```
#Importamos modulos
import io
import pandas as pd
import requests
import seaborn as sns
import timeit
import matplotlib.pyplot as plt
import numpy as np
```

```
#Cargamos el dataset de los pasajeros del Titanic
url="https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv"
s=requests.get(url).content
titanic=pd.read_csv(io.StringIO(s.decode('utf-8')))
titanic.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    object
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    object
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

```
#Renombra la columna class=clase y fare=tarifa
titanic.rename(columns={'class': 'clase'}, inplace=True)
titanic.rename(columns={'fare': 'tarifa'}, inplace=True)
#Muestra los valores distintos para class(clase)
titanic.clase.unique()
#Primeras 5 filas
titanic.head(5)
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	al
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	F
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	F
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	T
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	F
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	T

```
#Añadimos nuevas columnas: is_old, is_baby
###############################################
def is_old_func(row):
  return row['age'] > 60

titanic['is_old'] = titanic.apply(is_old_func, axis='columns')

#Otra forma de definir una nueva columna
titanic.eval ( ' is_baby = age< 15 ' , inplace = True)
titanic.head(5)
```

	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	al
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	F
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	F
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	T
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	F
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	T

```
#Define una variable numérica: class_num
def class_num_func(row):
  Clase={'Third':3,'First':1,'Second':2}
  return Clase[row.clase]
titanic['class_num'] = titanic.apply(class_num_func, axis='columns')
titanic.head(5)
```

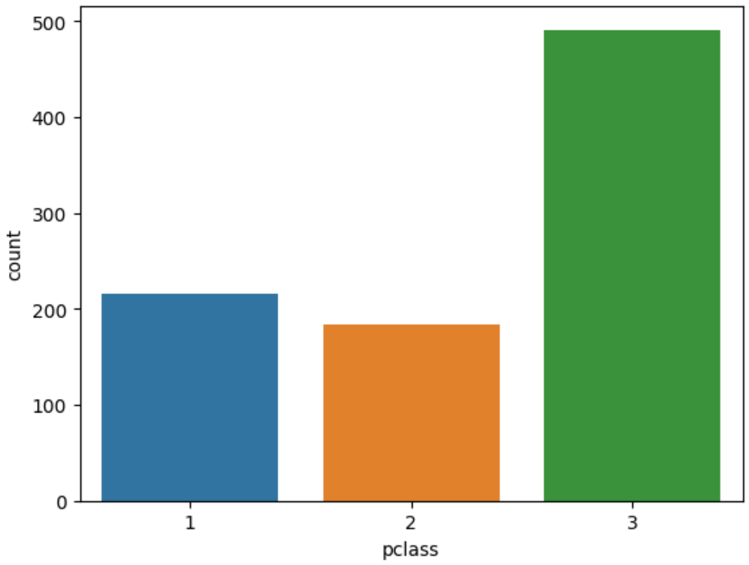
	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive	al
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	F
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	F
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	T
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	F
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	T

```
#Consulta con condiciones
titanic[
    (titanic.sex == 'female')
    & (titanic['clase'].isin(['First', 'Third']))
    & (titanic.age > 45 )
    & (titanic.survived == 0)
    ]
```

