

Reto Titanic

TC3006C.102

A00827826 Edgar Castillo

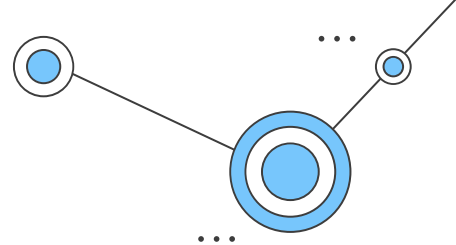
A01422876 Héctor San Román

A01197595 Roel de la Rosa

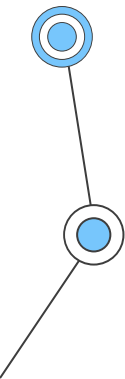
A00829598 Juan Pablo Yáñez

A00827757 Rodrigo Montelongo

Reto



Utilice el aprendizaje automático para crear un modelo que prediga qué pasajeros sobrevivieron al naufragio del Titanic.



Pasos principales

01

Análisis de la base de datos

02

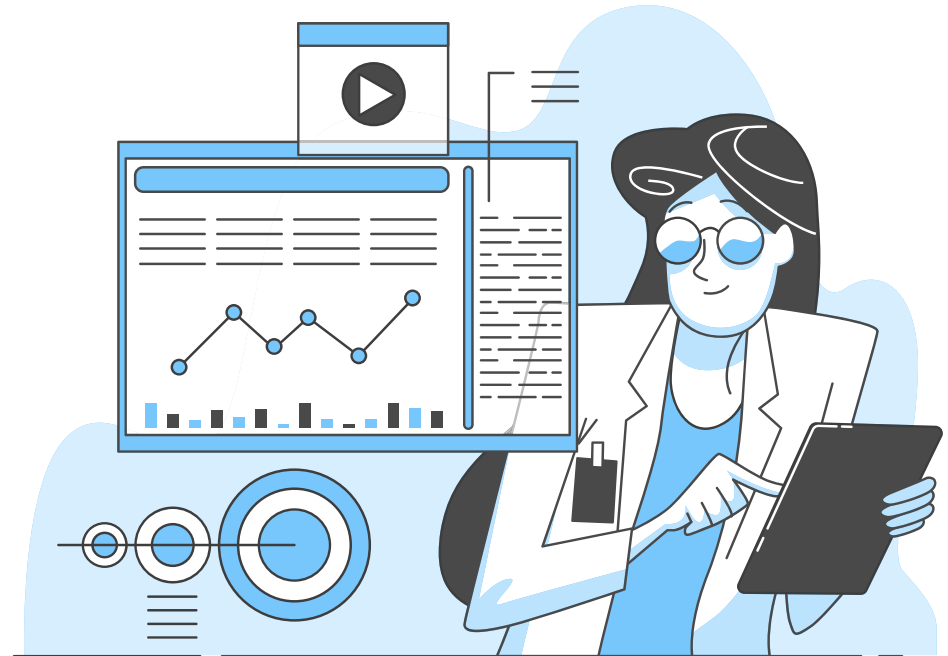
Depuración de datos

03

Selección de modelos

04

Submissions a Kaggle





01

Análisis de la base de datos



Se tienen un total de 12 variables. Si se clasifican en cuantitativas y cualitativas se obtiene la siguiente lista:

Cuantitativas:

- PassengerId
- Age
- Fare

Cualitativas:

- Survived
- Pclass
- Name
- Sex
- SibSp
- Parch
- Ticket
- Cabin
- Embarked



Valores faltantes

```
#Valores faltantes  
titanic.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
```

