ANA RAQUEL

DATA ANALYTICS

ANÁLISE DE SÉRIES TEMPORAIS

AULA 03

SUMÁRIO

O QUE VEM POR AÍ? 3
CONHEÇA SOBRE O ASSUNTO 4
HANDS ON 11
O QUE VOCÊ VIU NESTA AULA? 12
REFERÊNCIAS.....13
PALAVRAS-CHAVE.....14

O QUE VEM POR AÍ?

Nesta aula, você irá aprender quais são as principais bibliotecas em Python para trabalhar com séries temporais. Vamos lá?

CONHEÇA SOBRE O ASSUNTO

Quando trabalhamos com séries temporais, podemos encontrar alguns desafios, tanto para realizar o tratamento de dados, quanto para construir análises e modelos. Pensando em um conceito de modelos de Machine Learning, quando analisamos uma base de dados onde é possível identificar quais são as variáveis características e qual é a variável target, podemos pensar em solucionar problemas de negócio utilizando algoritmos de Machine Learning, como, por exemplo, é possível encontrar na biblioteca do Scikit-Learn. Mas e quando temos uma base de dados que possui valores sumarizados e datas? Como eu identifico a minha variável alvo? Qual variável eu posso analisar na análise exploratória dos dados?

O problema de séries temporais é diferente dos problemas supervisionados e não supervisionados. Podemos dizer que a estrutura dos dados é diferente, pois estamos lidando com dados que possuem uma certa frequência sobre um determinado tempo. Os modelos de séries temporais consideram a ordem temporal das observações e assumem que há alguma dependência entre elas. Nesse caso, podemos pensar que o dado que eu preciso encontrar é um valor que pode estar correlacionado com a dimensão temporal dos dados.

Variáveis características **Target**

Modelos supervisionados				
X1	X2	X3	X4	Y
123	6767	3445	34344	1
3455	7544	5567	344	1
55	32	4556	343	0
55	12	3456	34	0

Target

Séries temporais	
Date	Sales
13/04/2023	1200
14/04/2023	1300
15/04/2023	14030
16/04/2023	2339
17/04/2023	20000

Figura 1 - Diferença de bases de séries temporais
Fonte: Elaborado pela autora (2023)

Pensando nesses desafios, seguem algumas bibliotecas do Python que podem te auxiliar a trabalhar com modelos de séries temporais:

Pandas: O clássico Pandas pode te acompanhar em diversos tipos de problemas com os dados, como aprendemos na aula anterior. Além de ser uma ferramenta chave para instanciar dados no Python, dentro do Pandas temos opções para realizar transformações de datas em índices, agregação dos dados, resample... permitindo assim a facilidade em trabalhar com dados de séries temporais.

No exemplo da figura 2 – Exemplo de uso do Pandas para trabalhar com séries temporais, a biblioteca Pandas está sendo utilizada para instanciar dados no Python e criar transformações na base, como por exemplo, a transformação da coluna de data em índice (conforme vimos na aula passada).

```
[ ] dataset['Month'] = pd.to_datetime(dataset['Month'],infer_datetime_format=True) #convertendo a data para datetime
    indexedDataset = dataset.set_index(['Month']) #Indexando a coluna "month" como index do dataframe
    indexedDataset.head(5)
```

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

Figura 2 - Exemplo de uso do Pandas para trabalhar com séries temporais
Fonte: Elaborado pela autora (2023)

Numpy: O Numpy pode ser uma ferramenta muito útil quando precisamos realizar algumas análises estatísticas sobre as séries temporais, como, por exemplo, diversas funções para calcular estatísticas descritivas, como média, desvio padrão, correlação e covariância, que podem ser utilizadas para analisar as propriedades das séries temporais. No exemplo da figura 3 – Exemplo do uso de Numpy, para trabalhar com séries temporais, por exemplo, foi realizado uma transformação sobre a escala da série temporal para tipo logarítmica utilizando o comando np.log(). Você também pode explorar outras transformações matemáticas e estatísticas, tais como np.mean(), np.median(), np.std(), entre outras.

```
#Estimating trend
indexedDataset_logScale = np.log(indexedDataset) #Transformação logarítma
plt.plot(indexedDataset_logScale)
```

```
[<matplotlib.lines.Line2D at 0x1862d2b2490>]
```

