PRÁCTICA 2 - LIMPIEZA Y VALIDACIÓN DE LOS DATOS

DESCRIPCIÓN DEL DATASET

El conjunto de datos seleccionado para el análisis se ha conseguido en Kaggle. El dataset obtenido contiene registros sobre los pasajeros que abordaron el 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.

Este conjunto de datos de entrenamiento está compuesto por 12 atributos que representan 891 pasajeros que navegaban en el titanic.

Los campos de este conjunto de datos son los siguientes:

- PassengerId
- Survived: indica si el pasajero ha sobrevivido o no.
- Pclass: clase en la que viajaba el pasajero.
- Name: nombre del pasajero.
- Sex: sexo del pasajero.
- Age: edad del pasajero.
- SibSp: número de hermano/conyuge a bordo.
- Parch: número de padres/hijos a bordo.
- Ticket: tipo de ticket del pasajero.
- Fare: tarifa correspondiente al ticker del pasajero.
- Cabin: la cabina donde se hospeda el pasajero.
- Embarked: el puerto donde los pasajeros han embarcado.

Importancia y objetivos de los análisis

Con este conjunto de datos se plantea la problemática de determinar qué variables influyen más a la hora de determinar la probabilidad de supervivencia de un pasajero en el Titanic. Para esto, se crearán modelos de regresión que permitan predecir la probabilidad de supervivencia de un pasajero en función de sus características,así como contrastes de hipótesis que ayuden a identificar propiedades interesantes en las muestras que puedan ser inferidas con respecto a la población.

Estos análisis adquieren una gran relevancia en casi cualquier sector relacionado con la navegación. Un ejemplo de ello sería el de determinar el precio de el seguro de un pasajero en función de sus características. También se pueden generar modelos que ayuden a distribuir a los pasajeros por las distintas secciones del barco en función de sus características con el fín de conseguir salvar el máximo de pasajeros posibles en caso de emergencia.

INTEGRACIÓN Y SELECCIÓN DE LOS DATOS DE INTERÉS A ANALIZAR

Antes de comenzar con la limpieza de los datos, procedemos a realizar la lectura del fichero en formato CSV

en el que se encuentran. El resultado devuelto por la llamada a la función read.csv() será un objeto data.frame:

```
import pandas as pd
df = pd.read_csv('./train.csv', skipinitialspace=True)
df.head()
```

	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
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
# Tipo de dato asignado a cada campo
df.dtypes
```

```
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
```

```
df.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

Además, observamos cómo los tipos de datos asignados automáticamente por Python a las variablesse corresponden con el dominio de estas.

A la hora de realizar la selección de atributos, lo primero a tener en cuenta es que en principio se pueden eliminar las características que no son propiamente del pasajero, como valores asignados. Por lo tanto, podemos prescindir del primer campo (`PassengerId`) dado que no son atributos propios de los pasajeros y, por tanto no resulta relevante a la hora de resolver nuestro problema, y el nombre, ya que este tampoco es relevante.

Aunque la variable ('Cabin') podría haberse tenido en cuenta, prescindiremos de ella por dos motivos:

- No sabemos en que zona del barco se encontraba el pasajero en el momento de la tragedia.
- La variable contiene un alto porcentaje de valores perdidos.

```
# Eliminar la primera columns
df.drop('PassengerId', axis=1, inplace=True)
df.head()
```

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

Una vez eliminados estos atributos, vamos a realizar un primer análisis visual para determinar que variables pueden ser potencialmente más determinantes a la hora de generar los modelos. Observaremos distintos diagramas de barras para determinar que valores de los distintos atributos están relacionados con mayores probabilidades de sobrevivir.

