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()
```

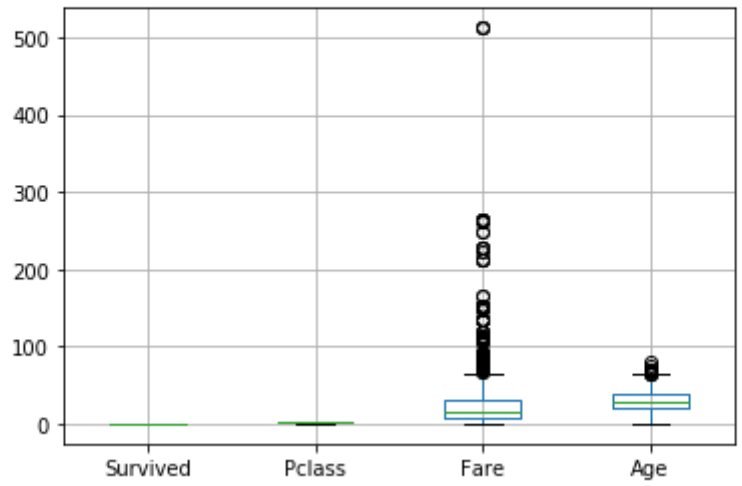
```
Out[42]: Pclass    0
         Fare      0
         Sex       0
         Age       0
         dtype: int64
```

3.2 Valores extremos

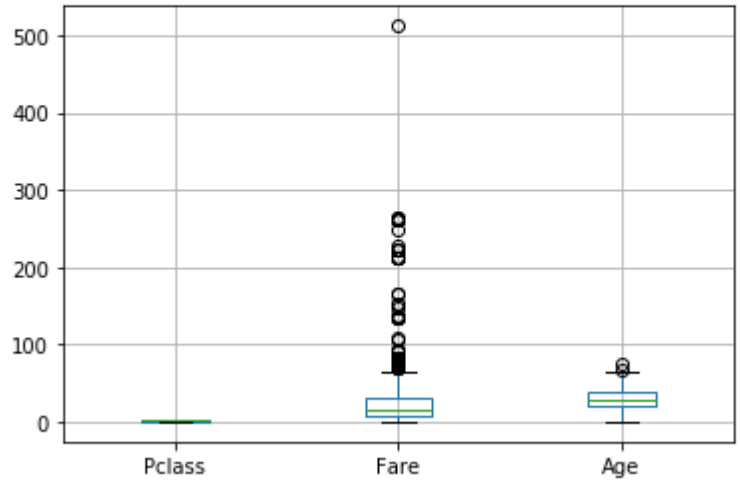
Los valores extremos son aquellos datos que se encuentran muy alejados de la distribución normal de una variable o población. Para poder identificar estos valores extremos utilizamos un boxplot.

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

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

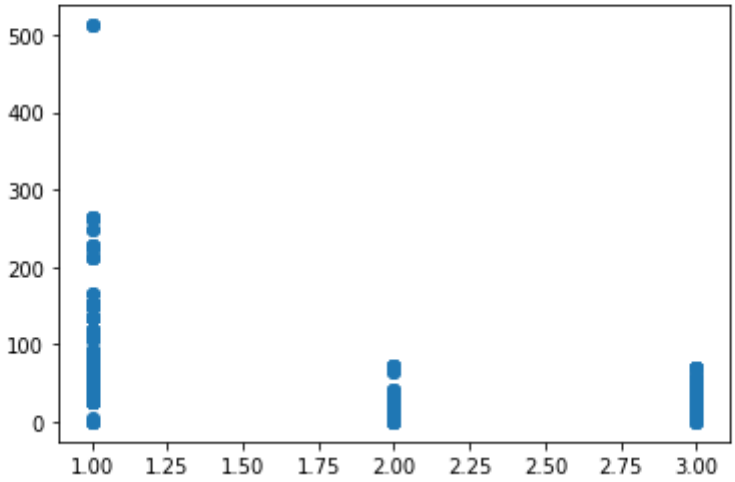


```
In [32]: df_test.boxplot()
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x22b46766668>
```



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 train 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 Levene, ya que debido a la gran cantidad de observaciones consideramos que nuestros datos siguen una distribución normal. La hipótesis nula asume igualdad de varianza.

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

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

```
levene(*edadvssupervivencia)
```

```
LeveneResult(statistic=9.990350319236496, pvalue=0.0016097545526084426)
```

```
levene(*edadvsclase)
```

```
LeveneResult(statistic=30.02550541238763, pvalue=1.7823695126007745e-13)
```

```
levene(*billetevssupervivencia)
```

```
LeveneResult(statistic=56.475996795156355, pvalue=1.0512418588066077e-13)
```

