Tipología de datos · Práctica 2
2021-6 · Máster universitario en Ciencia de datos (Data Science)
Estudios de Informática, Multimedia y Telecomunicación

Práctica 2: Limpieza y Análisis de Datos

En esta práctica se elabora un caso práctico orientado a aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas.

Objetivos

Los objetivos concretos de esta práctica son:

- Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares.
- Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico.
- Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos.
- Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico.
- Actuar con los principios éticos y legales relacionados con la manipulación de datos en el ámbito de aplicación.
- Desarrollar las habilidades de aprendizaje que les permitan continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo.
- Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

Descripción de la Práctica a realizar

El objetivo de esta actividad será el tratamiento de un dataset, que puede ser el creado en la práctica 1 o bien cualquier dataset libre disponible en Kaggle (https://www.kaggle.com). Algunos ejemplos de dataset con los que podéis trabajar son:

- Red Wine Quality (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)
- Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

El último ejemplo corresponde a una competición activa de Kaggle de manera que, opcionalmente, podéis aprovechar el trabajo realizado durante la práctica para entrar en esta competición.

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?

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

3. Limpieza de los datos.

```
 - 3.1.¿Los datos contienen ceros o elementos vacíos? ¿Cómo
gestionarías cada uno de estos casos?
 - 3.2.Identificación y tratamiento de valores extremos.
```

4. Análisis de los datos.

```
 - 4.1.Selección de los grupos de datos que se quieren
analizar/comparar (planificación de los análisis a aplicar).
 - 4.2.Comprobación de la normalidad y homogeneidad de la varianza.
 - 4.3.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 a partir de tablas y gráficas.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema? **7.** Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

> **LUIS ALBERTO PICO**

Configuración Inicial y Librerías

In [103…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
from statsmodels.graphics.gofplots import qqplot #Normalidad
from scipy import stats #Pruebas Estadísticas
from sklearn.preprocessing import OneHotEncoder
from sklearn import preprocessing
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

custom_palette = ["blue", "orange", "green", "purple","yellow","red"]
sns.set_palette(custom_palette)
```

```
sns.set_context("notebook")
hue_colors = {0: "red",1: "cyan",2: "yellow"}
```

1. Descripción del Dataset

Dataset

Titanic

Autor

Kaggle. Titanic - Machine Learning from Disaster. https://www.kaggle.com/c/titanic/overview

Descripción

Este dataset contiene dos conjuntos de datos similares que incluyen información del pasajero como nombre, edad, género, clase, etc. Un conjunto de datos se titula `train.csv` y el otro se titula `test.csv` .

train.csv contiene los atributos de un subconjunto de pasajeros a bordo (891) y la variable objetivo que indica si sobrevivieron o no.

El conjunto de datos **test.csv** contiene información de los atributos de los pasajeros, más no si sobreviviron o no y este conjunto de datos nos servirá para determinar la precisión del modelo de predicción.

Dimensiones

train.csv. El Dataset esta compuesto de 891 registros (pasajeros) y 10 atributos (9 variables de entrada y 1 variable de salida)
test.csv . El Dataset esta compuesto de 418 registros (pasajeros) y 9 atributos (9 variables de entrada)

Atributos

Atributo de salida:
survival. 1 si pasajero sobrevivió y 0 de lo contrario

Atributos de Entrada (pasajeros):

- PassengerId. Identificador único de pasajero
- Pclass. Clase asociada al boleto (1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex. Sexo del pasajero
- Age. Edad en años
- SibSp. Número de hermanos(as) /cónyuges a bordo.
- Parch. Número de padres/hijos a bordo.
- Ticket. Número de ticket
- Fare. tarifa
- Cabin. número de cabina
- Embarked. Puerto de embarque (C = Cherbourg, Q = Queenstown, S = Southampton)

Importancia

El hundimiento del Titanic es uno de los naufragios más infames de la historia.

El 15 de abril de 1912, durante su viaje inaugural, el RMS Titanic, ampliamente considerado "insumergible", se hundió después de chocar con un iceberg. Desafortunadamente, no había suficientes botes salvavidas para todos a bordo, lo que resultó en la muerte de 1,502 de los 2,224 pasajeros y la tripulación.

Si bien hubo algún elemento de suerte involucrado en sobrevivir, parece que algunos grupos de personas tenían más probabilidades de sobrevivir que otros, esta hipótesis será corroborrada creando un modelo predictivo que responda a la pregunta: "¿Qué tipo de personas tenían más probabilidades de sobrevivir?" utilizando los datos de los pasajeros.

2. Integración y selección de Datos

2.1 Carga del Conjunto de Datos

Procedemos a realizar la lectura de los ficheros en formato CSV `train.csv` y `test.csv` previamente descargado desde Kaggle, los almacenaremos en los dataframes **Titanic_train** y **Titanic_test**, finalmente visualizamos una muestra de los datos que contienen.

In [103…
```
titanic_train_original=pd.read_csv('train.csv')
titanic_test_original=pd.read_csv('test.csv')
titanic_train=titanic_train_original.copy()
titanic_test=titanic_test_original.copy()
titanic_train.head()
```

Out[103…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
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 [103…
```
titanic_test.head()
```

Out[103…

	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

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
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
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Procedemos a revisar la estructura y tipos de datos que contiene de los conjuntos de datos, además de los valores únicos de cada atributo.

