Análisis de Datos y Aprendizaje Máquina con Tensorflow 2.0: Clasificación

2019/09/30

1 Regresión Logística

Objetivo: Se pondran en práctica las herramientas para pre-procesamiento de datos. Obtener mínimo un 82% de precisión y matriz de confusión.

- Documentación: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.ht

Tiempo máximo: 2 horas

- Nota: Pre-procesar los datos para obtener un error menor

La regresión logística predice la probabilidad de que un elemento pertenezca a la clase 0 ó 1 aplicando la función sigmoide a una función lineal

$$y(x) = g(w^T x + b)$$

donde $g(z)$ es la función sigmoide

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

1.1 Importar bibliotecas y dataset

```
In [1]: import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
        import pandas as pd
        import numpy as np

        data = pd.read_csv('titanic_train.csv') # Dataset de entrenamiento
        data.head()
```

```
Out[1]:    PassengerId  Survived  Pclass  \
        0            1         0       3
        1            2         1       1
        2            3         1       3
        3            4         1       1
        4            5         0       3
```

```
                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

1.2 Análisis exploratorio

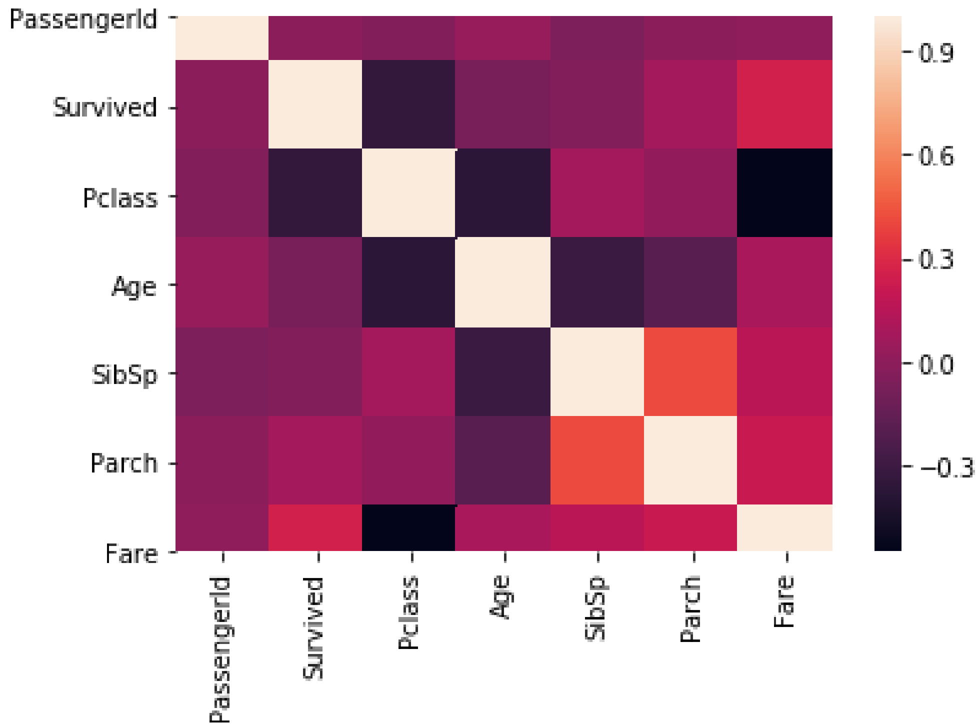
In [2]:
```
des = data.describe()
data.describe()
```

Out[2]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

In [3]:
```
corr = data.corr()
sns.heatmap(corr)
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4afd2079d0>
```



1.3 Solución

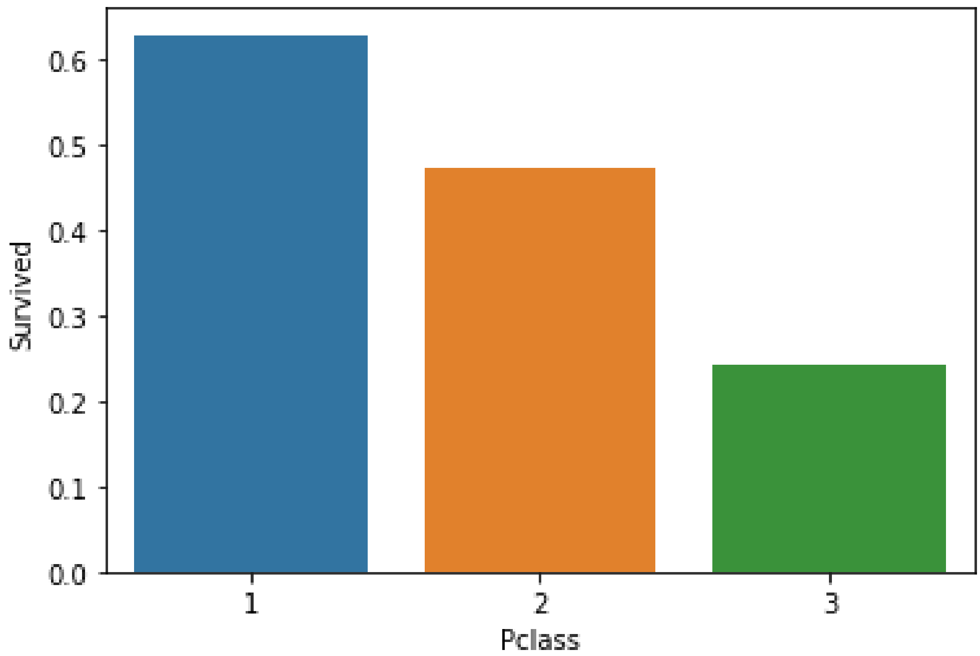
- Eliminar características no necesarias
- Convertir a variables numéricas

- Visualizando variables por clase y sexo