```
levene(*billetevsclase)
```

```
LeveneResult(statistic=193.3090836669573, pvalue=2.8964688997590964e-74)
```

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

- **Comparación de medias (t-test)**
  En este caso, podremos contrastar la diferencia de medias ya que los tamaños de las muestras son superiores a treinta. El teorema del límite central nos dice que si los tamaños de las muestras son superiores a 30, el estadístico de contraste es una observación de una variable aleatoria que se distribuye aproximadamente como una N(0,1).

$H_0 : \mu_1 = \mu_2$

Edad supervivientes vs no:

```
ttest_ind(*edadvssupervivencia)
```

```
Ttest_indResult(statistic=-1.7557529175197664, pvalue=0.07936482950320106)
```

Vemos que el P-valor es superior al nivel de significacia por lo que rechazaremos la hipótesis nula. Las medias no son iguales.

Edad 1era clase vs 2da clase:

```
ttest_ind(edadvsclase[0], edadvsclase[1])
```

```
Ttest_indResult(statistic=7.614911140265477, pvalue=1.0352048294386667e-13)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

Edad 2da clase vs 3ra clase:

```
ttest_ind(edadvsclase[1], edadvsclase[2])
```

```
Ttest_indResult(statistic=4.054943201088858, pvalue=5.410886536066226e-05)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

Edad 1era clase vs 3ra clase:

```
ttest_ind(edadvsclase[0], edadvsclase[2])
```

```
Ttest_indResult(statistic=15.081050594523997, pvalue=1.3816834689568628e-46)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

Billete supervivientes vs no:

```
ttest_ind(*billetevssupervivencia)
```

```
Ttest_indResult(statistic=8.683208834939236, pvalue=1.1275271059562358e-17)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

Billete 1ra clase vs 2da clase:

```
ttest_ind(billetevsclase[0], billetevsclase[1])
```

```
Ttest_indResult(statistic=13.555803811610469, pvalue=1.072059946201641e-36)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

Billete 2da clase vs 3ra clase:

```
ttest_ind(billetevsclase[1], billetevsclase[2])
```

```
Ttest_indResult(statistic=9.12026409262869, pvalue=4.122760404727597e-19)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

Billete 1ra clase vs 3ra clase:

```
ttest_ind(billetevsclase[0], billetevsclase[2])
```

```
Ttest_indResult(statistic=24.029707407024343, pvalue=1.1764693709683345e-101)
```

Vemos que el P-valor es inferior al nivel de significacia por lo que no rechazaremos la hipótesis nula. Las medias si son iguales.

- **Correlación**

```
df[['Age', 'Fare']].corr()
```

	Age	Fare
Age	1.000000	0.171301
Fare	0.171301	1.000000

Vemos que no existe correlación apreciable entre Fare y Age.

- **Comparación de más de dos grupos - ANOVA**
  Compararemos si hay diferencia entre la edad media de los supervivientes frente a los que no y clase, separando las observaciones por sexo:

  - H_0: Las medias poblacionales μk de Edad para las distintas clases entre si, los supervivientes frente a los no y el cruce de las dos categorias son iguales.
  - H_1: de Edad para las distintas clases entre si, los supervivientes frente a los no y el cruce de las dos categorias no son iguales.

```
mod = ols('Age ~ Survived * Pclass', data=df.loc[hombre, ['Age', 'Survived', 'Pclass']]).fit()
# do type 2 anova
aov_table = sm.stats.anova_lm(mod, typ=2)
print('ANOVA table for Male')
print('----------------------')
print(aov_table)
print()

mod = ols('Age ~ Survived * Pclass', data=df.loc[mujer, ['Age', 'Survived', 'Pclass']]).fit()
# do type 2 anova
aov_table = sm.stats.anova_lm(mod, typ=2)
print('ANOVA table for Female')
print('----------------------')
print(aov_table)
```

```
ANOVA table for Male
----------------------
                       sum_sq     df           F        PR(>F)
Survived           3037.121466    1.0   22.683898  2.249341e-06
Pclass            20742.214601    1.0  154.921125  9.328437e-33
Survived:Pclass      12.068556    1.0    0.090139  7.640754e-01
Residual         112332.763557  839.0         NaN           NaN

