Titanic data

Bruna Wundervald

October, 2018

```
# Packages and models import
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from pandas.api.types import is_object_dtype
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Intro

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

Data Dictionary

- `Survival`: Survival, or 0 = No, 1 = Yes
- `Pclass`: Ticket class, or 1 = 1st, 2 = 2nd, 3 = 3rd
- `Sex`: Sex
- `Age`: Age in years
- `Sibsp`: number of siblings/spouses aboard the Titanic
- `Parch`: number of parents/children aboard the Titanic
- `Ticket`: Ticket number
- `Fare`: Passenger fare
- `Cabin` Cabin number

- `Embarked`: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

```
# Reading data
da =  pd.read_csv('data/titanic/train.csv')

# Replacing NaNs with 0
da.fillna(0, inplace = True)
da.iloc[0]
```

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

```
da['TotalFamily'] = da['SibSp'] + da['Parch']

da.shape

# Proportion of the response variable
```

```
## (891, 13)
```

```
props = round(da['Survived'].value_counts()/891, 2)

print('The proportions are:', props[1]*100,'% (Survived) and', props[0]*100,'% (Did not survived)')
```

```
## The proportions are: 38.0 % (Survived) and 62.0 % (Did not survived)
```

Exploratory data analysis

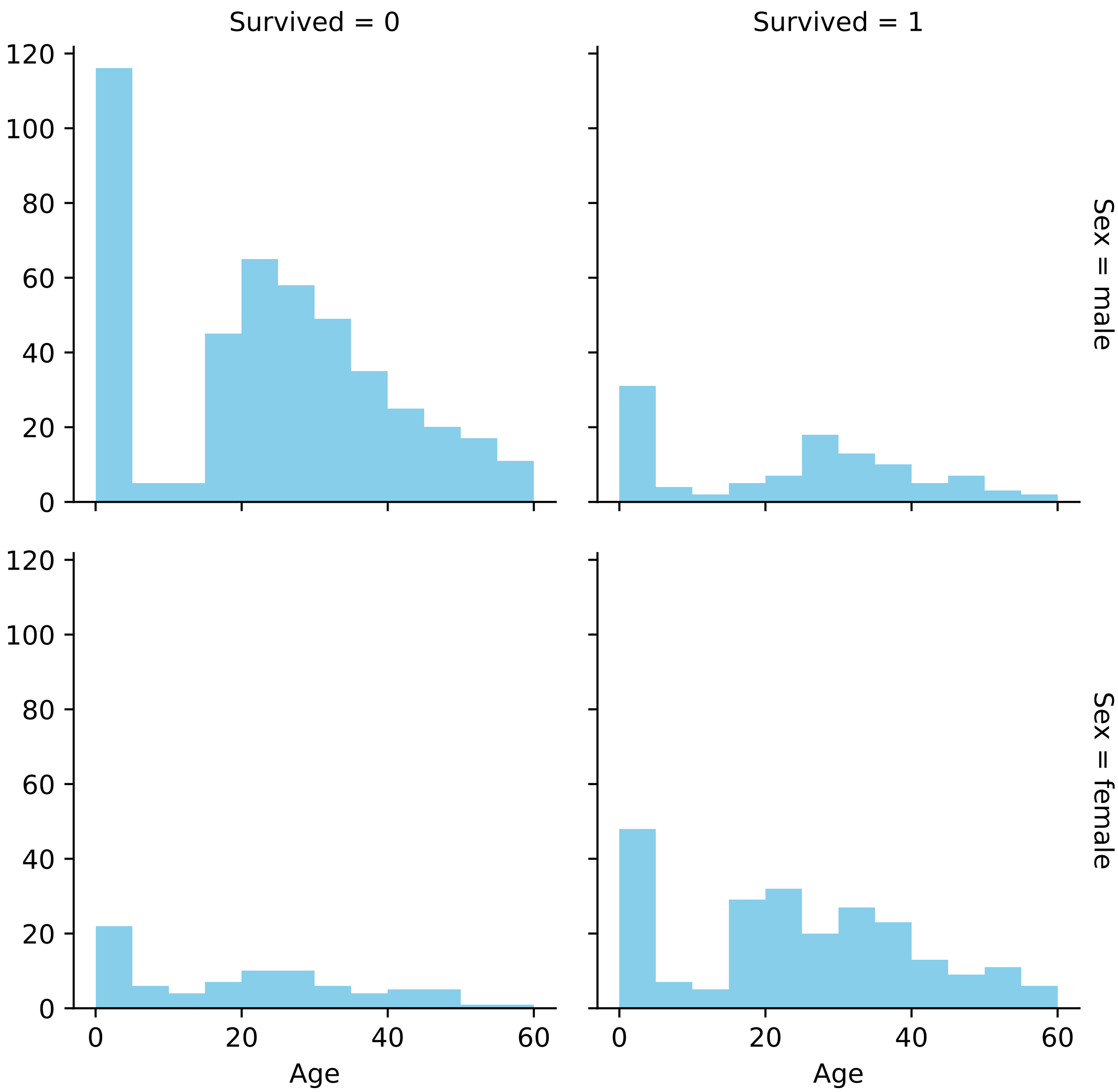
Initial checks for evidence of differences between the distributions of people who survived and who did not. The sex was also included.

