

# IMD0033 - Probabilidade

## Aula 09 - Pandas Avançado

Ivanovitch Silva  
Março, 2018



# Agenda

---

- Estudo de caso: Titanic
- Imputação de dados
- Higienização dos dados
- Pivoteamento de tabelas
- Aplicando funções sobre o DataFrame

# Atualizar o repositório

---

```
git clone https://github.com/ivanovitchm/imd0033_2018_1.git
```

Ou ....

```
git pull
```

# Estudo de Caso: Titanic

---



kaggle

<https://www.kaggle.com/c/titanic>

# Estudo de caso: Titanic



	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2		St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11		Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female		2	1	2	113781	151.5500	C22 C26	S		Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S		135	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.5500	C22 C26	S			Montreal, PQ / Chesterville,

# Imputação de dados

---

```
sex = titanic_survival["sex"]  
sex_is_null = pandas.isnull(sex)  
sex_null_true = sex[sex_is_null]
```

# Qual o ponto de discussão sobre imputação?

---

```
mean_age = sum(titanic_survival["age"]) / len(titanic_survival["age"])
```

Qual o valor da variável "mean\_age" se algum valor da coluna "age" estiver faltando?

# Algumas facilidades da API Pandas

---

```
correct_mean_age = titanic_survival["age"].mean()
```

Com sorte, a imputação de dados é bastante comum e uma grande maioria de métodos na API Pandas já realiza o filtro de dados faltantes.



# Limpendo os dados faltantes

---

```
drop_na_rows = titanic_survival.dropna(axis=0)
```

# Desafio

---

Qual o valor médio das passagens por classe?

- Exercício seção 5

Qual a idade média dos passageiros por classe?

# Pivoteamento de tabelas

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2		St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11		Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.5500	C22 C26	S			Montreal, PQ / Chesterville, ON

```
passenger_class_fares = titanic_survival.pivot_table(index="pclass",
values="fare", aggfunc=np.mean)
```

# ILOC vs LOC

---

	pclass	survived	name	sex	age
14	1.0	1.0	Barkworth, Mr. Algernon Henry Wilson	male	80.0
61	1.0	1.0	Cavendish, Mrs. Tyrell William (Julia Florence...	female	76.0
1235	3.0	0.0	Svensson, Mr. Johan	male	74.0
135	1.0	0.0	Goldschmidt, Mr. George B	male	71.0
9	1.0	0.0	Artagaveytia, Mr. Ramon	male	71.0

```
first_five_rows = new_titanic_survival.iloc[0:5]
```

# Filtrando dados

---

```
first_row_first_column = new_titanic_survival.iloc[0,0]  
all_rows_first_three_columns = new_titanic_survival.iloc[:,0:3]  
row_index_83_age = new_titanic_survival.loc[83,"age"]  
row_index_766_pclass = new_titanic_survival.loc[766,"pclass"]
```

# Aplicando funções sobre um DataFrame

---

```
# This function returns the hundredth item from a series  
def hundredth_row(column):  
    # Extract the hundredth item  
    hundredth_item = column.iloc[99]  
    return hundredth_item  
# Return the hundredth item from each column  
hundredth_row_var = titanic_survival.apply(hundredth_row)
```

Faça uma função que calcule o número de dados faltantes em todas as colunas

# Aplicando uma função sobre linhas

---

```
def which_class(row):  
    pclass = row['pclass']  
    if pd.isnull(pclass):  
        return "Unknown"  
    elif pclass == 1:  
        return "First Class"  
    elif pclass == 2:  
        return "Second Class"  
    else:  
        return "Third Class"  
classes = titanic_survivors.apply(which_class, axis=1)
```

# Desafio

---

Qual a percentagem de sobreviventes para grupos de diferentes idades?

- 0 - 5 (infantil)
- 6 - 10 (criança)
- 11 - 18 (adolescente)
- 19 - 30 (adulto jovem)
- 31 - 50 (adulto pleno)
- 51 - 65 (adulto senior)
- 66 - (idoso)





```
index.js
import React, { useState } from 'react';
import './index.css';

function App() {
  const [contacts, setContacts] = useState([
    { name: 'John Doe', phone: '123-456-7890' },
    { name: 'Jane Smith', phone: '987-654-3210' }
  ]);

  const handleClick = () => {
    // TODO: Implement the handleClick function
  };

  return (
    <div>
      <h1>React App</h1>
      <button onClick={handleClick}>Click Me</button>
    </div>
  );
}

export default App;
```

```
index.html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
    <script src="index.js"></script>
  </body>
</html>
```