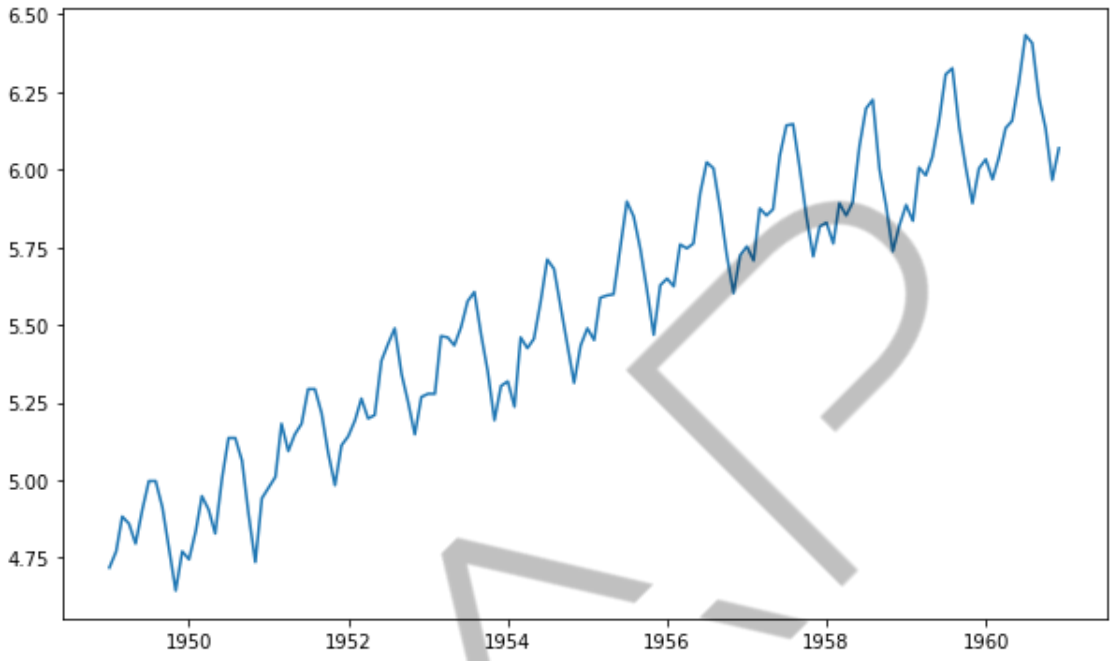

Figura 3 - Exemplo de uso do Numpy para trabalhar com séries temporais
Fonte: Elaborado pela autora (2023)

Matplotlib: A biblioteca do Matplotlib é uma das mais utilizadas e fáceis de ser trabalhadas quando pensamos em análise exploratória de dados. Na Matplotlib, podemos encontrar gráficos de linhas e dispersões que podem ser muito utilizados em análises de séries temporais. No exemplo da figura 4 – Exemplo de uso Matplotlib, para trabalhar com séries temporais, a biblioteca Matplotlib está sendo utilizada para gerar um gráfico de linha utilizando a data e os valores que compõem uma série temporal.

```
plt.xlabel('Date')
plt.ylabel('Number of air passengers')
plt.plot(indexedDataset)
```

```
[<matplotlib.lines.Line2D at 0x1862b24b250>]
```