```
# Exploratory data analysis -------------------------------------------
plt.clf()
g = sns.FacetGrid(da, row="Sex", col="Survived", margin_titles=True)
bins = np.linspace(0, 60, 13)
g.map(plt.hist, "Age", color="skyblue", bins=bins)
```

```
## <seaborn.axisgrid.FacetGrid object at 0x1a2667fcf8>
```

```
plt.show()
```

```
## 'titanic_files/figure-latex/unnamed-chunk-3-1.pdf'
```



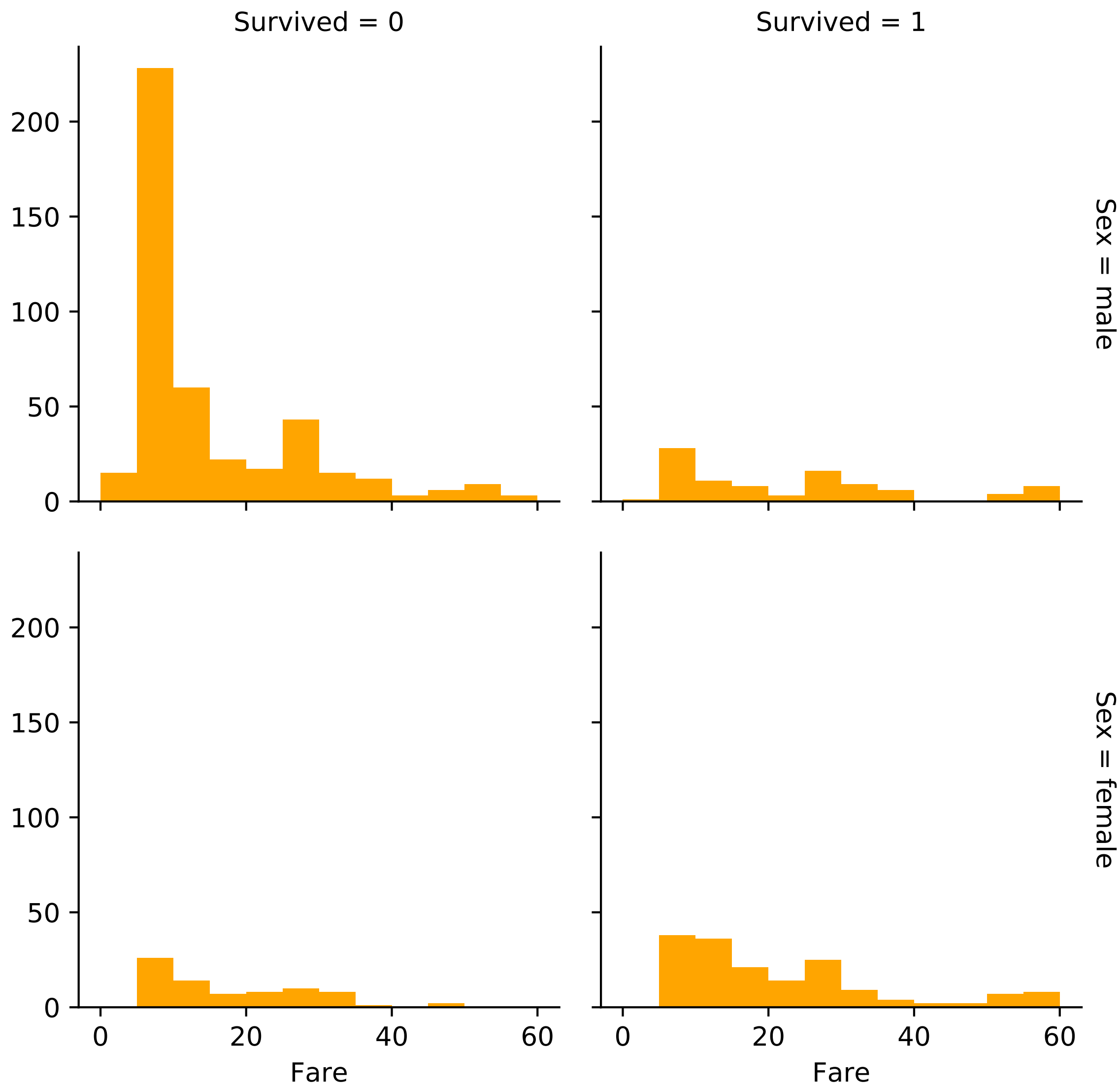
For the age, the majority of people who survived are women of all ages. Nevertheless, for both sexes, there is a concentration (a peak) of younger people who survived. Anyway, not all of the children survived: there is also a concentration in young men who did not survived.

