Preparing dataframe

# Basics:

1. Let’s start from basic imports:



Figure 1. Imports

From **seaborn** module we will get the “titanic”dataset*.* It is represented as **pandas.DataFrame** class, so we import **pandas** module for efficient work with this dataset. We will store the dataframe in variable named *titanic*:



Figure 2. Dataframe variable

# Learning data types and missing values:

Firstly, let’s look at our dataframe:

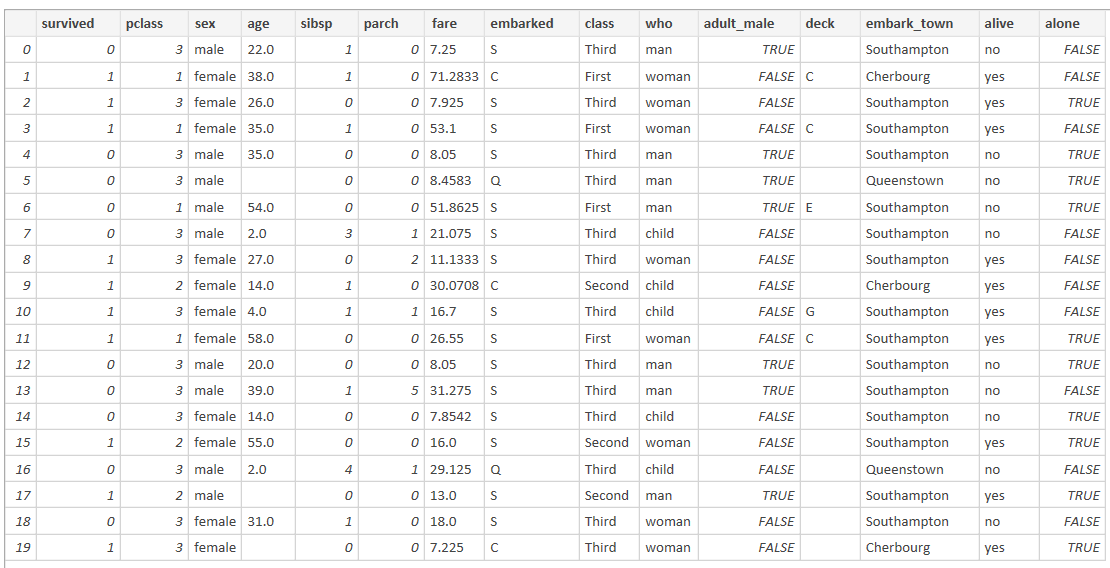


Figure 3. Titanic dataframe

Columns *‘alive’* and *‘survived’* are opposite, so I decided to remove the column ‘alive’. Columns *‘embark’* and *‘embark\_town’* are quite similar, too. **S** in *‘embark’* means **Southampton** in *‘embark\_town’* column, **C** means **Cherbourg** and **Q** – **Queenstown**. So, I decided to remove *‘embark\_town’* from our dataframe. Column *‘adult\_male’* can be got from kind of intersection between columns *‘sex’* and *‘who’*. Let me explain: only **male** and **man** combination makes the column value be **True**. I think it is reasonable to remove this column.



Figure 4. Removing 'useless' columns

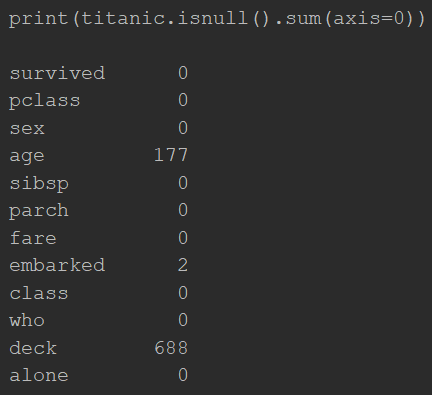
1. Let’s look on amount of missing values in each column:

Figure 5. Amount of missing values in each column

Total amount of people in our dataset is **891**. We have **688** missing values in *‘deck’*, it’s about **77%**! I decided to remove this column because we can’t get much information from it. Column *‘embarked’* have only **2** missing values. I decided to look on frequency of each element in this column.

To do this, I will use class **Counter** from **collections** module:

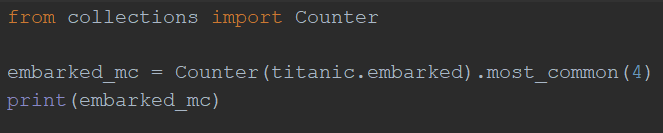


Figure 6. Using 'Counter' to find elements frequency



Figure 7. Using 'Counter' output

Here we can see that the most common value is **‘S’** (644 times, about 72%). Sounds logically to fill missing values with it.



Figure 8. Filling values in 'embarked'

The last column we need to think about is *‘age’*. We can see that *‘sex’* and *‘who’* columns don’t have missing values, and, using them, we can find some information about an **average age** of each group (like **male + man**, **female + child** and so on). I think it is a good idea to fill missing values according to the belonging to groups mentioned above and group’s average age. Here is a code:

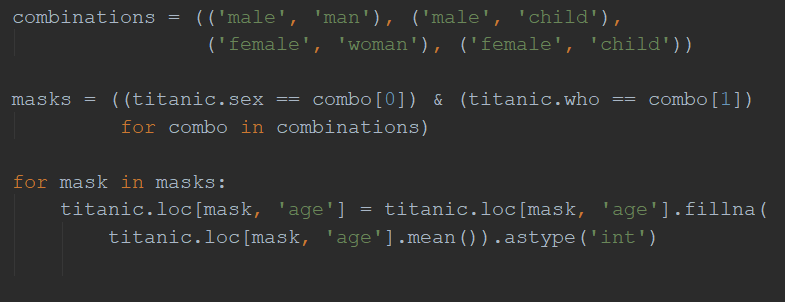


Figure 9. Filling values in 'age'

In **masks** variable we have a list of different groups. In **for** loop we fill **numpy.nan** values with **mean()**. Finally, we have a dataframe with no missing values in it. Next thing we will talk about is *features projecting*.

# Features projecting

To be decided…