ANA RAQUEL

DATA ANALYTICS

ANÁLISE DE SÉRIES TEMPORAIS

AULA 02

Sumário

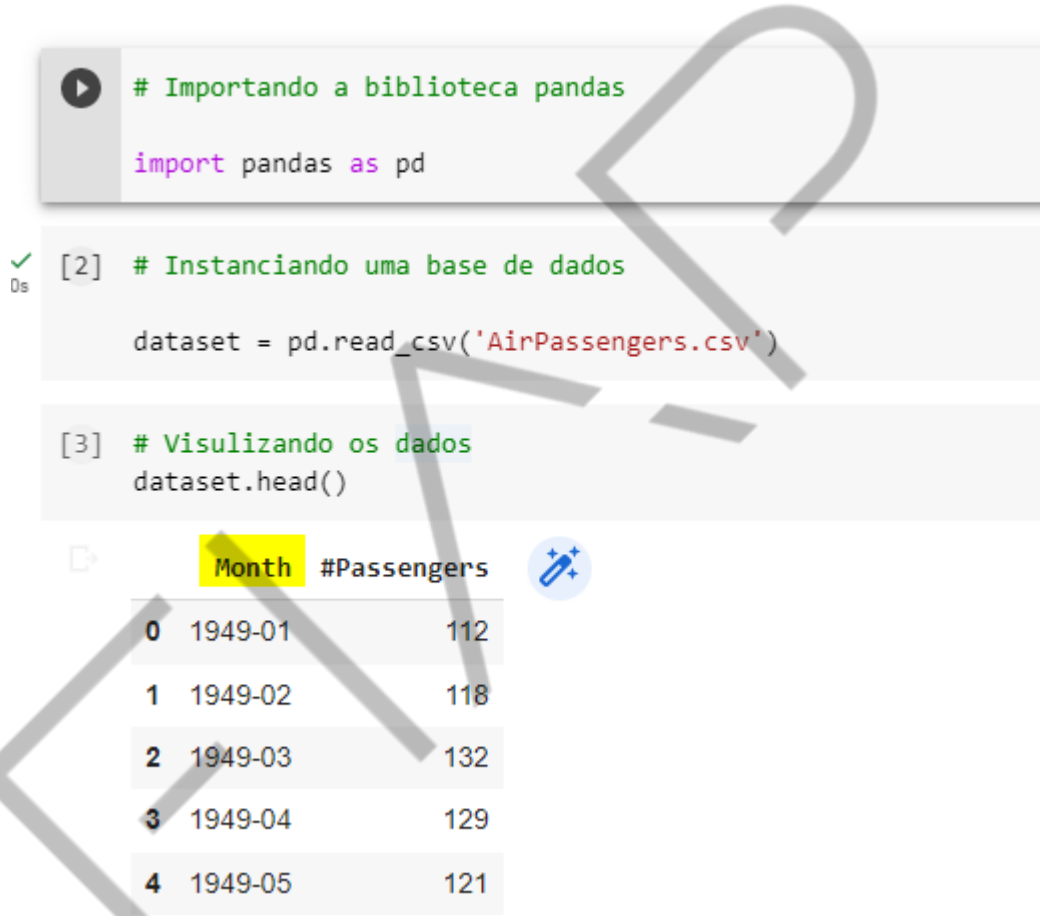
O QUE VEM POR AÍ? 3
CONHEÇA SOBRE O ASSUNTO 4
HANDS ON 10
O QUE VOCÊ VIU NESTA AULA? 11
REFERÊNCIAS.....12
PALAVRAS-CHAVE.....13

O QUE VEM POR AÍ?

Nesta aula, você irá aprender os benefícios e como utilizar a data como índice para a construção de análise de dados com séries temporais. Vamos lá?