```
plt.clf()
g = sns.FacetGrid(da, row="Sex", col="Survived", margin_titles=True)
bins = np.linspace(0, 60, 13)
g.map(plt.hist, "Fare", color="orange", bins=bins)
```

```
## <seaborn.axisgrid.FacetGrid object at 0x121383fd0>
```

```
plt.show()
```

```
## 'titanic_files/figure-latex/unnamed-chunk-4-1.pdf'
```



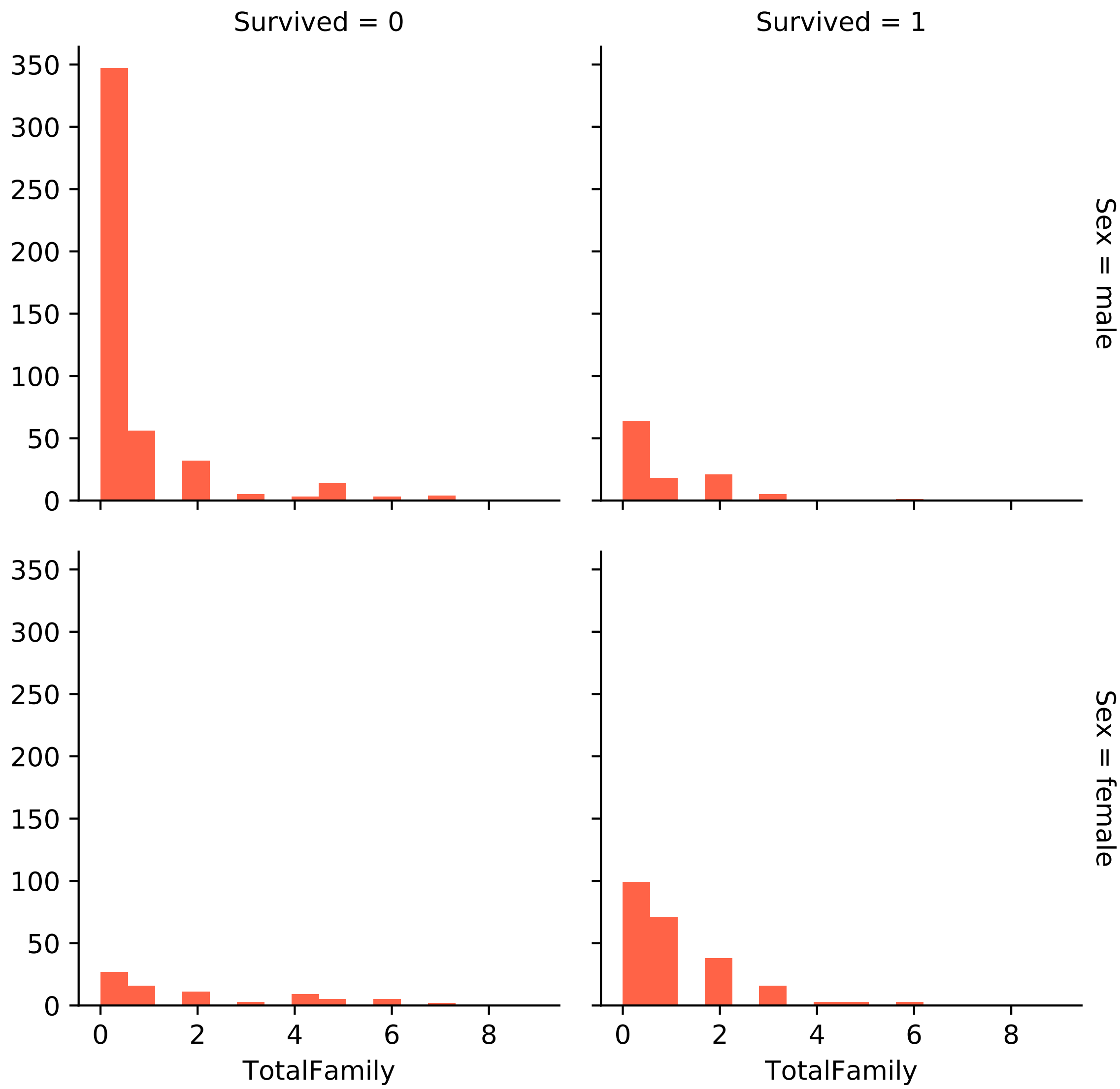
For the fares, we can clearly see that the ones who paid less are the majority of the ones who did not survived.

```
g = sns.FacetGrid(da, row="Sex", col="Survived", margin_titles=True)
bins = np.linspace(0, 9, 17)
g.map(plt.hist, "TotalFamily", color="tomato", bins=bins)
```

```
## <seaborn.axisgrid.FacetGrid object at 0x1a279770f0>
```

```
plt.show()
```

```
## 'titanic_files/figure-latex/unnamed-chunk-5-1.pdf'
```



Most people who survived had just a few family members in the ship, but this is a general characteristic of people in the Titanic.

```
# Selecting interesting variables
X = da.drop(['Survived', 'PassengerId', 'Name', 'Ticket', 'Cabin'], axis = 1)
y = da['Survived']

# Extracting the last name to use as a covariable
X['LastName'] = da['Name'].str.rsplit(',').str[0]
X.columns

# Columns chosen as predictors: 'Pclass', 'Sex', 'Age', 'SibSp',
# 'Parch', 'Fare', 'Embarked', 'LastName'

# Converting factors to dummies ------------------------------------
```

```
## Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked',
##        'TotalFamily', 'LastName'],
```

