Práctica 2: Limpieza y análisis de datos

Raquel Gómez Pérez y Jorge Serra Planelles

9 de junio de 2020

Resolución

1. Descripción del dataset

El conjunto de datos que vamos a analizar se ha obtenido a partir de este enlace https://www.kaggle.com/c/titanic/.
Este conjunto de datos contiene información de parte de los pasajeros que subieron a bordo del transatlántico Titanic, el 10 de abril de 1912 desde el puerto Southampton y los pasajeros que se incorporaron en los puertos de Cherburgo, Francia, y en Queenstown en Irlanda. Entre estos pasajeros se encuentran pasajeros de muy diferentes clases sociales. En el incidente murieron 1514 personas de las 2223 que abordaron. El estricto protocolo de salvamento que se utilizó seguía el principio "mujeres y niños primero".
En los dataset cada fila representa a una persona. Hay información de un total de 1309 pasajeros, divididos en dos dataset: 891 en el conjunto de train y 418 en el conjunto de test. Las columnas describen diferentes atributos sobre la persona, tenemos un total de 12 columnas en el dataset de train, y 11 columnas en el dataset de test, ya que no se incluye la columna survived que es la que se desea predecir.

- **PassengerId:** Número de identificación del pasajero.
- **Survived:** Indica si la persona sobrevivió o no al incidente (0 no sobrevivió, 1 sobrevivió).
- **Pclass:** Clase en la que viajaba el pasajero (1^{a}, 2^{a} o 3^{a}).
- **Name:** Nombre del pasajero.
- **Sex:** Sexo del pasajero, femenino o masculino.
- **Age:** Edad del pasajero.
- **SibSp:** Número de hermanos que el pasajero tenía a bordo.
- **Parch:** Número de padres (del pasajero) que estaban a bordo.
- **Ticket:** Número de ticket que el pasajero entregó al abordar.
- **Fare:** Indica el precio que el pasajero pagó para obtener su pasaje.

- **Cabin:** Indica la cabina que fue asignada al pasajero.
- **Embarked:** Indica el puerto donde el pasajero abordó (C = cherbourg, Q = Queenstown, S= Southampton).

A partir de este conjunto de datos se pretende responder a la pregunta de qué variables fueron las más determinantes sobre la supervivencia o no de los pasajeros, comprobando por ejemplo cuanto influyó la posición económica en el momento del rescate o si realmente se cumplió el protocolo "mujeres y niños primero.ª la hora de evacuar.

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

Comenzamos con la carga de datos, para realizar la selección de las variables de interés. Vamos a realizar la práctica con el lenguaje de programación Python.

```
import pandas as pd
df_train= pd.read_csv("train.csv")
df_train.head(4)
```

	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

```
df_test= pd.read_csv("test.csv")
df_test.head(4)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S

Mostramos el tipo de datos

```
In [19]: df_train.dtypes

Out[19]: 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
```

```
In [20]: df_test.dtypes

Out[20]: PassengerId      int64
         Pclass           int64
         Name            object
         Sex             object
         Age            float64
         SibSp            int64
         Parch            int64
         Ticket          object
         Fare           float64
         Cabin           object
         Embarked        object
         dtype: object
```

Vemos que las variables Survived, Pclass son números enteros, pero a pesar de ello estas variables son consideradas categóricas ya que representan categorías o grupos mutuamente excluyentes. Las variables de tipo object también son variables categóricas. PassengerId, Age, SibSp, Parch y Fare serían variables cuantitativas.

Mostramos también un resumen de las variables.

```
df_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

```
df_test.describe()
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200

Al analizar las variables hemos considerado que las variables mas representativas son:

- **Survived:** Es imprescindible para saber quien sobrevivió al accidente.
- **Pclass y Fare:** Nos puede dar información de la influencia del estatus social y económico a la hora de realizar el proceso de evacuación .
- **Sex y Age:** podremos comprobar si realmente al momento de evacuar se cumplió el protocolo de salvamento "mujeres y niños primero".

En cuanto a los atributos omitidos name y cabin no nos da ninguna información relevante, mas allá de la que ya obtenemos con Pclass y fare. Sibsp y parch no nos dan ningún dato que podamos analizar. Ticket es un dato irrelevante. Embarked tampoco influye en si sobrevivió o no, dado que en todos los puertos embarcaron personas de diferentes edades, sexo y clases sociales.

Seleccionamos las columnas que consideramos representativas:

```
In [23]: df_train = df_train[['Survived', 'Pclass', 'Fare', 'Sex', 'Age']]

         df_train.head(4)
```

Out[23]:

	Survived	Pclass	Fare	Sex	Age
0	0	3	7.2500	male	22.0
1	1	1	71.2833	female	38.0
2	1	3	7.9250	female	26.0
3	1	1	53.1000	female	35.0

```
In [25]: df_test = df_test[['Pclass', 'Fare', 'Sex', 'Age']]

         df_test.head(4)
```

Out[25]:

	Pclass	Fare	Sex	Age
0	3	7.8292	male	34.5
1	3	7.0000	female	47.0
2	2	9.6875	male	62.0
3	3	8.6625	male	27.0

3. Limpieza de los datos

3.1 Elementos vacíos

Primero observamos cuantos valores ceros tenemos por cada columna en los dos dataset

```
In [28]: (df_test == 0).sum()
```

```
Out[28]: Pclass    0
         Fare      2
         Sex       0
         Age       0
         dtype: int64
```

```
In [29]: (df_train == 0).sum()
```

```
Out[29]: Survived    549
         Pclass        0
         Fare         15
         Sex           0
         Age           0
         dtype: int64
```

Vemos que tenemos valores cero en los campos Survived y Fare. En el caso de Survived el valor 0 es un valor correcto ya que como hemos dicho anteriormente este campo tiene el valor 0 cuando la persona no sobrevivió y valor 1 cuando la persona no sobrevivió. En el caso de Fare habíamos visto que significa el precio que el pasajero pagó para obtener su pasaje, por tanto también podríamos considerar que es un valor válido, en caso de que el pasaje haya salido gratis por alguna promoción. Por tanto consideramos que no es necesario tratar estos elementos cero.

Observamos también los valores nulos

```
In [26]: df_train.isnull().sum()

Out[26]: Survived      0
         Pclass        0
         Fare          0
         Sex           0
         Age         177
         dtype: int64

In [27]: df_test.isnull().sum()

Out[27]: Pclass     0
         Fare       1
         Sex        0
         Age       86
         dtype: int64
```

Vemos que tenemos valores nulos en los campos Age y Fare. En este caso si consideramos que es necesario tratar estos datos nulos. Para tratarlos hemos decidido usar la media ya que se trata de valores numéricos.

```
In [40]: df_train['Age'] = df_train['Age'].fillna(df_train['Age'].mean())
         df_test['Age'] = df_test['Age'].fillna(df_test['Age'].mean())
         df_train['Fare'] = df_train['Fare'].fillna(df_train['Fare'].mean())
         df_test['Fare'] = df_test['Fare'].fillna(df_test['Fare'].mean())
```

```
In [41]: df_train.isnull().sum()
```

```
Out[41]: Survived    0
         Pclass      0
         Fare        0
         Sex         0
         Age         0
         dtype: int64
```

```
In [42]: df_test.isnull().sum()
```

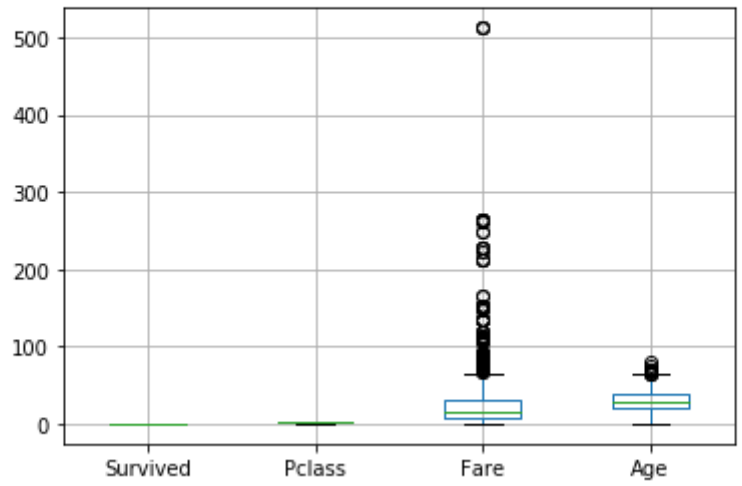
```
Out[42]: Pclass    0
         Fare      0
         Sex       0
         Age       0
         dtype: int64
```

3.2 Valores extremos

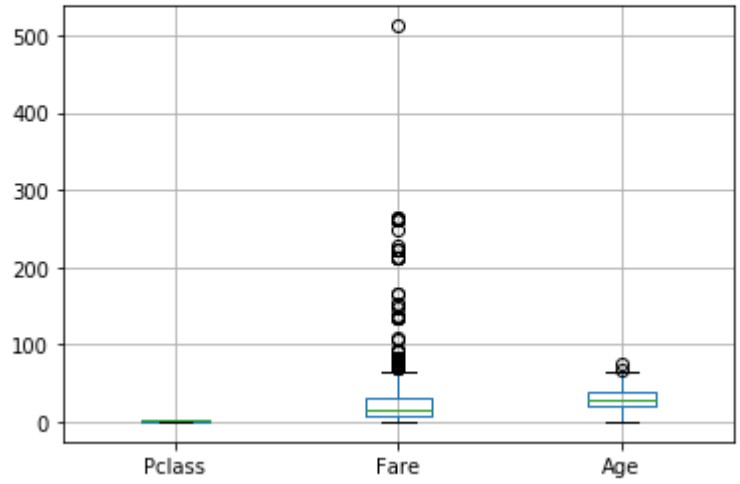
Los valores extremos son aquellos datos que se encuentran muy alejados de la distribución normal de una variable o población. Para poder identificar estos valores extremos utilizamos un boxplot.