Chart age Chart age Chart age Chart age Chart age Chart age Chart age

Observando los diagramas de barras creados llegamos a las siguientes conclusiones:

- Los pasajeros tienen una probabilidad de supervivencia proporcional a la clase en la que viajan, siendo los de primera clase los que tienen más probabilidad de sobrevivir y los de tercera clase los que menos.
- Las mujeres tienen mayor probabilidad de sobrevivir que los hombres.
- Los niños, y en especial los de teprana edad (entre 0 y 5 años) tienen mayor probabilidad de sobrevivir.
- Las personas con familiares a bordo tienen mayor probabilidad de sobrevivir que los que viajan sin familiares.
- El tratamiento que reciben en la variable ('Name') tiene gran influencia a la hora de determinar la probabilidad de supervivencia. Creamos una nueva variable ('Title') donde almacenaremos estos tratamientos.

LIMPIEZA DE LOS DATOS

¿Los datos contienen ceros o elementos vacíos?

```
df.isnull().sum()
```

```
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
```

¿Cómo gestionarías cada uno de estos casos?

Existen diferentes maneras de gestionar los elementos vacíos:

- Reemplazar el elemento vacío por un valor (constante, media, mediana...).
- Eliminar las filas con elementos vacíos.

En nuestro caso, reemplazaremos el elemento vacío por un valor:

- Para el atributo `Age`, reemplazaremos los elementos vacíos por la utilizando una imputación por el algoritmo KNN.
- Para el atributo `Embarked`, reemplazaremos los valores vacíos por el valor "S", ya que es el que tiene mayor porcentaje de aparición y hay pocos valores nulos (2).
- Prescindimos de la variable ('Cabin') ya que no será utilizada en los análisis.

```
df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Cabin"].fillna(" ", inplace=True)
df["Embarked"].fillna(" ", inplace=True)
```

```
df.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

Identificación y tratamiento de valores extremos

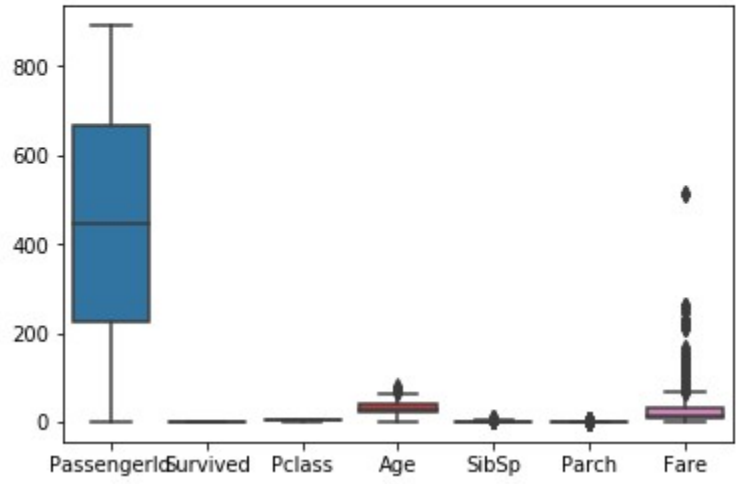
Los valores extremos o outliers son aquellos que parecen no ser coherentes si realizamos una comparación con el resto de los datos. Para identificarlos, podemos hacer uso de dos vías:

- Representar un diagrama de caja por cada variable y ver qué valores distan mucho del rango intercuartílico (la caja).
- Utilizar la función boxplots.stats() de R, la cual se emplea a continuación.

Así, se mostrarán sólo los valores atípicos para aquellas variables que los contienen:

```
import seaborn as sns
sns.boxplot(data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aef63e54a8>
```



Como podemos observar en el gráfico anterior, se detectan valores atípicos en los atributos `Age`, `SibSp`, `Parch` y `Fare`.

Trataremos estos valores atípicos en función de los valores intercuartílicos 1 y 3, y cogeríamos el valor máximo y mínimo dentro de ese rango. Si el valor supera el valor máximo, este valor se cambiara por este, de igual manera con el mínimo. Haremos las siguientes operaciones para evitar que estos valores extremos tengan un peso mayor al resto de valores, pudiendonos conducir a la generación de modelos sesgados, donde estos valores tendrán más influencia.

Función para eliminar valores atítipicos:

```python
def fixoutliers(x):
    xColumnNames=x.columns
    for j in xColumnNames:
        try:
            xy=x[j]
            updated=[]
            outliers = []
            Q1,Q3=np.percentile(xy,[25,75])
            IQR=Q3-Q1
            minimum=Q1-1.5*IQR
            maximum=Q3+1.5*IQR
            for i in xy:
                outlier = False
                if(i>maximum):
                    outlier = True
                    i=maximum
                elif(i<minimum):
                    outlier = True
                    i=minimum
                if(outlier):
                    outliers.append(i)
                updated.append(i)
            x[j]=updated
            print("Outliers de la columna", j, ":", outliers)
        except:
            continue
```

```
fixoutliers(df)
```

```
Outliers de la columna PassengerId : []
Outliers de la columna Survived : []
Outliers de la columna Pclass : []
Outliers de la columna Age : [61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0,
61.0, 61.0, 61.0, 61.0]
Outliers de la columna SibSp : [2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5,
2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5,
2.5, 2.5, 2.5, 2.5]
Outliers de la columna Parch : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0]
Outliers de la columna Fare : [65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344, 65.6344,
65.6344, 65.6344]
```

ANÁLISIS DE LOS DATOS

Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

A continuación, se seleccionan los grupos dentro de nuestro conjunto de datos que pueden resultar interesantes para analizar y/o comparar.

De acuerdo con las conclusiones obtenidas en el apartado de integración y selección de los datos de interés a analizar, generaremos modelos utilizando las siguientes variables: "Age", "Sex", "Title" (creada a partir de los tratamientos contenidos en "Name"), "Parch", "SibSp" y "Pclass". AUnque parece tener menos influencia que el resto de variables mencionadas, también incluiremos la variable "Embarked".

Comprobación de la normalidad y homogeneidad de la varianza

Normalidad

Para realizar la comprobación de que los valores que toman nuestras variables cuantitativas provienen de una población distribuida normalmente, utilizaremos la prueba de normalidad de Shapiro-Wilk. Así, se comprueba que para que cada prueba se obtiene un p-valor superior al nivel de significación prefijado alfa = 0, 05. Si esto se cumple, entonces se considera que variable en cuestión sigue una distribución normal.

Para el caso que nos atribuye comprobamos que no podemos asumir normalidad en los análisis y modelos a generar.

```
from scipy.stats import shapiro
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
newdf = df.select_dtypes(include=numerics)
statistic, pvalue = shapiro(newdf)
print("Shapiro Statistic " + str(statistic) + " and p-value " + str(pvalue))
if pvalue > 0.05:
    print("Normal")
else:
    print("Not normal")
```

```
Shapiro Statistic 0.6571232080459595 and p-value 0.0
Not normal
```

Podemos observar que los p-values son menores que 0,05, lo que indica que no hay una distribución normal de las variables analizadas.

Homogeneidad

Continuamos estudiando la homogeneidad de varianzas mediante la aplicación de un test de Levene y de un test de Fligner. En ambos caso, estudiaremos esta homogeneidad en cuanto a los grupos conformados por los pasajeros supervivientes y los no supervivientes. En el siguiente test, la hipótesis nula consiste en que ambas varianzas son iguales.

```
from scipy.stats import levene
maledf = df.male.select_dtypes(include=numerics)
femaledf = df.female.select_dtypes(include=numerics)
statistic, pvalue = levene(maledf["Survived"], femaledf["Survived"])
print("Levene Statistic " + str(statistic) + " and p-value " + str(pvalue))
if pvalue > 0.05:
    print("Homogeneidad")
else:
    print("No homogeneidad")
```

```
Levene Statistic 5.804100632617495 and p-value 0.016191248809611765
No homogeneidad
```

```
from scipy.stats import fligner
statistic, pvalue = fligner(df.survived["Survived"], df.nosurvived["Survived"], center='mean')
print("Fligner Statistic " + str(statistic) + " and p-value " + str(pvalue))
if pvalue > 0.05:
    print("Homogeneidad")
else:
    print("No homogeneidad")
```