CONHEÇA SOBRE O ASSUNTO

Quando vamos construir uma análise exploratória para séries temporais, uma ótima opção é transformar a coluna de data como índice do dataframe (base de dados). Vamos analisar na prática como essa transformação pode ser realizada, utilizando o Python:



```
# Importando a biblioteca pandas

import pandas as pd
```

```
[2]  # Instanciando uma base de dados

     dataset = pd.read_csv('AirPassengers.csv')
```

```
[3]  # Visulizando os dados
     dataset.head()
```

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

Figura 1 - Coluna com data no Dataframe
Fonte: Criado pela autora (2023)

Perceba que, na figura 1 – Coluna com data no Dataframe, a nossa base de dados “AirPassengers” possui uma coluna “Month”, que possui algumas datas, e outra coluna “#Passengers”, contabilizando o número de passageiros em uma companhia aérea. Veja que o índice da base de dados está no formato tradicional. Se eu tentar criar um gráfico de linhas, utilizando o eixo x a data e o eixo y o volume de passageiros, teríamos a visualização da figura 2 – Gráfico de linhas.

```
import matplotlib.pyplot as plt
```

```
plt.plot(dataset['Month'], dataset['#Passengers'])
plt.show()
```

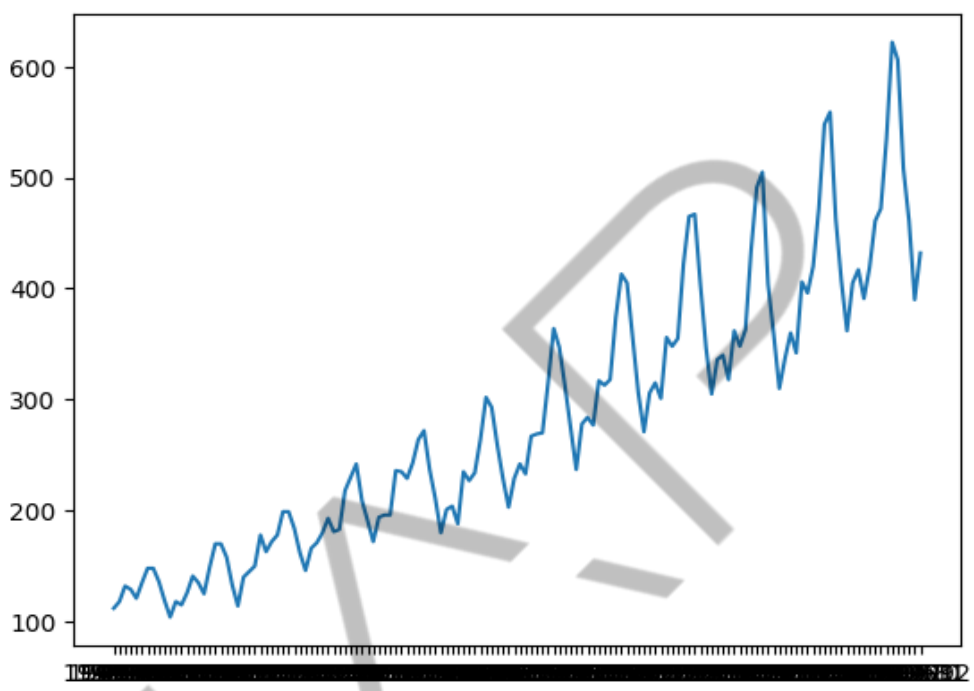


Figura 2 - Gráfico de linhas

Fonte: Elaborado pela autora (2023)

Perceba que temos algumas limitações: a data posicionada no eixo x não está evidente, impossibilitando a identificação dos dados. Esse tipo de situação pode ocorrer pois o tipo da coluna “Month” não está no formato **datetime**.

```
[7] dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Month        144 non-null    object
 1   #Passengers  144 non-null    int64
dtypes: int64(1), object(1)
memory usage: 2.4+ KB
```

Figura 3 - Analisando o formato dos dados

Fonte: Elaborado pela autora (2023)

Para resolver este problema, podemos realizar a transformação da coluna “Month” para o formato adequado de análise de tempo, com o tipo de dados **datetime**:

```
dataset['Month'] =
pd.to_datetime(dataset['Month'],infer_datetime_format=True)
```

Vamos visualizar como ficou essa transformação, repetindo o gráfico de linhas criado anteriormente, porém agora com o formato adequado:

```
plt.plot(dataset['Month'], dataset['#Passengers'])
plt.show()
```

