FICHA TRABALHO 5

INTERPOLAÇÃO E OUTLIERS

MODALIDADE:	Aprendizagem +	Não aplicável	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Limpeza e transformação de dados em Python	CÓDIGO UFCD:	10808
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Uso da técnica da interpolação na manipulação da informação e o saber o que são os Outliers

1 – Técnica de substituição de valores ausentes: Interpolação da informação

Em vez de eliminar linhas de informação (que podem ser importantes), podemos substituir os valores ausentes por uma estatística resumida ou um valor específico (dependendo do caso de uso), talvez possa ser o melhor caminho a seguir.

Para explicar melhor esta técnica, vamos assumir uma série de valores mensais com 12 observações, dos quais temos 4 dados ausentes:

[100, None, 120, 145, None 167, 189, None, None, 191, 210]

Questão:

Como preencher os valores omissos (None) desta série temporal?

A função **interpolate()** do Pandas dá resposta a este problema.

A interpolação através do alisamento dos dados é também uma técnica muito usada para estimada e preencher os valores ausentes (especialmente em casos de dados temporárias), com ou sem tendência, mas em caraterísticas sazonais.

A fórmula usada na interpolação foi a seguinte:

$$Yt = \text{Xt1} + (\text{t} - \text{t}_1)\ \frac{Xt2 - Xt1}{t2\ - t1}$$

Onde Y1 é o valor interpolado para o instante t em que se pretende realizar a interpolação, t_1 e $t2$ são os instantes temporais em observações conhecidas mais próximas do instante de interesse t tal que $t_1 \leq t \leq t2$, $Xt1$ e $Xt2$ são os valores conhecidos nos instantes t_1 e $t2$, respetivamente.

Por exemplo, o valor ausente do instante 2 foi recuperado através do seguinte cálculo:

$$\text{Y}_2 = \text{X}_1 + (2 - 1)\ \text{x}\ \frac{X3 - X1}{3-1} = 100 + 1\ \text{x}\ \frac{120 - 100}{2} = 110$$

Existem outras formas de interpolação disponíveis no Python como a interpolação polinomial, cúbica, entre outras. Para mais detalhes, deve consultar a documentação das funções:

- **pandas.DataFrame.interpolate()**
- **pandas.Series.Interpolate**

Mesmo na literatura ainda podemos encontrar ainda outros métodos de imputação de dados ausentes, como:

- **a imputação de valor fixos**, que consiste em substituir os valores ausentes por um valor fixo que faça sentido no contexto do problema em estudo (como por exemplo, não imputar o valor 0 ao valor de vendas de uma loja nos dias em que não realiza qualquer venda);
- **a imputação por um modelo estatístico**, que consiste em substituir os valores ausentes pelas estimativas ou previsões obtidas a partir de um modelo estatístico ou econométrico, como o modelo de regressão

Em qualquer dos casos, a imputação de dados deve ser realizada com cuidado para evitar distorções na análise ou resultados enganosos

2 – Exemplo prático com a Interpolação

Em vez de eliminar linhas de informação (que podem ser importantes), podemos substituir os valores ausentes por uma estatística resumida ou um valor específico (dependendo do caso de uso), talvez possa ser o melhor caminho a seguir.

Exercício 1 – Crie uma nova pasta com o nome ficha_5 e crie um novo ficheiro no laboratório do jupyter para começar a testar os próximos exercícios e renomeie o ficheiro com “ficha_5.ipnyb”;

Exercício 2 – Faça a importação da biblioteca Pandas:

```
#importação da biblioteca
import pandas as pd
```

Exercício 3 – Crie a variável dados e crie uma lista com o carregamento dos 12 valores a analisar (como visto anteriormente):

```
#Serie temporal com 12 observações (Incluindo os 4 valores nulos)
dados = [100, None, 120, 145, None, 167, 189, None, None, 191, 210]
```

Exercício 4 – Crie um variável com o nome Data e faça a conversão dos valores em um Dataframe com a instrução pd.DataFrame e dentro de parênteses e chavetas curvas dê o nome a coluna do dataframe "Serie" e associe os dados:

```
#Conversão da série num dataframe
data = pd.DataFrame({"Serie": dados})
```

Exercício 5 – Depois dos dados carregados como um dataframe, vamos utilizar a função da interpolação para fazer o cálculo dos valores em falta, utilizando o valor do antecessor e do sucessor para prever o valor.

Neste caso, foi usado o método da ***interpolação linear*** para estimar e preencher os valores ausentes, assumindo que existe uma relação linear entre os valores conhecidos da série temporal. Também usamos o **inplace=True** para colocar as alterações diretamente na variável data:

```
#Utilizar a interpolação para realizar o cálculo dos valores None
data["Serie"].interpolate(method="linear", inplace=True)
```

Exercício 6 – Vamos verificar quais foram os valores depois de utilizar a função da interpolação de dados. Como tal, vamos criar uma nova variável com o nome nova_data, criar um novo Dataframe com a instrução pd.DataFrame e dentro de parenteses e chavetas curvas, vamos colocar duas colunas:

- **Coluna** "Data_Original" e **atribuir** os valores iniciais;
- **Coluna** "Interpolação" e **atribuir** os valores da variável data (que só tem a coluna Serie);

```
#Juntar toda a informação em outra dataframe
nova_data = pd.DataFrame({"Data_Original": dados, "Interpolacao": data["Serie"]})
print(nova_data)
```

No final, vamos fazer mostrar os dados desse novo dataframe para poder comparar o antes e o depois:

```
   data["Serie"].interpolate(method="linear", inplace=True)
    Data_Original  Interpolacao
0           100.0    100.000000
1             NaN    110.000000
2           120.0    120.000000
3           145.0    145.000000
4             NaN    156.000000
5           167.0    167.000000
6           189.0    189.000000
7             NaN    189.666667
8             NaN    190.333333
9           191.0    191.000000
10          210.0    210.000000
```

2 – Técnica de substituição de valores ausentes: Deteção e correção de outliers

Os ***outliers*** (valores aberrantes) são observações de dados numéricos que se afastam significativamente das outras observações e que podem causar distrações ou anomalias na modelação dos dados e previsões estatísticas.

A existência de ***outliers*** pode ocorrer por vários motivos, como erros de recolha e medição dos dados, fenómenos artificiais ou experimentais, atípicos ou inesperados, ou até mesmo fenómenos naturais como guerras, greves, alterações legislativas, variações abruptas nos preços dos combustíveis, entre outras.

Os **outliers** não são necessariamente observações erradas e podem refletir características que se pretendem detetar e estudar separadamente nos dados, como por exemplo, operações bancárias fraudulentas, artigos defeituosos, doenças raras, falhas em máquinas, entre outras anomalias ou eventos raros.

Caso os **outliers** representem erros nas observações ou causem distorções no fenómeno em estudo, é comum procedermos à correção ou eliminar os dados usando diferentes técnicas estatísticas.

Em algumas situações, o analista de dados é confrontado com problemas de erros nos dados e duplicações.

Como detetar estes problemas e quais as medidas corretivas a tomar?

Caso de estudo

Suponha que tem uma série temporal com os valores da temperatura média mensal num país do sul da Europa em que alguns valores são claramente incorretos (como por exemplo, valores abaixo de -5 graus ou acima de 50 graus Celsius não fazem qualquer sentido para temperaturas médias mensais):

25, -20, 22, 30, 32, 36, 37, 39, 35, 29, 27, 124

Exercício 1 – Na pasta com o nome ficha_5 e crie um novo ficheiro com o nome no laboratório do jupyter para começar a testar os próximos exercícios com o nome "ficha_5_outliers.ipynb;

Exercício 2 – Faça a importação da biblioteca Pandas:

```
#importação da biblioteca
import pandas as pd
```

Exercício 3 – Crie a variável **dados** e **crie uma lista** com o carregamento dos **12 valores a analisar** (como visto anteriormente):

```
#Série temporal com erros nos dados
dados = [25, -20, 22, 30, 32, 36, 37, 39, 35, 29, 27, 124]
```

Exercício 4 – Crie um variável com o nome **Data** e faça a **conversão dos valores em um Dataframe** com a instrução **pd.DataFrame** e dentro de parênteses e chavetas curvas dê o nome a coluna do dataframe "**Serie**" e **associe os dados** da variável dados:

```
#Conversão da série num dataframe
data = pd.DataFrame({"Serie": dados})
```

Exercício 5 – Antes de começar por fazer a interpolação dos dados, temos de filtrar quais os valores que são uma anomalia na nossa lista de dados. Como tal, vamos usar duas condições em simultâneo, por forma a filtrar os valores fora de um intervalo razoável como por exemplo, valores abaixo de -5 graus ou acima de 50 graus Celsius não fazem qualquer sentido para temperaturas médias mensais).

Como tal, vamos criar uma nova variável com o nome **data_erros** e vamos pedir para:

- **filtrar os dados** apenas com os **valores abaixo** de -5 **ou** **superiores** a 50 erros:

```
#Filtrar valores fora de um intervalo razoavel
#como por exemplo entre -5 e 50
data_erros = data[(data["Temperatura"] < -5) | (data["Temperatura"] > 50)]
print(data_erros)
```

- **eliminar** essas observações e **substituir** os valores anómalos por valores ausentes:

```
#Substituir os valores anómalos por valores ausentes
data[data["Temperatura"].isin(data_erros["Temperatura"])] = None
```

- proceder à sua **interpolação com os valores** nos instantes temporais mais próximos. Depois dos dados carregados como um dataframe, vamos utilizar a função da interpolação para fazer o cálculo dos valores em falta, utilizando o valor do antecessor e do sucessor para prever o valor.

  Neste caso, foi usado o método da ***interpolação linear*** para estimar e preencher os valores ausentes, assumindo que existe uma relação linear entre os valores conhecidos da série temporal. Também usamos o **inplace=True** para colocar as alterações diretamente na variável data:

```
#Realizar uam interpolação linear para preencher os valores ausentes
data["Temperatura"].interpolate(method="linear", inplace=True)
```

Exercício 6 – Vamos verificar quais foram os valores depois de utilizar a função da interpolação de dados. Como tal, vamos criar uma nova variável com o nome nova_data, criar um novo Dataframe com a instrução pd.DataFrame e dentro de parenteses e chavetas curvas, vamos colocar duas colunas:

- **Coluna** “Data_Original” e **atribuir** os valores iniciais;
- **Coluna** “Interpolação” e **atribuir** os valores da variável data (que só tem a coluna Serie);

```
#Juntar toda a informação em outra dataframe
nova_data = pd.DataFrame({"Data_Original": dados, "Interpolacao": data["Temperatura"]})
print(nova_data)
```

No final, vamos fazer mostrar os dados desse novo dataframe para poder comparar o antes e o depois:

	Data_Original	Interpolacao
0	25	25.0
1	-20	23.5
2	22	22.0
3	30	30.0
4	32	32.0
5	36	36.0
6	37	37.0
7	39	39.0
8	35	35.0
9	29	29.0
10	27	27.0
11	124	27.0

Neste exemplo, os valores anómalos ou erros nos dados (-20 e 124 graus, hipoteticamente registados em vez de 20 e 24), foram corrigidos pelos valores 23.5 e 27.