In [103…
```
print('train \n')
titanic_train.info()
print('\n')
print('test \n')
titanic_test.info()
```

```
train

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB


test

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
```

```
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

In [104…
```
pd.DataFrame(titanic_train.nunique(),columns=['Valores Únicos'])
```

Out[104…

	Valores Únicos
PassengerId	891
Survived	2
Pclass	3
Name	891
Sex	2
Age	88
SibSp	7
Parch	7
Ticket	681
Fare	248
Cabin	147
Embarked	3

In [104…
```
pd.DataFrame(titanic_test.nunique(),columns=['Valores Únicos'])
```

Out[104…

	Valores Únicos
PassengerId	418
Pclass	3
Name	418
Sex	2
Age	79
SibSp	7
Parch	8
Ticket	363
Fare	169
Cabin	76
Embarked	3

Los datos del pasajero estan consituidos por 5 atributos de texto y 6 atributos numéricos, se considera como atributos categóricos a `Pclass` y `Survived` , los nombres de los atributos los guardaremos en dos listas que indiquen cuales son cualitativos y cuales son cuantitativos, adicional el conjunto de train contiene la variable objetivo que toma un valor numérico.

No se consideran para el análisis los atributos `Name`, `PassengerId` y `Ticket` por cuanto son identificadores únicos de los pasajeros y sus boletos que no aportarán nada al análisis.

In [104…
```
titanic_train['Pclass']=titanic_train['Pclass'].astype('category')
titanic_test['Pclass'] =titanic_test['Pclass'].astype('category')
titanic_train['Survived']=titanic_train['Survived'].astype('category')


atributos_cualitativos=['Sex','Cabin','Embarked','Pclass','Survived']
atributos_cuantitativos=['Age','SibSp','Parch','Fare']

