In [1]:
```
#datos de titanic de kaggel
import pandas as pd
datos=pd.read_csv("train.csv")
```

In [2]:
```
datos.head(5)
```

Out[2]:

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

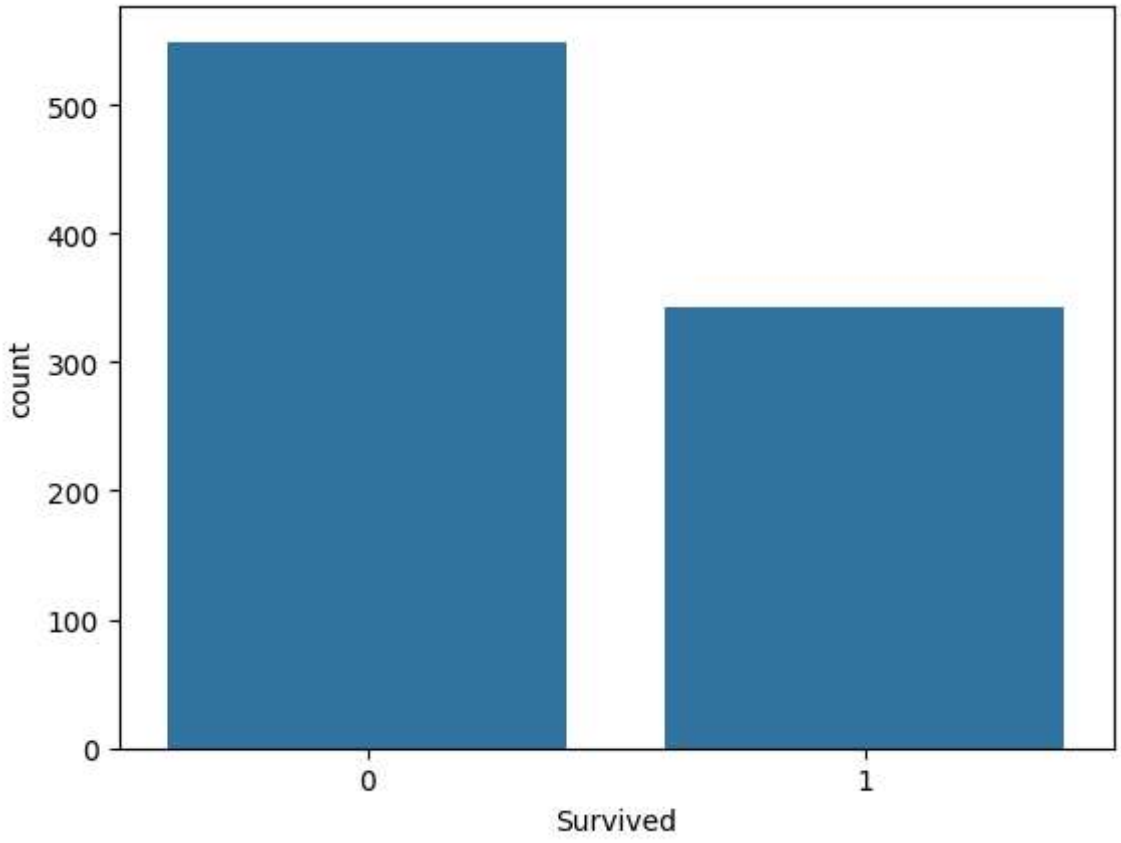
In [3]:
```
datos.describe()
```

Out[3]:

	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

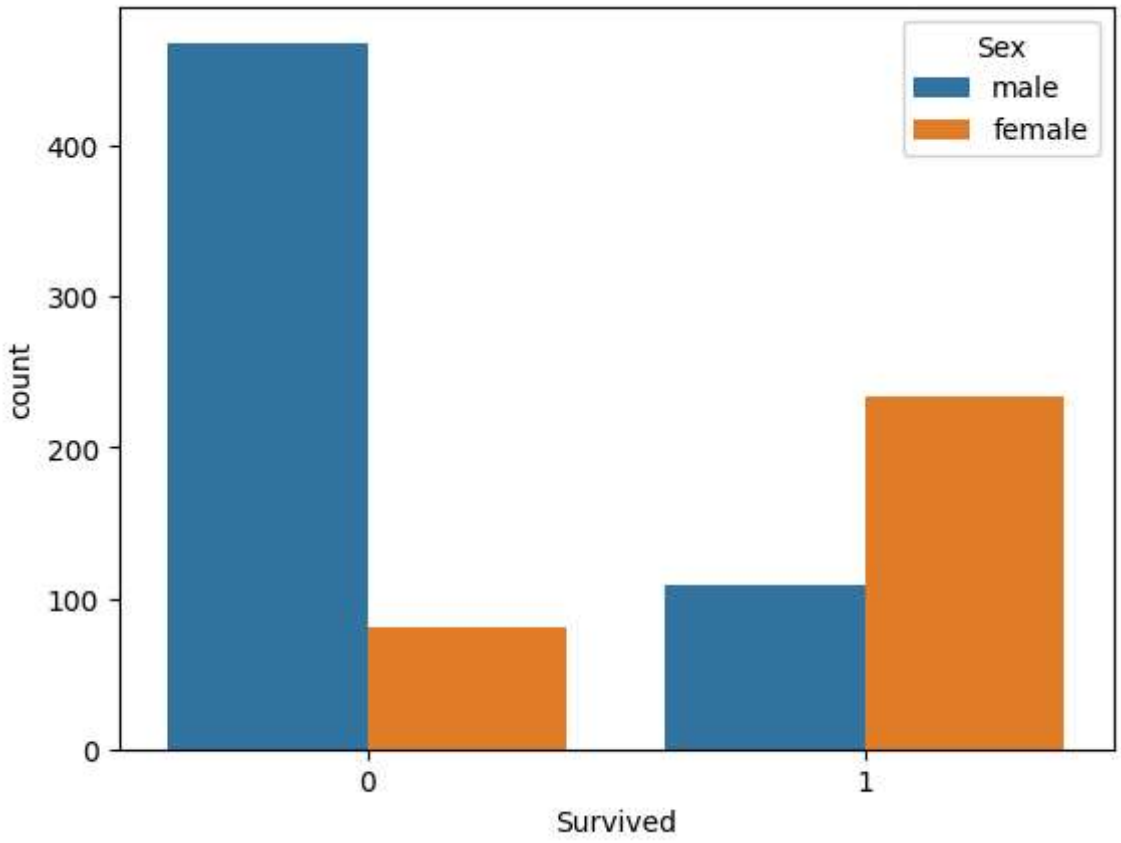
In [5]:
```
#vamos a graficar de la data cuantos sobrevivieron y cuantos no
import seaborn as sb
sb.countplot(x="Survived",data=datos)
```

Out[5]: <Axes: xlabel='Survived', ylabel='count'>



In [6]:
```
#con hue puedo pedir que me separe datos por sexo
sb.countplot(x="Survived",data=datos,hue="Sex")
```