```
In [4]: c = data[['Pclass','Survived']].groupby(['Pclass'], as_index=False).mean()
```

```
In [5]: sns.barplot('Pclass','Survived', data=c)
```

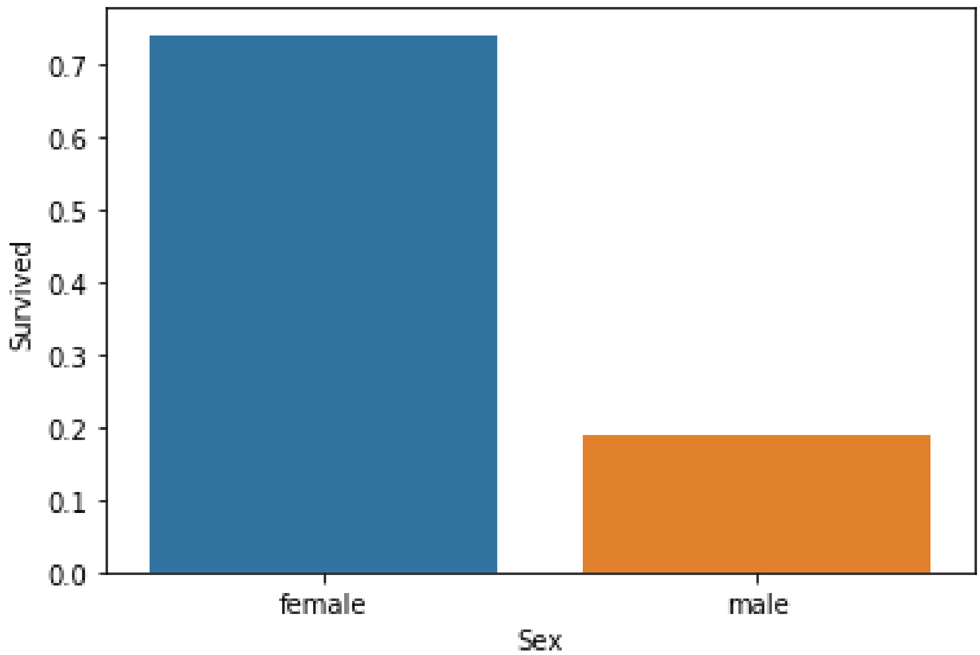
```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4afce774d0>
```




```
In [6]: d = data[['Sex','Survived']].groupby(['Sex'], as_index=False).mean()

In [7]: sns.barplot('Sex','Survived', data=d)