titanic_train=titanic_train.drop(columns=['Name','PassengerId','Ticket']).copy()
titanic_test =titanic_test.drop(columns=['Name','PassengerId','Ticket']).copy()
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype   
---  ------    --------------  -----   
 0   Survived  891 non-null    category
 1   Pclass    891 non-null    category
 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   Cabin     204 non-null    object  
 8   Embarked  889 non-null    object  
dtypes: category(2), float64(2), int64(2), object(3)
memory usage: 50.8+ KB
```

2.2 Análisis estadístico básico

Procedemos a visualizar estadístos básicos para los atributos cuantitativos `(media, mediana, desviación estándar, mínimo, máximo, cuartiles)` a través de la función `describe` del dataframe.

In [104…
```
titanic_train.describe()
```

Out[104…

	Age	SibSp	Parch	Fare
count	714.000000	891.000000	891.000000	891.000000
mean	29.699118	0.523008	0.381594	32.204208
std	14.526497	1.102743	0.806057	49.693429
min	0.420000	0.000000	0.000000	0.000000
25%	20.125000	0.000000	0.000000	7.910400
50%	28.000000	0.000000	0.000000	14.454200
75%	38.000000	1.000000	0.000000	31.000000
max	80.000000	8.000000	6.000000	512.329200

In [104…
```
titanic_test.describe()
```

Out[104…

	Age	SibSp	Parch	Fare
count	332.000000	418.000000	418.000000	417.000000
mean	30.272590	0.447368	0.392344	35.627188

	Age	SibSp	Parch	Fare
std	14.181209	0.896760	0.981429	55.907576
min	0.170000	0.000000	0.000000	0.000000
25%	21.000000	0.000000	0.000000	7.895800
50%	27.000000	0.000000	0.000000	14.454200
75%	39.000000	1.000000	0.000000	31.500000
max	76.000000	8.000000	9.000000	512.329200

Procedemos a visualizar estadístos básicos para los atributos cuanlitativos `unique(cantidad de valores únicos),top y frecuencia` a través de la función `describe` del dataframe.

In [104…
```
titanic_train[atributos_cualitativos].describe()
```

Out[104…

	Sex	Cabin	Embarked	Pclass	Survived
count	891	204	889	891	891
unique	2	147	3	3	2
top	male	G6	S	3	0
freq	577	4	644	491	549

In [104…
```
titanic_test[atributos_cualitativos[:-1]].describe()
```

Out[104…

	Sex	Cabin	Embarked	Pclass
count	418	91	418	418
unique	2	76	3	3
top	male	B57 B59 B63 B66	S	3
freq	266	3	270	218

2.3 Selección de Datos

Dado que el objetivo del análisis será generar una modelo predictivo que permita determinar si un pasajero sobrevivió o no en función de sus atributos sociodemográficos y del viaje, se utilizará como la variable objetivo o dependiente la variable `Survived`

- **Sobrevivió (1)**.
- **No Sobrevivió (0)**.

Las variables independientes se definen por los siguientes atributos del conjunto de datos: `Sex` , `Cabin` , `Embarked` , `Pclass` , `Survived` , `Age` , `SibSp` , `Parch` y `Fare` .

3. Limpieza de Datos

Procederemos en este apartado a determinar si los datos contienen ceros o elementos vacíos y gestionarlos en caso de existir alguno, luego identificaremos y trataremos en la medida de los posible los valores extremos.

3.1 Ceros o Elementos Vacíos

In [104…
```
pd.DataFrame(np.sum(titanic_train[atributos_cuantitativos]==0),columns=['Ceros'])
```

Out[104…

	Ceros
Age	0
SibSp	608
Parch	678
Fare	15

In [104…
```
pd.DataFrame(np.sum(titanic_test[atributos_cuantitativos]==0),columns=['Ceros'])
```

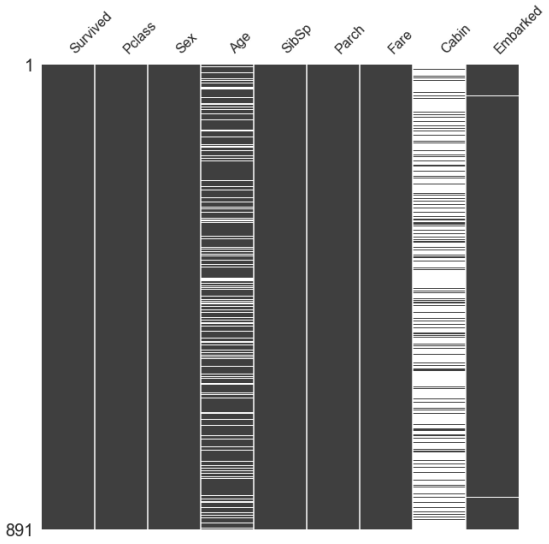
Out[104…

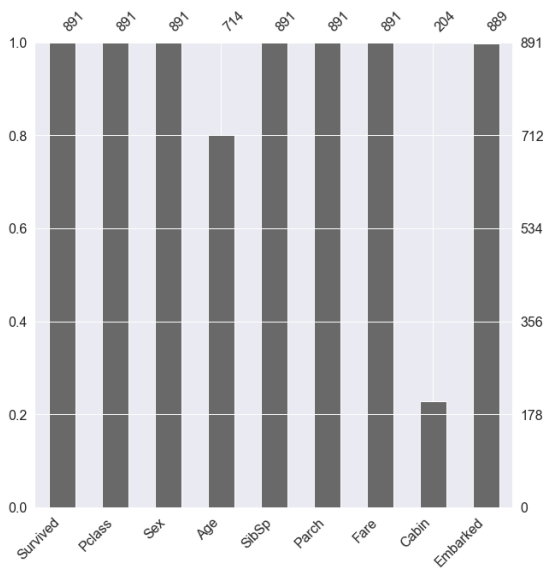
	Ceros
Age	0
SibSp	283
Parch	324
Fare	2

Las variables `SibSp` (Número de hermanos(as) /cónyuges a bordo) , `Parch` (Parch. Número de padres/hijos a bordo) y `Fare` (Tarifa) contienen valores cero, en función de sus definiciones el valor de cero es válido tanto para SibsSp y Parch, pero por la baja cantidad de valores cero se puede considerar que la tarifa fue cero para algunos invitados especiales, por lo que no se realizará ningún tipo de tratamiento para estos valores.

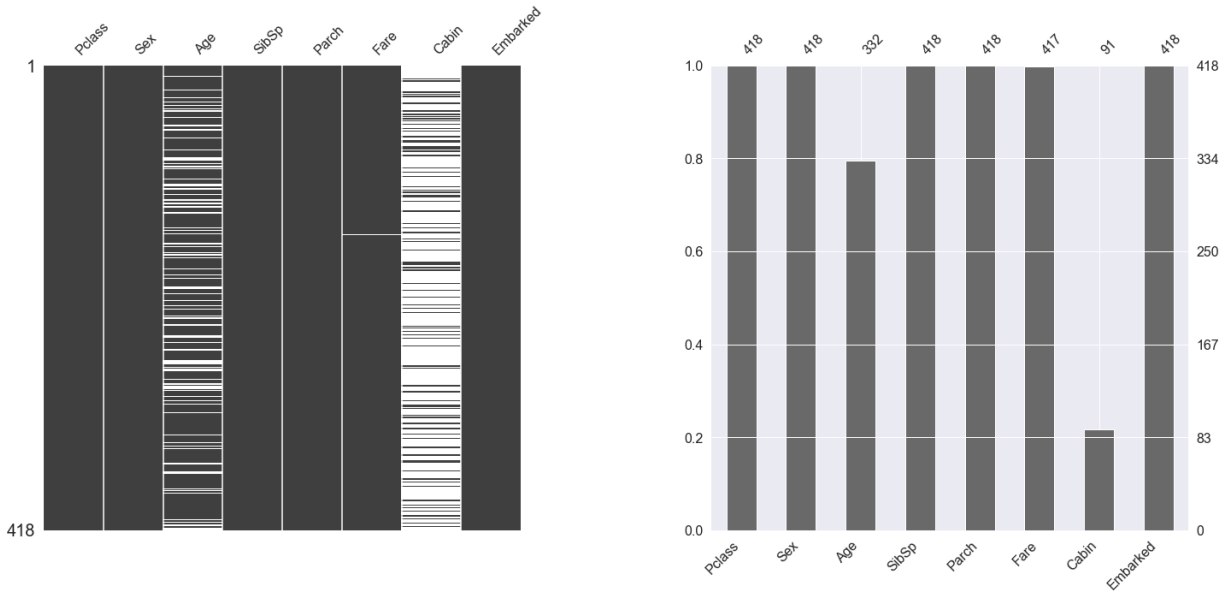
Para deteminar si los datos contienen los elementos vacíos utilizaremos la función `matrix` y `bar` de la librería **missingno** y la función `isna` del dataframe asociado a nuestros conjuntos de datos.

In [104…
```
fig = plt.figure(figsize=(4,2))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
ax = fig.add_subplot(1, 2, 1)
msno.matrix(titanic_train,ax=ax,sparkline=False)
ax = fig.add_subplot(1, 2, 2)
msno.bar(titanic_train)
plt.show()
```





In [105…
```
fig = plt.figure(figsize=(4,2))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
ax = fig.add_subplot(1, 2, 1)
msno.matrix(titanic_test,ax=ax,sparkline=False)
ax = fig.add_subplot(1, 2, 2)
msno.bar(titanic_test)
plt.show()
```



In [105…
```
pd.DataFrame(titanic_train.isna().sum(),columns=['Nulos'])
```

Out[105…

	Nulos
Survived	0
Pclass	0
Sex	0
Age	177
SibSp	0
Parch	0
Fare	0
Cabin	687
Embarked	2

In [105…
```
pd.DataFrame(titanic_test.isna().sum(),columns=['Nulos'])
```

Out[105…

	Nulos
Pclass	0
Sex	0
Age	86
SibSp	0
Parch	0
Fare	1
Cabin	327

	Nulos
Embarked	0

Las variables `Age` (Edad), `Cabin` (Número de cabina), `Fare` (Tarifa) y `Embarked` contienen valores nulos, para los atributos `Age` , `Fare` y `Embarked` realizaremos un proceso de imputación, en cambio la variable `Cabin` será excluida del análisis por cuanto el **77.81%** (687/891) de los pasajeros no tienen un valor, el **16.49%** (147/891) corresponde a valores únicos y por tanto dado que es una variable cualitativa su aporte no será significativo para el modelo.

In [105…
```
titanic_train=titanic_train.drop(columns=['Cabin']).copy()
titanic_test=titanic_test.drop(columns=['Cabin']).copy()
```

Imputación Atributo Age

El atributo `Age` lo imputaremos con el valor de la media de los pasajeros del conjunto de datos de train.

In [105…
```
missing_age = titanic_train[titanic_train['Age'].isna()]
complete_age = titanic_train[~titanic_train['Age'].isna()]
print('Missing')
print(missing_age.describe())
print('Complete')
print(complete_age.describe())
media_age=titanic_train['Age'].mean()
titanic_train = titanic_train.fillna({'Age': media_age})
titanic_test = titanic_test.fillna({'Age': media_age})
```

