# Herramientas de Machine Learning

# Preliminares


- vectores y matrices de N-dimensiones
- álgebra lineal, transformadas de Fourier y generación aleatoria
- madurez del proyecto y excelente documentación
- integración con código C/C++
- software libre (BSD)

http://www.numpy.org/

Version utilizada: 1.8.2 o superior


- procesamiento, visualización y predicción de datos
- orientado a usuarios finales
- madurez del proyecto y excelente documentación
- software libre (BSD)

http://scikit-learn.org/stable/

Version utilizada: 0.18.1


- procesamiento y manejo eficiente de datos
- fácil de usar
- excelente documentación
- software libre (BSD)

http://pandas.pydata.org/pandas-docs/stable/

Version utilizada: 0.19.2

# Documentación

http://scikit-learn.org/stable/documentation.html (0.18)

https://docs.scipy.org/doc/numpy/ (1.12)

http://pandas.pydata.org/pandas-docs/stable/ (0.19.2)

# Breve introducción a Pandas

# Introducción a DataFrames

```
>>> import pandas as pd


>>> df = pd.DataFrame({ 'A' : 1.,
   ....:                'B' : pd.Timestamp('20130102'),
   ....:                'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
   ....:                'D' : np.array([3] * 4,dtype='int32'),
   ....:                'E' : pd.Categorical(["test","train","test","train"]),
   ....:                'F' : 'foo' })
   ....:

>>> df
     A          B    C  D      E    F
0  1.0 2013-01-02  1.0  3   test  foo
1  1.0 2013-01-02  1.0  3  train  foo
2  1.0 2013-01-02  1.0  3   test  foo
3  1.0 2013-01-02  1.0  3  train  foo
```

# Introducción a DataFrames

```
>>> dates = pd.date_range('20130101', periods=6)

>>> dates
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

>>> df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))

>>> df
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

# Examinando datos

```
>>> df.head()
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401

>>> df.tail(3)
                   A         B         C         D
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988

>>> df.describe()
              A         B         C         D
count  6.000000  6.000000  6.000000  6.000000
mean   0.073711 -0.431125 -0.687758 -0.233103
std    0.843157  0.922818  0.779887  0.973118
min   -0.861849 -2.104569 -1.509059 -1.135632
25%   -0.611510 -0.600794 -1.368714 -1.076610
50%    0.022070 -0.228039 -0.767252 -0.386188
75%    0.658444  0.041933 -0.034326  0.461706
max    1.212112  0.567020  0.276232  1.071804
```

# Ordenando datos

```
>>> df.sort_index(axis=1, ascending=False)
                   D         C         B         A
2013-01-01 -1.135632 -1.509059 -0.282863  0.469112
2013-01-02 -1.044236  0.119209 -0.173215  1.212112
2013-01-03  1.071804 -0.494929 -2.104569 -0.861849
2013-01-04  0.271860 -1.039575 -0.706771  0.721555
2013-01-05 -1.087401  0.276232  0.567020 -0.424972
2013-01-06  0.524988 -1.478427  0.113648 -0.673690

>>> df.sort_values(by='B')
                   A         B         C         D
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
```

# Seleccionando columnas

```
>>> df['A'] # df.A

2013-01-01    0.469112
2013-01-02    1.212112
2013-01-03   -0.861849
2013-01-04    0.721555
2013-01-05   -0.424972
2013-01-06   -0.673690
Freq: D, Name: A, dtype: float64