Valores duplicados

```
[ ] #Valores duplicados  
titanic.duplicated().sum()
```

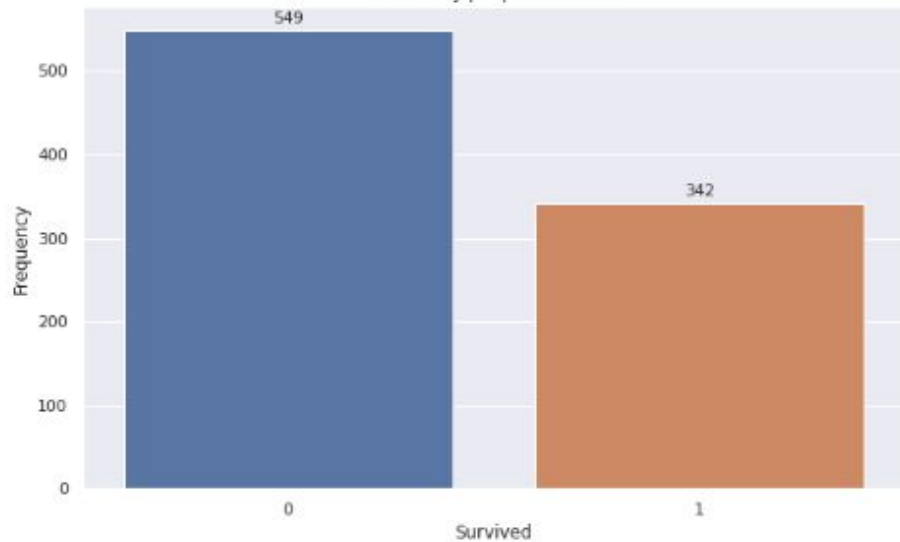
```
0
```



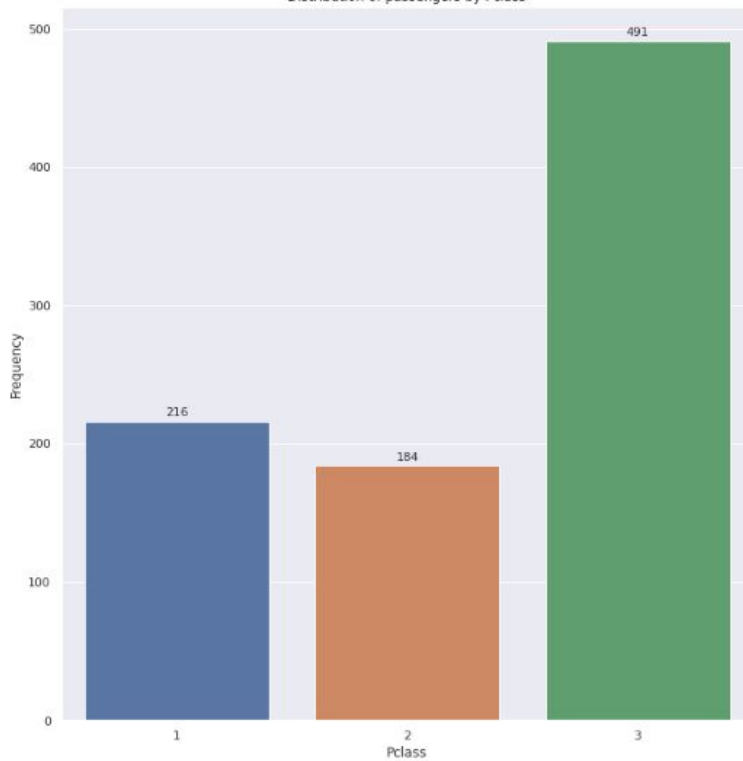


Medidas importantes

How many people survived?

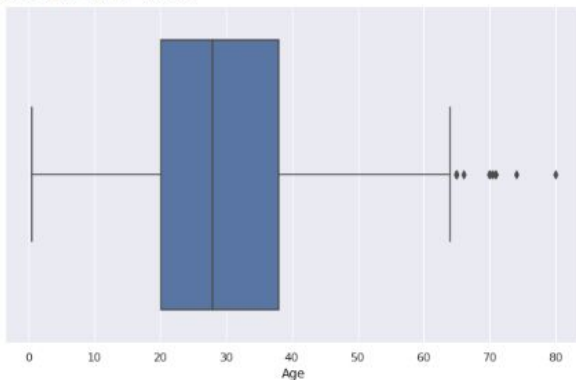


Distribution of passengers by Pclass



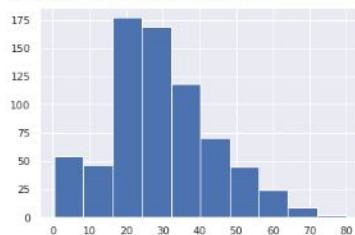
```
plt.figure(figsize=(10,6))
bp = sns.boxplot(x=titanic["Age"])
titanic["Age"].describe()
```

```
count    714.000000
mean     29.699118
std      14.526497
min       0.420000
25%      20.125000
50%      28.000000
75%      38.000000
max      80.000000
Name: Age, dtype: float64
```



