Lesson #05
Working with missing data

July. 2019

- Introduction to Pandas
- Exploring Data with Pandas
- Working with missing data
- Data Aggregation
- Combining Data with Pandas
- Transforming Data with Pandas
- Working with String in Pandas

- Case study: Titanic
- Imputation
- Sanitation
- Pivoting

Case Study: Titanic

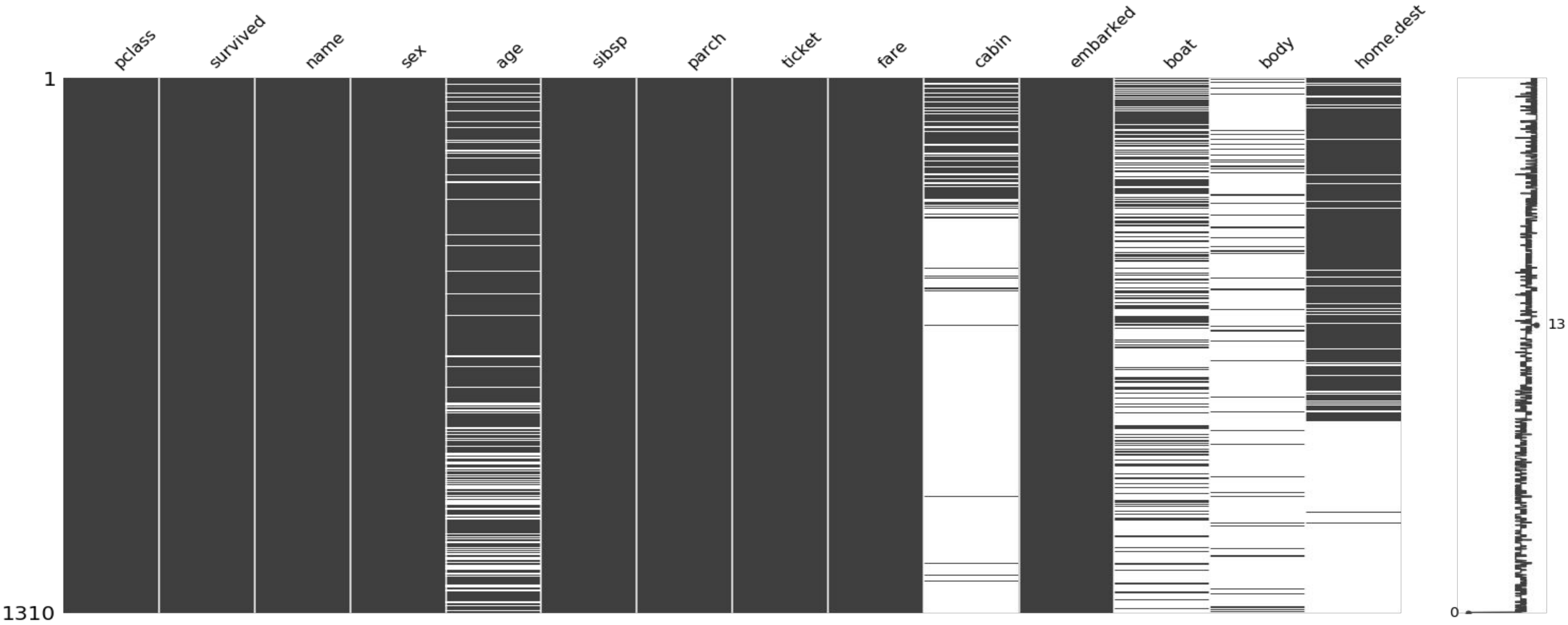
kaggle

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

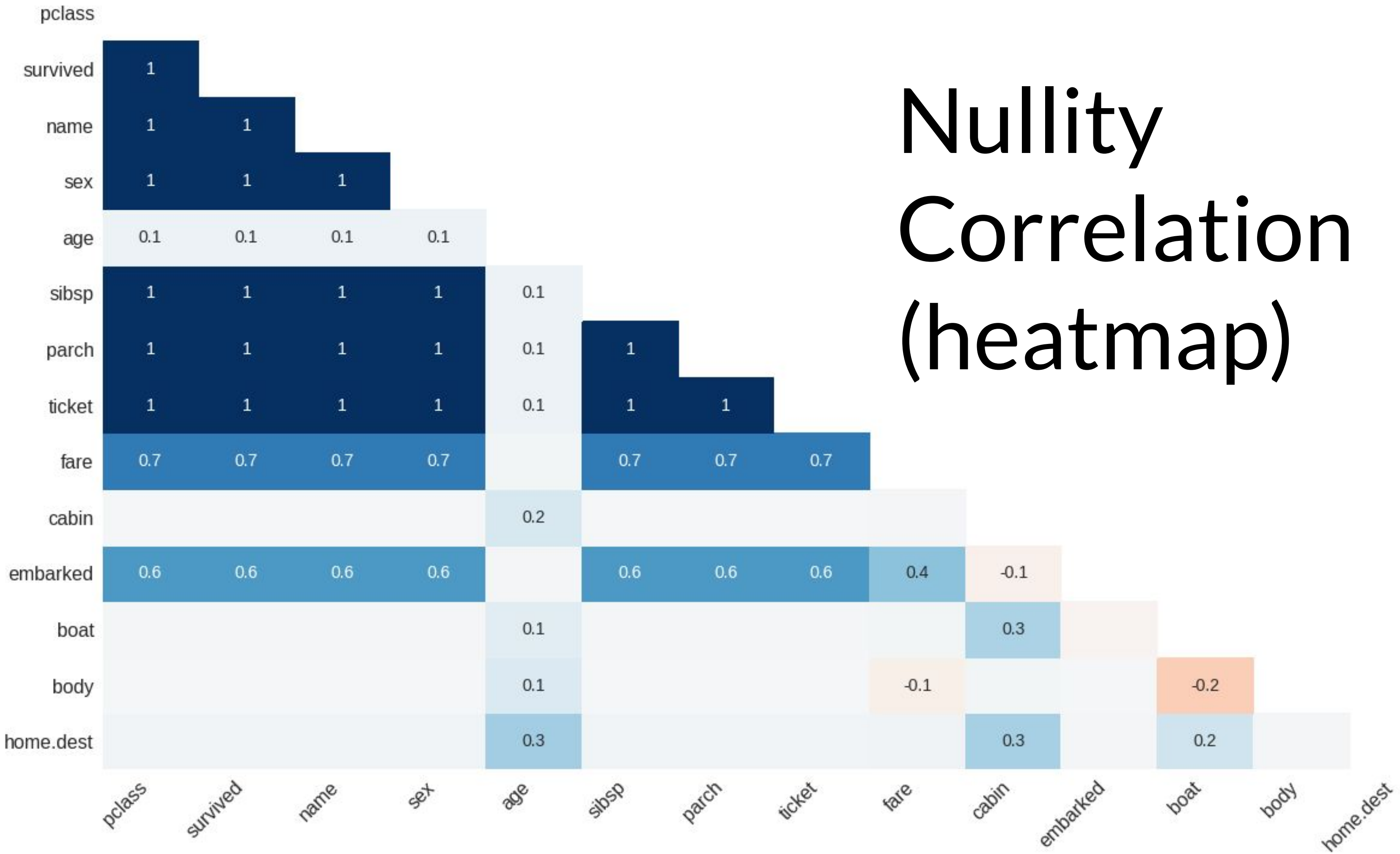
Case Study: Titanic

	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,

Visualizing Missing Data (matrix)



Nullity Correlation (heatmap)



Data Imputation

What do these codes do?

```
titanic_survival.loc[~titanic_survival.age.isnull(), "age"].shape

titanic_survival[~titanic_survival["age"].isnull()].shape
```

What is the discussion point about imputation?

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

What is the value of variable "mean_age" if any row in column "age" is missing?

Some Pandas API features

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

Luckily, data imputation is quite common and a large majority of methods in the Pandas API already filter missing data.

Challenge

What is the average value of tickets per class?
What is the average age of passengers per class?

Lesson 05 - Working with missing data.ipynb

Up to Section 5

Calculating Descriptive Statistics

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

```
fares_by_class = {i:titanic_survival[titanic_survival.pclass == i].fare.mean()
                  for i in titanic_survival.pclass.unique()
                 }
```

```
{1.0: 87.50899164086687, 2.0: 21.1791963898917, 3.0: 13.302888700564957}
```

Pivoting Tables