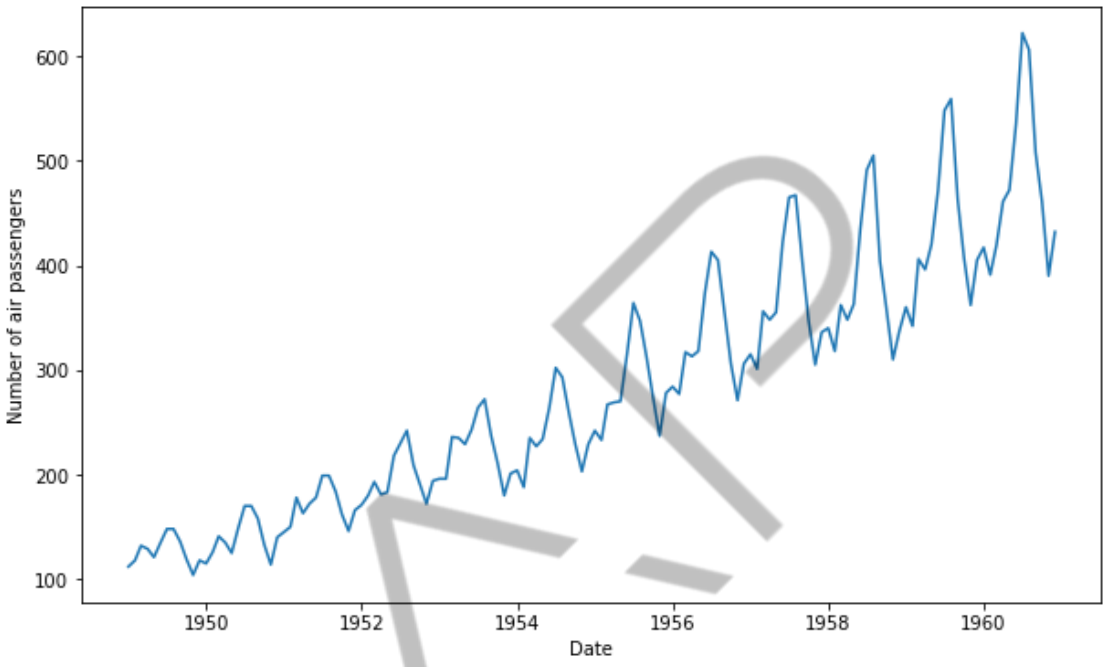

Figura 4 - Exemplo de uso do matplotlib para trabalhar com séries temporais
Fonte: Criado pela autora (2023)

Scikit-learn: Falando em modelos de séries temporais, temos que citar o modelo clássico de regressão que pode ser encontrado no Scikit-learn para a criação de modelos e análises de séries temporais. O gráfico é um resultado de previsão de um modelo de regressão linear simples da biblioteca Scikit-learn (LinearRegression).

