The Titanic

Giorgio De Nunzio
giorgio.denunzio@unisalento.it

RMS Titanic was a British passenger liner operated by the White Star Line that sank in the North Atlantic Ocean on 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City. Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making the sinking one of the deadliest of a single ship and the deadliest peacetime sinking of a superliner or cruise ship to date.[1]

The Kaggle Web site, well known reference for machine/deep learning competitions, proposed some time ago a dataset concerning the Titanic sinking, where each passenger was assigned some variables (features) concerning several pieces of information (e.g. age, sex, relatives on board, ticket class, and so on), and her/his survival to the sinking.

This small project consists in taking the data[2] (also attached to this text, titanic_train.csv) and applying a ML model to predict the survival state of the passengers from the given features. The dataset is in a csv file, i.e. rows contain comma-separated values. The first line accommodates the column headers.

For ease of reference, here is a short description of the data fields.

First two lines of the file (column headers and a line of data):

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
```

Here are the various fields n the file line (numbered from 0 in pythonic stile):

0. PassengerId (Identification number, int64)
1. Survived (did the passenger survive? 0 = No, 1 = Yes, int64: this is the target variable!)
2. Pclass (Ticket class: 1 = 1st, 2 = 2nd, 3 = 3rd, int64; A proxy for socio-economic status (SES))
3. Name (type: "object", actually a string)
4. Sex (female or male, type: "object", actually a string to be converted to 0/1)
5. Age (Age in years, int64; Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5))
6. SibSp (# of siblings / spouses aboard the Titanic, int64; Sibling = brother, sister, stepbrother, stepsister; Spouse = husband, wife (mistresses and fiancés were ignored))
7. Parch (# of parents / children aboard the Titanic, int64; Parent = mother, father; Child = daughter, son, stepdaughter, stepson; Some children travelled only with a nanny, so parch=0 for them.)
8. Ticket (Ticket number, type: "object", actually a string)
9. Fare (Passenger fare, float64)
10. Cabin (Cabin number, type: "object", actually a string)
11. Embarked (Port of Embarkation: C = Cherbourg, Q = Queenstown, S = Southampton; type: "object", actually a string)

Your mission, should you choose to accept it (!), is to build a prediction model in python.

[1] https://en.wikipedia.org/wiki/Titanic
[2] https://www.kaggle.com/c/titanic, https://www.kaggle.com/c/titanic/data

Remarks:

a. You can read the file inserting the data into a Panda dataframe (if you follow the guidelines I gave, but do whatever you wish); consider that the first line contains the headers.
b. You have to convert 'male', 'female' into some numeric values, e.g. 0, 1. You can use dataset.Sex.replace(...)
c. You have to convert 'C', 'Q', S' into some numeric values, e.g. 0, 1, 2. Ditto.
d. You have to use column 1 as the target variable.
e. In my opinion, you should only preserve columns 2, 4, 5, 6, 7, 9, 11 as the features. Do it (or change my proposal) motivating in your mind the reasons (I will maybe ask you about it).
f. Warning! When I put the target column into a numpy.ndarray called Y, I ended up with `Y.dtype` being 'O', object, and my ML model complained saying it was of type 'unknown'. This is visible e.g. by

```
from sklearn.utils.multiclass import type_of_target
type_of_target(Y)
```

which gives 'unknown'. I had to solve the problem by explicitly converting it to numbers:

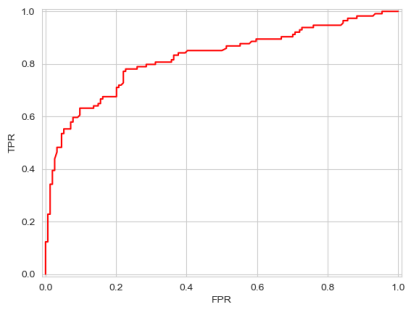
```
Y = Y.astype('int')
```

What to do?

a. In your code, you should before all show some statistics of the data, in particular the histograms of the single variables (per class) and the scatter plots of coupled features (per class). The purpose is to try and understand if these variables can be discriminant. Choose whatever pythonic graph-making library you like. Again, meditate on the graphs, maybe we shall discuss them later.
b. Then consider that there are some NaNs in the data: how can you check which columns contain NaNs? I did not say this, just google it, it is very easy to find...
c. Employ a XGBoost model, which already contains a procedure for missing-data imputation (so it works out-of-the-box with NaNs) and write some code which uses in sequence:
    i. hold out with feature normalization
    ii. k-fold cross validation with k = 5 (without feature normalization: implementing it involves some more work we did not see during the course[3])

   each time returning a ROC curve and its AUC. Are the AUCs similar? What happens if you run the code multiple times? Is one of the two approaches (hold out and k-fold cv) more stable? You might even accumulate the various AUCs and give a standard deviation, but this is not required.
d. Drop the columns containing NaNs (as found at step b) and change the model to a multilayer perceptron (play with the hidden layers). Any change?

Just as an example, here is the ROC I got by XGBoost with the mentioned features (AUC = 0.85).



Write your comments, insert your results and graphs into a document and send it to me!

Good work.

Giorgio De Nunzio

[3] https://stackoverflow.com/questions/44446501/how-to-standardize-data-with-sklearns-cross-val-score