```
Fligner Statistic 341.61616161616166 and p-value 2.837277250766588e-76
No homogeneidad
```

Podemos observar que el valor p es menor que 0,05, lo que indica que hay una diferencia significativa en las variaciones entre los grupos.

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

El primer paso que realizamos es comprobar la correlación entre la variable Survived y el resto de variables. Comprobamos como la variable más correlacionada es "Sex", seguida de "Pclass".

A continuación comprobamos que existen diferencias significativas en la probabilidad de sobrevivir en función del sexo. Aplicamos un t-test estableciendo como hipótesis nula (H0) que no existen diferencias significativas entre las probabilidades de supervivencia en función del sexo y como hipótesis alternativa (H1) que sí existen diferencias significativas y por lo tanto la media de la variable supervivencia para los dos sexos es distinta.

Observando los resultados obtenidos vemos como se obtiene un p-value muy inferior a 0,05, lo que nos indica que no podemos aceptar la hipótesis nula y debemos quedarnos con la alternativa. Aplicamos el test de Wilcoxon para asegurar los resultados al tratarse de una población de la cual no podemos asumir la condición de normalidad. Una vez más obtenemos un p-value muy por debajo de 0,05. Cabe comentar que para realizar los tests escogemos muestras de datos por encima de los 30 registros de tamaño para poder trabajar asumiendo que trabajamos con muestras de gran tamaño.