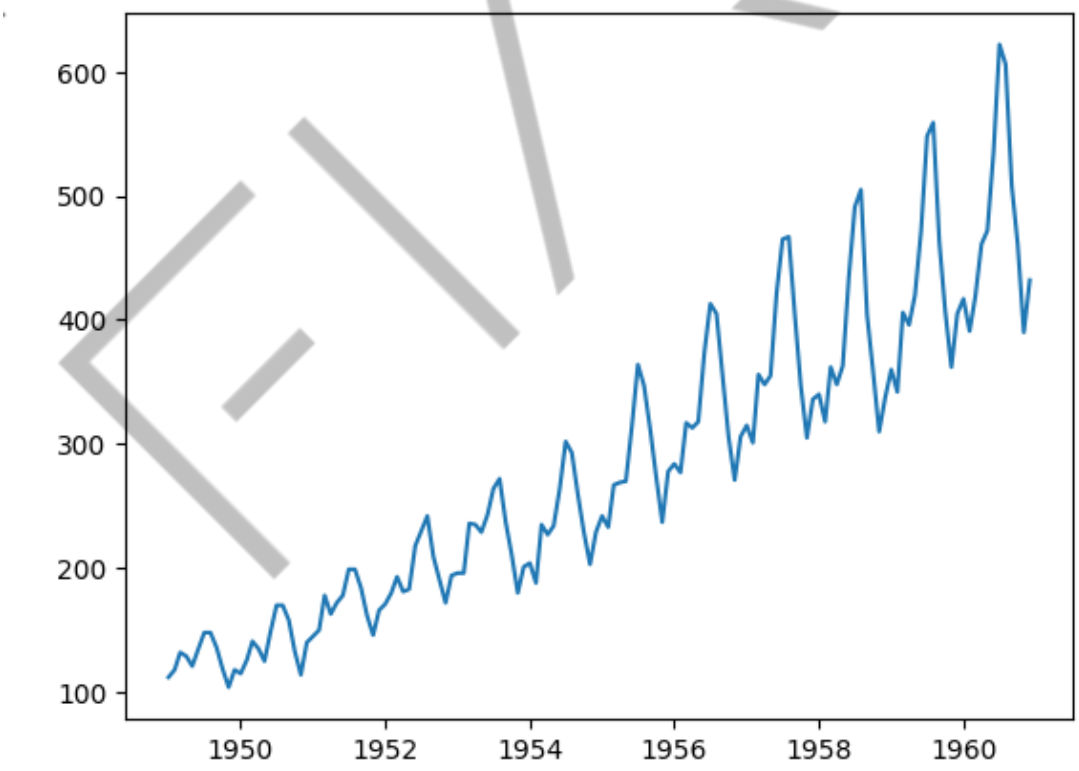


Figura 4 - Gráfico de linhas com o formato de data

Fonte: Elaborado pela autora (2023)

Perceba que agora conseguimos visualizar os anos da coluna “Month” de forma clara e objetiva.

Entendemos que a transformação do tipo de dados para datetime é essencial para a análise de séries temporais, mas qual pode ser o benefício de transformar essa coluna de data em índice e como eu posso aplicá-la?

Temos alguns benefícios, tais como:

- Facilidade na manipulação dos dados;
- Melhorar a eficiência da busca de dados, tal como por exemplo, ao utilizar o método loc[] para buscar dados com base em datas (podendo melhorar assim a indexação dos dados, pois a data está configurada no índice);
- Facilita o uso de técnicas, como resamples e agregação de dados.

```
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 5 - Utilizando data como índice

Fonte: Criado pela autora (2023)

A partir dessa transformação, podemos criar gráficos de linhas com os dados já indexados:

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

```
[<matplotlib.lines.Line2D at 0x7ff029d931f0>]
```

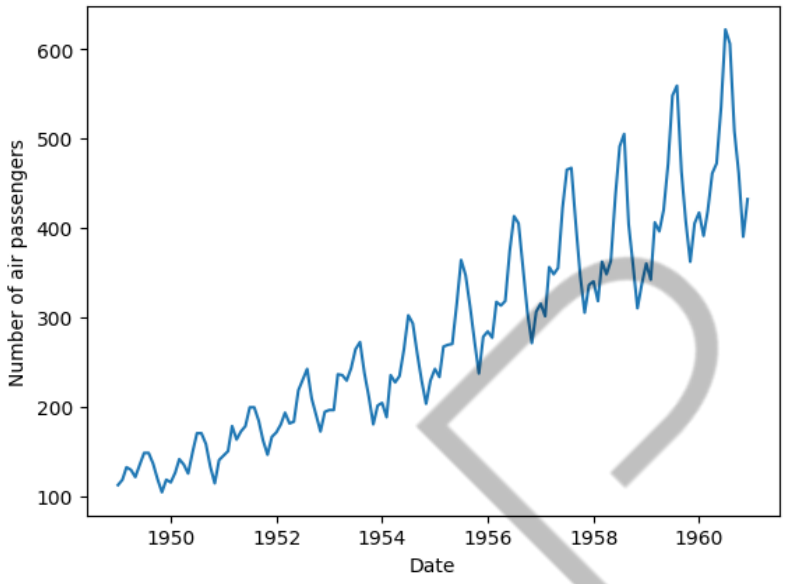


Figura 6 - Utilizando data como índice em gráfico de linhas.

Fonte: Criado pela autora (2023)

Veja como podemos utilizar o comando loc[] para realizar filtros:

```
[28] # Filtrando os dados e criando um novo dataframe
     df_filtrado = dataset.loc[(dataset['Month'].dt.year > 1958)]

     # Alterando coluna para tipo índice
     df_filtrado = df_filtrado.set_index(['Month']) #Indexando a coluna "month" como index do dataframe
```

Figura 7 - Filtrando datas como índice.

Fonte: Criado pela autora (2023)

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

```
[<matplotlib.lines.Line2D at 0x7ff029b55220>]
```

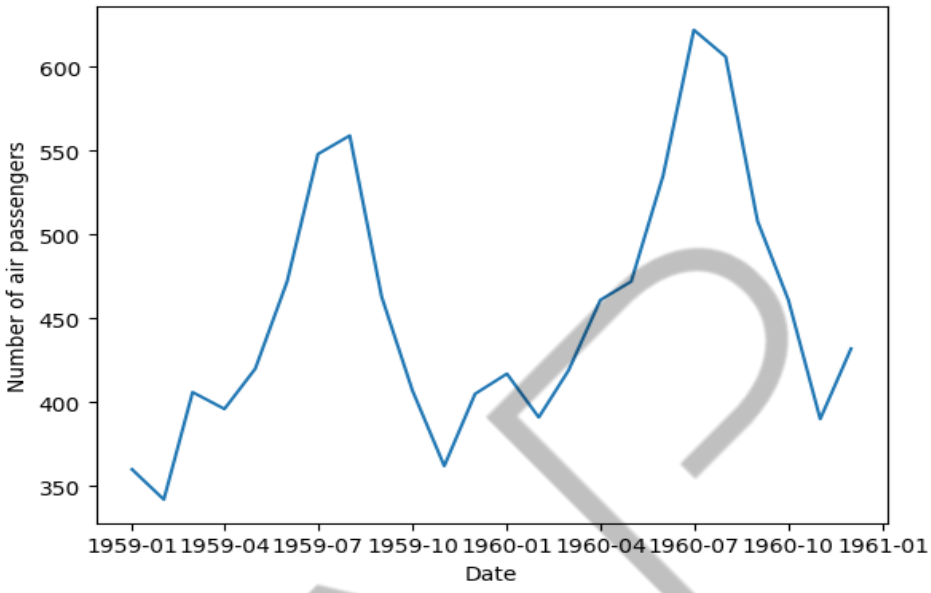


Figura 8 - Visualizando os dados após filtro.
Fonte: Criado pela autora (2023)

HANDS ON

Agora, chegou o momento de ver, na prática, como podemos lidar com o problema de séries temporárias utilizando o Python.

O QUE VOCÊ VIU NESTA AULA?

O desafio de lidar com séries temporais e os principais componentes.

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.

PALAVRAS-CHAVE

Palavras-Chave: Datetime, Index, Python.