```
Missing
       Age       SibSp       Parch        Fare
count  0.0  177.000000  177.000000  177.000000
mean   NaN    0.564972    0.180791   22.158567
std    NaN    1.626316    0.534145   31.874608
min    NaN    0.000000    0.000000    0.000000
25%    NaN    0.000000    0.000000    7.750000
50%    NaN    0.000000    0.000000    8.050000
75%    NaN    0.000000    0.000000   24.150000
max    NaN    8.000000    2.000000  227.525000
Complete
              Age       SibSp       Parch        Fare
count  714.000000  714.000000  714.000000  714.000000
mean    29.699118    0.512605    0.431373   34.694514
std     14.526497    0.929783    0.853289   52.918930
min      0.420000    0.000000    0.000000    0.000000
25%     20.125000    0.000000    0.000000    8.050000
50%     28.000000    0.000000    0.000000   15.741700
75%     38.000000    1.000000    1.000000   33.375000
max     80.000000    5.000000    6.000000  512.329200
```

Imputación Atributo Fare

El atributo `Fare` lo imputaremos con el valor de la media de los pasajeros del conjunto de datos de train.

In [105…
```
missing_fare = titanic_test[titanic_test['Fare'].isna()]
complete_fare = titanic_test[~titanic_test['Fare'].isna()]
print('Missing')
print(missing_fare.describe())
print('Complete')
print(complete_fare.describe())
media_fare=titanic_train['Fare'].mean()
titanic_test = titanic_test.fillna({'Fare': media_fare})
```

```
Missing
        Age  SibSp  Parch  Fare
count   1.0    1.0    1.0   0.0
mean   60.5    0.0    0.0   NaN
std     NaN    NaN    NaN   NaN
min    60.5    0.0    0.0   NaN
25%    60.5    0.0    0.0   NaN
50%    60.5    0.0    0.0   NaN
75%    60.5    0.0    0.0   NaN
max    60.5    0.0    0.0   NaN
Complete
              Age       SibSp       Parch        Fare
count  417.000000  417.000000  417.000000  417.000000
mean    30.081832    0.448441    0.393285   35.627188
std     12.563849    0.897568    0.982419   55.907576
min      0.170000    0.000000    0.000000    0.000000
25%     23.000000    0.000000    0.000000    7.895800
50%     29.699118    0.000000    0.000000   14.454200
75%     35.000000    1.000000    0.000000   31.500000
max     76.000000    8.000000    9.000000  512.329200
```

Imputación Atributo Embarked

El atributo `Embarked` lo imputaremos con el valor de la moda de los pasajeros del conjunto de datos de train.

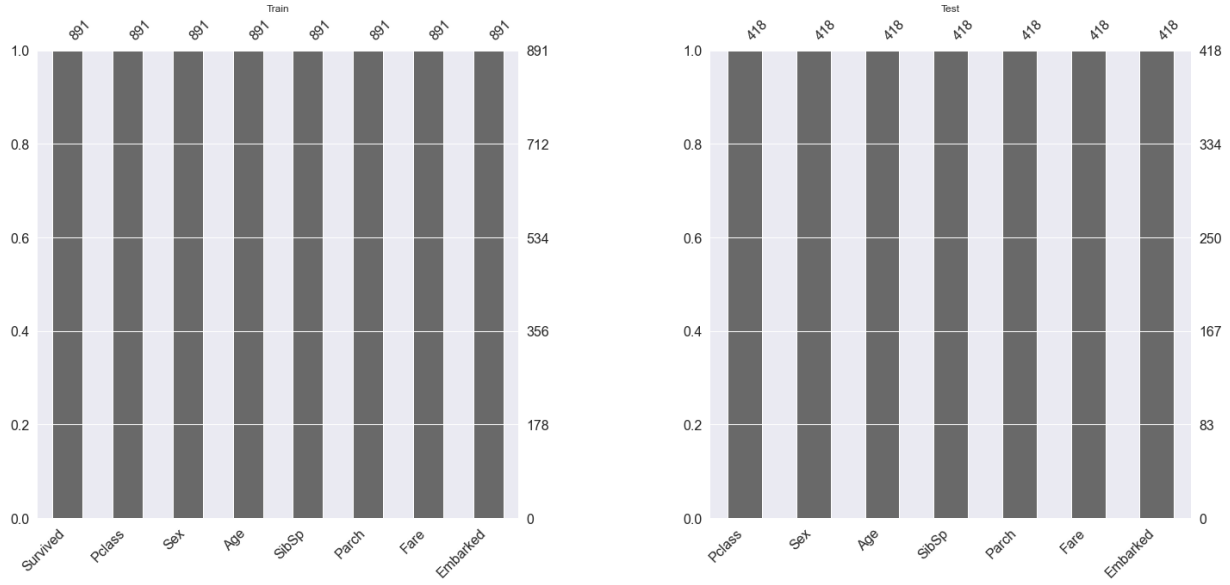
In [105…

```
missing_embarked = titanic_train[titanic_train['Embarked'].isna()]
complete_embarked = titanic_train[~titanic_train['Embarked'].isna()]
print('Missing')
print(missing_embarked.describe())
print('Complete')
print(complete_embarked.describe())
moda_embarked='S'
titanic_train = titanic_train.fillna({'Embarked': moda_embarked})
```