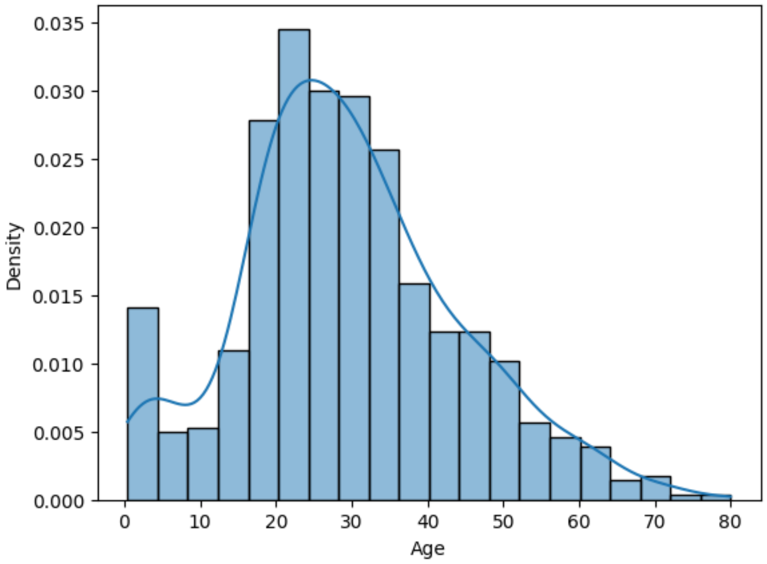
	survived	pclass	sex	age	sibsp	parch	tarifa	embarked	clase	who	adult_male	deck	embark_town	alive
132	0	3	female	47.0	1	0	14.5000	S	Third	woman	False	NaN	Southampton	no
177	0	1	female	50.0	0	0	28.7125	C	First	woman	False	C	Cherbourg	no
736	0	3	female	48.0	1	3	34.3750	S	Third	woman	False	NaN	Southampton	no

```
#Distribución de las clases
sns.countplot(x="pclass", data=titanic)
```

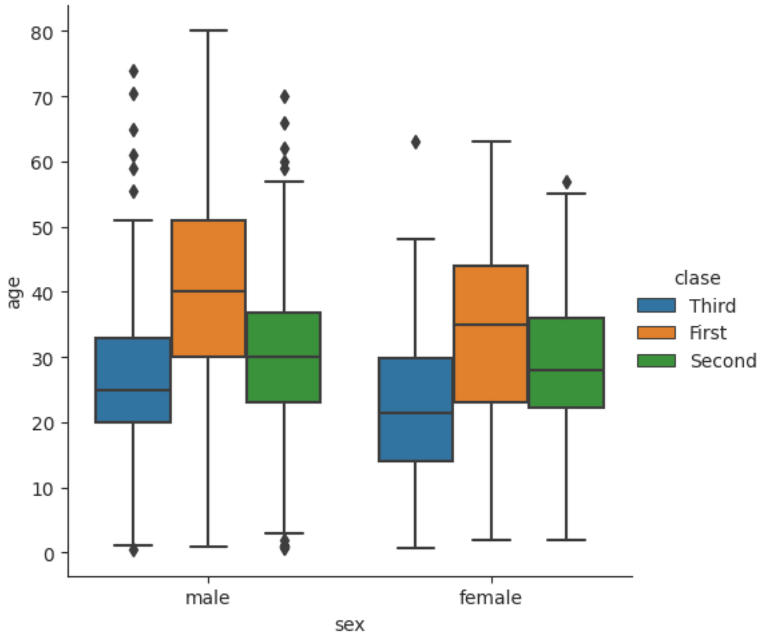
```
<AxesSubplot: xlabel='pclass', ylabel='count'>
```



```
#Distribución de la edad(ege)
sns.histplot(titanic.age.dropna(), stat="density", kde=True)
plt.xlabel("Age")
plt.ylabel("Density")
plt.show( )
```

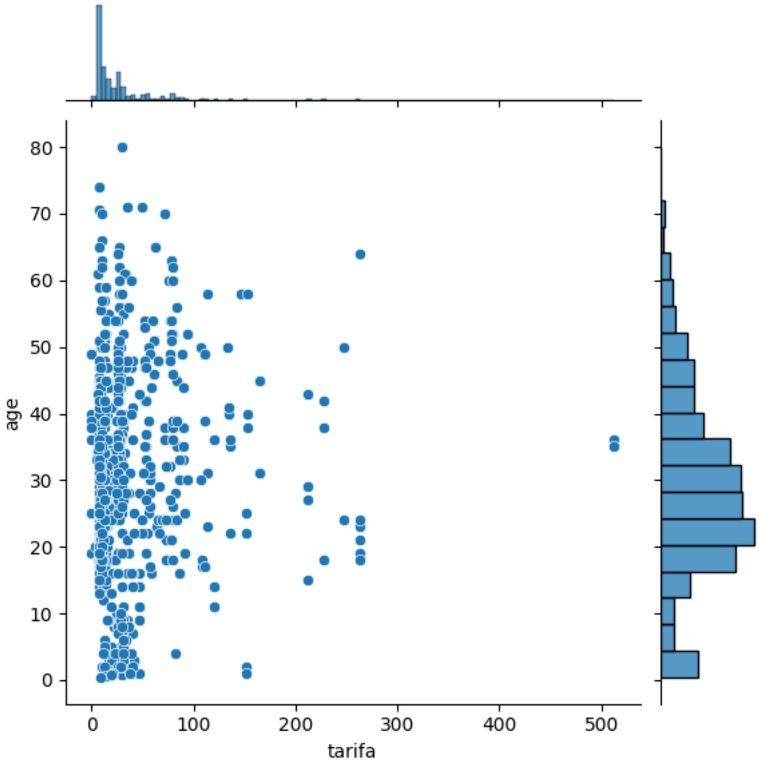


```
#BoxPlot de la edad por sexo y clase
with sns.axes_style(style='ticks'):
  ax = sns.catplot(data=titanic, x="sex", y="age", hue="clase", kind="box")
```