>>> df[0:3]
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804

>>> df['20130102':'20130104']
                   A         B         C         D
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
```

# Filtrando columnas

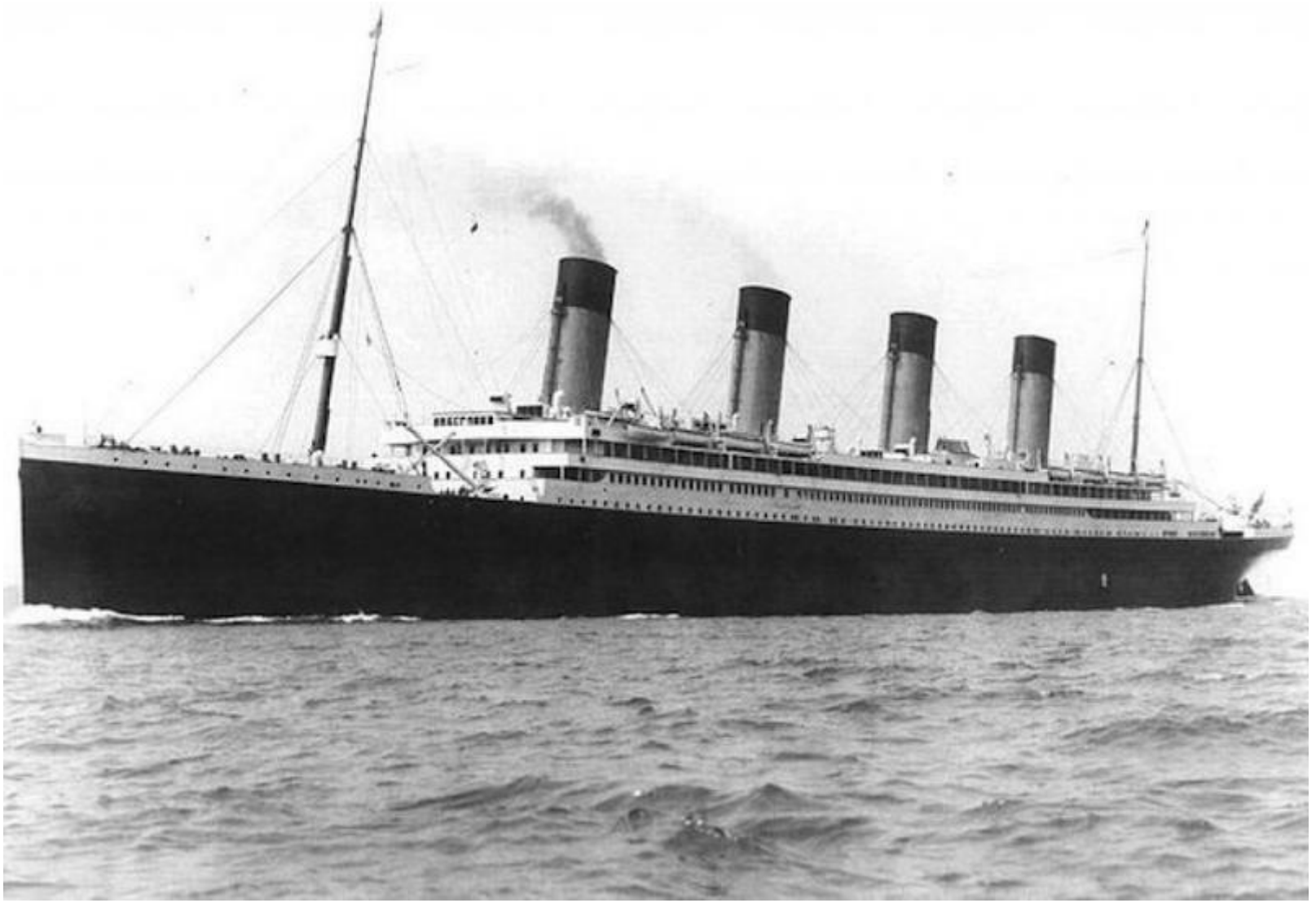
```
>>> df[df.A > 0]
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-04  0.721555 -0.706771 -1.039575  0.271860

>>> df[df.A > 0 & df.C > 0]
                   A         B         C         D
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
```

# Conceptos Básicos de Machine Learning

# El plan

1. Lectura de datos
2. Limpieza
3. Exploración de datos mediante gráficos
4. Formulación de hipótesis de los datos

# Dataset de pasajeros del Titanic



# Cargamos los módulos e importamos los datos en crudo

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

data = pd.read_csv('data/titanic.csv')
```

