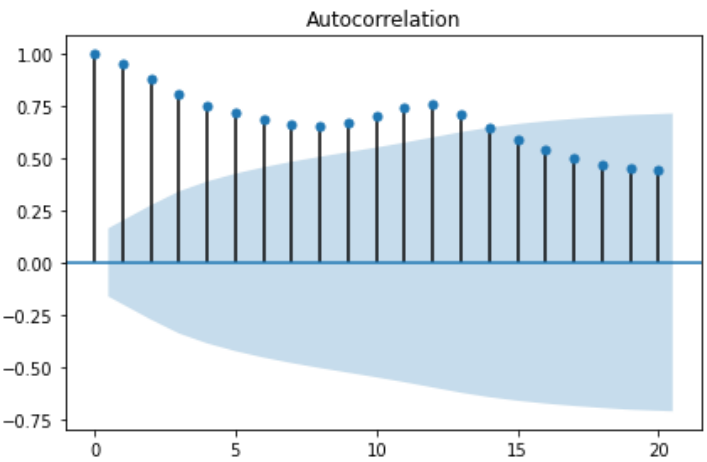

```
[151] # Criando o gráfico com a previsão e treino
      plt.figure(figsize=(12,8))
      plt.plot( Base_Dados['#Passengers'], label='Treino', color='red')
      plt.plot( Previsao, label='Previsão', color='blue', linestyle='--')
      plt.xlabel('Periodo')
      plt.ylabel('Quantidade')
      plt.legend(loc=0)
```

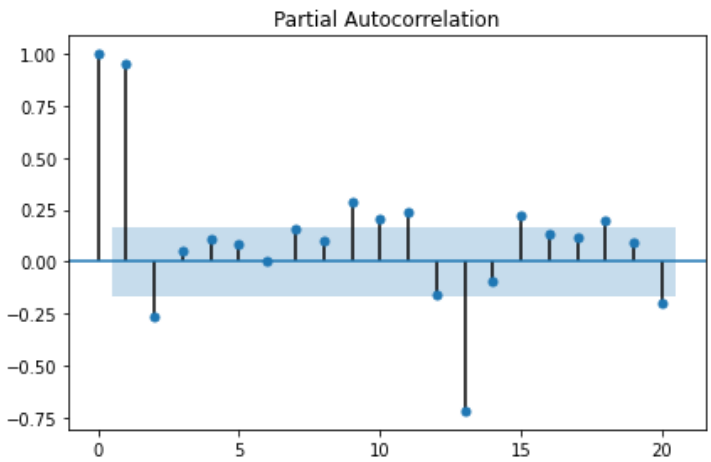


```
[152] # Verificando o diagrama de Autocorrelação
      # Verificando o diagrama de Autocorrelação parcial

      # Importando a função smt para gerar as correlações
      import statsmodels.tsa.api as smt

      # Definindo uma figura de 1 linha e 2 colunas
      fig, axes = plt.subplots(1, 2)
      # Fixando o tamanho dos gráficos
      fig.set_figwidth(12)
      fig.set_figheight(4)
      # Plotando o gráfico de AutoCorrelação
      smt.graphics.plot_acf(Base_Dados, lags=20, ax=axes[0])
      # Plotando o gráfico de AutoCorrelação PArcial
      smt.graphics.plot_pacf(Base_Dados, lags=20, ax=axes[1])
      plt.tight_layout()
```





Conclusão.

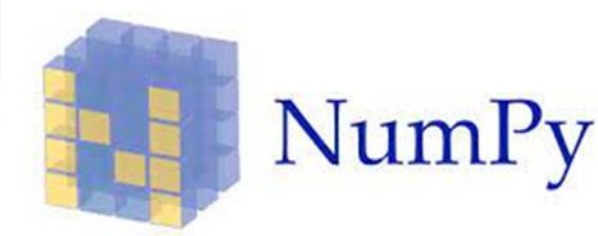
Nossa serie teve uma performance muito boa, devido os dados terem um padrão sequencial (Tendência).

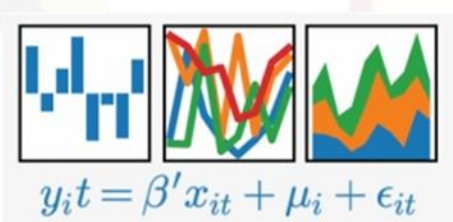
Mas ressalto que nem sempre será assim rsrsrs.

Final

Esse guia sobre como usar uma serie temporal no Python.

Guia da documentação caso queira mais detalhes
https://www.statsmodels.org/stable/examples/notebooks/generated/exponential_smoothing.html





Odemir Depieri Jr

Software Engineer Sr
Tech Lead
Specialization AI