Out[6]: <Axes: xlabel='Survived', ylabel='count'>

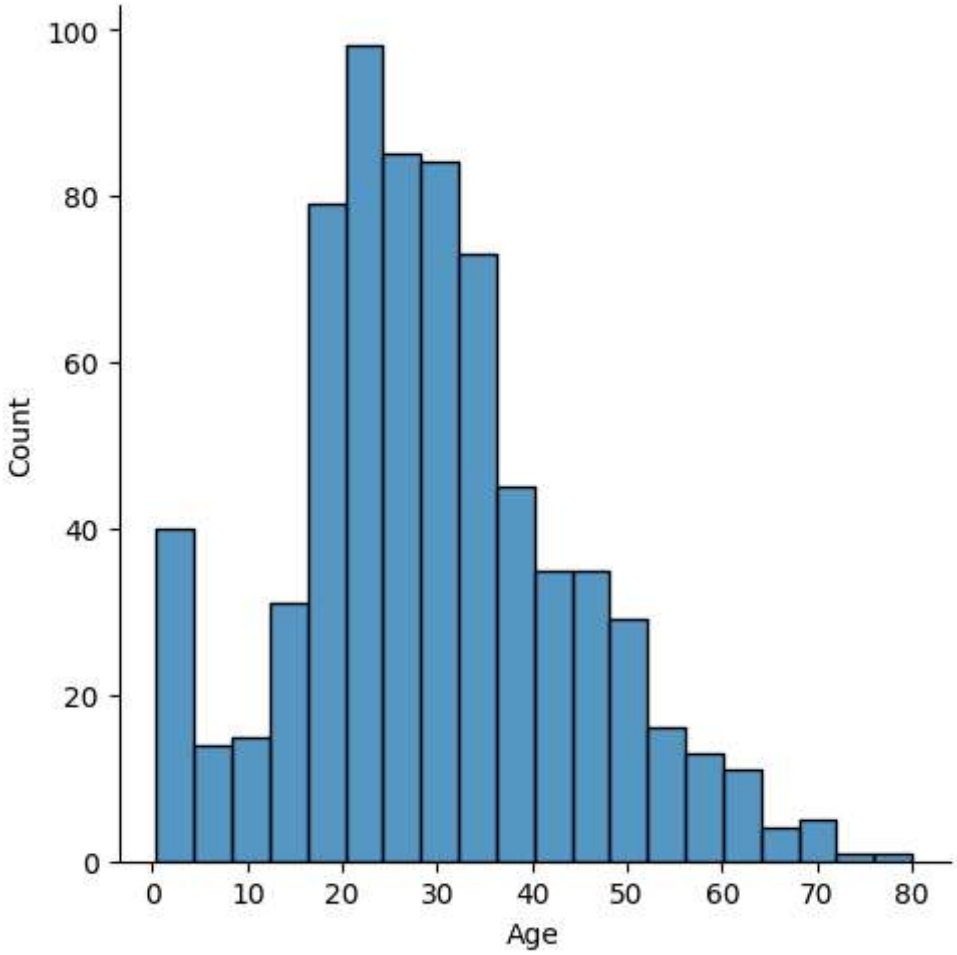


In [8]:
```
#verificamos en que columnas tenemos datos na y cuantos
datos.isna().sum()
```

Out[8]:
```
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
```

In [9]:
```
sb.displot(x="Age",data=datos)
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x134f4767860>
```



In [10]:
```
#llenar los 177 datos na con el promedio de edades
#fillna llena los datos na  y dentro de parentesis se pone el promedio para que lo lle
datos["Age"].fillna(datos["Age"].mean())  #esta instruccion no actualiza el dataset
datos["Age"]=datos["Age"].fillna(datos["Age"].mean())  #esta instruccion si actualiza
datos["Age"]
```

Out[10]:
```
0      22.000000
1      38.000000
2      26.000000
3      35.000000
4      35.000000
         ...
886    27.000000
887    19.000000
888    29.699118
889    26.000000
890    32.000000
Name: Age, Length: 891, dtype: float64
```

In [11]:
```
#verificamos en que columnas tenemos datos na y cuantos
datos.isna().sum()
```

Out[11]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [12]:
```
#los de cabina si vamos a quitar la caracteristica(columna) ya que no nos aporta valor
datos=datos.drop("Cabin",axis=1)
datos.isna().sum()
```

Out[12]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

In [15]:
```
#Ahora con los embarked solo vamos a borrar esas dos filas que tienen na en el campo e
datos=datos.dropna(subset=["Embarked"])
datos.isna().sum() #ahora ya no tenemos na
```

Out[15]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [16]:
```
datos.head(5)
```

Out[16]:

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

In [18]:
```
#ahora quitamos las columnas o caracteristicas que no nos sirven.
datos=datos.drop(["PassengerId","Name","Ticket"],axis=1)
datos.head(5)
```

Out[18]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [21]:
```
#ahora tenemos que convertir a numero las columnas categoricas como por ejemplo Sex
pd.get_dummies(datos["Sex"]).astype(int)   #el astype(int) fue necesario porque me reg
```

Out[21]:

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
886	0	1
887	1	0
888	1	0
889	0	1
890	0	1

889 rows × 2 columns

In [23]:
```
#pero en este caso no necesitamos las dos columnas ya que una es 0 y la otra 1 siempre
pd.get_dummies(datos["Sex"],drop_first=True).astype(int)   #drop_first quita la primer
#vamos a agregarlo a que el resultado lo ponga en una variable
dummies_sex=pd.get_dummies(datos["Sex"],drop_first=True).astype(int)
```