Una vez realizados los tests estadísticos anteriores, procedemos a generar un modelo de regresión logística. Antes del entrenamiento necesitamos aplicar una serie de preprocesados a los datos para dejarlos listos para el análisis.

El primer paso es aplicar una normalización a los valores de las variables "Age" y "Fare". Aplicamos un StandardScaler() con el que dejamos todos los valores dentro del rango [-1,1] y las variables pasan a tener media 0 y desviación 1.

Acto seguido, creamos nuevas variables binarias a partir de los valores de las variables "Title", "Pclass", "Embarked", "SibSp", "Parch" y "Sex". Utilizamos el comando get_dummies de la librería Pandas.

Una vez tenemos los datos listos, dividimos el conjunto en train y test, dejando un 20% de los datos para testear.

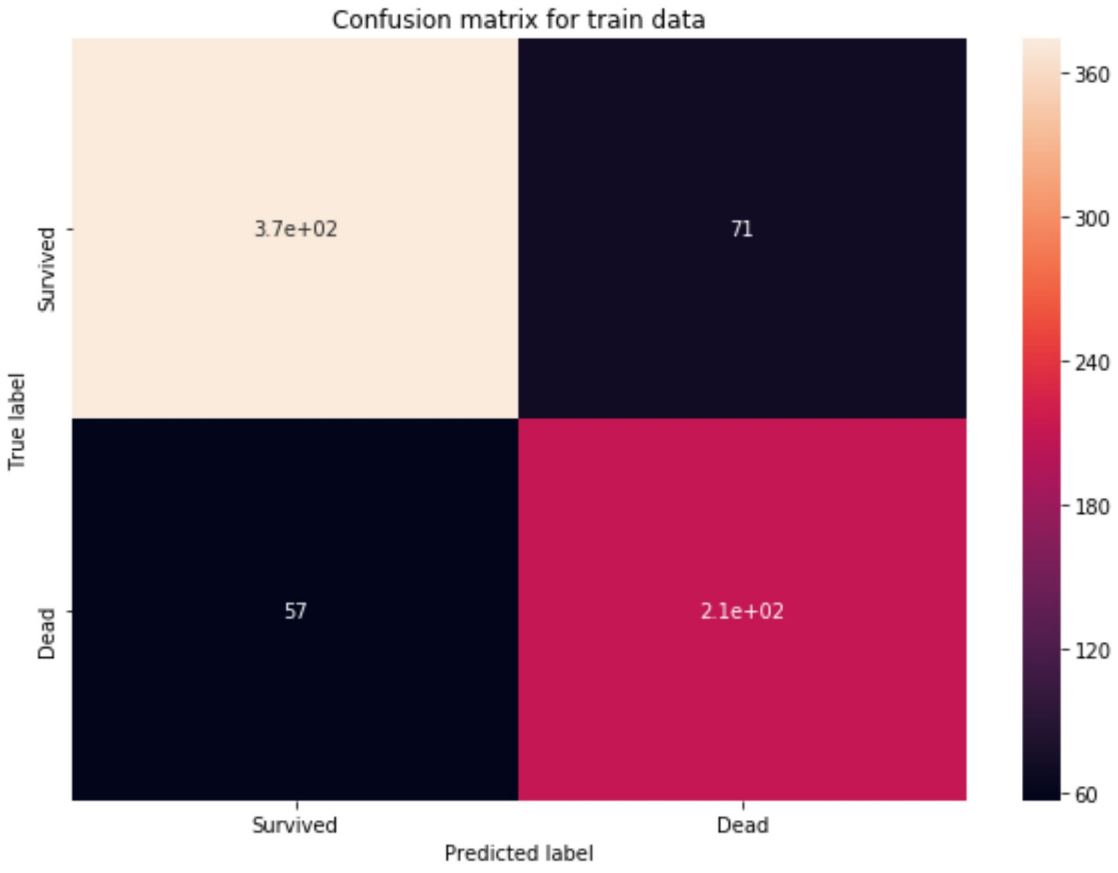
Finalmente generamos el modelo de regresión logística con los datos de train y comprobamos sus prestaciones tanto para el conjunto de train como para el de test. Vemos que obtenemos una precisión de predicción de la variable "Survived" superior al 80% y los resultados obtenidos para train y test no varían en exceso, por lo que podemos considerar que los resultados son buenos y el modelo ofrece buenas prestaciones.

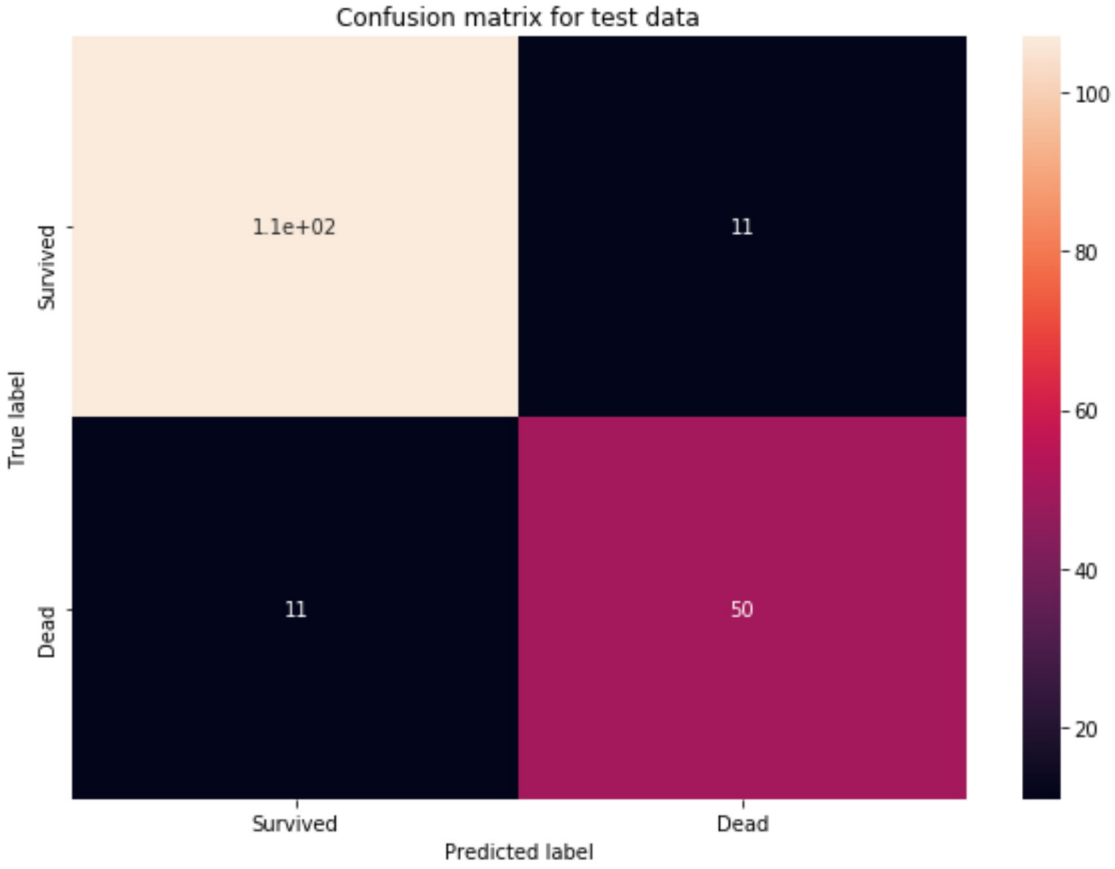
