Práctica 2

Limpieza y validación de los datos

Gisele Guadalupe Almeida dos Santos Maia

Decembre 2019.

Universitat Oberta de Catalunya.

Máster de Ciencia de Datos (Data Science).

Tipología y ciclo de Vida de los datos

Siguiendo las principales etapas de un proyecto analítico, las diferentes tareas a realizar (y justificar) son las siguientes:

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

Respuesta: He elegido el data set Titanic: Machine Learning from Disaster disponible en Kaggle: (https://www.kaggle.com/c/titanic).

*Tenemos 2 data set principales, el de teste (**test.csv**) y el de entrenamiento (**train.csv**).*

El archivo train.csv tenemos 12 variables (columnas) y 891 registros (filas) y en archivo test.csv tenemos 11 variables y 419 registros. La columna de nombre Survived es la diferencia entre las columnas de los dos archivos.

La descripción de las variables:

PassengerId: el número, la clave de identificación de la persona;

Survived: es la sobrevivencia, se el pasajero ha sobrevivido o no. Se = 1 el pasajero ha sobrevivido, se = 0 no ha sobrevivido;

Pclass: es la clase del billete: 1º, 2º o 3º donde se igual a 1 es de la 1º clase, se igual a 2 es de la 2º clase y se igual a 3 es de la 3º clase;

Name: es el nombre del pasajero, ejemplo: Harris, Mr. Walter, se es mujer y esposa de algun pasajero, tiene el nombre de los dos. Ejemplo: Braf, Miss. Elin Ester Maria;

Sex: el sexo del pasajero: female = mujer o male=hombre;

Age: la edad del pasajero: 26 años;

SibSp: número de hermanos o cónyuge a bordo en Titanic;

Parch: número de parientes (niños) a bordo en Titanic;

Ticket: el número del billete. Ejemplo: 330911;

Fare: el precio pago por el billete;

Cabin: el número de la cabina en Titanic;

Embarked: el nombre del puerto de embarcación. C = Cherboung, Q = Queenstorwn, S = Southampton.

Cómo en el archivo de teste no hay la variable de Survived, la idea es intentar hacer la previsibilidad de las personas del archivo de teste sobrevivieran o no. Y para esto vamos a llevar en consideración características como: género, clase social, edad y las variables que sean posible o que sean importantes en el facto de sobrevivir o no al desastre, o sea, identificar las variables que más influyen en la sobrevivencia. Para esto es importante hacer un análisis de las variables para identificar si es posible o no su utilización.

2. Integración y selección de los datos de interés a analizar.

Para el análisis de los datos, vamos a utilizar solamente el archivo train donde contiene las 12 variables.

Hacemos la importación de las librarías que vamos a utilizar, miramos la cantidad de líneas y columnas:

```
# Leer lo archivo
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
train = pd.read_csv('train.csv')
```

```
train.shape
```

```
(891, 12)
```

Las variables que tenemos y sus formatos:

```
train.head()
```

	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

```
train.dtypes
```

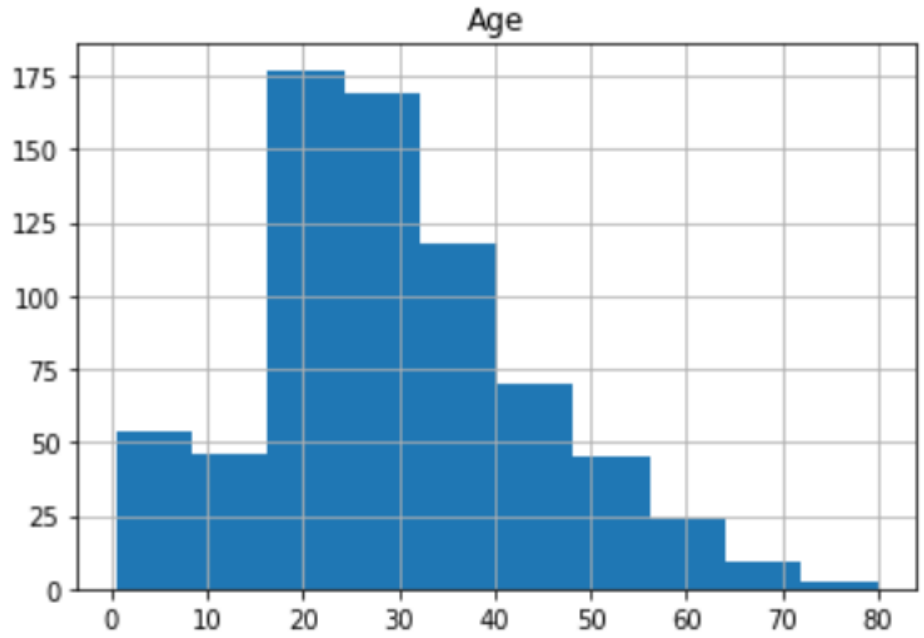
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

```
# Miramos se hay valores nulos y en cual cantidad
train.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Podemos comprobar que en la variable Age hay 177 registros nulos, variable Cabin 687 y variable Embarked 2. Cómo la variable cabin hay 77,1% de las variables vacías, no vas a ser una variable muy útil ya que tenemos pocos registros, así podemos borrarla.

Con un histograma de la variable edad (age) miramos edades de 0 hasta 80 años y la grand concentración es entre los 18 y 40 años. Así vamos a hacer intervalos de edad



< 18 años;

>= 18 y < 25 años;

>= 25 y < 35 años;

>= 35 y <45 años;

>= 45 años;

Los que no hay registros, dejamos por ahora cómo: sin información. Así, quedamos con la distribución:

```
def f(x):
  if x['Age'] >= 45: return '>= 45 anõs'
  elif x['Age'] < 45  and x['Age'] >= 35: return '>= 35 y <45 años'
  elif x['Age'] < 35 and x['Age'] >= 25: return '>= 25 y < 35 años'
  elif x['Age'] < 25  and x['Age'] >= 18: return '>= 18 y < 25 años'
  elif x['Age'] < 18: return '< 18 años'
  else: return 'Sin inforamción'

train['Age_2'] = train.apply(f, axis=1)
```

```
Age_2 = pd.crosstab(index=train["Age_2"],
                                columns="count")
Age_2/Age_2.sum()
```

col_0	count
Age_2	
< 18 años	0.126824
>= 18 y < 25 años	0.185185
>= 25 y < 35 años	0.225589
>= 35 y <45 años	0.134680
>= 45 anõs	0.129068
Sin inforamción	0.198653

3. Limpieza de los datos.

3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Los datos vacíos en la edad, creamos el registro: sin información.

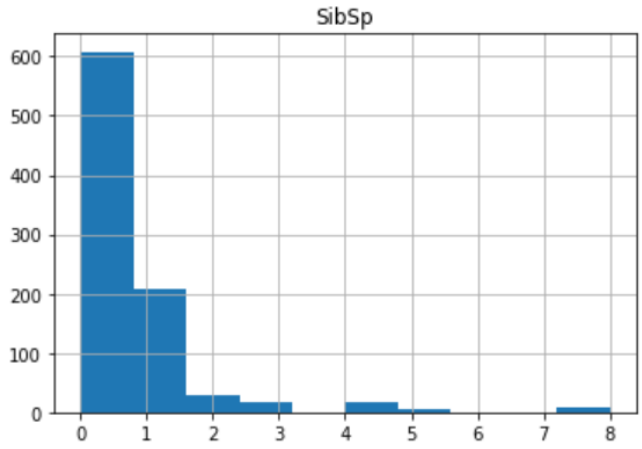
La variable SibSp (hermanos o cónyuge), hay la siguiente distribución:

```
SibSp = pd.crosstab(index=train["SibSp"],
                    columns="count")
SibSp/SibSp.sum()
```

col_0	count
SibSp	
0	0.682379
1	0.234568
2	0.031425
3	0.017957
4	0.020202
5	0.005612
8	0.007856

```
train.hist(column='SibSp')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x
000001E0160775F8>]],
      dtype=object)
```



68% de los registros están cómo 0 que no sabemos se es porque no son casados o no tiene hermanos o porque no tenía su marido o esposa o hermanos a bordo.

Así, vamos a hacer una integración donde 0 es: no tenía hermanos o cónyuge a bordo y > 0 con hermanos o cónyuge a bordo.