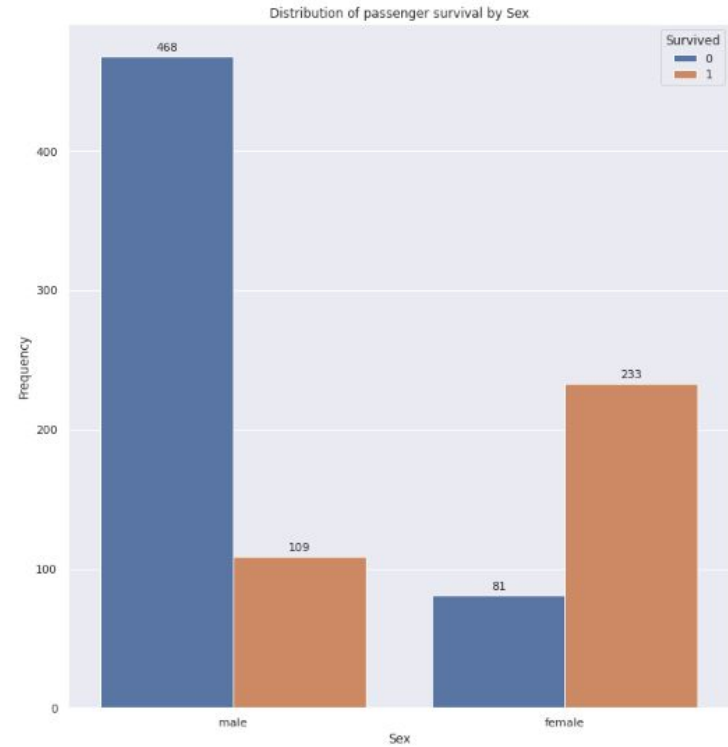
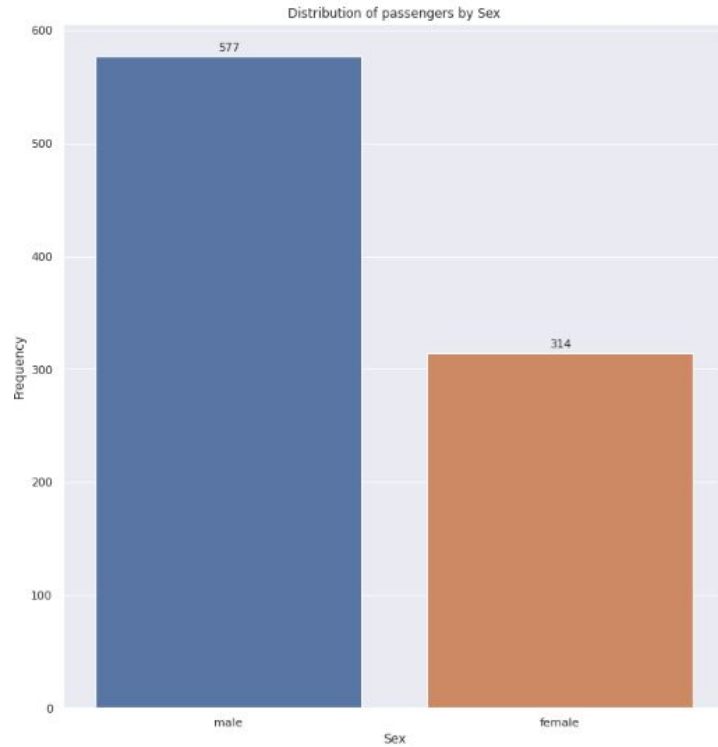
```
plt.hist(titanic["Age"])
```