Encontramos el código en el script MAIN.py

Representación de los resultados a partir de tablas y gráficas

A continuación adjuntamos distintos gráficos y tablas que reflejan los resultados obtenidos.

En primer lugar ploteamos las matrices de confusión obtenidas.

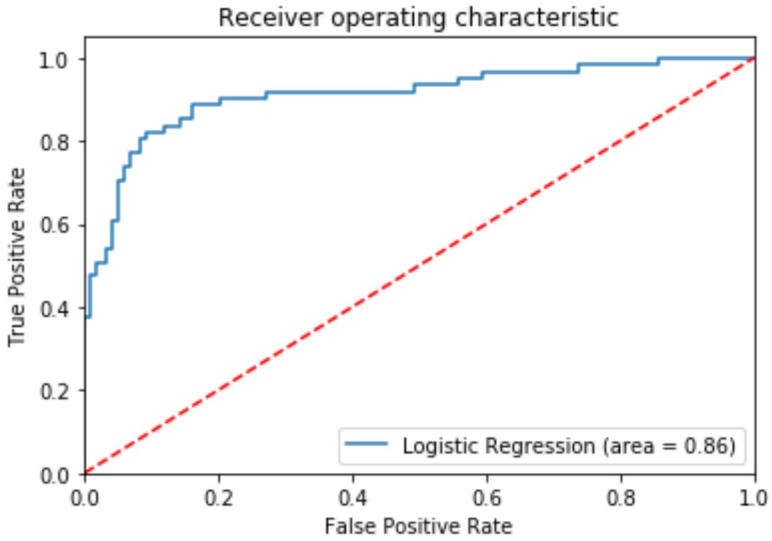




Debido al imbalanceo de clases, no es suficiente con mirar los valores de accuracy obtenidos. Debido a esto comprobamos como obtenemos buenas prestaciones en términos de f-score, una métrica donde se tiene en cuenta este imbalanceo.

	precision	recall	f1-score	support
0	0.91	0.91	0.91	118
1	0.82	0.82	0.82	61
avg / total	0.88	0.88	0.88	179

Finalmente, observamos como obtenemos una curva ROC donde los resultados son buenos, ya que tenemos una área por debajo de la curva (AUC) DE 0.86, un valor más que aceptable.



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

A partir de los resultados obtenidos, llegamos a la conclusión de que las variables utilizadas son útiles para determinar las probablidades de supervivencia de un pasajero, por lo tanto conseguimos dar respuesta al problema planteado.

Por lo tanto, el modelo de regresión generado podría ser útil para conseguir los objetivos propuestos en el apartado de objetivos.

CONTRIBUCIONES AL TRABAJO

Contribuciones	**Firma**
Investigación previa	PFA, MAC
Redacción de las respuestas	PFA, MAC
Desarrollo código	PFA, MAC