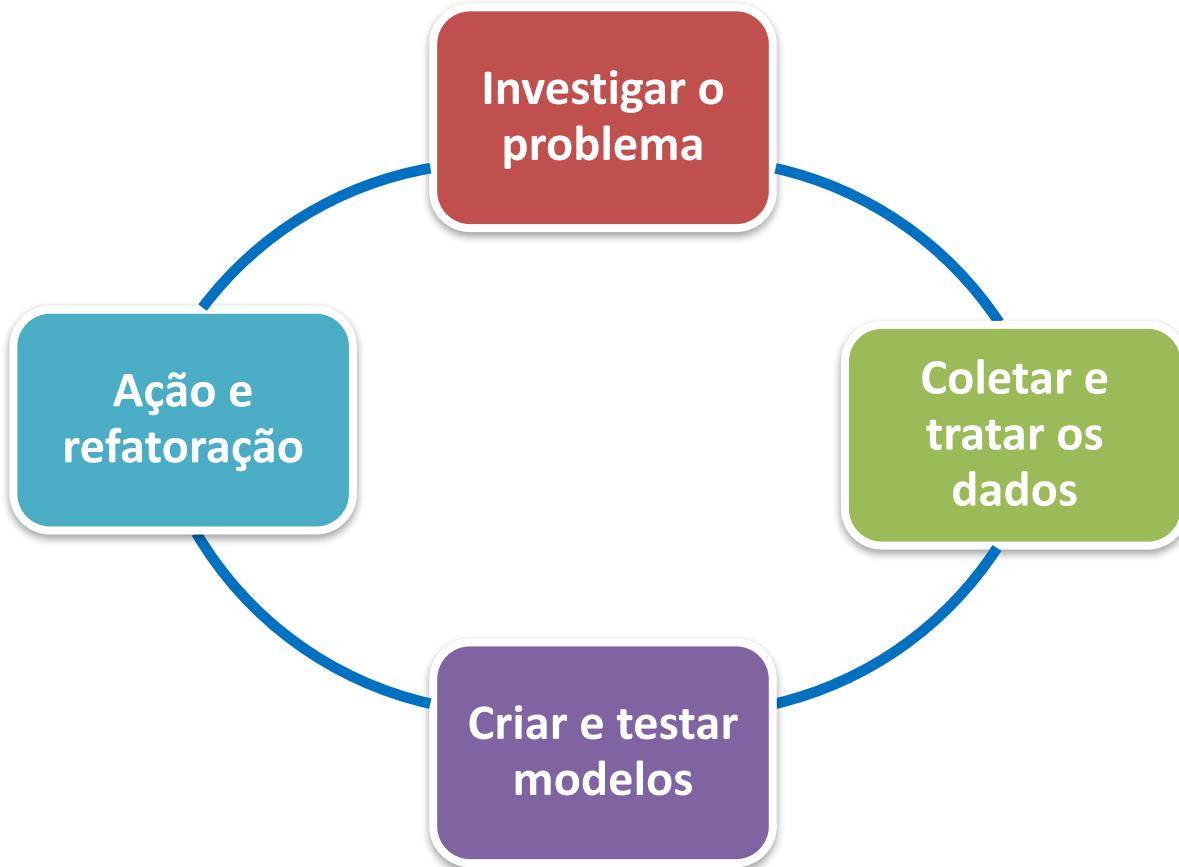




Descoberta de Conhecimento em Bancos de Dados (KDD)

Prof. Juliano Gomes da Silveira

KDD



KDD



Investigação do Problema

- **Não há receita de bolo!**
- Não há uma única forma investigar o problema e também não há uma melhor forma...
- Há algumas formas de fazer essa investigação, mas essa deve customizada empresa-a-empresa, ou ainda, caso-a-caso.
- Depende muito da cultura da empresa...

Investigação do Problema

- Lembra daqueles métodos de **levantamento de requisitos**? Pois é, aqui NÃO vai funcionar...
- As pessoas (clientes) simplesmente não sabem o que querem. E isso é completamente normal.
- Só se sabe que se quer **melhorar** algo que não está bom o suficiente, **resolver** um problema específico ou **prever** algo.
- O requisito é uma frase: “*Quero resolver o problema X!*”

Investigação do Problema

- Diferente do desenvolvimento de software padrão, o Cientista de Dados NÃO é uma entidade passiva.
 - “*Me diga o que você precisa que eu faço!*” [ERRADO!]
 - “*Me explica o problema...*” [CERTO!]



Investigação do Problema

- Acostume-se, pois a partir do momento que você estiver fazendo ciência de dados, o **jogo inverte**, e as pessoas vão fazer aquilo que você diz, ou melhor, aquilo que o resultado do seu trabalho a partir dos dados está sugerindo.
- Quem dita o jogo são os **dados**. A **capacidade técnica** e o **talento** do cientista demonstram os caminhos a partir dos dados.

Investigação do Problema

- Faça entrevistas
 - Prepare algumas perguntas antes e faça todas as perguntas que surgirem durante
 - Entreviste pessoas de **áreas diferentes** que estejam relacionadas com o problema. Tome nota das diferentes percepções
- Faça *brain-storm* com as pessoas envolvidas e com outros cientistas. Tome nota de tudo!

Investigação do Problema

- Simule o problema, fingindo que você é um usuário que enfrenta esse problema.
- Busque o que há de material a respeito:
 - Políticas, procedimentos, intranet...
- Em qualquer interação, pergunte e tome nota:
 - *“Quais dados você acha que estão relacionados ao problema?”*
 - *“Você obtém essas informações de onde?”*
 - Isso será útil na etapa de **coletar os dados**.

Investigação do Problema

- Faça brain-storm com o time de Ciência de Dados
 - Cientistas mais experientes podem dar dicas, sugestões e ajudar a levantar hipóteses.
 - Trabalhos anteriores semelhantes podem ajudar a acelerar o novo trabalho.



Investigação do Problema

- Seja qual for a estratégia, tire os fones de ouvido e **CONVERSE** com as pessoas. Não há melhor forma que essa!



KDD



Coletar Dados

- Se chegamos até aqui, já temos:
 - O **problema**, ou ainda, o **objetivo** do trabalho definido (nem sempre será um problema literal).
 - Uma lista de possíveis **hipóteses** e “chutes”.
 - Uma lista de possíveis **dados** que estão relacionados ao problema em questão.
- Agora é preciso encontrar esses dados.

Coletar Dados

- Se sua empresa possui um ***Data Warehouse*** (*DW*), essa etapa será imensamente mais fácil.
- Verifique se os dados que você procura podem ser extraídos do DW. Fale com o time que construiu ou sustenta este DW, verifique se há documentação disponível. Se não houver, fale com os colegas mais antigos, eles devem saber...

Coletar Dados

- Se não há um DW, essa etapa fica mais complexa e demorada, mas NÃO é impedimento para que os trabalhos de Ciência de Dados prossiga...
- **Como fazer essa coleta e para onde serão enviados os dados para serem trabalhados???**
 - Isso depende do ferramental que sua empresa dispõe para Ciência de Dados, do volume e do tipo de dados que será trabalhado.

Coletar Dados

- Formas e ferramentas comuns para coleta:
 - **Export e import:** faça a extração dos dados em um ou mais arquivos e suba no servidor de Ciência de Dados
 - Ex.: Jupyter:

```
# lendo de um csv
df = pd.read_csv('arquivo.csv', sep=',')
```

Coletar Dados

- Formas e ferramentas comuns para coleta:
 - **Sqoop**: importação via jdbc para um server Hadoop (HDFS). Permite transformação preliminar via SQL de extração e pode ser usado também para exportação.
 - Ex.: Sqoop:

```
scoop import
--connect jdbc: <endereço jdbc e porta da origem>
--username <usuario>
--password <senha>
--table <tabela_origem ou SQL>
--target-dir <endereço no HDFS, ex: hdfs://localhost:8020/>
```

Tratar os Dados

Saiba que você vai usar muito tempo de trabalho nessa etapa!!!



Tratar os Dados – Concatenar

- Consideremos os seguintes DataFrames:

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                   index=[0, 1, 2, 3])
```

```
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                   index=[4, 5, 6, 7])
```

```
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                    'B': ['B8', 'B9', 'B10', 'B11'],
                    'C': ['C8', 'C9', 'C10', 'C11'],
                    'D': ['D8', 'D9', 'D10', 'D11']},
                   index=[8, 9, 10, 11])
```

df1.head()				
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2.head()				
	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3.head()				
	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Tratar os Dados – Concatenar

- Em **pandas**, podemos usar o método ***concat()*** para unir os DataFrames, um abaixo do outro.
- Basta passar uma lista de DataFrames para a função.
- A **ordem da lista** é a mesma que será usada para montar o DataFrame resultado.

```
pd.concat([df2, df1, df3],  
          axis=0)
```

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Tratar os Dados – Mesclar

- Consideremos os seguintes DataFrames:

```
esquerda = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                         'key2': ['K0', 'K1', 'K0', 'K1'],
                         'A': ['A0', 'A1', 'A2', 'A3'],
                         'B': ['B0', 'B1', 'B2', 'B3']})
```



esquerda.head()				
	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

```
direita = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                         'key2': ['K0', 'K0', 'K0', 'K0'],
                         'C': ['C0', 'C1', 'C2', 'C3'],
                         'D': ['D0', 'D1', 'D2', 'D3']})
```



direita.head()				
	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Tratar os Dados – Mesclar

- O método ***merge()*** é semelhante à **SQL**, mescla com base em uma **chave** de um ou mais campos.
- Especifica o tipo de mescla (parâmetro ***how***). Por padrão ***how=inner***, mas pode ser ***outer***, ***left*** ou ***right***.
 - ***inner*** → retorna os registros em comum da chave.
 - ***outer*** → retorna todos, informando nulo para os registros que não houver correspondência na chave.
 - ***left, right*** → retorna todos de um dos lados, mesmo não havendo correspondência no lado oposto. Quando não há correspondência no lado oposto, preenche com nulo.

Tratar os Dados – Mesclar

Ex.: *inner merge()* usando duas chaves.

Retorna apenas os registros correspondentes de ambos os lados:

esquerda.head()				
	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

direita.head()				
	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3



```
pd.merge(esquerda, direita,  
        on=['key1', 'key2'])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Tratar os Dados – Mesclar

Ex.: *outer merge()*

usando duas chaves.

Retorna todos registros,
preenchendo nulo
quando não há
correspondência:

esquerda.head()				
	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

direita.head()				
	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3



```
pd.merge(esquerda, direita,  
        how='outer',  
        on=['key1', 'key2'])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

Tratar os Dados – Mesclar

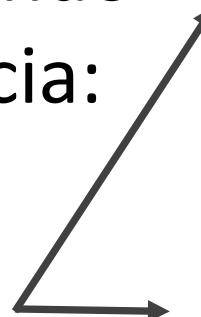
Ex.: *left, right merge()*

usando duas chaves.

Retorna todos registros, de
um dos lados,
preenchendo nulo quando
não há correspondência:

esquerda.head ()				
	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

direita.head ()				
	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3



```
pd.merge(esquerda, direita, how='left',  
        on=['key1', 'key2'])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN

```
pd.merge(esquerda, direita, how='right',  
        on=['key1', 'key2'])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2
3	K2	K0	NaN	NaN	C3	D3

Tratar os Dados – Juntar

- Consideremos os seguintes DataFrames:

```
esquerda = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                         'B': ['B0', 'B1', 'B2']},
                         index=['K0', 'K1', 'K2'])
```

esquerda.head()

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

```
direita = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                         'D': ['D0', 'D2', 'D3']},
                         index=['K0', 'K2', 'K3'])
```

direita.head()

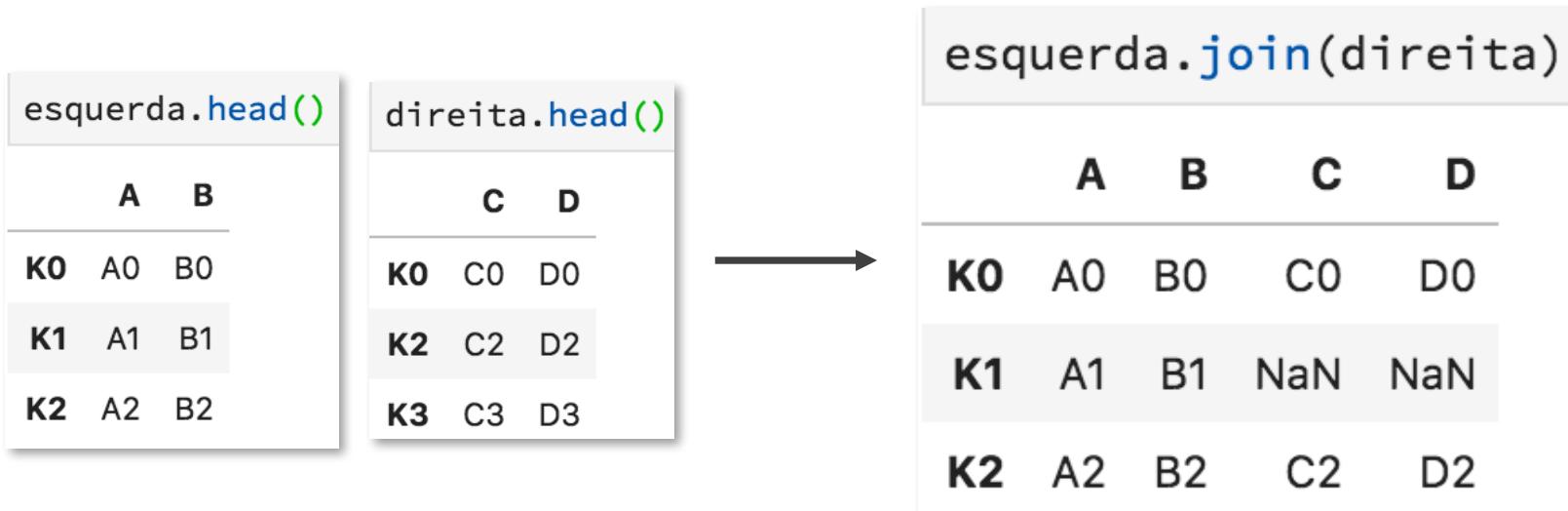
	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

Tratar os Dados – Juntar

- O método *join()* é também originado da sintaxe SQL ANSI.
- Idem ao *merge()*, especifica o tipo de mescla (parâmetro *how*).
- Por padrão *how=left*, mas pode ser *outer*, *left* ou *right*.

Tratar os Dados – Juntar

Ex.: *join()*. Retorna todos registros da esquerda e os que possuem correspondência de chave da direita:



Tratar os Dados – Juntar

Ex.: ***join() how=right***. Retorna todos registros da direita e os que possuem correspondência de chave da esquerda:

The diagram illustrates the join operation using three boxes. On the left, two boxes represent input DataFrames: 'esquerda.head()' and 'direita.head()'. The 'esquerda' box has columns A and B, with rows K0, K1, and K2. The 'direita' box has columns C and D, with rows K0, K2, and K3. An arrow points from these two boxes to a third box on the right, which represents the resulting DataFrame from 'esquerda.join(direita, how='right')'. This resulting box has columns A, B, C, and D, with rows K0, K2, and K3. Row K0 has values A0, B0, NaN, and D0. Row K2 has values A2, B2, C2, and D2. Row K3 has values NaN, NaN, C3, and D3.

esquerda.head()		direita.head()		esquerda.join(direita, how='right')			
A	B	C	D	A	B	C	D
K0	A0	B0		K0	A0	NaN	D0
K1	A1	B1		K2	B2	C2	D2
K2	A2	B2		K3	Nan	Nan	C3

Tratar os Dados – Juntar

Ex.: ***join() how=outer***. Retorna todos registros, incluindo nulo quando não há correspondência de chave:

The diagram illustrates a left outer join operation. On the left, two DataFrames are shown: 'esquerda' (left) and 'direita' (right). The 'esquerda' DataFrame has columns A and B, with keys K0, K1, and K2. The 'direita' DataFrame has columns C and D, with keys K0, K2, and K3. An arrow points from the inputs to the resulting DataFrame on the right, which is generated by the code 'esquerda.join(direita, how='outer')'. This resulting DataFrame has columns A, B, C, and D, containing the values from both DataFrames where they overlap, and NaN where there is no match.

esquerda.head()		direita.head()		esquerda.join(direita, how='outer')			
				A	B	C	D
K0	A0	B0	K0	A0	B0	C0	D0
K1	A1	B1	K2	C2	D2		
K2	A2	B2	K3	C3	D3		



“Em dez/89 estava procurando um hobby que me mantivesse ocupado... Então, criei o Python”

Guido Van Rossum