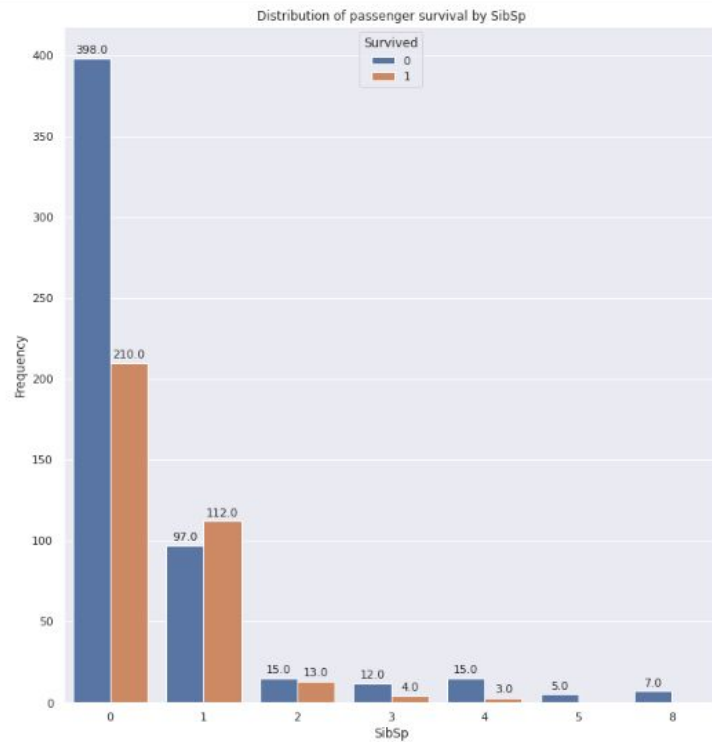
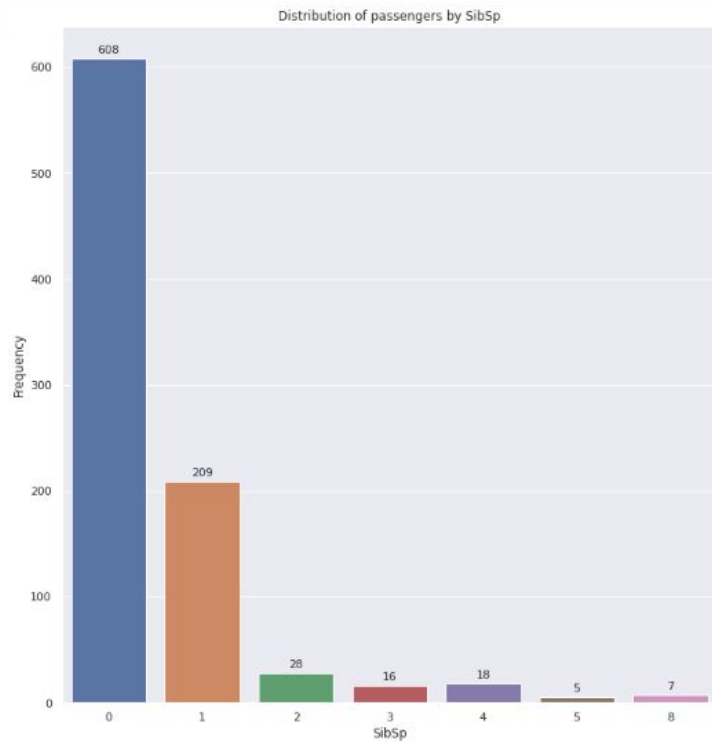
```
(array([ 54.,  46., 177., 169., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <BarContainer object of 10 artists>)
```