# Examinando los datos

```
print data.shape
print data.head()
# sumario de columnas numéricas
print data.describe()
print data.isnull().any()
```

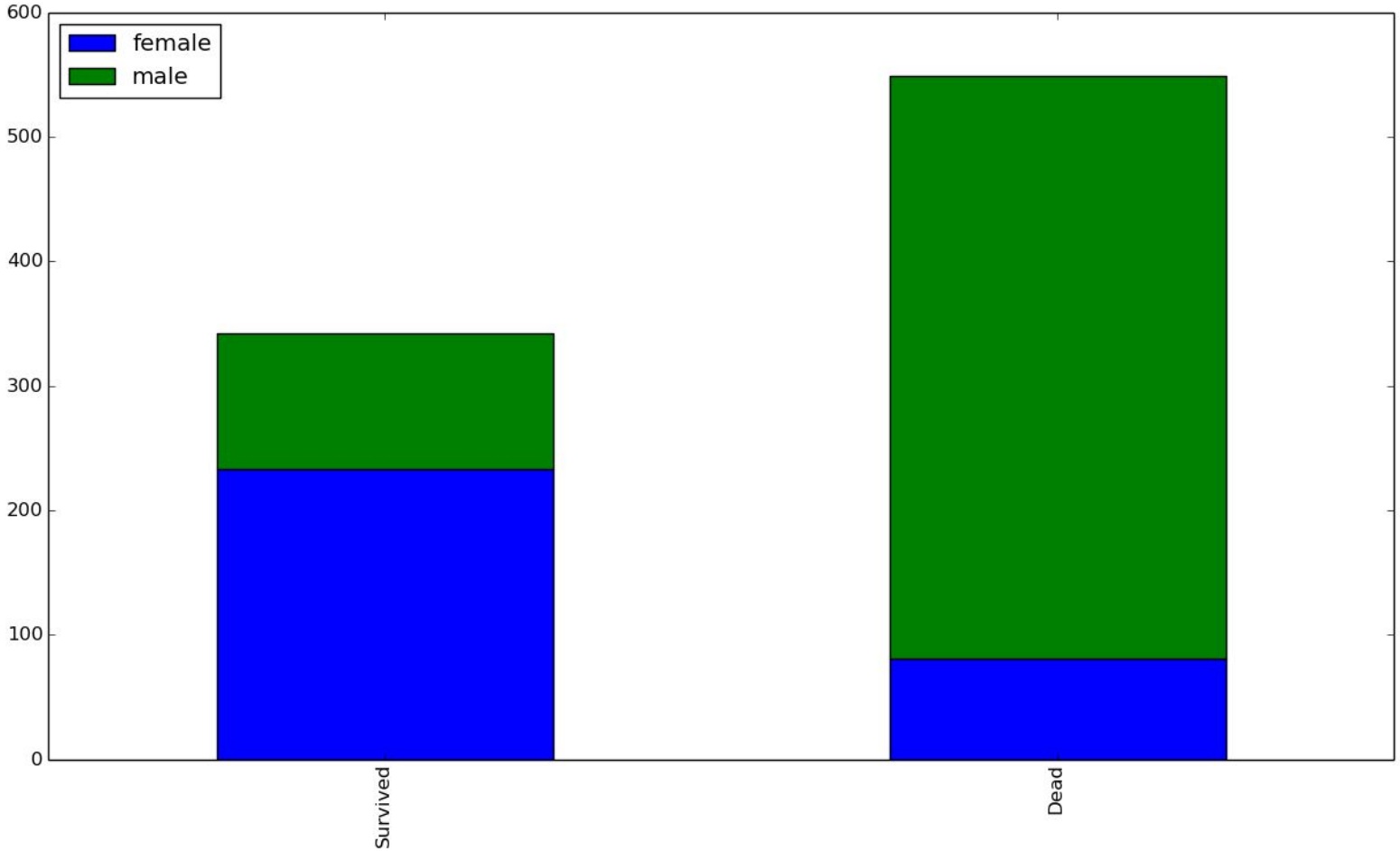
| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