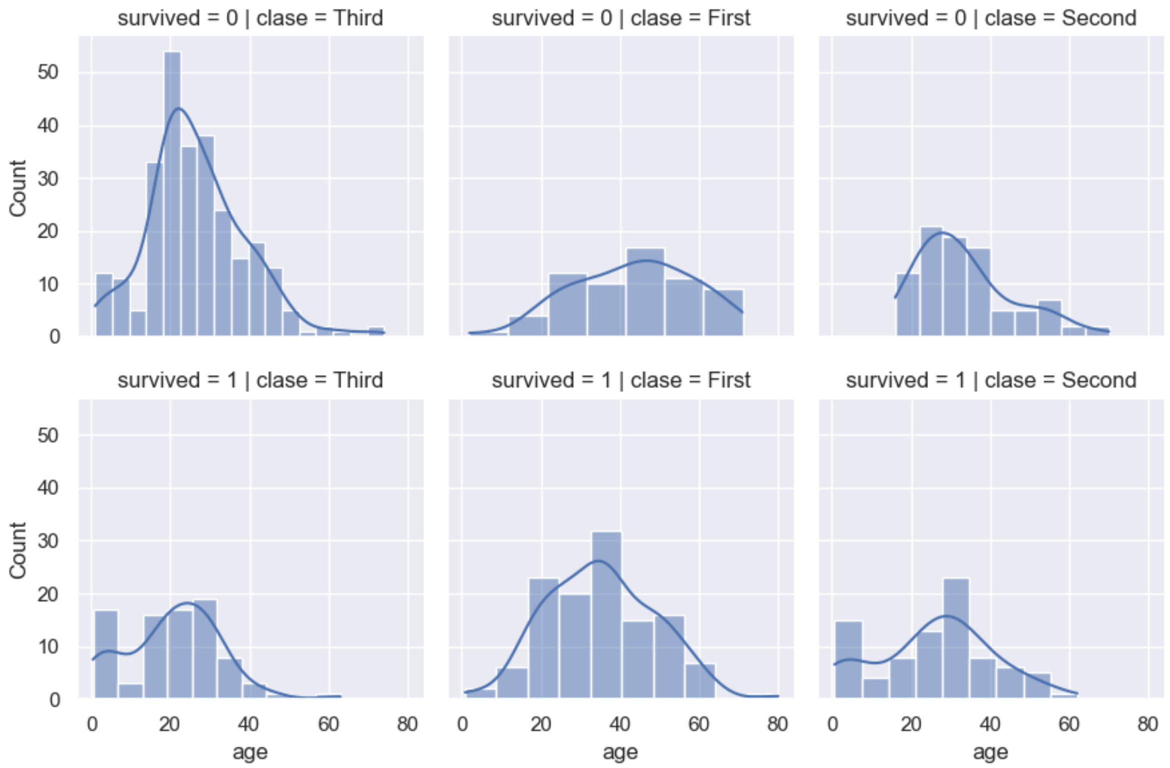


```
#Distribución cruzada de Edad y Tarifa
sns.jointplot(x='tarifa',y='age',data=titanic)
```

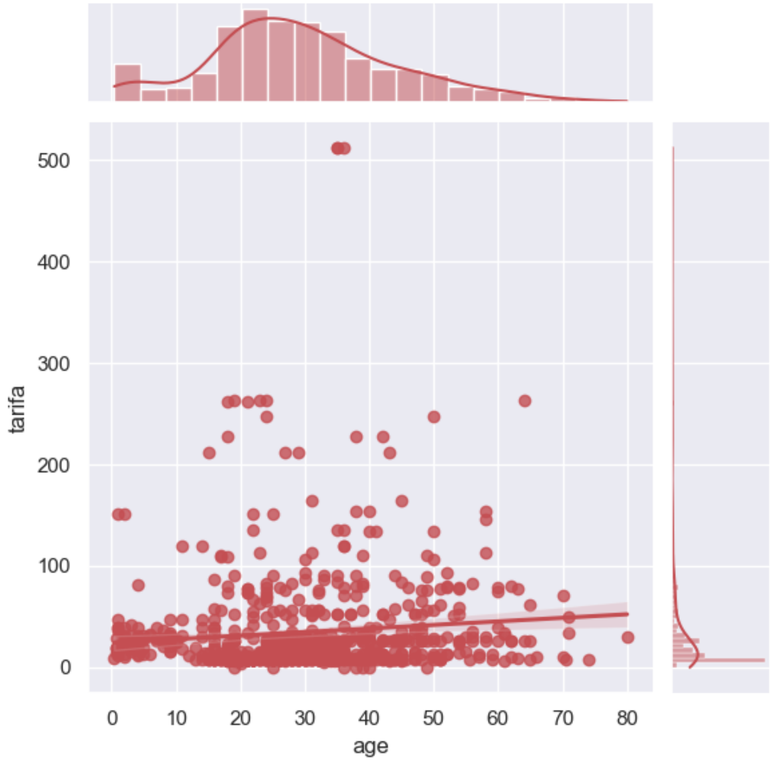
```
<seaborn.axisgrid.JointGrid at 0x13bb34970>
```



```
#Cambiamos el font
sns.set(font_scale=1)
#FacetGrid - Construir una matriz de gráficos
g = sns.FacetGrid(titanic, row='survived',col='clase' )
g.map(sns.histplot, "age", kde=True)
plt.show()
```

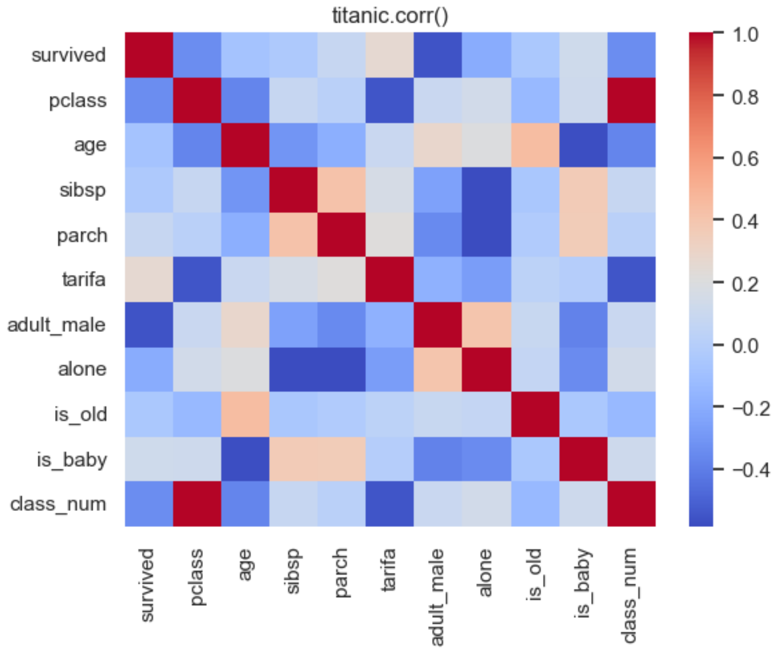


```
#Diagrama de dispersion con Distribucion de cada variable: fare(precio)/age(edad)
sns.jointplot(data=titanic, x='age', y='tarifa', kind='reg', color='r')
plt.show()
```

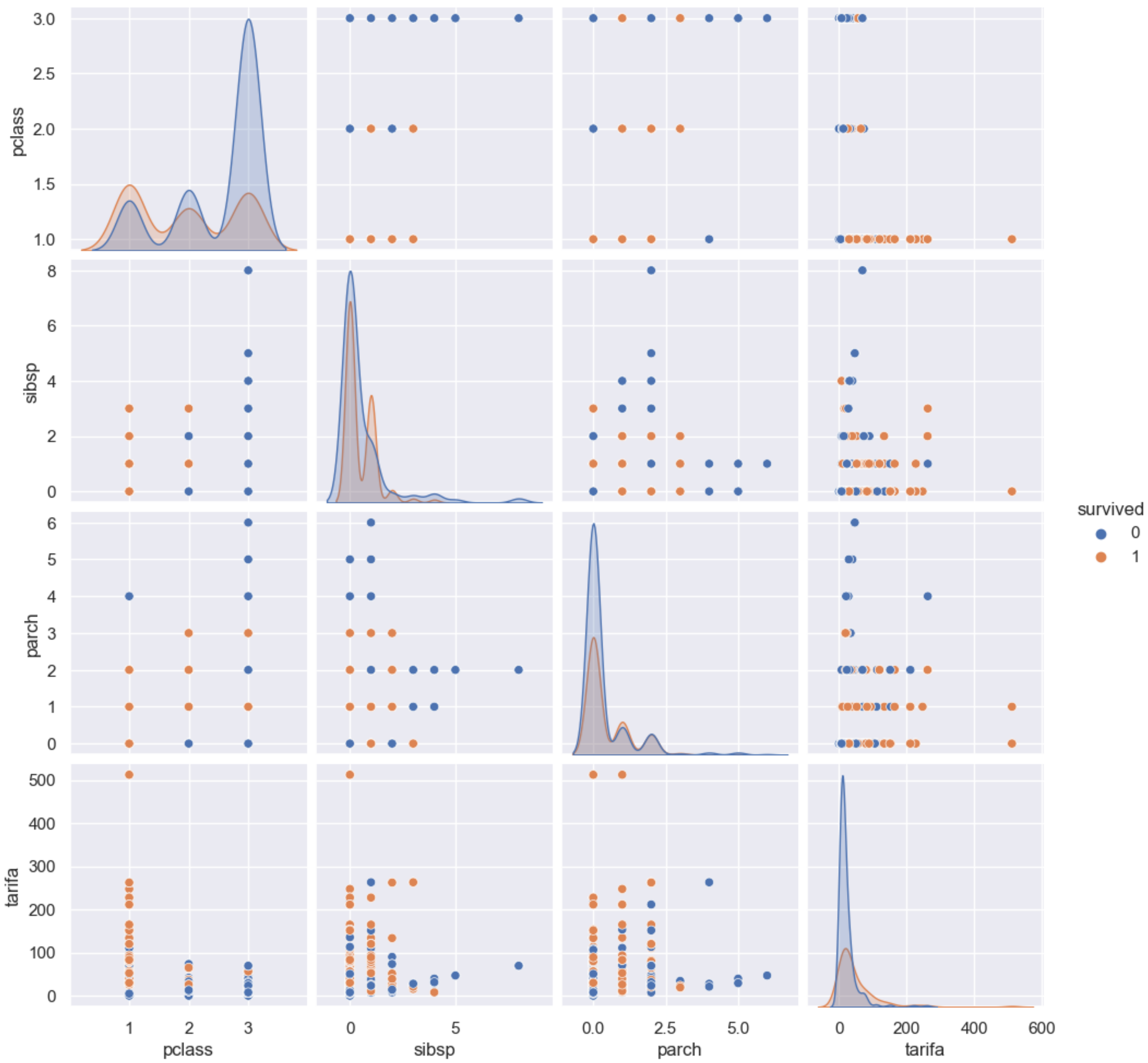


```
# Mapa de calor de correlaciones
tc = titanic.corr()
sns.heatmap(tc,cmap='coolwarm')
plt.title('titanic.corr()')
```

```
/var/folders/s_/4s3r9xbj6vbcd1v786z79f9c0000gn/T/ipykernel_5397/936654290.py:2: FutureWarning: The default value of numeric_only
  tc = titanic.corr()




Text(0.5, 1.0, 'titanic.corr()')
```

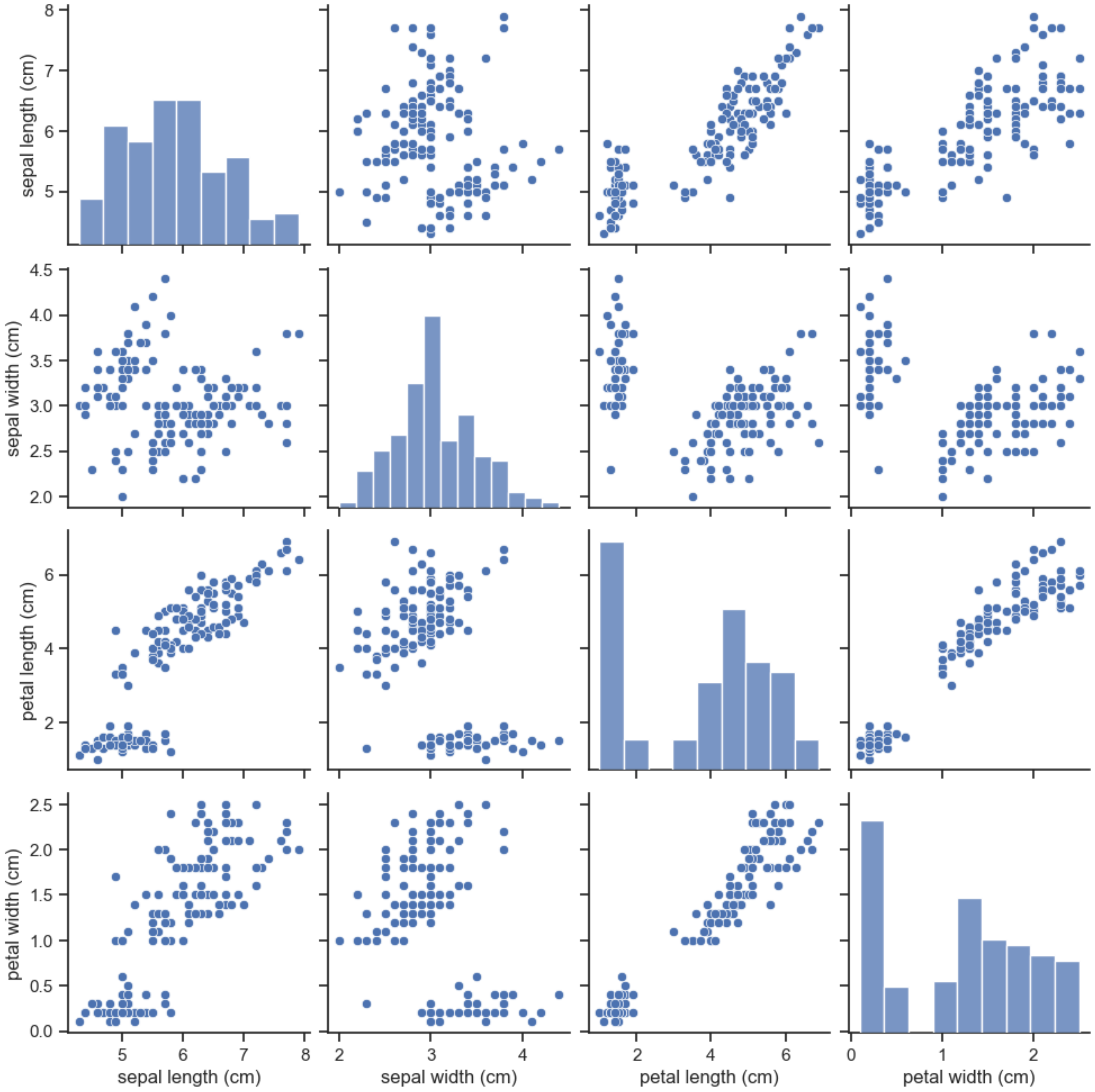


```
#Define un subconjuto de datos con las variables numéricas
titanic_num = titanic[['survived','pclass','sibsp','parch','tarifa']]
#Hace una matriz de diagramas de dispersión de parejas de variables.
sns.pairplot(titanic_num, hue="survived")
plt.show()
```

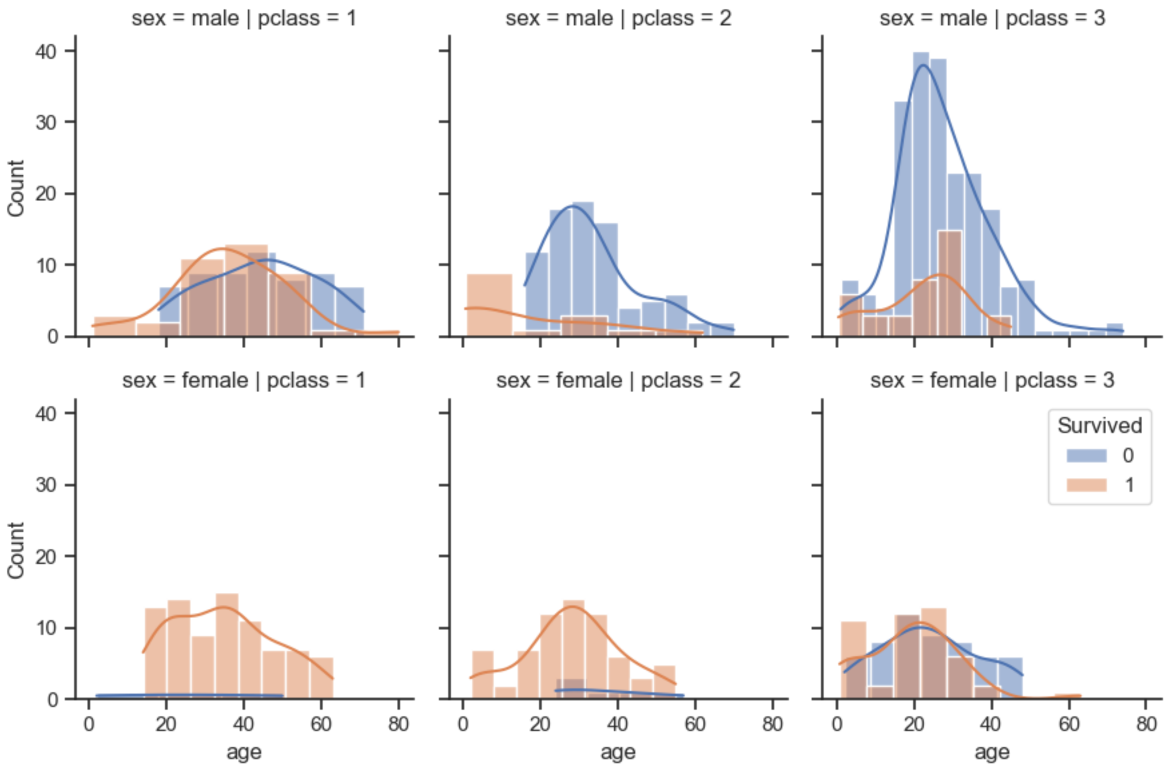


```
#Carga el data set
from sklearn import datasets

#Establece el estilo estético de las tramas
sns.set(style="ticks")

data = datasets.load_iris()
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df.head()
#df = sns.load_dataset(iris_file)
#matriz de diagramas de dispersion
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x28559af50>
```



```
#Ampliación de la práctica

#FacetGrid - Construir una matriz de gráficos
g = sns.FacetGrid(titanic, row='sex',col='pclass' , hue="survived")
g.map(sns.histplot, "age", kde=True)
plt.legend(title="Survived")
plt.show()
```


sex = male | pclass = 1
sex = male | pclass = 2
sex = male | pclass = 3
sex = female | pclass = 1
sex = female | pclass = 2
sex = female | pclass = 3
Count
Count
0
10
20
30
40
0
20
40
60
80
age
Survived
0
1