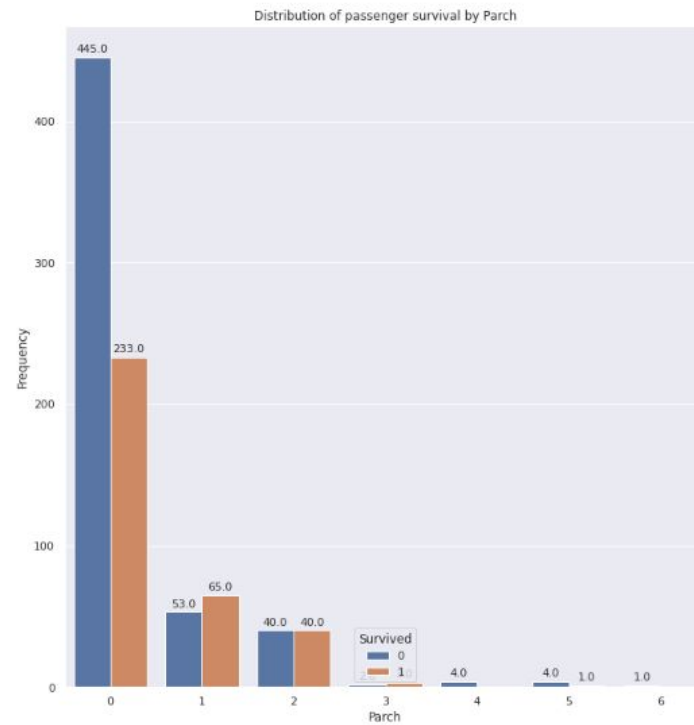
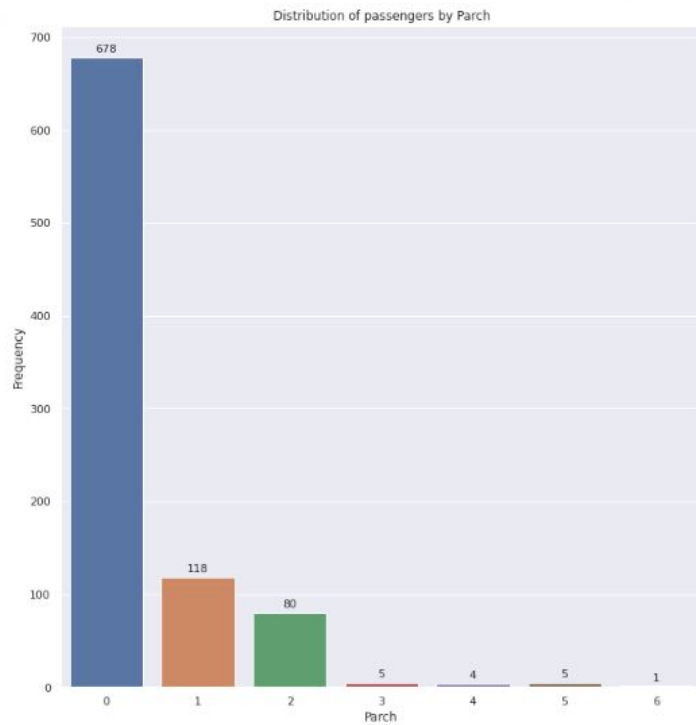
Una gran cantidad de los pasajeros se encuentran debajo de los 40 años, se tienen algunos valores atípicos que deberán ser interpretados y manejados.

Medidas importantes



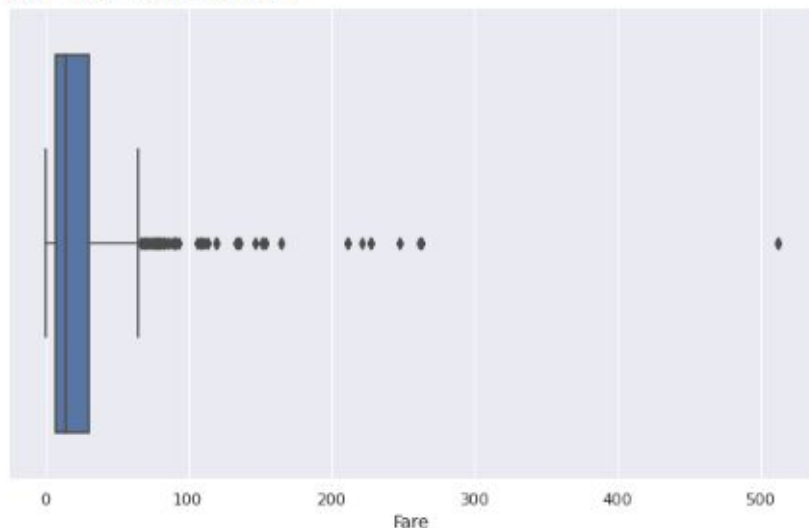


Se puede ver que en su mayoría, no había personas con hermanos o esposas en el barco.