ANOVA table for Female
----------------------
                      sum_sq     df          F        PR(>F)
Survived          171.811205    1.0   1.133635  2.875574e-01
Pclass          11392.210811    1.0  75.167458  7.375751e-17
Survived:Pclass   115.184009    1.0   0.760001  3.837811e-01
Residual        70019.680290  462.0        NaN           NaN
```

Para los hombres:

1. vemos que la media entre supervivientes y no supervivientes no es igual (p-valor = 2.25 * 10^-6, muy inferior al nivel de significancia 0.05).
2. vemos que la media entre clases tampoco es igual entre ellas.(p-valor = 9.33 * 10^-33, muy inferior al nivel de significancia 0.05).
3. por último, vemos que la intersección de supervivencia con clase si que tiene medias iguales entre clases (p-valor = 0.76... >0.05).

Para las mujeres:

1. vemos que la hipóstesis nula entre supervivientes y no supervivientes no se puede rechazar, por lo que las medias son iguales (p-valor = 0.29..., superior al nivel de significancia 0.05).
2. vemos que la media entre clases no es igual entre ellas (p-valor = 7.36 * 10^-17, muy inferior al nivel de significancia 0.05).
3. por último, vemos que la intersección de supervivencia con clase si que tiene medias iguales entre clases (p-valor = 0.38... >0.05).

- **Regresión**

  Para finalizar podemos crear un modelo de regresión logística que teniendo en cuenta las variables explicativas que hemos estado estudiando nos sirva para predecir/clasificar si un pasajero sobreviviría o moriría en el accidente:

```
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

```
# Crear variables binarias para las variables categóricas
categoricas = ['Pclass', 'Sex']
for var in categoricas:
    cat_list = pd.get_dummies(df[var], prefix=var)
    df_aux = df.join(cat_list)
    df = df_aux
```

```
df.columns
```

```
Index(['Age', 'Fare', 'Pclass', 'Sex', 'Survived', 'Pclass_1', 'Pclass_2',
       'Pclass_3', 'Sex_female', 'Sex_male'],
      dtype='object')
```

```
y = df.loc[df['Age'].notnull(), 'Survived']
X = df.loc[df['Age'].notnull(), ['Age', 'Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male']]
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.583154
         Iterations 6
                         Results: Logit
=================================================================
Model:                 Logit             Pseudo R-squared:  0.120
Dependent Variable:    Survived          AIC:               8361.9276
Date:                  2020-05-31 20:04  BIC:               8396.3096
No. Observations:      7161              Log-Likelihood:    -4176.0
Df Model:              4                 LL-Null:           -4744.4
Df Residuals:          7156              LLR p-value:       8.0257e-245
Converged:             1.0000            Scale:             1.0000
No. Iterations:        6.0000
-----------------------------------------------------------------
              Coef.    Std.Err.      z    P>|z|      [0.025       0.975]
-----------------------------------------------------------------
Age         -0.0114      0.0022 -5.3215 0.0000       -0.0157      -0.0072
Pclass_1     0.2271 1181219.8534  0.0000 1.0000 -2315148.1433 2315148.5975
Pclass_2     0.0117 1181219.8534  0.0000 1.0000 -2315148.3587 2315148.3822
Pclass_3    -0.1802 1181219.8534 -0.0000 1.0000 -2315148.5507 2315148.1902
Sex_female   0.8799 1181219.8534  0.0000 1.0000 -2315147.4905 2315149.2503
Sex_male    -0.8213 1181219.8534 -0.0000 1.0000 -2315149.1917 2315147.5491
=================================================================
```

Age tiene un P-valor <a 0.05 por lo que no es una variable que influya en el resultado en comparación con el resto de categorías, la podemos sacar del modelo.

```
y = df.loc[:, 'Survived']
X = df.loc[:, ['Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male']]
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.585151
         Iterations: 35
                            Results: Logit
=================================================================
Model:                 Logit              Pseudo R-squared:   0.117
Dependent Variable:    Survived           AIC:                8388.5380
Date:                  2020-05-31 20:06   BIC:                8416.0436
No. Observations:      7161               Log-Likelihood:     -4190.3
Df Model:              3                  LL-Null:            -4744.4
Df Residuals:          7157               LLR p-value:        6.1145e-240
Converged:             0.0000             Scale:              1.0000
No. Iterations:        35.0000
-----------------------------------------------------------------
              Coef.    Std.Err.      z    P>|z|      [0.025       0.975]
-----------------------------------------------------------------
Pclass_1     0.0491 1233061.8064  0.0000 1.0000 -2416756.6821 2416756.7804
Pclass_2    -0.1188 1233061.8064 -0.0000 1.0000 -2416756.8500 2416756.6125
Pclass_3    -0.2876 1233061.8064 -0.0000 1.0000 -2416757.0188 2416756.4436
Sex_female   0.6697 1233061.8064  0.0000 1.0000 -2416756.0615 2416757.4009
Sex_male    -1.0269 1233061.8064 -0.0000 1.0000 -2416757.7582 2416755.7043
=================================================================
```

```
logreg = LogisticRegression(solver='liblinear')
logreg.fit(X, y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
# Crear variables binarias para las variables categóricas de test
categoricas = ['Pclass', 'Sex']
for var in categoricas:
    cat_list = pd.get_dummies(df_test[var], prefix=var)
    df_aux = df_test.join(cat_list)
    df_test = df_aux

X_test = df_test.loc[:, ['Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male']]
y_test = df_test['Survived']

y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 1.00
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[266   0]
 [  0 152]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

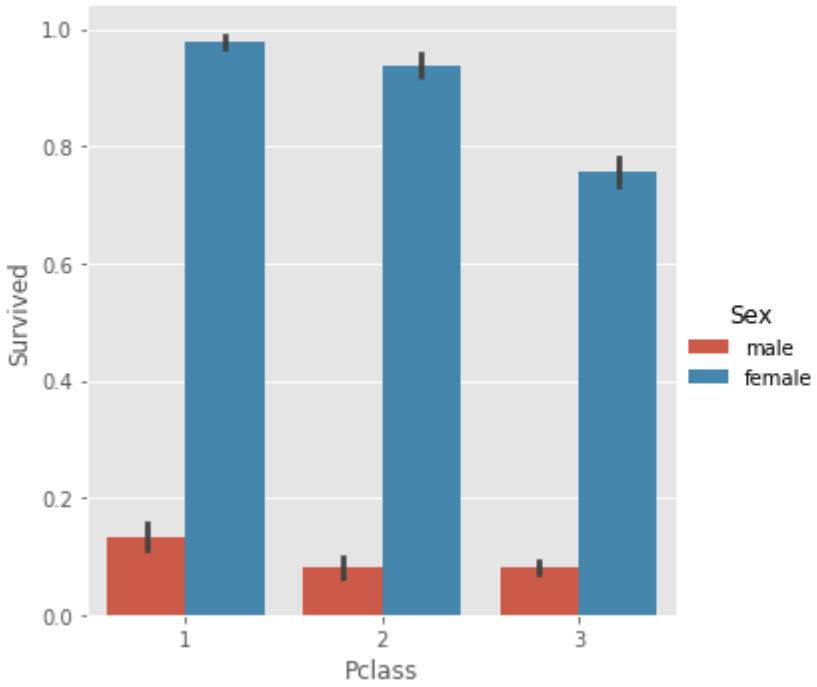
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       266
           1       1.00      1.00      1.00       152

    accuracy                           1.00       418
   macro avg       1.00      1.00      1.00       418
weighted avg       1.00      1.00      1.00       418
```

El modelo obtenido tiene un rendimiento perfecto. Consigue clasificar sin errores los casos proporcionados.

5. Representación de los resultados

Supervivencia por clase y sexo:

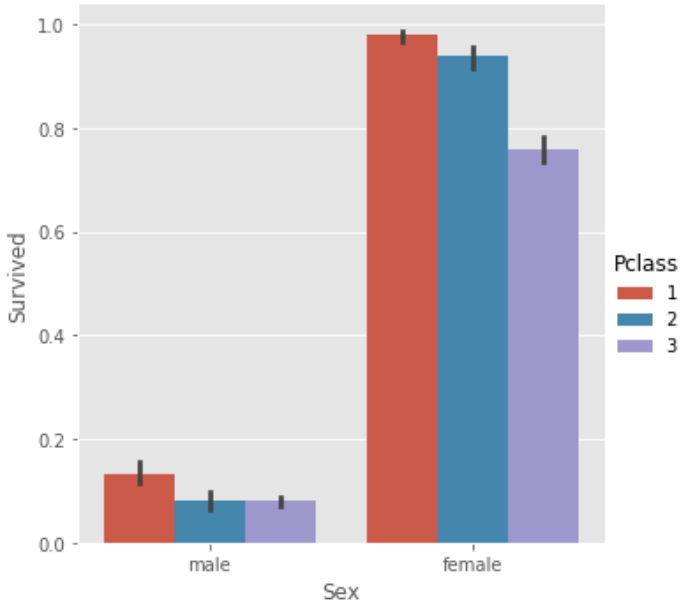
```
sns.catplot(x="Pclass", y="Survived", hue="Sex", kind="bar", data=df);
```



Se puede observar resultados en línea con los análisis estadísticos del apartado anterior.

Supervivencia por sexo y clase

```
sns.catplot(x="Sex", y="Survived", hue="Pclass", kind="bar", data=df);
```

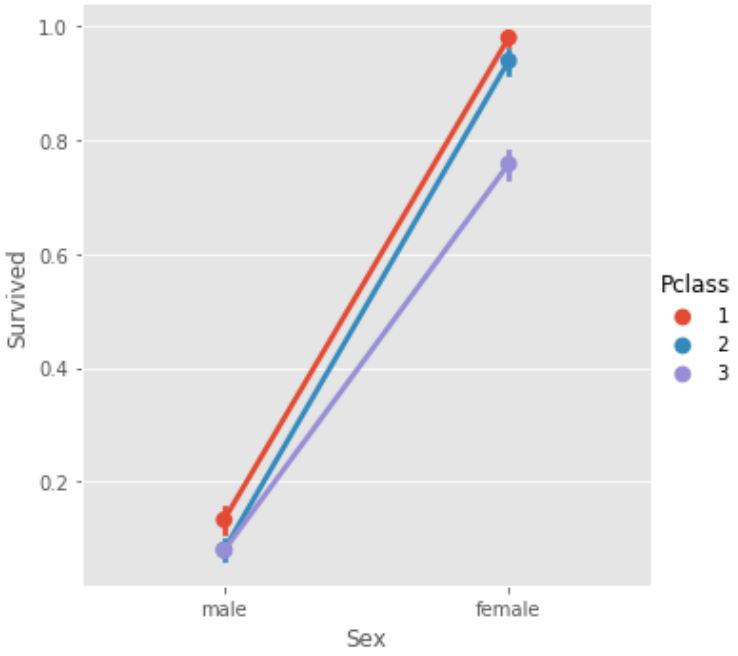


Los hombres fueron la mayoría de las victimas del accidente. Sin embargo, la clase también influyo en las víctimas femeninas.

Gráficos de perfil:

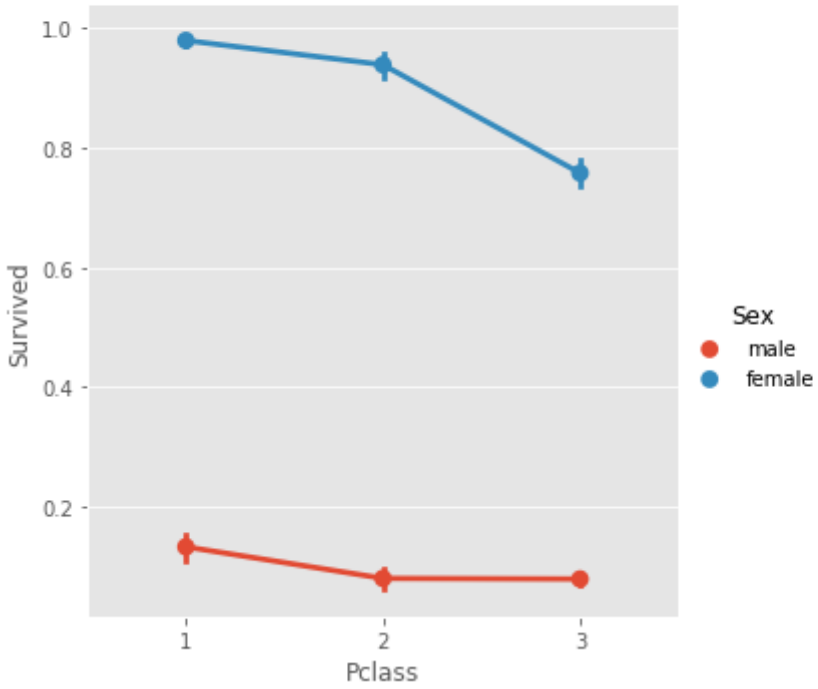
Supervivencia por sexo y clase:

```
sns.catplot(x="Sex", y="Survived", hue="Pclass", kind="point", data=df);
```



Supervivencia por clase y sexo:

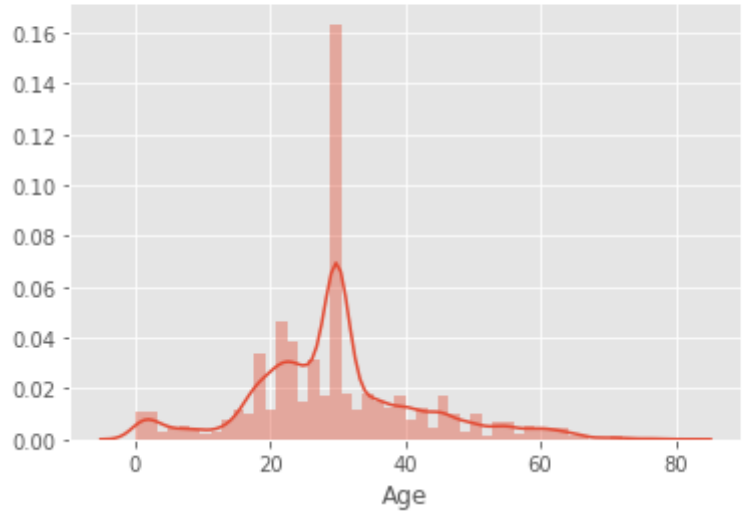
```
sns.catplot(x="Pclass", y="Survived", hue="Sex",
            kind="point", data=df);
```



Unas visualizaciones que nos permiten ver la misma conclusión que los gráficos anteriores. Rectas paralelas nos indican disociación entre las variables explicativas. Quiebros de las rectas indican diferencias de las medias.

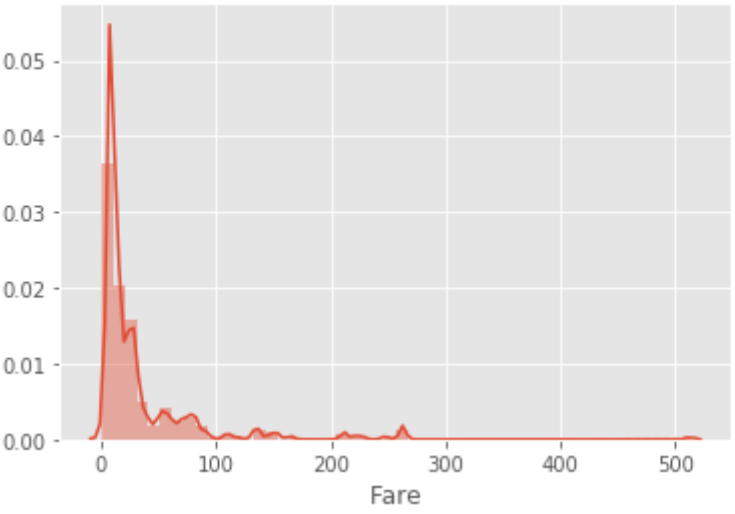
Distribuciones de las variables numéricas (Hisotgramas)

```
sns.distplot(df['Age']);
```



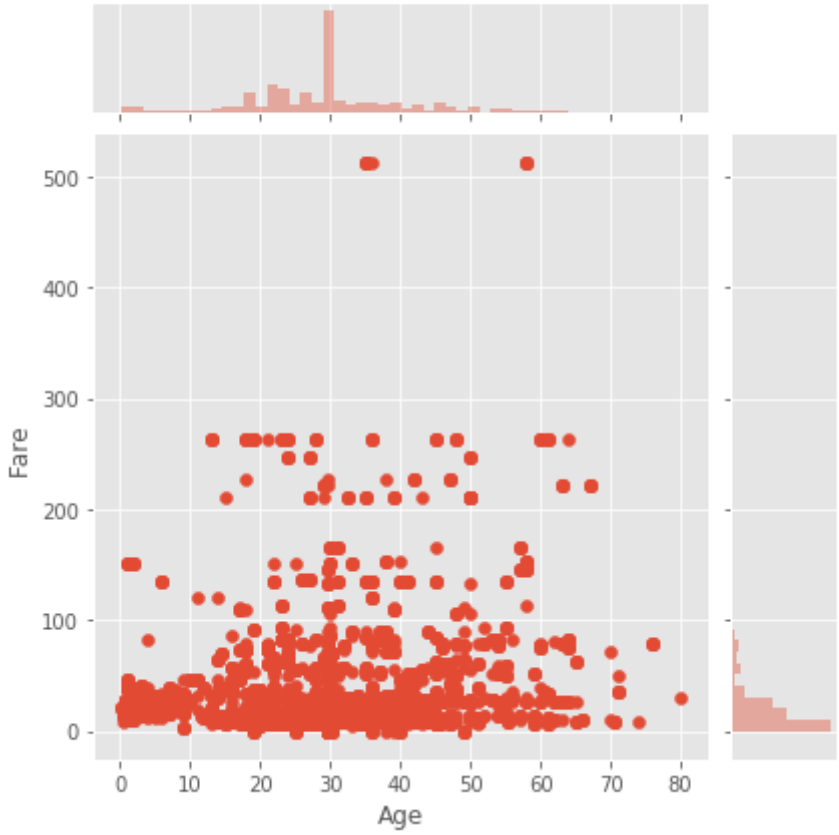
Se puede apreciar claramente la sustitución de los valores nulos por la media (la barra alta en el centro de la gráfica)

```
sns.distplot(df['Fare']);
```



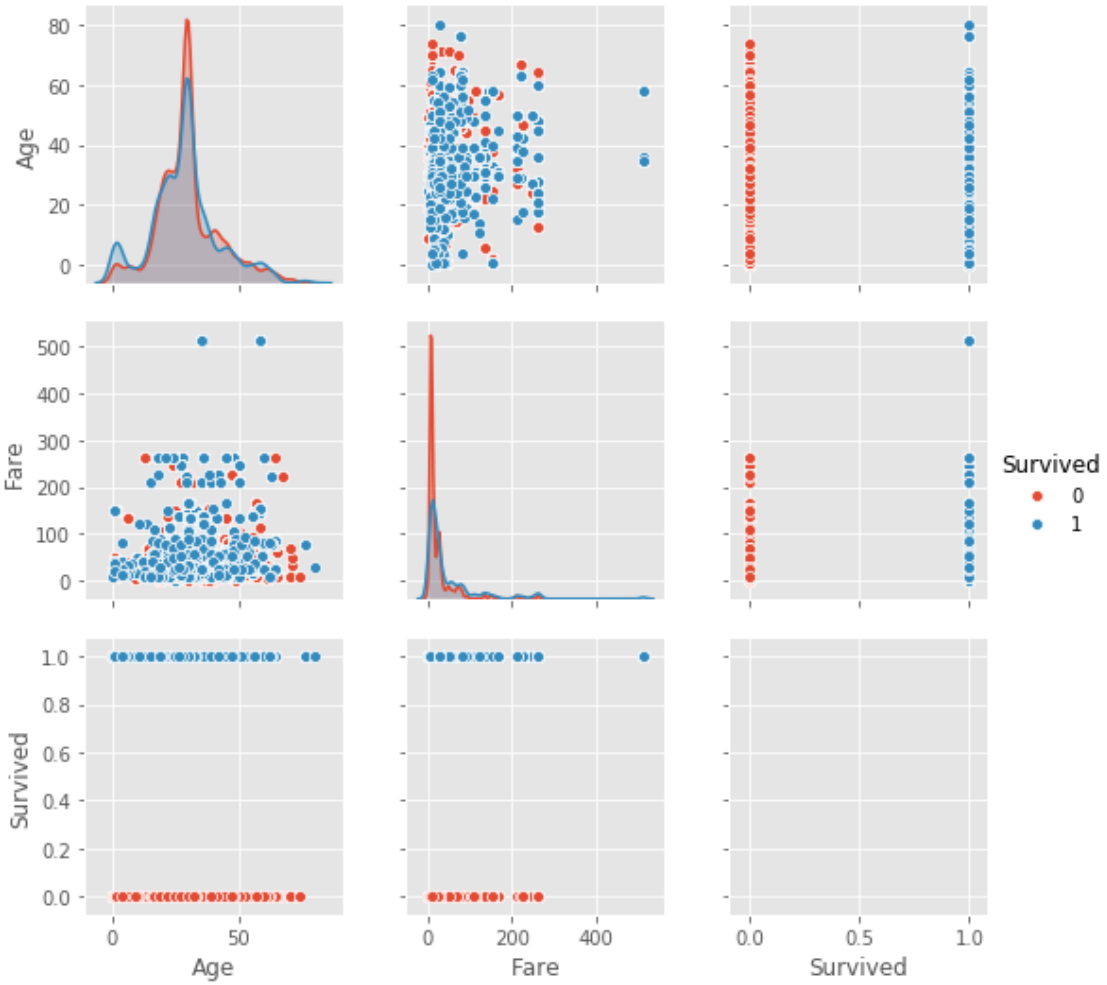
Gráfica de correlación entre edad y tarifa

```
sns.jointplot(x="Age", y="Fare", data=df);
```



Añadimos diferenciación mediante color entre los supervivientes y no supervivientes:

```
sns.pairplot(df[['Age', 'Fare', 'Survived']], hue="Survived");
```



El línea con los estudios estadísticos realizados anteriormente, vemos que ninguna de estas dos variables aporta una diferenciación clara a la hora de determinar la supervivencia.

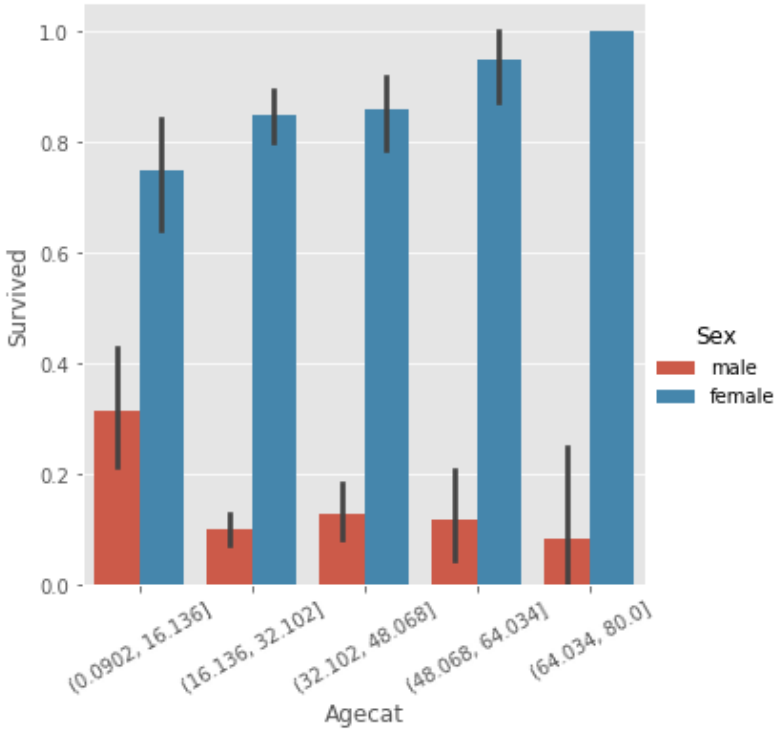
Gráficas de supervivencia según franja de edad, sexo y clase

```
df['Age'].describe()
```

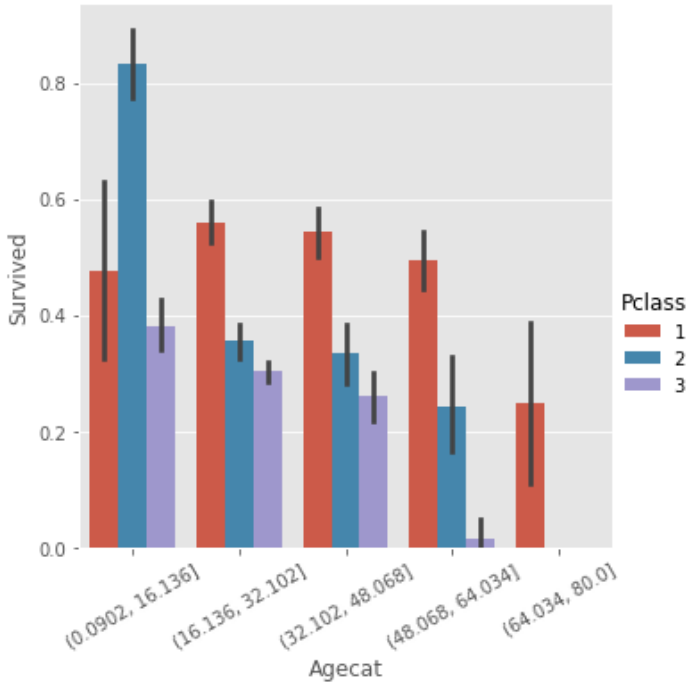
```
count    7161.000000
mean       29.633137
std        12.698897
min         0.170000
25%        22.000000
50%        29.699118
75%        35.000000
max        80.000000
Name: Age, dtype: float64
```

```
df['Agecat'] = pd.cut(df['Age'], 5)
```

```
ax = sns.catplot(x="Agecat", y="Survived", hue="Sex", kind="bar", data=df)
ax.set_xticklabels(rotation=30);
```



```
ax = sns.catplot(x="Agecat", y="Survived", hue="Pclass", kind="bar", data=df)
ax.set_xticklabels(rotation=30);
```



Podemos apreciar, que en el caso de las mujeres, cuanto más jovenes menos proporción de supervivientes. En el caso de los hombres. Los niños y adolescentes tuvieron una proporción de supervivientes superior a la del resto de franjas de edad. Por clase, podemos ver que (con la excepción de la primera clase) mayor edad supuso menor proporción de supervivencia.

6. Resolución del problema

Los resultados obtenidos nos han permitido responder al problema que comentábamos al principio, cuales fueron las variables más determinantes para la supervivencia y comprobar por ejemplo si se cumplió el protocolo "mujeres y niños primero". Como hemos podido ver en el análisis y la representación gráfica, la población femenina tuvo una mayor proporción de población superviviente que la población masculina. También hemos podido comprobar, que dentro de la población masculina, la edad influía en la supervivencia, siendo los niños y los adolescentes los que tenían mayor porcentaje de supervivencia. De este modo podemos concluir que se cumplió el protocolo "mujeres y niños primero.ª la hora de evacuar. También hemos podido comprobar que la clase también influyó en la supervivencia, siendo los de clases más altas los que tienen mas porcentaje de supervivencia. Por tanto podemos concluir que las variables seleccionadas para el análisis fueron las más determinantes para la supervivencia o no de los pasajeros.

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