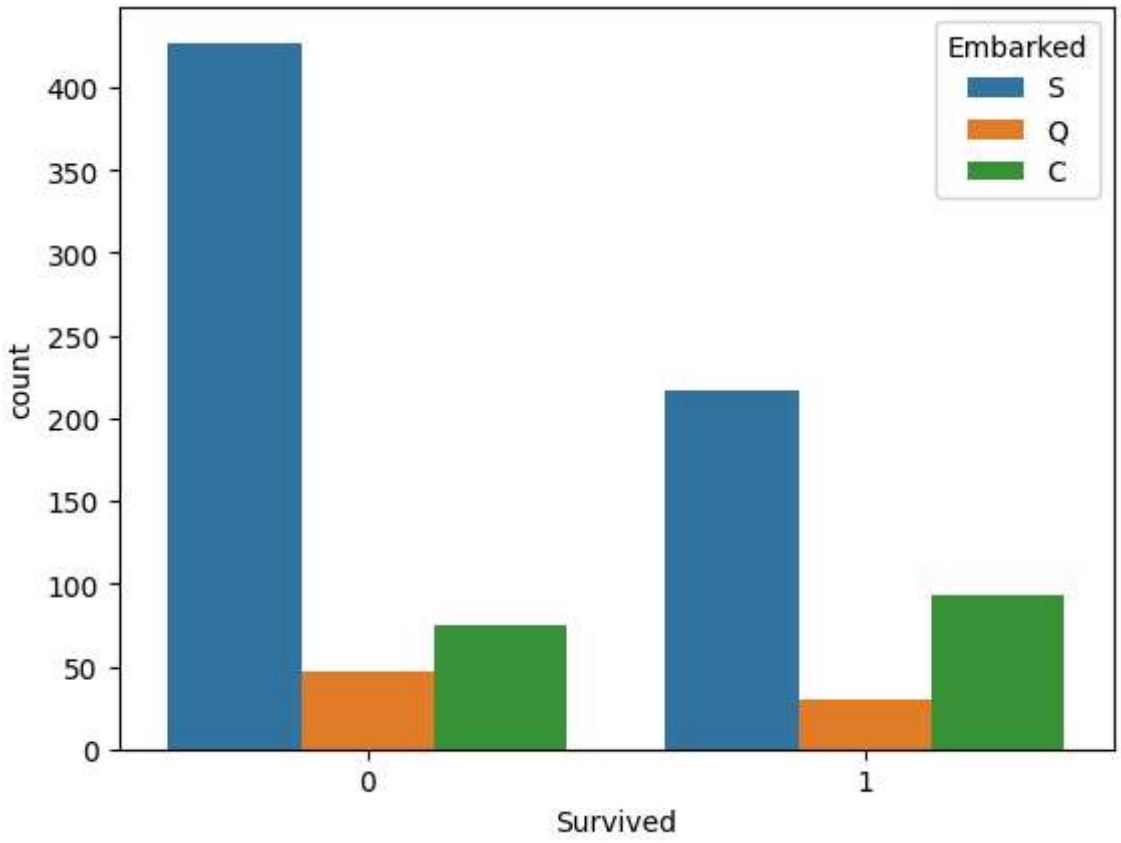
In [25]:
```
#ahora debo agregar esa columna dummiesSex a datos y quitar la de sex
datos=datos.join(dummies_sex) #agrega la columna nueva
datos=datos.drop('Sex', axis=1) #quita la columna sex
datos.head(5)
```

Out[25]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Embarked	male
0	0	3	22.0	1	0	7.2500	S	1
1	1	1	38.0	1	0	71.2833	C	0
2	1	3	26.0	0	0	7.9250	S	0
3	1	1	35.0	1	0	53.1000	S	0
4	0	3	35.0	0	0	8.0500	S	1

In [27]:
```
#ahora debemos ver que hacemos con embarked, y veremos si es relevante mantenerla
sb.countplot(x="Survived",data=datos,hue="Embarked")
```

Out[27]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



In [30]:
```
#pero en este caso no necesitamos las tres columnas ya que una es 0 y la otra 1 siempr
pd.get_dummies(datos["Embarked"],drop_first=True).astype(int)   #drop_first quita la p
#vamos a agregarlo a que el resultado lo ponga en una variable
dummies_embarked=pd.get_dummies(datos["Embarked"],drop_first=True).astype(int)
dummies_embarked
```

Out[30]:

	Q	S
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1
...	...	...
886	0	1
887	0	1
888	0	1
889	0	0
890	1	0

889 rows × 2 columns

In [31]:
```
#ahora debo agregar esas 2 columna dummiesembarked a datos y quitar la de embarked
datos=datos.join(dummies_embarked) #agrega la columna nueva
```

```
datos=datos.drop('Embarked', axis=1) #quita la columna embarked
datos.head(5)
```

Out[31]:

	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	0	3	22.0	1	0	7.2500	1	0	1
1	1	1	38.0	1	0	71.2833	0	0	0
2	1	3	26.0	0	0	7.9250	0	0	1
3	1	1	35.0	1	0	53.1000	0	0	1
4	0	3	35.0	0	0	8.0500	1	0	1

In [33]:
```
#ahora veamos las correlaciones
#como la tabla de arriba cuesta entenderla mejor generemos una grafica con esos datos
sb.set(rc={'figure.figsize':(12,6)}) #esta linea configura el tamanyo de la grafica pa
sb.heatmap(datos.corr(),annot=True,cmap="YlGnBu") #el cmap le damos los tres colores q
#la grafica lo que nos da es la correlacion entre todas las variables donde 0 signific
#y 1 significa que tienen una relacion perfecta
#los positivos tienen relacion y los negativos tienen una relacion inversa
# por ejemplo el la relacion -0.54 de si es male dice qu esi eres hombre hay alta prob
# por ejemplo pclas tambien mientras mas baja la clase mas probabilidad tienen de sobr
# sepuee leer ente menos seas hombre mas probabilidad de sobrevivir
# entre menos clase tengas mas probabilidad de sobrevivir ojo que es ente mas peque;o
# es decir que los de primera clase mas probabilidad tienen de sobrevivir
# entre mas pages (fare) mas probabilidad tienes de sobrevir, si fuera negativo fuera
```

Out[33]:
```
<Axes: >
```

Empezamos el entrenamiento

In [34]:
```
X=datos.drop(["Survived"],axis=1)
y=datos["Survived"]
```

In [35]:
```
from sklearn.model_selection import train_test_split
X_ent,X_pru,y_ent,y_prue = train_test_split(X,y,test_size=0.2)
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
#el maximo de iteraciones fue necesrio ponerlo porque daba error diciendo que se llego
# y este numero es la cantidad de veces que le va dar la vuelta tratando de ajustar el
modelo = LogisticRegression(max_iter=1000)
modelo.fit(X_ent,y_ent)
```

Out[37]:

LogisticRegression

LogisticRegression(max_iter=1000)

In [38]:
```
#ahora hacemos preducciones
predicciones=modelo.predict(X_pru)
```

In [40]:
```
#ahora midamos la exactitud de nuestro modelo o el accuraty
from sklearn.metrics import accuracy_score
accuracy_score(y_prue,predicciones)
```

Out[40]:
```
0.8370786516853933
```

In [43]:
```
from sklearn.metrics import classification_report
print(classification_report(y_prue,predicciones))
```

```
              precision    recall  f1-score   support

           0       0.87      0.88      0.87       114
           1       0.78      0.77      0.77        64

    accuracy                           0.84       178
   macro avg       0.82      0.82      0.82       178
weighted avg       0.84      0.84      0.84       178
```

In [46]:
```
#vamos a usar la matriz de confusion para revisar la exactitud del modelo
from sklearn.metrics import confusion_matrix
confusion_matrix(y_prue,predicciones)
```

Out[46]:
```
array([[100,  14],
       [ 15,  49]], dtype=int64)
```

In [47]:
```
#mejoremos la data de arriba para entenderla
pd.DataFrame(confusion_matrix(y_prue,predicciones),columns=["pred: No","Pred: Si"],inc
#esta matriz nos dice que el modelo predijo 100 veces que no correctamente pero predij
#predijo el modelo 14(se equivoco 14 veces) veces que si cuando era no y 49 veces si q
```

Out[47]:

	pred: No	Pred: Si
Real: No	100	14
Real: Si	15	49

In [54]:
```
X_pru.head(5)
```

Out[54]:

	Pclass	Age	SibSp	Parch	Fare	male	Q	S
63	3	4.0	3	2	27.90	1	0	1
426	2	28.0	1	0	26.00	0	0	1
221	2	27.0	0	0	13.00	1	0	1
465	3	38.0	0	0	7.05	1	0	1
461	3	34.0	0	0	8.05	1	0	1

In [57]:
```
#ahora metamos a una nueva versona que el modelo nunca ha visto
import numpy as np

nueva_persona = np.array([1, 35, 0, 0, 80, 1, 0, 0])
prediccion = modelo.predict(nueva_persona.reshape(1, -1))
if prediccion[0]==1:
    print("sobreviviste")
else:
    print("no sobreviviste")
```

```
sobreviviste
c:\Users\Erick\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```