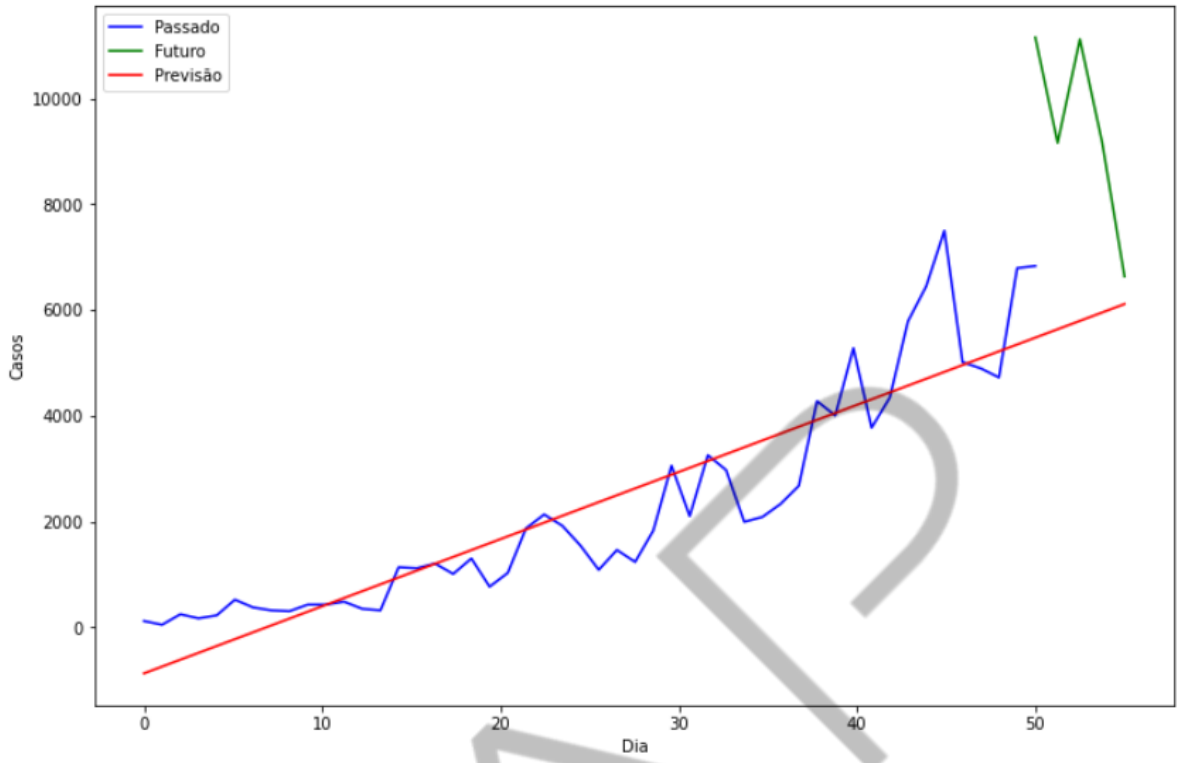

Figura 5 - Exemplo de uso da biblioteca Scikit-Learn para trabalhar com séries temporais, modelo de regressão linear.
Fonte: Kaggle [s.d.].

Statsmodels: A biblioteca Statsmodels é clássica para análise de séries temporais, sendo composta por vários modelos estatísticos para resolver problemas de séries temporais, como, por exemplo, modelos de média móvel, modelos autoregressivos, testes estatísticos (como o teste ADF) e etc.

```
# AR+I+MA = ARIMA model
model = ARIMA(indexedDataset_logScale, order=(2,1,2)) #(p,d,q)
results_ARIMA = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f'%sum((results_ARIMA.fittedvalues - datasetLogDiffShifting['#Passengers'])**2))
print('Plotting ARIMA model')
```

```
Plotting ARIMA model
```

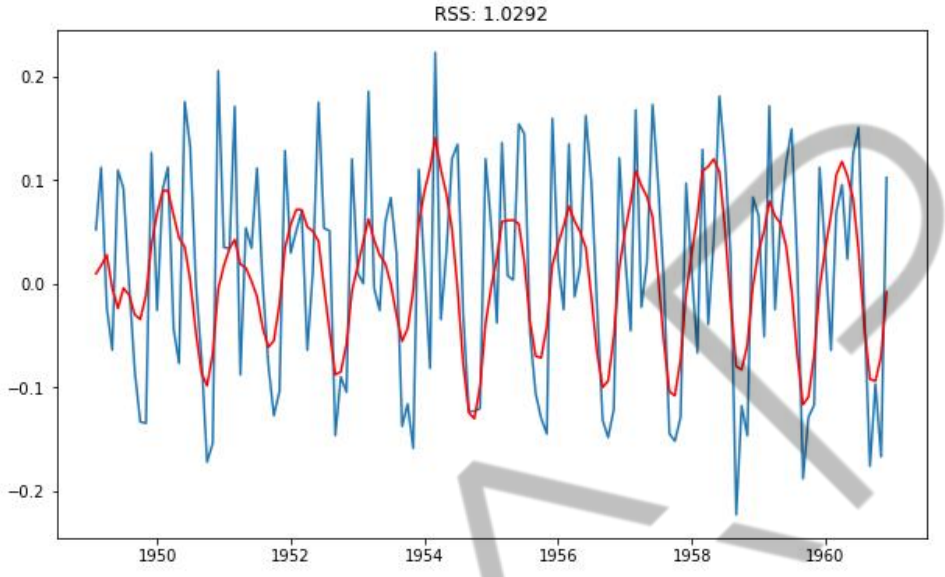

Figura 6 - Exemplo de uso da biblioteca statsmodels para trabalhar com séries temporais, modelo ARIMA

Fonte: Elaborado pela autora (2023)

Prophet: é uma biblioteca da empresa Meta para modelagem de séries temporais, que implementa um procedimento para prever dados de séries temporais com base em um modelo aditivo em que tendências não lineares são ajustadas à sazonalidade anual, semanal e diária, além de efeitos de feriados. Na figura 7 – Exemplo de uso da biblioteca Prophet para trabalhar com séries temporais, o gráfico é resultado de um modelo de série temporal construído pela biblioteca Prophet.