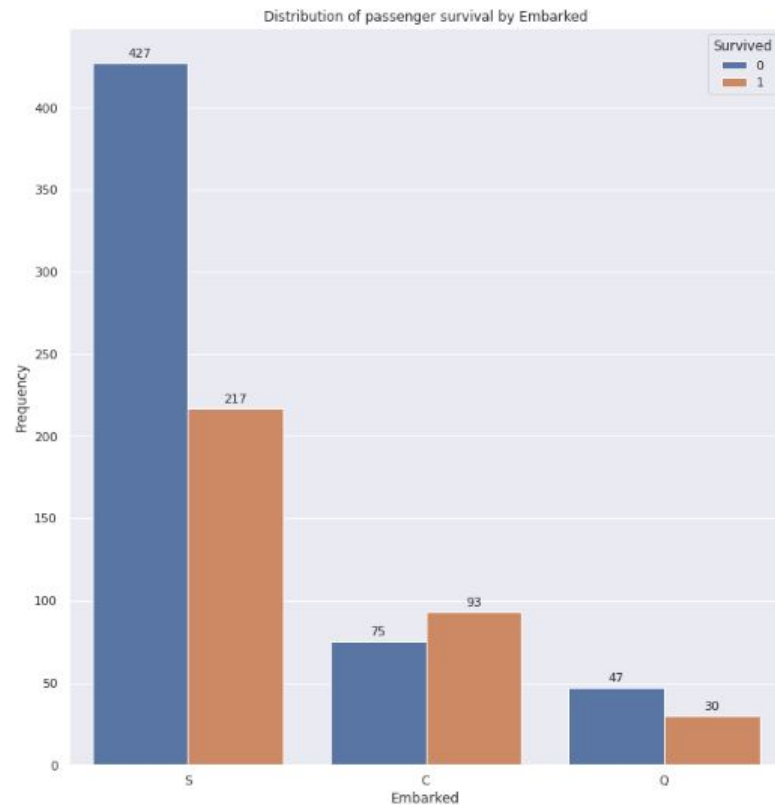
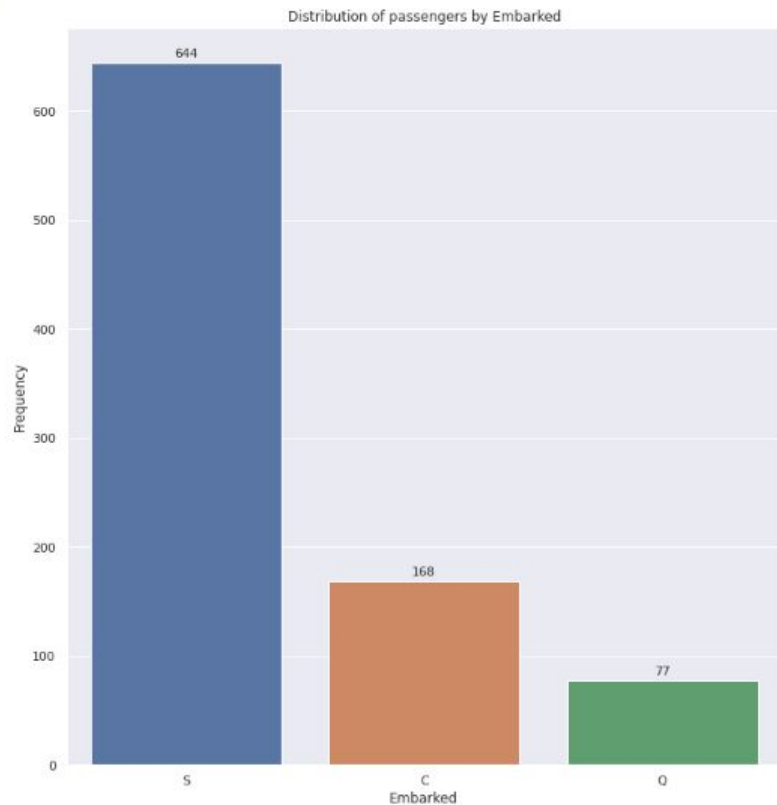


Gran parte de los pasajeros no eran padres e hijos.

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```



En este caso se puede observar que los boletos del Titanic no costaban (en su tiempo) un precio mayor a 100 dólares en su mayoría. Sin embargo, hay valores atípicos, como la persona que pagó 512 unidades. Puede estar sesgada esta gráfica, ya que aparecen valores en 0. No se sabría si son valores incorrectos o tal vez se refieran a los boletos de los niños.





Se puede observar que gran parte de los pasajeros eran provenientes de Southampton, posteriormente Cherburgo y finalmente Queenstown. Esto tiene una explicación lógica, ya que el puerto de abordaje inicial fue Southampton, después tuvo una escala en Cherburgo y finalmente en Queenstown.



02

Depuración de datos





Después de analizar los datos, se ha decidido retirar algunas de las variables que contiene originalmente el dataframe:

Ticket

Ubicación en la que los pasajeros se encontraban, pero resulta innecesario ya que esto mismo nos lo especifica la variable pclass.

PassengerId

No aporta mucha información ya que es un índice para contar los registros.

Name

Se identificó que los nombres en el dataframe contienen un título como Mr, Mrs, Miss, entre otros. Sin embargo, para saber el sexo y edad de una persona ya se tienen variables específicas.

Cabin

Cuenta con una gran cantidad de datos faltantes (687 registros, aproximadamente 78%) y la información que nos puede brindar puede ser de igual manera que el Ticket, la ubicación de los pasajeros.

Transformación de datos

- Ischild: Esta variable nos ayudará a saber si una persona es menor de edad o no.
- Family: Esta variable apoyará para saber si la persona tenía familiares a bordo.

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

Además, se tienen que transformar las variables de Sex y Embarked a numéricas.

Cambio de variables categóricas a numéricas



	Survived	Pclass	Age	Fare	Is_child	Family	female	male	C	Q	S
0	0	3	22.0	7.2500	0	1	0	1	0	0	1
1	1	1	38.0	71.2833	0	1	1	0	1	0	0
2	1	3	26.0	7.9250	0	0	1	0	0	0	1
3	1	1	35.0	53.1000	0	1	1	0	0	0	1
4	0	3	35.0	8.0500	0	0	0	1	0	0	1

Hay que recordar que había valores faltantes en Age(177) y Embarked (2). A continuación serán manejadas ambas situaciones.

Llenado de variables nulas

Age

```
guess_ages = np.zeros((2,3))

for i in range(0, 2):
    for j in range(0, 3):
        guess_df = cleanDf[(cleanDf['Sex'] == i) & \
                             (cleanDf['Pclass'] == j+1)][ 'Age' ].dropna()
        age_guess = guess_df.median()

        guess_ages[i,j] = int( age_guess/0.5 + 0.5 ) * 0.5

for i in range(0, 2):
    for j in range(0, 3):
        cleanDf.loc[ (cleanDf.Age.isnull()) & (cleanDf.Sex == i) & (cleanDf.Pclass == j+1),
                     'Age' ] = guess_ages[i,j]

cleanDf['Age'] = cleanDf['Age'].astype(int)
```

Llenado de variables nulas

Embarked y Fare

```
cleanDf['Embarked'] = cleanDf.Embarked.fillna(cleanDf.Embarked.dropna().max())  
cleanDf['Fare'] = cleanDf.Fare.fillna(cleanDf.Fare.dropna().mean())
```

03

Selección de modelos

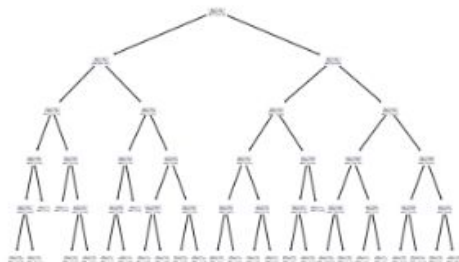
Modelo de Árboles de Decisión

+ Code

+ Markdown

```
from sklearn import tree
```

```
Decision_tree = tree.DecisionTreeClassifier(max_depth=5 , max_features=3 , random_state=42)  
Decision_tree.fit(X_train, Y_train)  
tree.plot_tree(Decision_tree)  
plt.show()
```



```
from sklearn.model_selection import cross_val_score  
print("La precision de nuestro modelo es: ", Decision_tree.score(X_test, Y_test))  
  
CV_scores = cross_val_score(Decision_tree, X_test, Y_test, cv=10)  
  
sns.heatmap(confusion_matrix(Y_test, y_pred_tree), annot=True, fmt = 'd')  
plt.plot()
```

La precision de nuestro modelo es: 0.7988826815642458

[62]: []



Regresión logística

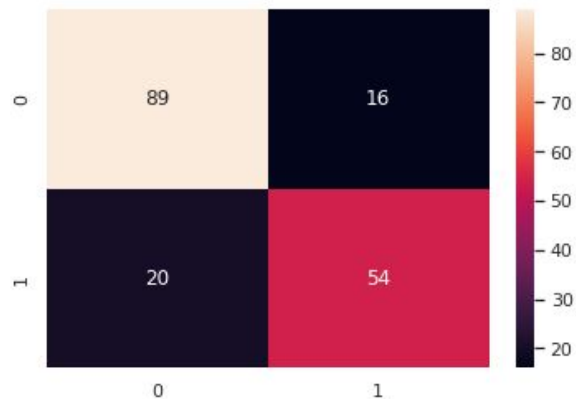
```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 42).fit(X_train_scaled, Y_train)
y_pred_logreg = log_reg.predict(X_test_scaled)
print("La precision de nuestro modelo es: ", log_reg.score( X_test_scaled, Y_test))

CV_scores = cross_val_score(log_reg, X_test_scaled, Y_test, cv=10)

sns.heatmap(confusion_matrix(Y_test, y_pred_logreg), annot=True, fmt = 'd')
plt.plot()
```

La precision de nuestro modelo es: 0.7988826815642458

: []



Support Vector Machines

```
from sklearn import svm

mod_svm = svm.SVC(kernel = 'rbf')
mod_svm.fit(X_train, Y_train)

y_pred_svm = mod_svm.predict(X_test)

print("La precision de nuestro modelo es: ", mod_svm.score(X_test, Y_test))

CV_scores = cross_val_score(mod_svm, X_test, Y_test, cv=10)

sns.heatmap(confusion_matrix(Y_test, y_pred_svm), annot=True, fmt = 'd')
plt.plot()
```

La precision de nuestro modelo es: 0.6536312849162011

[]



Random Forest

```
from sklearn.ensemble import RandomForestClassifier

mod_randomf = RandomForestClassifier(n_estimators= 150, max_features=7 , max_depth=5 ,min_samples_split=3, random_state=35)
mod_randomf.fit(X_train, Y_train)
y_pred_randf = mod_randomf.predict(X_test)

print("La precision de nuestro modelo es: ", mod_randomf.score(X_test, Y_test))

CV_scores = cross_val_score(mod_randomf, X_test, Y_test, cv=10)

sns.heatmap(confusion_matrix(Y_test, y_pred_randf), annot=True, fmt = 'd')
plt.plot()
```

La precision de nuestro modelo es: 0.8156424581005587

[]





04

Submissions a
Kaggle





Decision_tree.csv

20 hours ago by [Roeldlr](#)

[add submission details](#)

0.72966

reg_log.csv

20 hours ago by [Roeldlr](#)

[add submission details](#)

0.76555

reg_log.csv

21 hours ago by [Roeldlr](#)

[add submission details](#)

0.76794

reg_log.csv

21 hours ago by [Roeldlr](#)

Predicción regresión logística

0.75837

svm.csv

21 hours ago by [Roeldlr](#)

Predicciones SVM

0.66267

Decision_tree.csv

21 hours ago by [Roeldlr](#)

Predicciones del Árbol de Decisión

0.73205





Submission and Description	Public Score
rand (11).csv 12 hours ago by kuchao add submission details	0.78229
rand (10).csv 12 hours ago by kuchao add submission details	0.78947
Decision_tree (4).csv 12 hours ago by kuchao add submission details	0.77990
rand (9).csv 12 hours ago by kuchao add submission details	0.78708
rand (8).csv 12 hours ago by kuchao add submission details	0.77033
Decision_tree (3).csv 12 hours ago by kuchao add submission details	0.78229
rand (7).csv 12 hours ago by kuchao add submission details	0.78229

Recomendaciones a Futuro

Jugar con los hiper parámetros

Observar que hiper parámetros pueden mejorar el rendimiento de nuestro modelo

Buscar formas de mejorar los datos

Transformar "Fare" a una categórica, usando los cuartiles podemos reducir el riesgo de los valores atípicos.

Reentrenar el modelo

Entrenar el modelo de nuevo, pero esta vez con todo el set de datos, sin hacer el train_test

Valores nulos

Investigar las mejores formas para tratar los valores nulos

Gracias