```
In [31]: df_train.boxplot()
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x22b466b56a0>
```

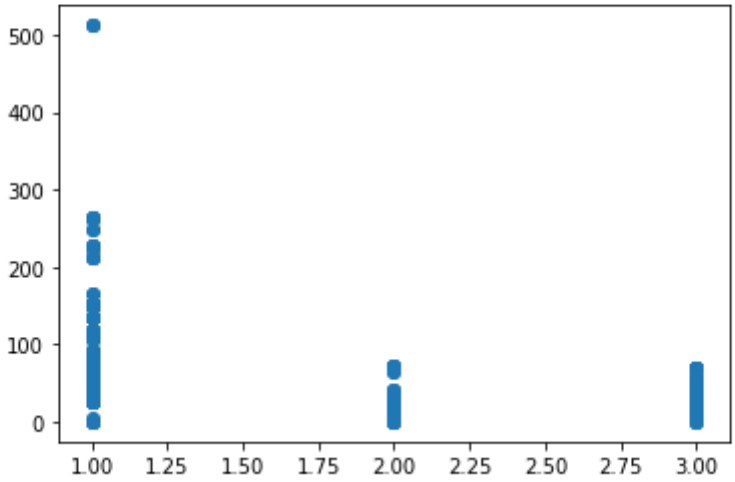


```
In [32]: df_test.boxplot()
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x22b46766668>
```



Vemos que parecen existir valores extremos en Fare y Age. En Age vemos que los puntos más extremos en el gráfico se encuentran por debajo de 100, por lo que no se podrían considerar errores en los datos ya que se encuentran dentro de la distribución normal de la población, aunque la edad media de los pasajeros sea alrededor de los 30 años.
Para hacer un análisis más en profundidad de los valores Fare para comprobar si se tratan o no de valores extremos, vamos a representar el valor Fare dependiendo de la clase en la que viajaban los pasajeros. Vemos que el valor más alto de tarifa es aproximadamente 510, y además pertenece un pasajero de primera clase, por lo que parece que se encuentra dentro de la distribución normal de la población y no lo consideraríamos un error en los datos. Por tanto, no es necesario realizar ninguna transformación en los datos para tratar los valores extremos.

```
In [36]: import matplotlib.pyplot as plt
         x = df_train['Pclass']
         y = df_train['Fare']
         plt.scatter(x,y)
         plt.show()
```



4. Análisis de los datos

4.1 Selección del grupo de datos

Para hacer el análisis de datos vamos a utilizar los dataset obtenido en los pasos anteriores, con la selección de las variables representativas y la corrección de los valores nulos. Para el dataset de test lo vamos a unir con el csv gender submission que tiene la información del campo Survived para el dataset de test. A continuación concatenamos el dataset de traing con el dataset de test

In [83]:
```
import pandas as pd
df_resultados= pd.read_csv("gender_submission.csv")

df_test= pd.read_csv("test.csv")

df_test = pd.merge(df_test, df_resultados, on='PassengerId')

df_test = df_test[['Pclass', 'Fare', 'Sex', 'Age', 'Survived']]

df_test['Age'] = df_test['Age'].fillna(df_test['Age'].mean())
df_test['Fare'] = df_test['Fare'].fillna(df_test['Fare'].mean())

df = pd.concat([df_train, df_test])

df.head(4)
```

Out[83]:

	Age	Fare	Pclass	Sex	Survived
0	22.000000	7.2500	3	male	0
1	38.000000	71.2833	1	female	1
2	26.000000	7.9250	3	female	1
3	35.000000	53.1000	1	female	1

4.2 Normalidad y homogeneidad de la varianza

Para comprobar la **normalidad** de las variables numéricas Age y Fare aplicaremos el test Shapiro-Wilk. La hipótesis nula es que la variable sigue una distribución normal. El nivel de significancia será α=0.05, si el p-valor es inferior a este, rechazaremos la hipótesis nula.

In [192]:
```
from scipy.stats import shapiro, fligner, ttest_ind

