**Get the Data with Pandas**

100xp

When the Titanic sank, 1502 of the 2224 passengers and crew were killed. One of the main reasons for this high level of casualties was the lack of lifeboats on this self-proclaimed "unsinkable" ship.

Those that have seen the movie know that some individuals were more likely to survive the sinking (lucky Rose) than others (poor Jack). In this course, you will learn how to apply machine learning techniques to predict a passenger's chance of surviving using Python.

Let's start with loading in the training and testing set into your Python environment. You will use the training set to build your model, and the test set to validate it. The data is stored on the web as csv files; their URLs are already available as character strings in the sample code. You can load this data with the read\_csv() method from the Pandas library.

**Understanding your data**

50xp

Before starting with the actual analysis, it's important to understand the structure of your data. Both test and train are DataFrame objects, the way pandas represent datasets. You can easily explore a DataFrame using the .describe() method. .describe() summarizes the columns/features of the DataFrame, including the count of observations, mean, max and so on. Another useful trick is to look at the dimensions of the DataFrame. This is done by requesting the .shape attribute of your DataFrame object. (ex. your\_data.shape)

The training and test set are already available in the workspace, as train and test. Apply .describe() method and print the .shape attribute of the training set.

**Rose vs Jack, or Female vs Male**

100xp

How many people in your training set survived the disaster with the Titanic? To see this, you can use the value\_counts() method in combination with standard bracket notation to select a single column of a DataFrame:

# absolute numbers

train["Survived"].value\_counts()

# percentages

train["Survived"].value\_counts(normalize = True)

If you run these commands in the console, you'll see that 549 individuals died (62%) and 342 survived (38%). A simple way to predict heuristically could be: "majority wins". This would mean that you will predict every unseen observation to not survive.

To dive in a little deeper we can perform similar counts and percentage calculations on subsets of the Survived column. For example, maybe gender could play a role as well? You can explore this using the .value\_counts() method for a two-way comparison on the number of males and females that survived, with this syntax:

train["Survived"][train["Sex"] == 'male'].value\_counts()

train["Survived"][train["Sex"] == 'female'].value\_counts()

To get proportions, you can again pass in the argument normalize = True to the .value\_counts() method.

**Instructions**

* Calculate and print the survival rates in absolute numbers using values\_counts() method.
* Calculate and print the survival rates as proportions by setting the normalize argument to True.
* Repeat the same calculations but on subsets of survivals based on Sex.

# Does age play a role?

100xp

Another variable that could influence survival is age; since it's probable that children were saved first. You can test this by creating a new column with a categorical variable Child. Child will take the value 1 in cases where age is less than 18, and a value of 0 in cases where age is greater than or equal to 18.

To add this new variable you need to do two things (i) create a new column, and (ii) provide the values for each observation (i.e., row) based on the age of the passenger.

Adding a new column with Pandas in Python is easy and can be done via the following syntax:

your\_data["new\_var"] = 0

This code would create a new column in the train DataFrame titled new\_var with 0 for each observation.

To set the values based on the age of the passenger, you make use of a boolean test inside the square bracket operator. With the []-operator you create a subset of rows and assign a value to a certain variable of that subset of observations. For example,

train["new\_var"][train["Fare"] > 10] = 1

would give a value of 1 to the variable new\_var for the subset of passengers whose fares greater than 10. Remember that new\_var has a value of 0 for all other values (including missing values).

A new column called Child in the train data frame has been created for you that takes the value NaN for all observations.

## Instructions

* Set the values of Child to 1 is the passenger's age is less than 18 years.
* Then assign the value 0 to observations where the passenger is greater than or equal to 18 years in the new Child column.
* Compare the normalized survival rates for those who are <18 and those who are older. Use code similar to what you had in the previous exercise.