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4afcde77d0>
```



1.4 Faltan valores en edad

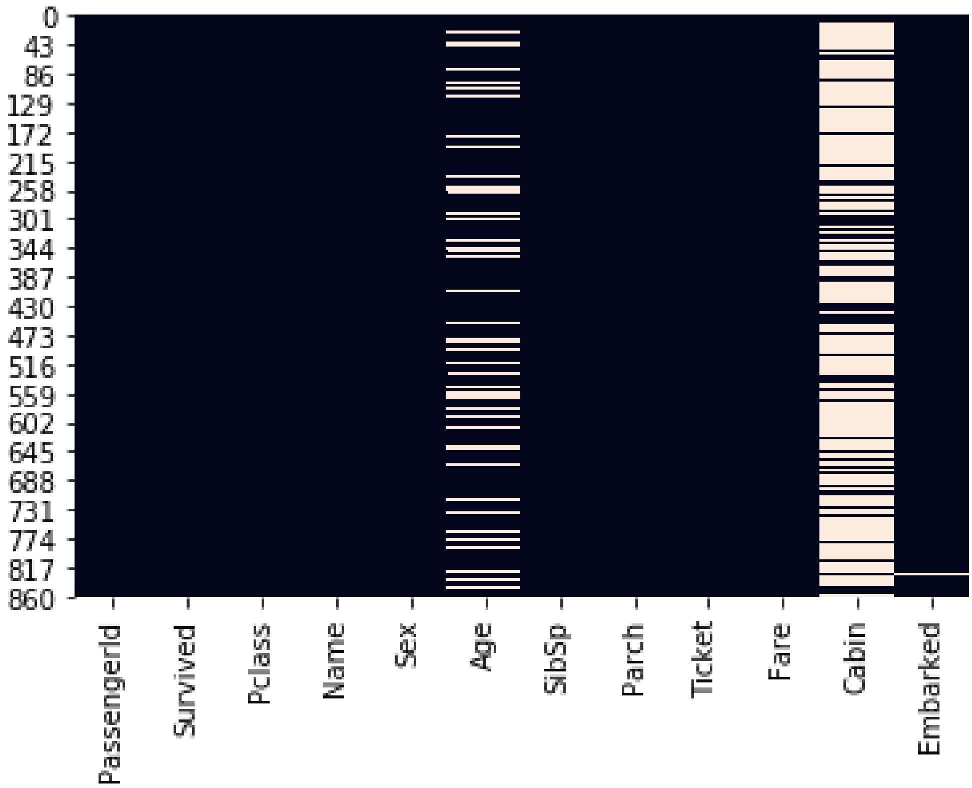
- Conocer valores nulos con 'info()'
- El método de 'heatmap' recibe 'data.isnull()' para visualizar los valores nulos

```
In [8]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
In [9]: sns.heatmap(data.isnull(), cbar=False)

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4afcdd3910>
```




```
In [10]: data[:30]
```

Out[10]:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	
5	6	0	3	
6	7	0	1	
7	8	0	3	
8	9	1	3	
9	10	1	2	
10	11	1	3	
11	12	1	1	
12	13	0	3	
13	14	0	3	

```
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3
20           21         0       2
21           22         1       2
22           23         1       3
23           24         1       1
24           25         0       3
25           26         1       3
26           27         0       3
27           28         0       1
28           29         1       3
29           30         0       3
```

```
                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
5                                    Moran, Mr. James    male   NaN      0
6                             McCarthy, Mr. Timothy J    male  54.0      0
7                      Palsson, Master. Gosta Leonard    male   2.0      3
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                 Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1
10                    Sandstrom, Miss. Marguerite Rut  female   4.0      1
11                           Bonnell, Miss. Elizabeth  female  58.0      0
12                     Saundercock, Mr. William Henry    male  20.0      0
13                        Andersson, Mr. Anders Johan    male  39.0      1
14               Vestrom, Miss. Hulda Amanda Adolfina  female  14.0      0
15                   Hewlett, Mrs. (Mary D Kingcome)   female  55.0      0
16                               Rice, Master. Eugene    male   2.0      4
17                       Williams, Mr. Charles Eugene    male   NaN      0
18  Vander Planke, Mrs. Julius (Emelia Maria Vande...  female  31.0      1
19                            Masselmani, Mrs. Fatima  female   NaN      0
20                               Fynney, Mr. Joseph J    male  35.0      0
21                              Beesley, Mr. Lawrence    male  34.0      0
22                        McGowan, Miss. Anna "Annie"  female  15.0      0
23                       Sloper, Mr. William Thompson    male  28.0      0
24                      Palsson, Miss. Torborg Danira  female   8.0      3
25  Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...  female  38.0      1
26                            Emir, Mr. Farred Chehab    male   NaN      0
27                     Fortune, Mr. Charles Alexander    male  19.0      3
28                      O'Dwyer, Miss. Ellen "Nellie"  female   NaN      0
29                                Todoroff, Mr. Lalio    male   NaN      0
```

```
    Parch              Ticket      Fare        Cabin Embarked
0       0           A/5 21171    7.2500          NaN        S
1       0            PC 17599   71.2833          C85        C
2       0  STON/O2. 3101282    7.9250          NaN        S
3       0              113803   53.1000         C123        S
4       0              373450    8.0500          NaN        S
5       0              330877    8.4583          NaN        Q
6       0               17463   51.8625          E46        S
7       1              349909   21.0750          NaN        S
8       2              347742   11.1333          NaN        S
9       0              237736   30.0708          NaN        C
10      1             PP 9549   16.7000           G6        S
11      0              113783   26.5500         C103        S
12      0           A/5. 2151    8.0500          NaN        S
13      5              347082   31.2750          NaN        S
14      0              350406    7.8542          NaN        S
15      0              248706   16.0000          NaN        S
16      1              382652   29.1250          NaN        Q
17      0              244373   13.0000          NaN        S
18      0              345763   18.0000          NaN        S
19      0                2649    7.2250          NaN        C
20      0              239865   26.0000          NaN        S
21      0              248698   13.0000          D56        S
22      0              330923    8.0292          NaN        Q
23      0              113788   35.5000           A6        S
24      1              349909   21.0750          NaN        S
25      5              347077   31.3875          NaN        S
26      0                2631    7.2250          NaN        C
27      2               19950  263.0000  C23 C25 C27        S
28      0              330959    7.8792          NaN        Q
29      0              349216    7.8958          NaN        S
```

- La variable de edad está incompleta

1.4.1 Llenar valores perdidos

- Se asigna la media de la edad o se completan los valores con SimpleImputer

In [11]:
```
data.head(7)
```

Out[11]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1

                                                Name     Sex   Age  SibSp  \
```

```
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
5                                   Moran, Mr. James    male   NaN      0
6                            McCarthy, Mr. Timothy J    male  54.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
```

1.5 Asigna los valores con 'fillna' y 'median'

In [12]:
```
#data['Age'].fillna(data['Age'].median(),inplace=True)
```

In [13]:
```
data.tail(7)
```

Out[13]:
```
     PassengerId  Survived  Pclass                                      Name  \
884          885         0       3                     Sutehall, Mr. Henry Jr
885          886         0       3      Rice, Mrs. William (Margaret Norton)
886          887         0       2                      Montvila, Rev. Juozas
887          888         1       1               Graham, Miss. Margaret Edith
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                      Behr, Mr. Karl Howell
890          891         0       3                        Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch            Ticket    Fare Cabin Embarked
884    male  25.0      0      0  SOTON/OQ 392076   7.050   NaN        S
885  female  39.0      0      5            382652  29.125   NaN        Q
886    male  27.0      0      0            211536  13.000   NaN        S
887  female  19.0      0      0            112053  30.000   B42        S
888  female   NaN      1      2        W./C. 6607  23.450   NaN        S
889    male  26.0      0      0            111369  30.000  C148        C
890    male  32.0      0      0            370376   7.750   NaN        Q
```

1.5.1 Eliminar la características que no aportan información

In [14]:
```
data.drop(['PassengerId','Name','Ticket', 'Cabin'],axis=1,inplace=True)
data.head()
```

Out[14]:
```
   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
```

1.5.2 Convierte variables tipo caracter a valores numéricos

- Usando 'replace' para convertir variables

```
In [15]: data = data.replace({'Sex':{'male':0, 'female':1}})
```

```
In [16]: data = data.replace({'Embarked': {'S':0, 'C':1, 'Q':2}})
         data
```

```
Out[16]:      Survived  Pclass  Sex   Age  SibSp  Parch     Fare  Embarked
         0           0       3    0  22.0      1      0   7.2500       0.0
         1           1       1    1  38.0      1      0  71.2833       1.0
         2           1       3    1  26.0      0      0   7.9250       0.0
         3           1       1    1  35.0      1      0  53.1000       0.0
         4           0       3    0  35.0      0      0   8.0500       0.0
         ..        ...     ...  ...   ...    ...    ...      ...       ...
         886         0       2    0  27.0      0      0  13.0000       0.0
         887         1       1    1  19.0      0      0  30.0000       0.0
         888         0       3    1   NaN      1      2  23.4500       0.0
         889         1       1    0  26.0      0      0  30.0000       1.0
         890         0       3    0  32.0      0      0   7.7500       2.0

         [891 rows x 8 columns]
```

1.6 SimpleImputer para valores faltantes de edad

```
In [17]: from sklearn.impute import SimpleImputer

         imputer = SimpleImputer()
         data_complete = imputer.fit_transform(data.drop('Survived',axis=1))
```

```
In [18]: data
```

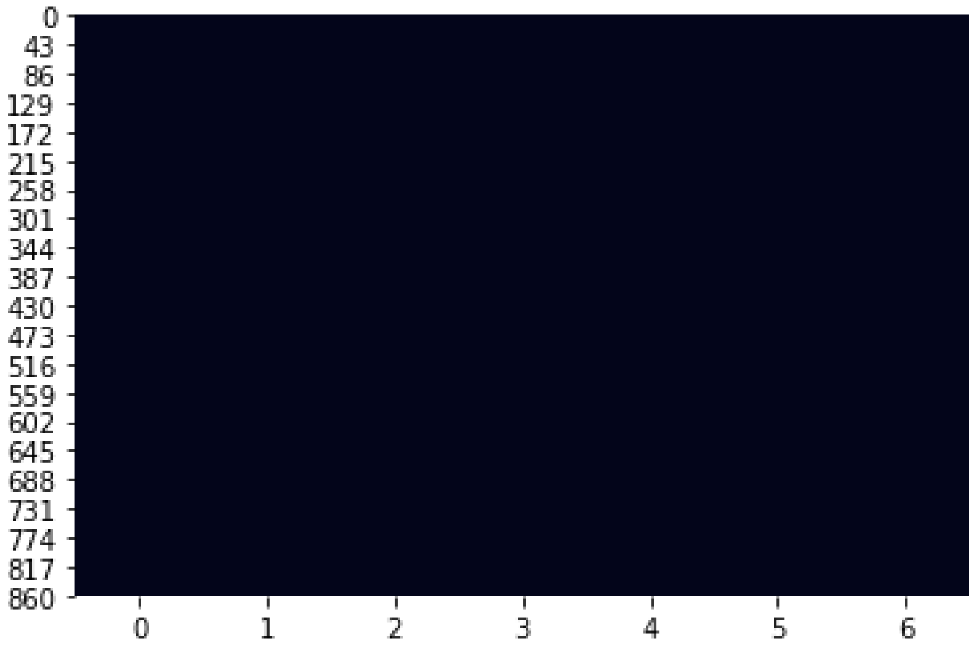
```
Out[18]:      Survived  Pclass  Sex   Age  SibSp  Parch     Fare  Embarked
         0           0       3    0  22.0      1      0   7.2500       0.0
         1           1       1    1  38.0      1      0  71.2833       1.0
         2           1       3    1  26.0      0      0   7.9250       0.0
         3           1       1    1  35.0      1      0  53.1000       0.0
         4           0       3    0  35.0      0      0   8.0500       0.0
         ..        ...     ...  ...   ...    ...    ...      ...       ...
         886         0       2    0  27.0      0      0  13.0000       0.0
         887         1       1    1  19.0      0      0  30.0000       0.0
         888         0       3    1   NaN      1      2  23.4500       0.0
         889         1       1    0  26.0      0      0  30.0000       1.0
         890         0       3    0  32.0      0      0   7.7500       2.0

         [891 rows x 8 columns]
```

1.7 Verificar datos nulos con 'heatmap'

In [19]:
```
dc = pd.DataFrame(data_complete)
sns.heatmap(dc.isnull(), cbar=False)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4afcbb4890>



1.8 Entrenamiento y predicción con Regresión Logística

- Importar de sklearn el modelo

In [20]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_complete,
                                                    data['Survived'], test_size=0.3,
                                                    random_state=11)
import warnings
warnings.filterwarnings('ignore')
```

In [26]:
```
X_train.shape, X_test.shape
```

Out[26]: ((623, 7), (268, 7))

In [21]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
Out[21]: 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='warn', tol=0.0001, verbose=0,
                             warm_start=False)
```

```
In [27]: logmodel.score(X_test, y_test)
```

```
Out[27]: 0.8395522388059702
```

```
In [22]: predictions = logmodel.predict(X_test)
         from sklearn.metrics import classification_report
         print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       176
           1       0.79      0.73      0.76        92

    accuracy                           0.84       268
   macro avg       0.83      0.81      0.82       268
weighted avg       0.84      0.84      0.84       268
```

```
In [23]: from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test, predictions)
```

```
Out[23]: array([[158,  18],
                [ 25,  67]])
```

- True positive: 158 (Predicción positiva que es positiva)
- True negative: 67 (Predicción negativa que es negativa)
- False negative: 18 (Predicción negativa que es positiva)
- False positive: 25 (Predicción positiva que es negativa)

- El resultado depende de la partición del conjunto de entrenamiento y prueba. Usar 'random_state' para repetir experimento con mismo conjunto de datos