```
Missing
             Age  SibSp  Parch  Fare
count   2.000000    2.0    2.0   2.0
mean   50.000000    0.0    0.0  80.0
std    16.970563    0.0    0.0   0.0
min    38.000000    0.0    0.0  80.0
25%    44.000000    0.0    0.0  80.0
50%    50.000000    0.0    0.0  80.0
75%    56.000000    0.0    0.0  80.0
max    62.000000    0.0    0.0  80.0
Complete
              Age       SibSp       Parch        Fare
count  889.000000  889.000000  889.000000  889.000000
mean    29.653446    0.524184    0.382452   32.096681
std     12.968366    1.103705    0.806761   49.697504
min      0.420000    0.000000    0.000000    0.000000
25%     22.000000    0.000000    0.000000    7.895800
50%     29.699118    0.000000    0.000000   14.454200
75%     35.000000    1.000000    0.000000   31.000000
max     80.000000    8.000000    6.000000  512.329200
```

In [105…

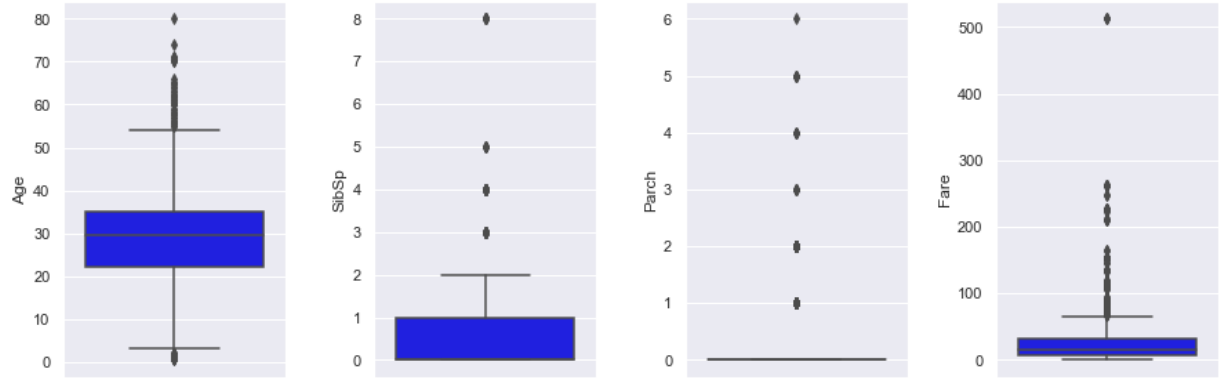
```
fig = plt.figure(figsize=(4,2))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
ax = fig.add_subplot(1, 2, 1)
ax.set_title('Train')
msno.bar(titanic_train)
ax = fig.add_subplot(1, 2, 2)
ax.set_title('Test')
msno.bar(titanic_test)
plt.show()
```



3.2 Valores Extremos

Procedemos a identificar y dar tratamiento en la medidad de lo posible a los valores extremos que se identifiquen en el conjunto de datos, para esto se utilizará el diagrama de cajas para cada una de los atributos del dataset.

In [105…
```
fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
for i,atributo in enumerate(titanic_train[atributos_cuantitativos]):
    ax = fig.add_subplot(1, 4, i+1)
    sns.boxplot(data=titanic_train,y=atributo,ax=ax)
plt.show()
```



En función de lo observado se identifica que los datos de todas las variables se encuentran en una escala de valores adecuado, por lo que no se sugiere realizar un tratamiento de valores extremos.

In [105…
```
titanic_train.to_csv('titanic_train_clean.csv')
titanic_test.to_csv('titanic_test_clean.csv')
```

4. Análisis de los Datos

Dado que nuestro objetivo de análisis será generar un modelo que permita clasificar las personas que sobrevieron o no en función de sus atributos y con esto determinar si existían grupos de personas que tenían más probabilidades de sobrevivir que otros, procederemos a seleccionar los grupos de datos que se quieren para realizar el análisis y aplicaremos algunas pruebas estadísticas para comparar estos grupos.

4.1 Selección de los grupos de Datos

Seleccionamos los grupos definidos inicialmente:

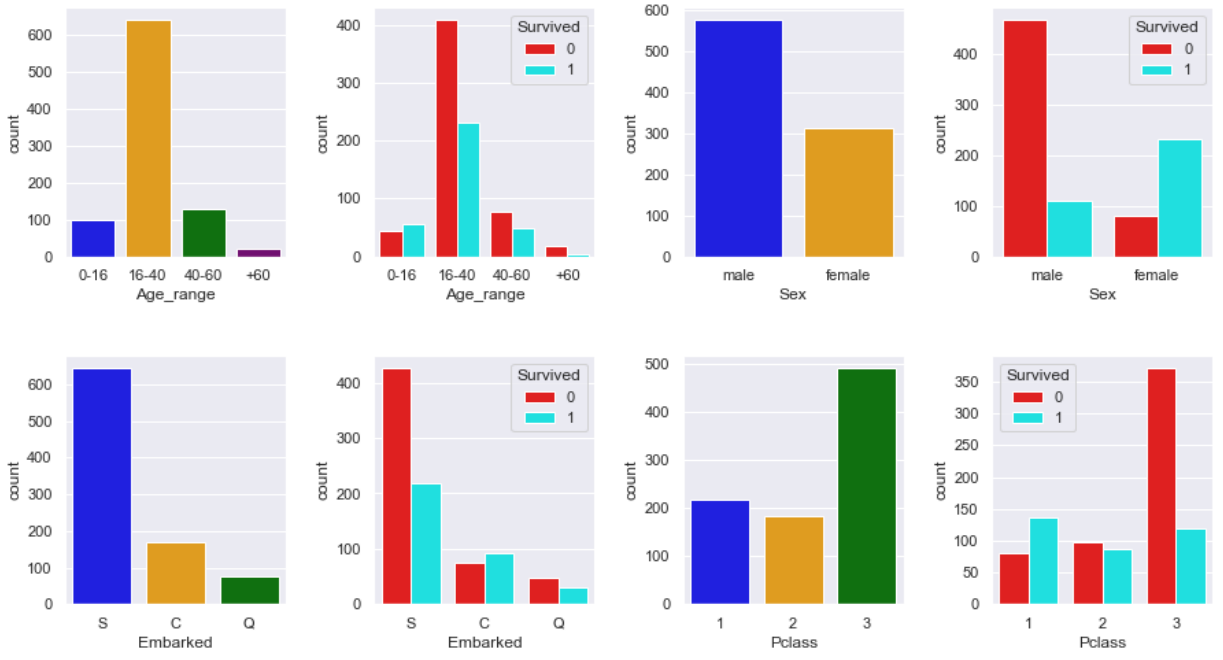
- **Survived (1)**. Pasajero sobrevivió `sobrevivio_si` .
- **Survived (0)**. Pasajero no sobrevivió `sobrevivio_no` .

Además se discretizará el atributo edad en un nuevo atributo `age_range`

In [106…
```
rangos = [0,16,40,60,np.inf]
categorias = ['0-16', '16-40', '40-60', '+60']
titanic_train['Age_range'] = pd.cut(titanic_train['Age'], bins=ranges,labels=group_n
titanic_test['Age_range'] = pd.cut(titanic_test['Age'], bins=ranges,labels=group_nam

