



Machine Learning Project from Titanic DB Using Python



Luigi Mennella

Project Description

Starting data: DB Titanic, version available in the Seaborn library .

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Purpose: To train a machine learning algorithm to predict the behavior of the target variable *survived* with respect to the features. Records = 891.

	survived	pclass	age	sibsp	parch	fare	adult_male	alone	sex_female	sex_male	...	deck_A	deck_B	deck_C
0	0	3	22.0	1	0	7.2500	True	False	False	True	...	False	False	False
1	1	1	38.0	1	0	71.2833	False	False	True	False	...	False	False	True
2	1	3	26.0	0	0	7.9250	False	True	True	False	...	False	False	False
3	1	1	35.0	1	0	53.1000	False	False	True	False	...	False	False	True
4	0	3	35.0	0	0	8.0500	True	True	False	True	...	False	False	False
...
886	0	2	27.0	0	0	13.0000	True	True	False	True	...	False	False	False
887	1	1	19.0	0	0	30.0000	False	True	True	False	...	False	True	False
888	0	3	NaN	1	2	23.4500	False	False	True	False	...	False	False	False
889	1	1	26.0	0	0	30.0000	True	True	False	True	...	False	False	True
890	0	3	32.0	0	0	7.7500	True	True	False	True	...	False	False	False

891 rows × 29 columns

Features description

pclass : passenger's travel class (1, 2, 3)

sex: gender of the passenger (male, female)

age: age of the passenger

sibsp : number of siblings or spouses

parch : number of parents or children

fare: ticket price

embarked : place of embarkation

class: travel class (e.g. first)

who : man/woman/child

adult_male : true = adult male; false = other

deck: ship's deck (A, B, C...G, NaN)

embark_town : embarkation city (full name)

alive : survived (yes/no)

alone: alone (true) or accompanied (false)

Redundant Features Analysis

Qualitative analysis of features highlights redundancy between the following variables:

`alive` : redundant with the target variable `survived`

`pclass` : redundant to `class`

`embark_town` : redundant to `embarked`

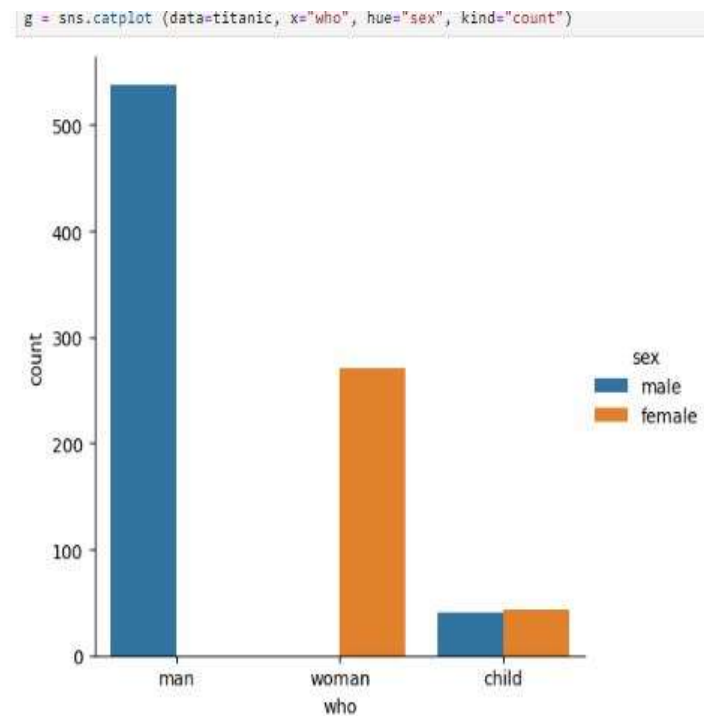
`sex` and `adult_male` : both redundant with `who`

```
titanic.groupby(by=["survived", "alive"],
                 as_index=False,
                 dropna=False) \
    .agg(conteggio = ("survived",
                     np.size))
```

	survived	alive	conteggio
0	0	no	549
1	1	yes	342

```
#Ridondanza anche tra class e pclass
pd.crosstab(titanic["class"], titanic[
```