```
def g(x):
  if x['SibSp'] > 0: return 'Con hermanos o cónyuge'
  else: return 'Sin hermanos o cónyuge'

train['SibSp_2'] = train.apply(g, axis=1)
```

```
SibSp_2 = pd.crosstab(index=train["SibSp_2"],
                              columns="count")
SibSp_2/SibSp_2.sum()
```

col_0	count
SibSp_2	
Con hermanos o cónyuge	0.317621
Sin hermanos o cónyuge	0.682379

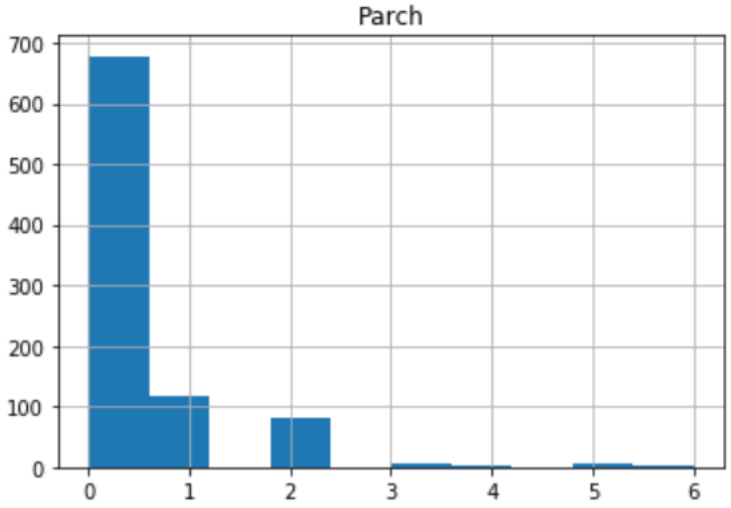
Miramos la distribución de la variable Parch donde tenemos el número de niños a bordo

```
Parch = pd.crosstab(index=train["Parch"],
                              columns="count")
Parch/Parch.sum()
```

col_0	count
Parch	
0	0.760943
1	0.132435
2	0.089787
3	0.005612
4	0.004489
5	0.005612
6	0.001122

```
train.hist(column='Parch')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x
000001E016077DA0>]],
      dtype=object)
```



76% de los pasajeros no tienen hijos o no están con ellos a bordo.

Por lo tanto, vamos a hacer también una integración, donde 0 no están con hijos a bordo y > 0 están con hijos a bordo. Ya que tampoco sabemos si 0 es que no tiene hijos o no están con ellos.

```
def h(x):
  if x['Parch'] > 0: return 'Con hijos a bordo'
  else: return 'Sin hijos a bordo'

train['Parch_2'] = train.apply(h, axis=1)
```

```
Parch_2 = pd.crosstab(index=train["Parch_2"],
                              columns="count")
Parch_2/Parch_2.sum()
```

col_0	count
Parch_2	
Con hijos a bordo	0.239057
Sin hijos a bordo	0.760943

3.2. Identificación y tratamiento de valores extremos.

Para las variables numéricas, vamos a comprobar los valores y los valores máximos y mínimos:

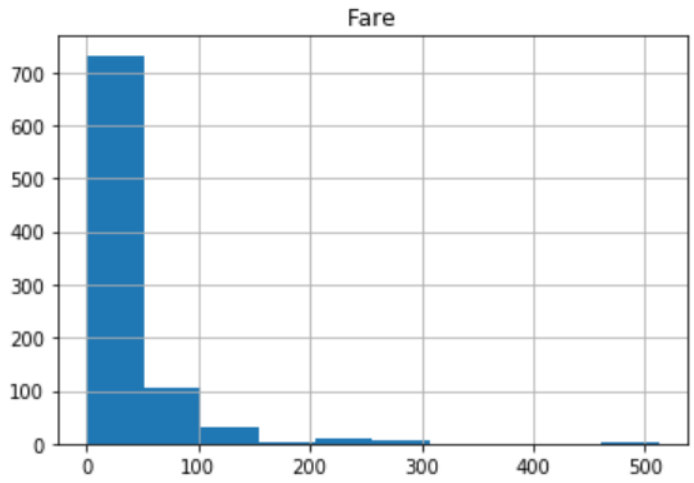
```
train.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

El precio (Fare) tiene un rango muy grande, el mínimo es 0 y el máximo es 512.32992, y, el 75% de la base de datos tiene el valor de 31.000. Así, vamos mirar su distribución y se hay diferencia de los precios por las clases:

```
train.hist(column='Fare')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E015B97B38>]],
      dtype=object)
```




```
train.groupby('Pclass')['Fare'].describe()
```

Pclass	count	mean	std	min	25%	50%	75%	max
1	216.0	84.154687	78.380373	0.0	30.92395	60.2875	93.5	512.3292
2	184.0	20.662183	13.417399	0.0	13.00000	14.2500	26.0	73.5000
3	491.0	13.675550	11.778142	0.0	7.75000	8.0500	15.5	69.5500