```
##       dtype='object')
```

```
def dummy(var):
  X[var] = X[var].astype('category')
  X[var] = X[var].cat.codes
  return X

columns = X.columns

for index in range(0, len(columns)):
  if is_object_dtype(X[columns[index]]) == True:
    X = dummy(columns[index])
  else:
    X = X

X.dtypes # All ok

# Train and test split (automatic function)
```

```
## Pclass           int64
## Sex               int8
## Age            float64
## SibSp            int64
## Parch            int64
## Fare           float64
## Embarked          int8
## TotalFamily      int64
## LastName         int16
## dtype: object
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4)


# Setting the model(s)
# -----------------------------
# 1. Logistic Regression
# 2. SVM
# 3. Linear Discriminant Analysis
# --------------------------------

# 1. Logistic regression
model_lr = LogisticRegression(random_state=0, solver='lbfgs')
model_lr.fit(X_train, y_train)

# % of correct classifications
```

```
## LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
##           intercept_scaling=1, max_iter=100, multi_class='warn',
##           n_jobs=None, penalty='l2', random_state=0, solver='lbfgs',
##           tol=0.0001, verbose=0, warm_start=False)
##
## /Users/brunawundervald/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:757: C
##   "of iterations.", ConvergenceWarning)
```

```
model_lr.score(X_test, y_test)
```

```
## 0.8470149253731343
```

```
pred_lr = model_lr.predict(X_test)

# Confusion matrix
np.round(confusion_matrix(pred_lr, y_test)/2.68, 1)

# Logistic regression: gets most "non-survivals" right

# SVM
```

```
## array([[61.9, 10.8],
##        [ 4.5, 22.8]])
```

```
model_svm = svm.SVC()
model_svm.fit(X_train, y_train)

# % of correct classifications
```

```
## SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
##   decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
##   kernel='rbf', max_iter=-1, probability=False, random_state=None,
##   shrinking=True, tol=0.001, verbose=False)
##
## /Users/brunawundervald/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:196: FutureWarning:
##   "avoid this warning.", FutureWarning)
```

```
model_svm.score(X_test, y_test)
```

```
## 0.6716417910447762
```

```
pred_svm = model_svm.predict(X_test)

# Confusion matrix
np.round(confusion_matrix(pred_svm, y_test)/2.68, 1)

# SVM: gets almost all "non-survivals" right, but also gets most survivals wrong
# Why?

# LDA
```

```
## array([[64.2, 30.6],
##        [ 2.2,  3. ]])
```

```
model_lda = LinearDiscriminantAnalysis()
model_lda.fit(X_train, y_train)

# % of correct classifications
```

```
## LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
##               solver='svd', store_covariance=False, tol=0.0001)
##
## /Users/brunawundervald/anaconda3/lib/python3.7/site-packages/sklearn/discriminant_analysis.py:388: U
##   warnings.warn("Variables are collinear.")
```

```
model_lda.score(X_test, y_test)
```

```
## 0.8134328358208955
```

```
pred_lda = model_lda.predict(X_test)

# Confusion matrix
np.round(confusion_matrix(pred_lda, y_test)/2.68, 1)

# LDA: gets most "non-survivals" right, but also gets some survivals wrong
```

```
## array([[58.2, 10.4],
##        [ 8.2, 23.1]])
```

Conclusions

The logistic regression showed the best results so far, because it actually gets the survivals right, while the other methods fail to accomplish this task properly. We can then combine the best of two models, for example:

```
# The "semi-ensemble" model
y_test2 = np.array(y_test[:, ])
final_pred = pred_svm

for index in range(0, len(y_test)):
  if  pred_lr[index] == 1:
    final_pred[index] = pred_lr[index]
  else:
    final_pred[index] = pred_svm[index]

# Confusion matrix
mat = np.round(confusion_matrix(final_pred, y_test2)/2.68, 1)
mat
# Accuracy is higher:
```

```
## array([[60.1,  9.7],
##        [ 6.3, 23.9]])
```

```
mat[1, 1] + mat[0,0]
```

```
## 84.0
```