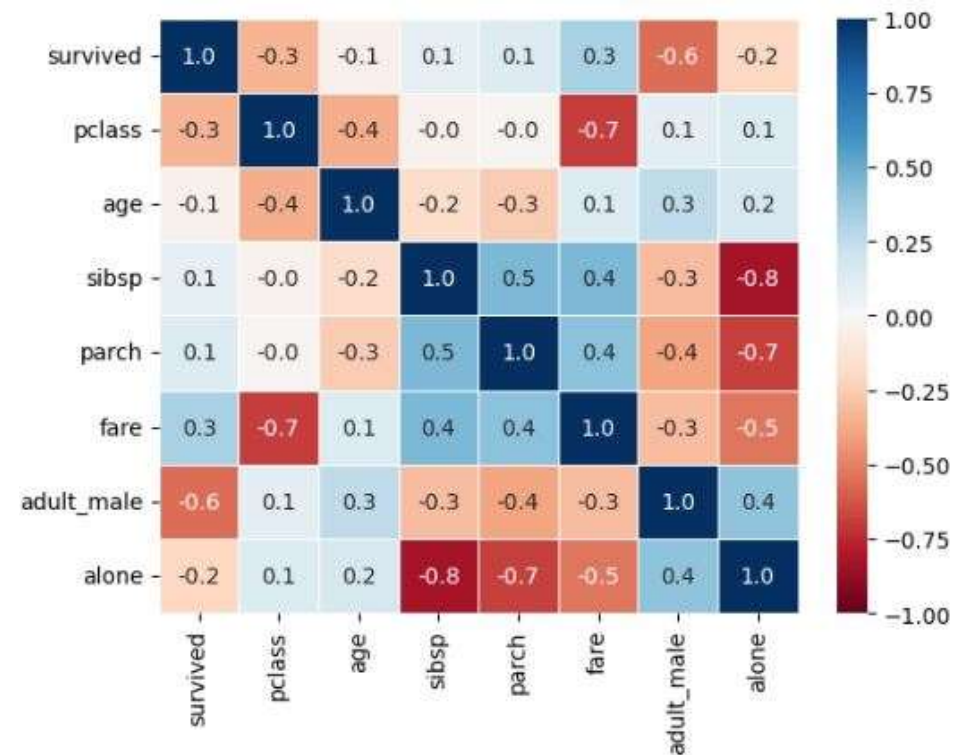
pclass	1	2	3
class			
First	216	0	0
Second	0	184	0
Third	0	0	491



Correlation analysis

The correlation analysis, also performed with the aid of heat maps (linear and Spearman), highlights the presence of some strongly correlated variables.

In particular, the correlation between the *alone variables* with respect to *sibsp* and *parch* is very strong .



Elimination of redundant or missing features

Qualitative feature analysis suggests eliminating the following variables:

alive : redundant with the target variable

sex and adult_male : : redundant to *who*

pclass : redundant to *class*

embark_town : redundant to *embarked*

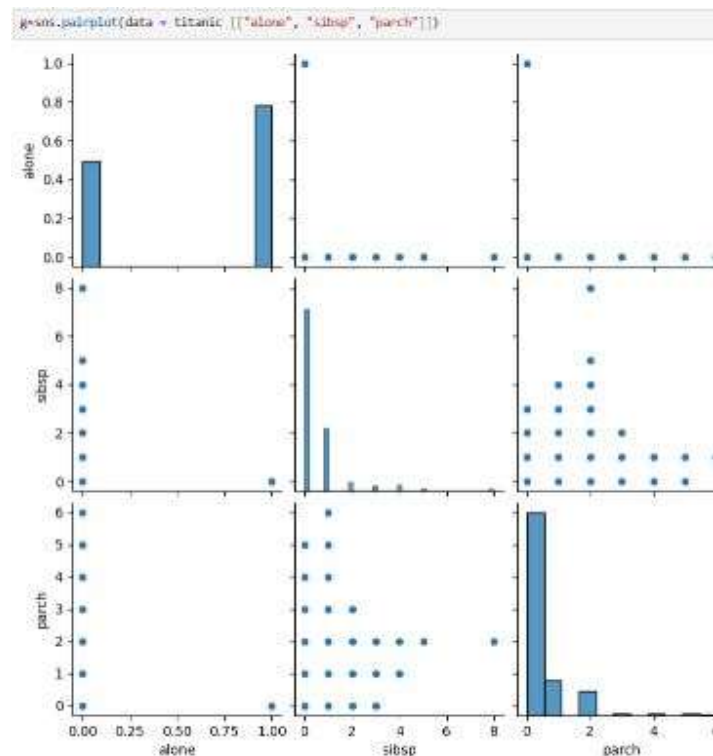
alone: redundant with respect to *sibsp*

parch

deck: has many nulls.

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 15 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   survived    891 non-null    int64  
1   pclass      891 non-null    int64  
2   sex         891 non-null    object  
3   age         714 non-null    float64  
4   sibsp       891 non-null    int64  
5   parch       891 non-null    int64  
6   fare        891 non-null    float64  
7   embarked    889 non-null    object  
8   class       891 non-null    category  
9   who         891 non-null    object  
10  adult_male   891 non-null    bool  
11  deck         203 non-null    category  
12  embark_town  889 non-null    object  
13  alive        891 non-null    object  
14  alone        891 non-null    bool
```



Splitting data into training and testing

Thanks to the `train_test_split` class of `scikit-learn` the DB is divided into a training part and a testing part.

```
X = titanic.drop("survived",axis=1).values
Xcolumns = titanic.drop("survived",axis=1).columns
y = titanic["survived"].values
```

```
[51]: # creiamo i db di training e test
      x1,x2,y1,y2 = train_test_split(X,
                                   y,
                                   test_size = 0.3,
                                   shuffle = True,
                                   random_state = 1)
```

```
[52]: Xcolumns
```

```
[52]: Index(['pclass', 'age', 'sibsp', 'parch', 'fare', 'embarked', 'who'], dtype='object')
```

Null management

Among the residual variables, only *age* and *embarked* present some null values.

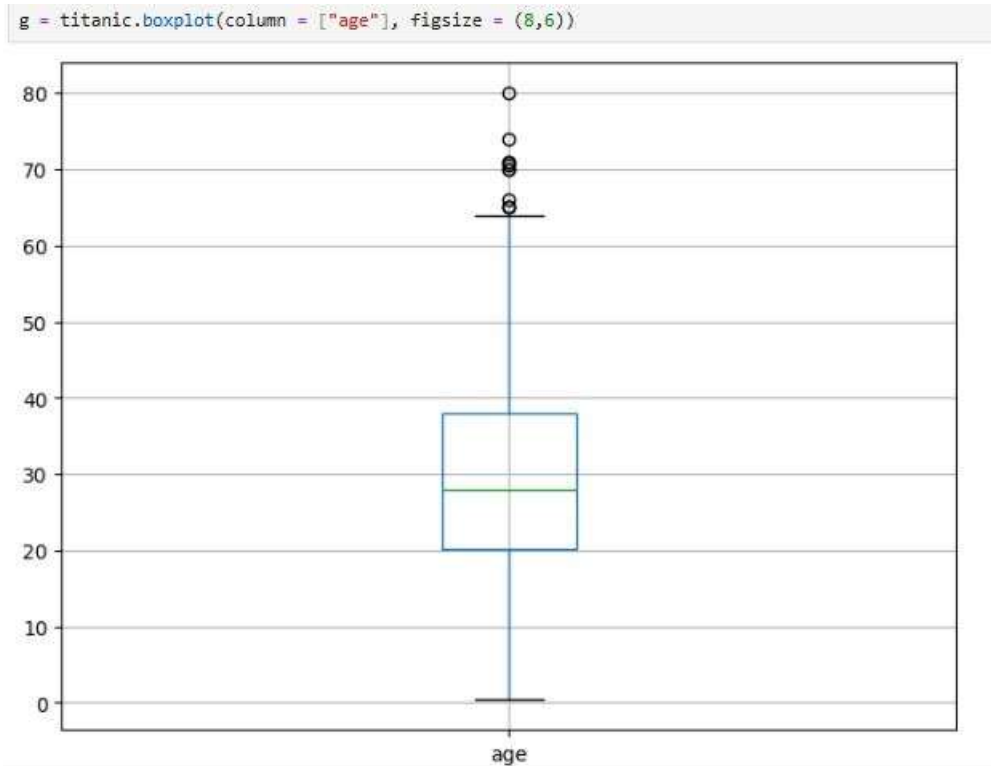
After studying the variables, we decide to replace them like this:

age: median («28»)

embarked : fashion

(«S»).

	embarked	size
0	C	168
1	Q	77
2	S	644
3	NaN	2



Creating the pipeline

A pipeline is created using the following classes from the scikit-learn library :

- ✓ SimpleImputer : to valorize *nulls*
- ✓ StandardScaler to normalize quantitative variables
- ✓ OneHotEncoder for encoding nominal qualitative variables
- ✓ OrdinalEncoder to encode ordinal qualitative variables
- ✓ LogisticRegression as classification algorithm
- ✓ MakeColumnTransformer to apply methods to different columns

```
null_handling = make_column_transformer(  
    (SimpleImputer(strategy='median'), [0,1,2,3]),  
    (SimpleImputer(strategy='most_frequent'), [4,5,6]),  
    remainder='passthrough')
```

```
pipeline = Pipeline([('vn',valorizza_null),  
                     ('pr',preprocessing),  
                     ('lr', LogisticRegression(random_state=0))  
                     ])
```

```
preprocessing = make_column_transformer(  
    (StandardScaler(), [0,1,2,3]),  
    (OneHotEncoder(handle_unknown='ignore',drop = 'first'), [4,6]),  
    (OrdinalEncoder(categories=[["First","Second","Third"]],handle_unknown="use_encoded_value",unknown_value=4),[5]),  
    remainder='passthrough')
```

Parameter tuning and model accuracy

GridSearchCV class from scikit-learn to verify the choice of the median to replace the nulls of the quantitative parameters. The choice is confirmed.

- ✓ The fitting on the training sample is 0.836
- ✓ The fitting value that would be obtained with a model without pre -processing is slightly lower (0.833), so the operations performed are validated.
- ✓ The fit to the test data is 0.791, which is lower but still good.

Confusion matrix

A further element to evaluate the goodness of the model is the confusion matrix.

- ✓ On the main diagonal there are the **correct solutions**, that is 133 non-survivors (0;0) and 79 non-survivors (1;1)
- ✓ 20 are the **errors of the first type**, that is those predicted as not surviving but actually surviving (0;1)
- ✓ 36 are the **second type errors**, that is, those expected to survive but in reality did not survive (1;0).

```
array([[133, 20],  
       [ 36, 79]])
```