sobrevivio_si=titanic_train[titanic_train['Survived']==1].copy()
sobrevivio_no=titanic_train[titanic_train['Survived']==0].copy()
```

Análisis de Atributos Cualitativos

In [106…
```
atributos_cualitativos=['Age_range','Sex','Embarked','Pclass']

fig = plt.figure(figsize=(15,8))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
i=1
for atributo in atributos_cualitativos:
    ax = fig.add_subplot(2,4,i)
    sns.countplot(data=titanic_train,x=atributo, ax=ax)
    ax = fig.add_subplot(2,4,i+1)
    sns.countplot(data=titanic_train,x=atributo,hue='Survived',ax=ax,palette=hue_col
    i=i+2
plt.show()
sns.set()
```



- **Rango de Edad** `range_Age`
  En función del rango de edad podemos determinar que más del 50% de los pasajeros menores a 16 años sobrevivieron, en contraste con el resto de segmentos de edad, siendo las personas de más de 60 años las que en mayor proporción murieron.

- **Sexo** `Sex`
  En función del sexo podemos determinar que en proporción las mujeres sobrevivieron mucho más que en los hombres.

- **Puerto de Embarqued** `Embarked`
  En función del puerto de embarque podemos determinar que en proporción los pasajeros que embarcaron en el puerto **Cherbourg** sobrevivieron en mayor proporción que los embarcados en el resto de puertos.

- **Calse** `Pclass`
  En función de la clase en el que viajaron los pasajeros podemos determinar que los pasajeros de primera clase sobrevivieron en mayor proporción, en contraste con el resto de clases, siendo los pasajeros de tercera clase los que en mayor proporción murieron.

Análisis de Atributos Cuantitativos

In [106…
```
sns.pairplot(titanic_train,hue='Survived',palette=hue_colors)
plt.show()
```

- **Edad** `Age`
  En función de la edad del pasajero se ratifica que a menor edad existieron mayor probabilidad de sobrevivir.

- **Número de hermanos/cónyuge a bordo** `Sibsip`
  En función del número de hermanos/cónyuge a bordo podemos determinar que aquellos que tenían 1 tuvieron más probabilidad de sobrevivir.

- **Número de padres/hijos a bordo** `Parch`
  En función del número de padres/hijos a bordo podemos determinar que aquellos que tenían 1 tuvieron más probabilidad de sobrevivir.

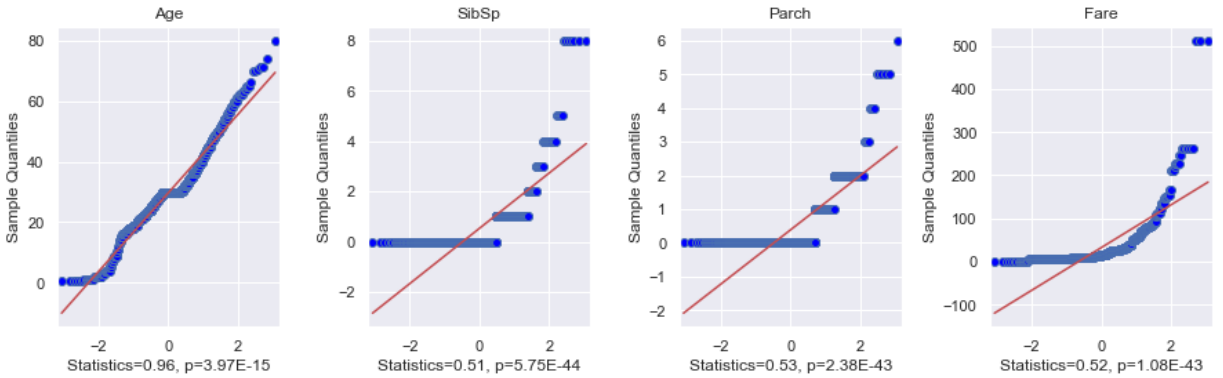
- **Tarifa** `Fare`
  En función de la tarifa que pago el pasajero podemos determinar que los pasajeros cuyo boleto de mayor valor tuvieron mayor probabilidad de sobrevivir.

4.2 Normalidad y Homogeneidad de la Varianza de los Datos

4.2.1 Normalidad

Determinamos la normalidad de las variables dependientes para esto se utilizará el método visual q-q plot y el método estadístico Shapiro-Wilk test.

In [106…
```
fig = plt.figure(figsize=(15,4))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
for i,variable in enumerate(atributos_cuantitativos):
    ax = fig.add_subplot(1, 4, i+1)
    ax.set_title(variable)
    qqplot(titanic_train[variable], line='s',ax=ax) # q-q plot
    stat, p = stats.shapiro(titanic_train[variable])
    ax.set_xlabel('Statistics=%.2f, p=%.2E' % (stat, p))
plt.show()
```



Dado que para todas las variable en el test de `Shapiro-Wilk` se obtiene un *p-valor* **inferior** al nivel de significancia **α = 0.05**, entonces se determina que ninguna variable analizada sigue una distribución normal.

4.2.2 Homogeneidad de la Varianza de los Datos

Se realizará el test de homogeneidad de la varianza para los atributos `Age` y `Fare` con realción a si sobrevivió o no el pasajero, para esto se utilizará la mediana como métrica dado que sus distribuciones no son normales.

In [106…
```
statistic,pvalue = stats.levene(titanic_train.loc[sobrevivio_si.index,'Age'],titanic
print('Age : Statistics=%.2f, p-value=%.2f' % (statistic,pvalue))

statistic,pvalue = stats.levene(titanic_train.loc[sobrevivio_si.index,'Fare'],titani
print('Fare : Statistics=%.2f, p-value=%.2f' % (statistic,pvalue))
```