	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)
```

```python
import numpy as np
df_pivot = titanic_survival.pivot_table(index="pclass",
                                        values=["fare","age"],
                                        aggfunc=[np.mean,len])
```

	mean		len	
	age	fare	age	fare
pclass				
1.0	39.159918	87.508992	323.0	323.0
2.0	29.506705	21.179196	277.0	277.0
3.0	24.816367	13.302889	709.0	709.0

```python
df_pivot["mean"]["age"][1.0]
39.159918
```

Cleaning Data

```
print(df.dropna())
```

	A	B	C	D
w	6.0	3.0	7.0	4.0
y	4.0	3.0	7.0	7.0

```
print(df.dropna(axis=1))
```

	A	B	D
w	6.0	3.0	4.0
x	6.0	2.0	7.0
y	4.0	3.0	7.0
z	2.0	5.0	1.0

	A	B	C	D
w	6.0	3.0	7.0	4.0
x	6.0	2.0	NaN	7.0
y	4.0	3.0	7.0	7.0
z	2.0	5.0	NaN	1.0

Challenge

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)

		count
agecat	**survived**	
Infant	0.0	19
	1.0	37
Child	0.0	17
	1.0	13
Teenager	0.0	62
	1.0	45
Young adult	0.0	263
	1.0	153
Adult	0.0	201
	1.0	141
Senior adult	0.0	49
	1.0	36
Senior	0.0	8
	1.0	2

Another way to aggregate data

```
titanic_survival["agecat"] = pd.cut(titanic_survival.age,
                            bins=[0,5,10,18,30,50,65,100],
                            labels=["Infant","Child","Teenager",
                                    "Young adult","Adult","Senior adult","Senior"])
```

	agecat	age
0	Young adult	29.0000
1	Infant	0.9167
2	Infant	2.0000
3	Young adult	30.0000
4	Young adult	25.0000
5	Adult	48.0000

```
titanic_survival.pivot_table(index=["agecat","survived"],
                             values="age",
                             aggfunc="count")
```

```
aggfunc=lambda x: len(x)/len(titanic_survival[~titanic_survival.age.isnull()])
```


import React from "react";
export default FindContact;
MacBook Pro