# Limpieza de datos: reemplazando valores faltantes

```
print data[data['Age'].isnull()].count()
print data[data['Age'].isnull()].head()
data['Age'].fillna(data['Age'].median(), inplace=True)
print data.describe()
```

# Exploración de datos mediante gráficos

```
survived_sex = data[data['Survived']==1]['Sex'].value_counts()
dead_sex = data[data['Survived']==0]['Sex'].value_counts()
df = pd.DataFrame([survived_sex,dead_sex])
df.index = ['Survived','Dead']
df.plot(kind='bar', stacked=True, figsize=(15,8))

plt.show()
```


female
male
600
500
400
300
200
100
0
Survived
Dead

# matplotlib.pyplot.hist(..)

Parámetros:

x, bins=None, range=None, normed=False,

weights=None, cumulative=False, bottom=None,

histtype='bar', align='mid', orientation='vertical',

rwidth=None, log=False, color=None, label=None,

stacked=False, hold=None, data=None

# matplotlib.pyplot.hist(..)

Parámetros:

**x**, **bins=None**, range=None, normed=False,

weights=None, cumulative=False, bottom=None,

histtype='bar', align='mid', orientation='vertical',

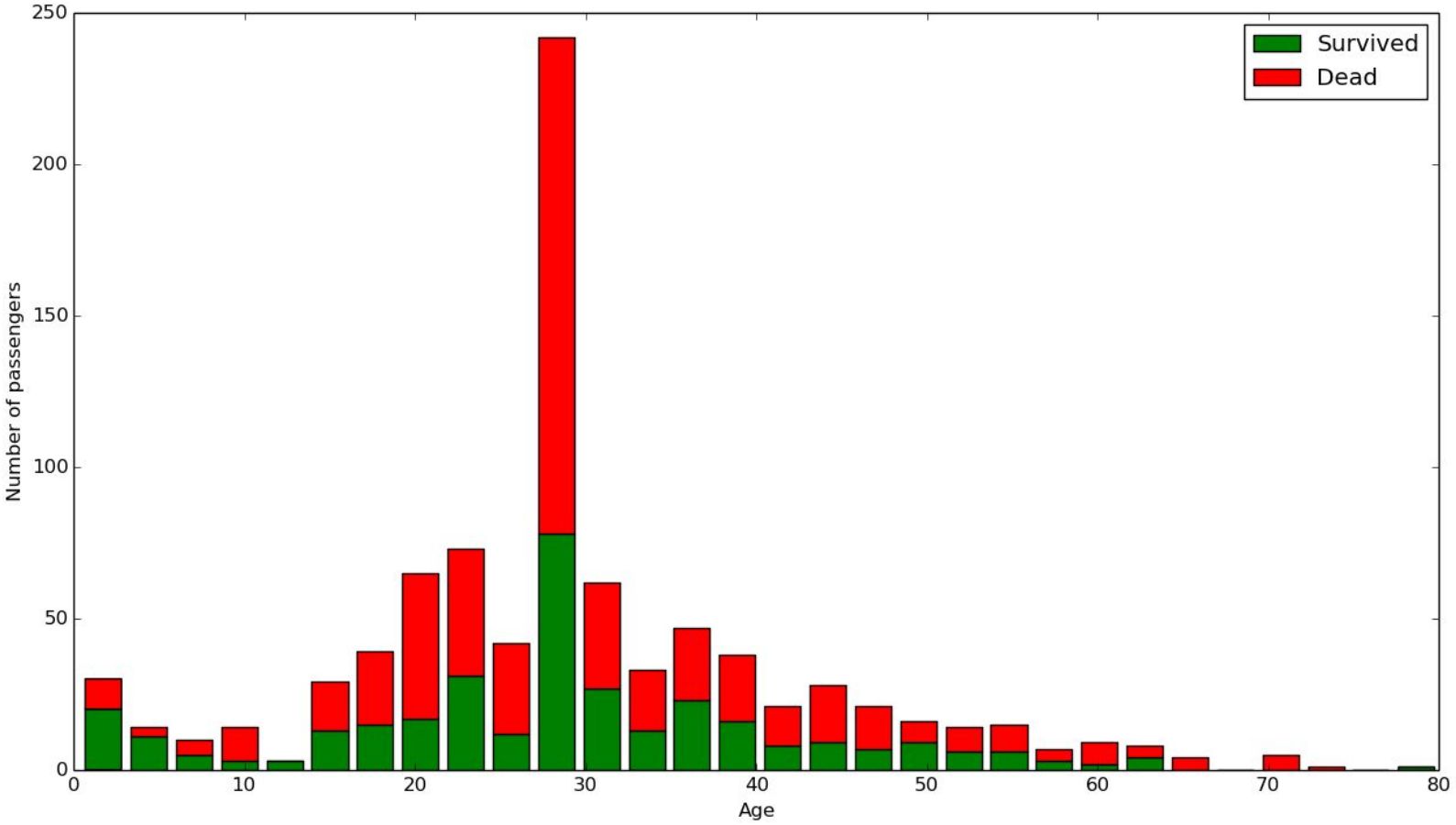
rwidth=None, log=False, **color=None**, **label=None**,

**stacked=False**, hold=None, data=None

# Exploración de datos mediante gráficos

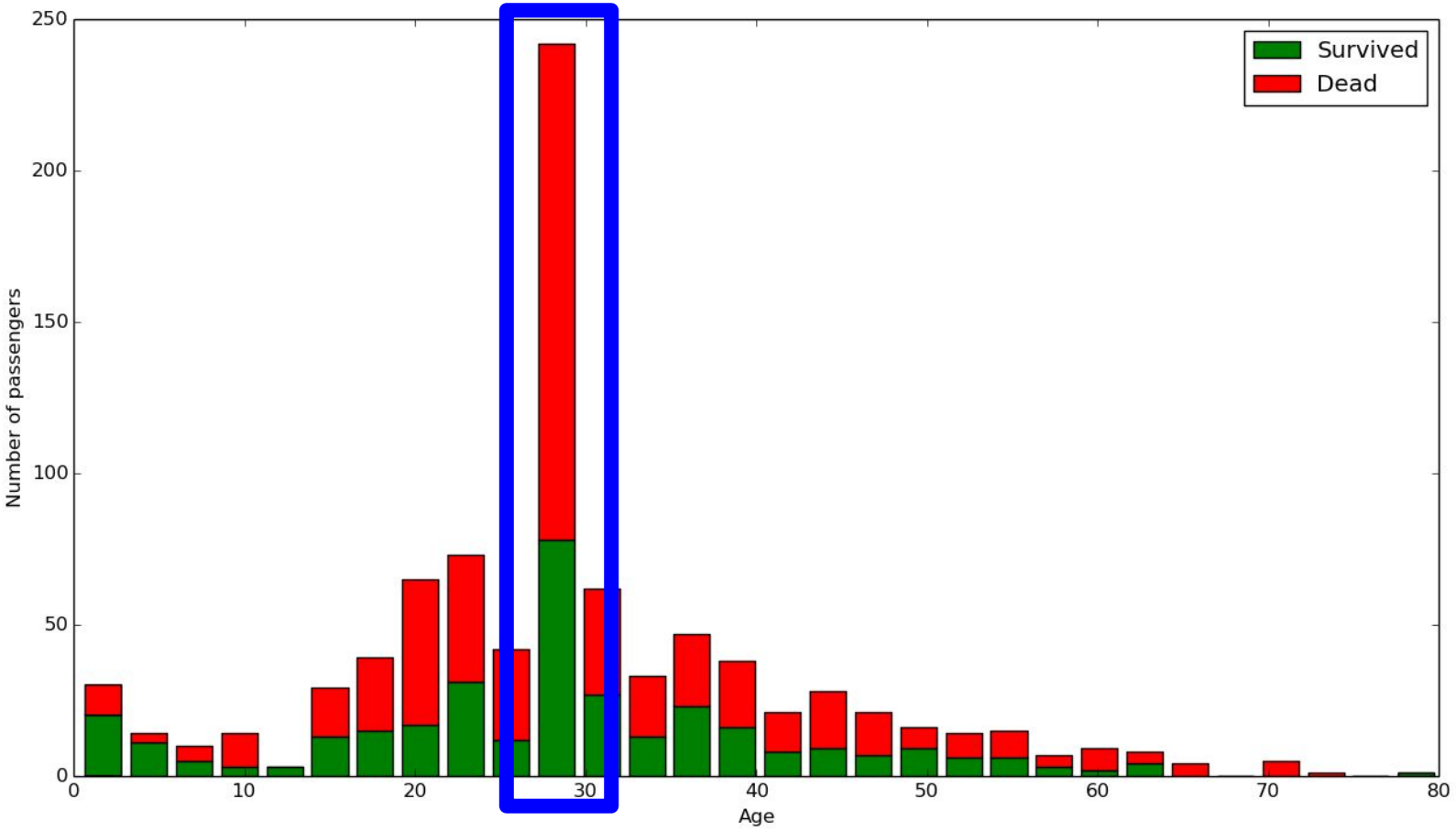
```
figure = plt.figure(figsize=(15,8))
plt.hist([data[data['Survived']==1]['Age'],
          data[data['Survived']==0]['Age']],
         stacked=True, color = ['g','r'],
         bins = 30, label = ['Survived','Dead'])

plt.xlabel('Age')
plt.ylabel('Number of passengers')
plt.legend()

plt.show()
```

# ¿Qué pueden decirme de este gráfico?



# Hipótesis: ¿Mujeres y niños primero?

# ¿Una anomalía?



# Ejercicio 1

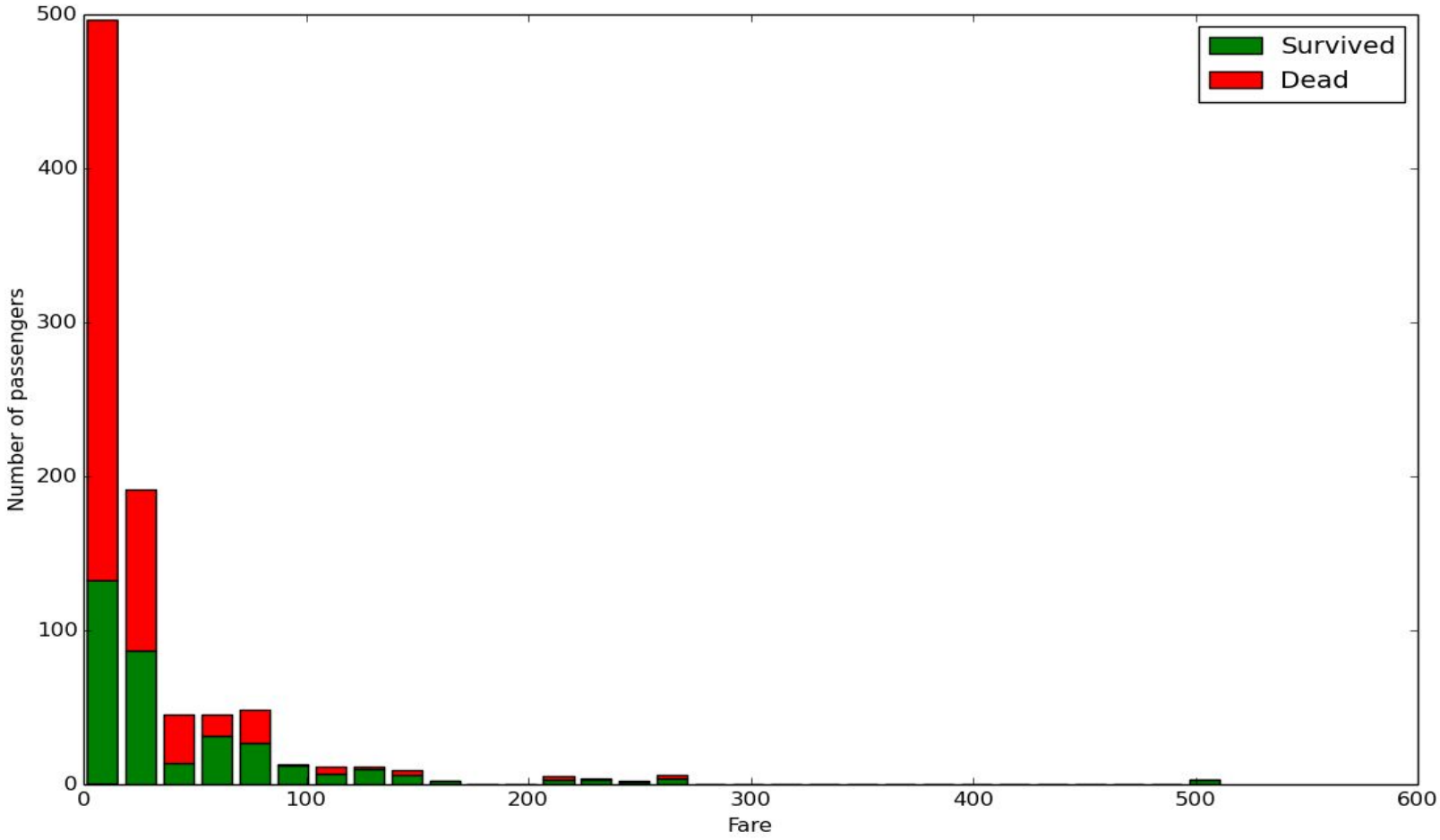
Ejercicio:

1. Filtre los pasajeros cuyas edades sean faltantes.
2. Utilizando los datos del punto anterior, re-haga el gráfico anterior.
3. Responda: ¿el completado de edades faltantes introdujo una anomalía?

# Exploración de datos mediante gráficos

```
figure = plt.figure(figsize=(13,8))
plt.hist([data[data['Survived']==1]['Fare'],
          data[data['Survived']==0]['Fare']],
          stacked=True, color = ['g','r'],
          bins = 30,label = ['Survived','Dead'])
plt.xlabel('Fare')
plt.ylabel('Number of passengers')
plt.legend()

plt.show()
```

# Conceptos Básicos de Machine Learning



# Hipótesis: Sálvese quien pueda (pagarlo)

# Ejercicio 2

Los pasajeros pudieron haber embarcado en Cherbourg (C), Queenstown (Q) o Southampton (S)

Ejercicio:

1. Investigue la co-relación entre el punto de embarque (campo 'Embarked') y la supervivencia de los pasajeros (campo 'Suvived'). Para esto, grafique estos datos en un histograma.
2. ¿Hay correlación? ¿De que tipo?

# Dataset de ventas de Rossmann

# Dataset de ventas de Rossmann

- La empresa Rossmann tiene aproximadamente 3,000 farmacias en 7 países de Europa.
- En los datos se proporcionan 1115 farmacias y se busca un modelo que estime las ventas diarias de 6 semanas.
- Algunas farmacias están cerradas por refacción.

# Ejercicio 3

Examine el dataset de ventas de Rossmann:

1. Cargue los datos de Rossmann usando pandas.
2. Examine las primeras filas, cuente el número de valores únicos por columna. ¿Existen valores faltantes?
3. Grafique las ventas en orden cronológico de la sucursal número 150 durante el año 2013.
4. Grafique las ventas de la sucursal número 150 durante ese año en función a la existencia (o no) de una promoción. ¿Están correlacionados estos datos?
5. Averigue las medias de las ventas cuando hay y no hay promociones.