```
Age : Statistics=5.48, p-value=0.02
```

```
Fare : Statistics=45.10, p-value=0.00
```

En función de los test realizados determinamos que los atributos `Age` y `Fare` no tienen homogeneidad de la varianza en sus datos con relación a si el pasajero sobrevivió o no, por cuanto su estadístico *p-valor* es **inferior** al nivel de significancia **α <= 0.05**.

4.3 Pruebas Estadisticas

En función del objetivo del estudio procederemos a realizar pruebas estadísticas para comparar los grupos de datos definidos.

4.3.1 Test de Igualdad de Medianas

Dado que los atributos `Age` y `Fare` no siguen una distribución normal se utliza la prueba H de Kruskal-Wallis que prueba la hipótesis nula de que la mediana de la población de todos los grupos es igual, utilizamos el parámetro `equal_var=False` dado que las varianzas no son iguales en estos atributos para los grupos analizados.

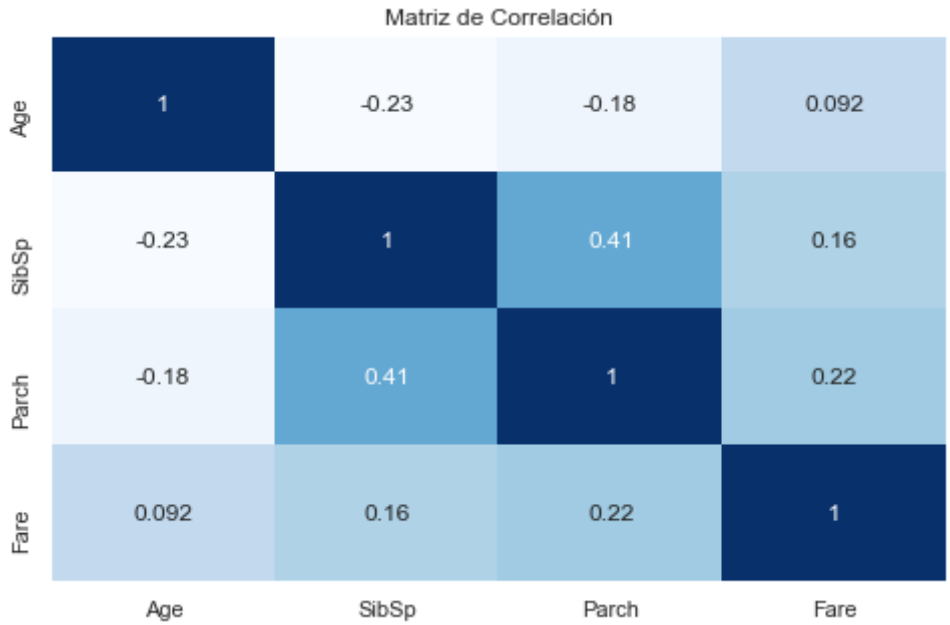
In [106…
```
statistic,pvalue = stats.kruskal(titanic_train.loc[sobrevivio_si.index,'Age'],titani
print('Age : Statistics=%.2f, p-value=%.2f' % (statistic,pvalue))

statistic,pvalue = stats.kruskal(titanic_train.loc[sobrevivio_si.index,'Fare'],titan
print('Fare : Statistics=%.2f, p-value=%.2f' % (statistic,pvalue))
```

```
Age : Statistics=1.36, p-value=0.24
Fare : Statistics=93.28, p-value=0.00
```

En función del test realizado determinamos que el atributo `Age` tiene medianas iguales entre los dos grupos de análisis (sobrevivio_si, sobrevivio_no) por cuanto su estadístico *p-valor* es **superior** al nivel de significancia **α >= 0.05**, no así el atributo `Fare` .

4.3.2 Correlación de Variables

In [106…
```
fig = plt.figure(figsize=(8,5))
sns.heatmap(titanic_train.corr(),cmap='Blues',annot=True,cbar=False)
plt.title('Matriz de Correlación')
plt.show()
```



Como se puede visualizar no existe alguna correlación fuerte entre las variables analizadas.

4.4 Regresión Logística

Procederemos a realizar el modelo predictivo a través de una regresión logística, para esto realizaremos el proceso para codificar los atributos cuanlitativos a datos numéricos a través de `OneHotEncoder` y luego normalizaremos todas las variables a través de `StandardScaler`.

In [106…
```
encoder = OneHotEncoder(drop='first')
codificacion=encoder.fit_transform(titanic_train[['Pclass','Sex','Embarked']]).toarr
titanic_train_encoding = pd.DataFrame(codificacion,columns=np.hstack(['2','3','male'
titanic_train=titanic_train.join(titanic_train_encoding)
titanic_train.drop(['Pclass','Sex','Embarked','Age_range'],axis=1,inplace=True)

codificacion=encoder.fit_transform(titanic_test[['Pclass','Sex','Embarked']]).toarra
titanic_test_encoding = pd.DataFrame(codificacion,columns=np.hstack(['2','3','male',
titanic_test=titanic_test.join(titanic_test_encoding)
titanic_test.drop(['Pclass','Sex','Embarked','Age_range'],axis=1,inplace=True)

survived=titanic_train['Survived']
titanic_train.drop(['Survived'],axis=1,inplace=True)

titanic_train = pd.DataFrame(preprocessing.StandardScaler().fit_transform(titanic_tr
titanic_test = pd.DataFrame(preprocessing.StandardScaler().fit_transform(titanic_tes
```

Para determinar seleccionar los atributos que mayor aportación generen al modelo utilizaremos el proceso de eliminación de atributos recursivo `RFE (feature_selection)`.

In [106…
```
logisticRegression = LogisticRegression()
recursiveFeatureElimination = RFE(logisticRegression)
recursiveFeatureElimination = recursiveFeatureElimination.fit(titanic_train, survive
print(recursiveFeatureElimination.support_)
print(recursiveFeatureElimination.ranking_)
```

```
[ True False False False  True  True  True False False]
[1 2 5 4 1 1 1 6 3]
```

In [106…
```
titanic_train=titanic_train.loc[:,recursiveFeatureElimination.support_].copy()
titanic_test=titanic_test.loc[:,recursiveFeatureElimination.support_].copy()
titanic_train.head()
```

Out[106…

	Age	2	3	male
0	-0.592481	-0.510152	0.902587	0.737695
1	0.638789	-0.510152	-1.107926	-1.355574
2	-0.284663	-0.510152	0.902587	-1.355574
3	0.407926	-0.510152	-1.107926	-1.355574
4	0.407926	-0.510152	0.902587	0.737695

Como resultado de la selección de atributos, se seleccionaron: `Age` , `Pclass` y `Sex` .

Coeficientes y odds

Utilizaremos el modelo Logit para determinar los coeficientes del modelo.

In [107…
```
logit_model=sm.Logit(survived,titanic_train)
resultado=logit_model.fit()
print(resultado.summary2())
print('Odds Ratios')
print(np.exp(resultado.params))
```