# test Saphiro Wilk para comprobar la normalidad de Age
print("Nº de observaciones:",  len(df['Age']))
shapiro(df['Age'])
```

```
Nº de observaciones: 1309
```

Out[192]:
```
(0.9587069153785706, 8.605683104126587e-19)
```

In [194]:
```
# test Saphiro Wilk para comprobar la normalidad de Age
print("Nº de observaciones:",  len(df['Fare']))
shapiro(df['Fare'])
```

```
Nº de observaciones: 1309
```

Out[194]:
```
(0.5279001593589783, 0.0)
```

Ninguna de las dos variables estudiadas sigue una distribución normal. Sin embargo, dado que tenemos un gran número de observaciones (1309 >20) podemos asumir el teorema central del límite. La distribución de la media de las variables estudiadas, gracias a la gran cantidad de observaciones, seguirá una distribución normal con una media de población μ y una varianza $\frac{\sigma^2}{N}$.

Para comprobar la **homocedasticidad** de las variables usaremos el test de Fligner-Killeen, dado que nuestros datos no siguen una distribución normal. La hipótesis nula asume igualdad de varianza.

In [305]:
```
sobrevive = df['Survived'] == 1
nop = df['Survived'] == 0

hombre = df['Sex'] == 'male'
mujer = df['Sex'] == 'female'

niño = df['Age'] <= 12
adolescente = (df['Age'] > 12) & (df['Age'] < 18)
adulto = df['Age'] >= 18

primera = df['Pclass'] == 1
segunda = df['Pclass'] == 2
tercera = df['Pclass'] == 3
```

In [306]:
```
edadvssupervivencia = [df.loc[sobrevive, 'Age'] , df.loc[nop, 'Age']]
edadvsclase = [df.loc[primera, 'Age'] , df.loc[segunda, 'Age'], df.loc[tercera, 'Age']]
```

In [307]:
```
billetevssupervivencia = [df.loc[sobrevive, 'Fare'] , df.loc[nop, 'Fare']]
billetevsclase = [df.loc[primera, 'Fare'] , df.loc[segunda, 'Fare'], df.loc[tercera, 'Fare']]
```

In [308]:
```
fligner(*edadvssupervivencia)
```

Out[308]:
```
FlignerResult(statistic=9.673447446371325, pvalue=0.0018695018312961695)
```

In [309]:
```
fligner(*edadvsclase)
```

Out[309]:
```
FlignerResult(statistic=67.91688247355259, pvalue=1.7866371363834e-15)
```

In [310]:
```
fligner(*billetevssupervivencia)
```

Out[310]:
```
FlignerResult(statistic=128.37617589290807, pvalue=9.286369113501898e-30)
```

In [311]:
```
fligner(*billetevsclase)
```

Out[311]:
```
FlignerResult(statistic=550.5462872041451, pvalue=2.820932649973658e-120)
```

Dado que todas las combinaciones resultan en un p-valor inferior al nivel de significancia (<0,05), se rechaza la hipótesis nula de homocedasticidad y se concluye que las variables presentan varianzas estadísticamente diferentes para los diferentes grupos.

4.3 Pruebas estadísticas para comparar los grupos de datos

Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

5. Representación de los resultados

Representación de los resultados a partir de tablas y gráficas.

6. Resolución del problema

Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

7. Código

Tanto el código fuente escrito para la extracción de datos como el dataset generado pueden ser accedidos a través de este enlace.

Recursos

1. Comparación de media (t-test).
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html.
    - https://towardsdatascience.com/inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f.
2. Comprobación de normalidad.
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html.
3. Comparación de homocedasticidad.
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.fligner.html.
    - https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.stats.fligner.html.
4. Comparaciones de distr no parametricos.
    - https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/.

- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html.
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html.

5. ANOVA.
   - https://ariepratama.github.io/How-to-Use-1-Way-Anova-in-Python/.
   - https://www.marsja.se/four-ways-to-conduct-one-way-anovas-using-python/.
   - https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html.

Contribuciones al trabajo

Contribuciones	Firma
Investigación previa	RGP, JSP
Redacción de las respuestas	RGP, JSP
Desarrollo código	RGP, JSP