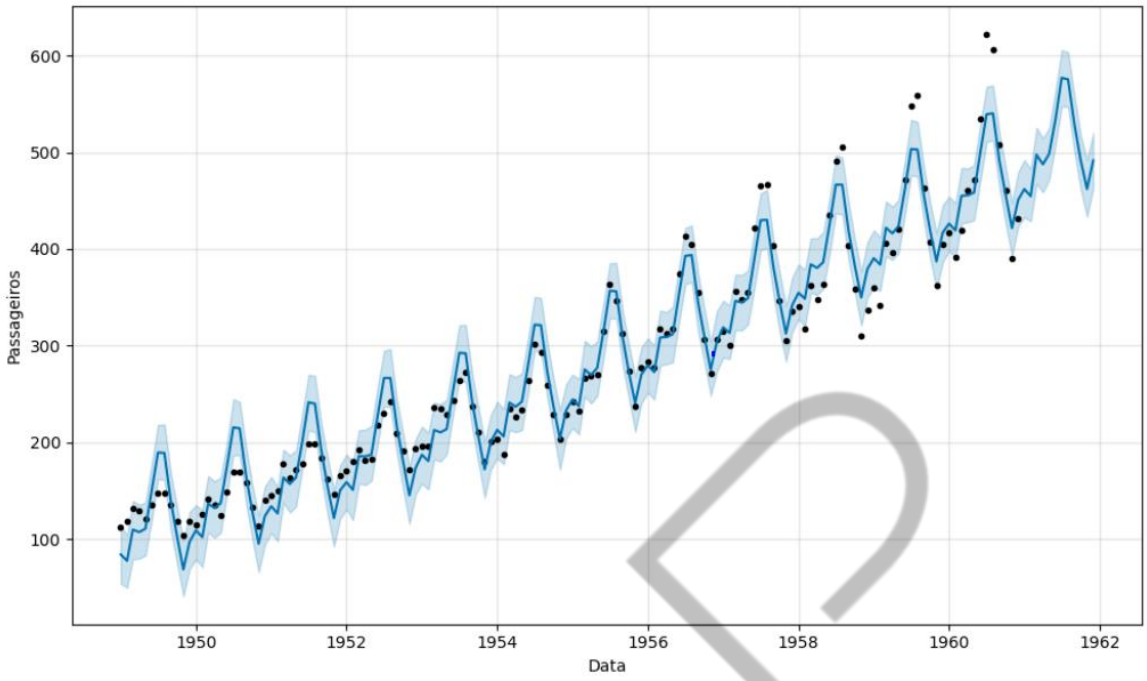

Figura 7 - Exemplo de uso da biblioteca Prophet para trabalhar com séries temporais

Fonte: Kaggle (s.d.)

HANDS ON

Agora chegou o momento de ver, na prática, como podemos trabalhar com a modelagem temporal dos dados utilizando as principais bibliotecas para trabalhar com séries temporais no Python.

O QUE VOCÊ VIU NESTA AULA?

Principais bibliotecas em Python para análise de séries temporais e suas utilidades.

Daqui em diante, é importante que você replique os conhecimentos adquiridos para fortalecer mais suas bases e conhecimentos.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que assim você possa aprimorar os seus conhecimentos!

Você não está sozinho(a) nesta jornada! Te esperamos no Discord e nas *lives* com os(as) professores(as) especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

NIELSEN, Aileen. **Análise Prática de Séries Temporais**: Predição com Estatística e Aprendizado de Máquina. [s.l.]: O'Reilly Media, Inc., 2021.

STATSMODELS. **Statistical models, hypothesis tests, and data exploration**. [s.d]. Disponível em: <https://www.statsmodels.org/stable/index.html>. Acesso em: 09 mai 2023.

PALAVRAS-CHAVE

Palavras-Chave: Statsmodels, Prophet, Pandas.