```
Optimization terminated successfully.
```

```
         Current function value: 0.483180
         Iterations 6
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.274
Dependent Variable: Survived         AIC:              869.0267
Date:               2021-06-05 18:38 BIC:              888.1961
No. Observations:   891              Log-Likelihood:   -430.51
Df Model:           3                LL-Null:          -593.33
Df Residuals:       887              LLR p-value:      2.8205e-70
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
----------------------------------------------------------------
          Coef.    Std.Err.      z       P>|z|     [0.025    0.975]
----------------------------------------------------------------
Age      -0.4131     0.0929    -4.4477   0.0000   -0.5951   -0.2311
2        -0.4623     0.1050    -4.4035   0.0000   -0.6681   -0.2565
3        -1.1064     0.1175    -9.4131   0.0000   -1.3368   -0.8760
male     -1.2518     0.0908   -13.7811   0.0000   -1.4298   -1.0738
================================================================

Odds Ratios
Age     0.661592
2       0.629827
3       0.330750
male    0.285995
dtype: float64
```

Interpretación de Odds Ratio

En función de los odds ratio del modelo de regresión logística, se puede concluir que:

- **Age**. Por cada año de incremento en la edad, la probabilidad de sobrevivir es 0.66 veces menor.
- **Pclass (2)**. La probabilidad de sobrevivir es 0.62 veces menor para la pasajeros de segunda clase, en relación con los pasajeros de las otras clases.
- **Pclass (3)**. La probabilidad de sobrevivir es 0.33 veces menor para la pasajeros de tercera clase, en relación con los pasajeros de las otras clases.
- **Sex (male)**. La probabilidad de sobrevivir es 0.28 veces menor para la pasajeros de sexo masculino, en relación con los pasajeros de sexo femenino.

In [107…

```
X_train, X_test, y_train, y_test = train_test_split(titanic_train,survived,test_size

logisticRegression = LogisticRegression()
logisticRegression.fit(X_train, y_train)
print('Accuracy of logistic regression classifier on train set: {:.2f} %'.format(log
```

```
Accuracy of logistic regression classifier on train set: 79.61 %
```

Como resultado del entrenamiento del modelo se obtuvo un accuracy en el conjunto de entrenamiento de **79.61%**

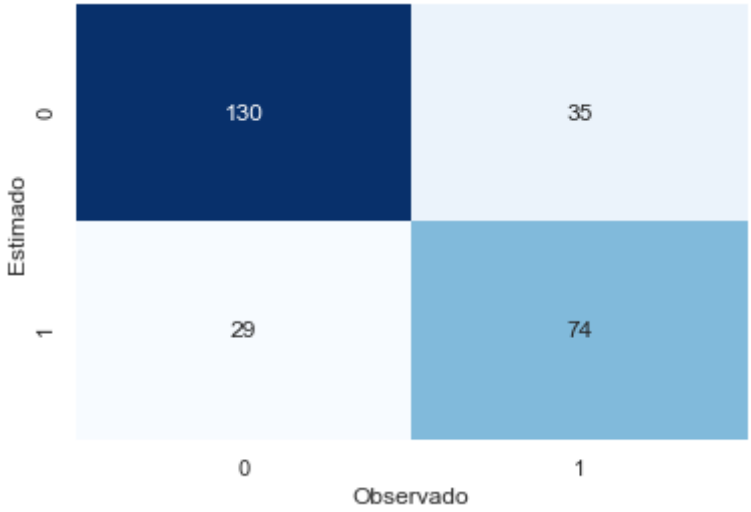
5. Resultados

Con el modelo entrenado determinamos:

- El accuracy en el conjunto de test.
- La matriz de confusión de los resultados del modelo.

In [107…
```
print('Accuracy of logistic regression classifier on test set: {:.2f}%'.format(logis
matriz_confusion = confusion_matrix(y_test, logisticRegression.predict(X_test))
print('Matriz de Confusión:')
sns.heatmap(matriz_confusion,annot=True,fmt="d",cbar=False,cmap="Blues")
plt.xlabel('Observado')
plt.ylabel('Estimado')
plt.show()
```

```
Accuracy of logistic regression classifier on test set: 76.12%
Matriz de Confusión:
```



Como el conjunto de test se obtuvo un accuracy de **76.12%**, finalmente procedemos a calcular si sobrevivieron o no los pasajeros del conjunto `titanic_test` .

In [107…
```
titanic_test_original['Survived_Prediction']=logisticRegression.predict(titanic_test
titanic_test_original
```

Out[107…

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	
…	…	…	…	…	…	…	…	…	…	…	…
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	

418 rows × 12 columns

6. Conclusiones

En función de los resultados, podemos concluir que se obtiene un modelo relativamente bueno para clasificar si un pasajero sobrevivió o no, la precisión global del modelo es del **76.12%**. Esta precisión podría mejorarse utilizando modelos más avanzados bsados en árboles o redes neuronales.

Además hemos determinado que los atributos más importantes a la hora de clasificar a los pasajeros si sobrevivieron o no son: `Age` , `Pclass` y `Fare` .

Finalmente en función de los odds ratio del modelo de regresión logística, se puede concluir que:

- **Age**. Por cada año de incremento en la edad, la probabilidad de sobrevivir es 0.66 veces menor.
- **Pclass (2)**. La probabilidad de sobrevivir es 0.62 veces menor para la pasajeros de segunda clase, en relación con los pasajeros de las otras clases.
- **Pclass (3)**. La probabilidad de sobrevivir es 0.33 veces menor para la pasajeros de tercera clase, en relación con los pasajeros de las otras clases.
- **Sex (male)**. La probabilidad de sobrevivir es 0.28 veces menor para la pasajeros de sexo masculino, en relación con los pasajeros de sexo femenino.

In [107…
```
pd.DataFrame({'CONTRIBUCIONES':['Investigación Previa','Redacción de las Respuestas'
 'FIRMA':['LP','LP','LP']})
```

Out[107…

	CONTRIBUCIONES	FIRMA
0	Investigación Previa	LP
1	Redacción de las Respuestas	LP
2	Desarrollo código	LP

In [ ]: