Reto - Titanic: Machine Learning from Disaster

Lautaro Gabriel Coteja - A01571214

Daniela Jiménez Téllez - A01654798

Andrés Villarreal González - A00833915

Héctor Hibran Tapia Fernández - A01661114

Link del repositorio:

https://github.com/Lautaro000/Inteligencia-Artificial-Avanzada-para-la-Ciencia-de-Datos-I

Importamos librerias

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
import plotly.graph_objects as go
```

In [2]:
```
plt.style.use('dark_background')
```

Importamos dataset (train)

In [3]:
```
df = pd.read_csv('/content/train.csv')
pd.reset_option('display.max_rows')
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN

891 rows × 12 columns

Recordamos que se descompone de la siguiente forma el dataset:

- train: Contiene las 12 columnas del dataset, solo tenemos del pasajero 1 al 891.
- test: Contiene 11 columnas del dataset, se quitó "Survived", tenemos del pasajero 892 al 1309.
- gender_submission: Contiene la columna "Survived" del test, con los mismos pasarejos, se usará como comparación de respuesta al finalizar el modelo.

Documentación

Variable	Definición	Clave
survival	Supervivencia	0 = No, 1 = Sí
pclass	Clase de ticket	1 = 1ª, 2 = 2ª, 3 = 3ª
sex	Sexo	
Age	Edad en años	
sibsp	Nº de hermanos / cónyuges a bordo	
parch	Nº de padres / hijos a bordo	
ticket	Número de ticket	
fare	Tarifa del pasajero	
cabin	Número de camarote	
embarked	Puerto de embarque	C = Cherburgo, Q = Queenstown, S = Southampton

Notas sobre las variables:

- **pclass:** Un indicador de estatus socioeconómico (SES)
  - 1ª = Alta
  - 2ª = Media
  - 3ª = Baja
- **edad:** La edad es fraccionaria si es menor de 1. Si la edad es estimada, se presenta en la forma xx.5
- **sibsp:** El conjunto de datos define las relaciones familiares de la siguiente manera:
  - Hermano/a = hermano, hermana, hermanastro/a
  - Cónyuge = esposo, esposa (se ignoran amantes y prometidos)
- **parch:** El conjunto de datos define las relaciones familiares de la siguiente manera:
  - Padre/madre = madre, padre
  - Hijo/a = hija, hijo, hijastra/o
  - Algunos niños viajaban solo con una niñera, por lo que parch=0 para ellos.

In [4]:
```
df.describe()
```

Out[4]:

	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

Procesamientos de Datos

In [5]:
```
df.isnull().sum()
```

Out[5]:

	0
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

Aquí "Cabin," está sospechoso como llenarlo, por ahora no la borraremos, pero la tendremos sí es necesario borrarla en el futuro.

Vamos a hacer experimentos...

Vamos a Rellenar los datos con interpolación

In [6]:
```
df["Title"] = df["Name"].apply(lambda x: x.split(",")[1].split(".")[0].strip())
df_mr = df[df["Title"] == "Mr"]
df_mr[df_mr.isnull().any(axis = 1)]
df_mr.loc[:, "Age"] = df_mr["Age"].interpolate(method = "linear")
df_miss = df[df["Title"] == "Miss"]
df_miss.loc[:, "Age"] = df_miss["Age"].interpolate(method = "linear")
df_mrs = df[df["Title"] == "Mrs"]
df_mrs.loc[:, "Age"] = df_mrs["Age"].interpolate(method = "linear")
df_master = df[df["Title"] == "Master"]
df_master.loc[:, "Age"] = df_master["Age"].interpolate(method = "linear")
df_extra = df[~df["Title"].isin(["Mr", "Miss", "Mrs", "Master"])]
mean_age_dr = df_extra[df_extra["Title"] == "Dr"]["Age"].mean()
df_extra.loc[(df["Title"] == "Dr") & (df["Age"].isnull()), "Age"] = mean_age_dr
df_list = [df_mr, df_miss, df_mrs, df_master, df_extra]
df = pd.concat(df_list, ignore_index = True)
```

In [7]:
```
pd.set_option('display.max_rows', None)
df.head(10)
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
2	6	0	3	Moran, Mr. James	male	44.5	0	0	330877	8.4583	NaN	
3	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	
4	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	
5	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	
6	18	1	2	Williams, Mr. Charles Eugene	male	37.0	0	0	244373	13.0000	NaN	
7	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	
8	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	
9	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	

In [8]:
```
df['Cabin'] = df['Cabin'].fillna(df['Cabin'].mode()[0])
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
df["Family"] = df["SibSp"] + df["Parch"] + 1
```

In [9]:
```
from sklearn.preprocessing import OneHotEncoder

def transform_dataframe(df):
    encoder = OneHotEncoder(sparse_output = False, handle_unknown = 'ignore')

    matrix = encoder.fit_transform(df[['Embarked']])
    column_names = ["C", "Q", "S"]

    for i in range(len(matrix.T)):
        df[column_names[i]] = matrix.T[i]

    matrix = encoder.fit_transform(df[['Sex']])
    column_names = ["Female", "Male"]

    for i in range(len(matrix.T)):
        df[column_names[i]] = matrix.T[i]

    return df
```

In [10]:
```
pd.reset_option('display.max_rows')
transform_dataframe(df)
```

Out[10]:

	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	B96 B98
1	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	B96 B98
2	6	0	3	Moran, Mr. James	male	44.5	0	0	330877	8.4583	B96 B98
3	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46
4	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	B96 B98
...	...	...	...	...	...	...	...	...	...	...	...
886	767	0	1	Brewe, Dr. Arthur Jackson	male	42.0	0	0	112379	39.6000	B96 B98
887	797	1	1	Leader, Dr. Alice (Farnham)	female	49.0	0	0	17465	25.9292	D17
888	823	0	1	Reuchlin, Jonkheer. John George	male	38.0	0	0	19972	0.0000	B96 B98
889	849	0	2	Harper, Rev. John	male	28.0	0	1	248727	33.0000	B96 B98
890	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	B96 B98

891 rows × 19 columns

In [11]:
```
df.isnull().sum()
```

Out[11]:

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	0
Embarked	0
Title	0
Family	0
C	0
Q	0
S	0
Female	0
Male	0

dtype: int64

Guardamos el Dataset limpio de "train.csv"

In [14]:
```
df.to_csv('/content/train_cleaned_INTER.csv', index = False)
df_train_cleaned = pd.read_csv('/content/train_cleaned_INTER.csv')
```

In [15]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Ya que tenemos 0 valores nulos, podemos iniciar con el análisis.

Análisis de Datos

In [16]:
```
df_train_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 19 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          891 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        891 non-null    object 
 11  Embarked     891 non-null    object 
 12  Title        891 non-null    object 
 13  Family       891 non-null    int64  
 14  C            891 non-null    float64
 15  Q            891 non-null    float64
 16  S            891 non-null    float64
 17  Female       891 non-null    float64
 18  Male         891 non-null    float64
dtypes: float64(7), int64(6), object(6)
memory usage: 132.4+ KB
```

In [17]:
```
df_train_cleaned.describe(include = 'all')
```

Out[17]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
count	891.000000	891.000000	891.000000	891	891	891.000000	891.000000	891.000000	8
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	6
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	NaN	3470
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	
mean	446.000000	0.383838	2.308642	NaN	NaN	29.807054	0.523008	0.381594	Na
std	257.353842	0.486592	0.836071	NaN	NaN	13.883697	1.102743	0.806057	Na
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	Na
25%	223.500000	0.000000	2.000000	NaN	NaN	21.000000	0.000000	0.000000	Na
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	Na
75%	668.500000	1.000000	3.000000	NaN	NaN	38.125000	1.000000	0.000000	Na
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	Na

In [18]:
```
fig = px.histogram(df_train_cleaned, x = "Survived", color = "Sex", barmode = "group",
                   color_discrete_map = {'male': '#00FFFF', 'female': '#FF69B4'})

fig.update_xaxes(tickvals = [0, 1], ticktext = ['Not Survived', 'Survived'], title_tex

fig.update_traces(textposition = 'outside', textfont_color = 'white')

fig.update_layout(
    title_text = 'Survivors by Sex',
    title_x = 0.5,
    plot_bgcolor = 'black',
    paper_bgcolor = 'black',
    font_color = 'white',
    xaxis = dict(
        color = 'white'
    ),
    yaxis = dict(
        color = 'white'
    ),
    legend = dict(
        font = dict(
            color = 'white'
        )
    )
)

fig.show()
```

In [19]:
```
fig = px.histogram(
    df_train_cleaned,
    x = "Fare",
    color = "Survived",
    marginal = "box",
    nbins = 50,
    title = "Distribution of Survival Fees",
    color_discrete_map = {0: "#ff073a", 1: "#39ff14"},
    category_orders = {"Survived": [0, 1]}
)

fig.for_each_trace(lambda t: t.update(name=t.name.replace("0", "Not Survived").replace

fig.update_layout(
    title_x = 0.5,
    plot_bgcolor = 'black',
    paper_bgcolor = 'black',
    font_color = 'white',
    title_font = dict(color = 'white'),
    legend_bgcolor = 'black',
    legend_font = dict(color = 'white')
)

fig.show()
```

In [20]:
```
fig = px.histogram(
    df_train_cleaned,
    x = "Age",
    color = "Survived",
    marginal = "box",
    nbins = 50,
    title = "Age Distribution",
    color_discrete_map = {0: "#ff073a", 1: "#39ff14"},
    category_orders = {"Survived": [0, 1]}
)

fig.for_each_trace(lambda t: t.update(name=t.name.replace("0", "Not Survived").replace

fig.update_layout(
    title_x = 0.5,
    plot_bgcolor = 'black',
    paper_bgcolor = 'black',
    font_color = 'white',
    title_font = dict(color = 'white'),
    legend_bgcolor = 'black',
    legend_font = dict(color = 'white')
)

fig.show()
```

In [21]:
```
pclass_counts = df_train_cleaned['Pclass'].value_counts().sort_index()

fig = go.Figure(data = [go.Bar(
    x = pclass_counts.index,
    y = pclass_counts.values,
    text = pclass_counts.values,
    textposition = 'outside',
    marker_color = ['#FFD700', '#C0C0C0', '#CD7F32'],
    marker_line_color = ['#FFD700', '#C0C0C0', '#CD7F32'],
    marker_line_width = 2
)])

fig.update_traces(
    textposition = 'outside',
    textfont = dict(size = 14),
    cliponaxis = False,
    offsetgroup = 0
)

fig.update_traces(y = [v + 20 for v in pclass_counts.values])

fig.update_layout(
    title = 'Number of People per Class',
    xaxis_title = ' ',
    yaxis_title = 'Count',
    xaxis = dict(
```

```
        tickmode = 'array',
        tickvals = [1, 2, 3],
        ticktext = ['1st Class', '2nd Class', '3rd Class'],
        linecolor = 'gray',
        gridcolor = 'black'
    ),
    yaxis = dict(
        range = [0, 520],
        linecolor = 'gray',
        gridcolor = 'gray'
    ),
    bargap = 0.2,
    width = 700,
    height = 500,
    plot_bgcolor = 'black',
    paper_bgcolor = 'black',
    font = dict(color = 'white'),
    title_font = dict(color = 'white')
)

fig.show()
```

In [22]:
```
fig = px.histogram(df_train_cleaned, x = 'Pclass', color = 'Sex', barmode = 'group', t
                   title = 'Number of Women and Men per Class',
                   color_discrete_map = {'male': '#00FFFF', 'female': '#FF69B4'})
```

```
fig.update_xaxes(title_text = ' ', tickvals = [1, 2, 3], ticktext = ['1st Class', '2nd

fig.update_traces(textposition = 'outside', textfont_color = 'white')

fig.update_layout(
    width = 1000,
    height = 500,
    plot_bgcolor = 'black',
    paper_bgcolor = 'black',
    font_color = 'white',
    xaxis = dict(
        color = 'white'
    ),
    yaxis = dict(
        color = 'white'
    ),
    legend = dict(
        font = dict(
            color = 'white'
        )
    )
)

fig.show()
```

In [23]:
```
df_count = df_train_cleaned.groupby(['Survived', 'Pclass']).size().reset_index(name =

fig = go.Figure()

pclass_names = {1: '1st Class', 2: '2nd Class', 3: '3rd Class'}
survival_labels = {0: 'Not Survived', 1: 'Survived'}

for pclass in sorted(df_train_cleaned['Pclass'].unique()):
    filtered_df = df_count[df_count['Pclass'] == pclass]
    filtered_df['Survived'] = filtered_df['Survived'].map(survival_labels)

    fig.add_trace(go.Bar(
        x = filtered_df['Survived'],
        y = filtered_df['count'],
        name = pclass_names[pclass],
        text = filtered_df['count'],
        textposition = 'outside',
        marker_color = ['#FFD700', '#C0C0C0', '#CD7F32'][pclass-1],
        marker_line_color = ['#FFD700', '#C0C0C0', '#CD7F32'][pclass-1],
        marker_line_width = 2
    ))

fig.update_layout(
    title_text = 'Survivors per Class',
    xaxis_title = ' ',
    barmode = 'group',
    plot_bgcolor = 'black',
    paper_bgcolor = 'black',
    font = dict(color = 'white'),
    title_font = dict(color = 'white')
)

fig.show()
```

```
<ipython-input-23-a8f29d6a8b23>:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
```

In [24]:
```
df_train_cleaned['Título'] = df_train_cleaned['Name'].str.extract(r'([A-Za-z]+)\.')

title_counts = df_train_cleaned['Título'].value_counts().reset_index()
title_counts.columns = ['Título', 'Count']
title_counts = title_counts.iloc[::-1].reset_index(drop = True)

neon_colors = [
    '#39FF14', '#0AFF0A', '#00FFFF', '#00BFFF', '#FF00FF', '#FF1493',
    '#FF4500', '#FFD700', '#ADFF2F', '#7FFF00', '#00FF7F', '#32CD32',
    '#00FA9A', '#00CED1', '#1E90FF', '#8A2BE2', '#FF69B4'
]

fig = go.Figure()

for i, row in title_counts.iterrows():
    fig.add_trace(go.Bar(
        y = [row['Título']],
        x = [row['Count']],
        orientation = 'h',
        name = row['Título'],
        marker = dict(color = neon_colors[i % len(neon_colors)]),
        text = row['Count'],
        textposition = 'outside'
    ))
```

```
fig.update_layout(
    template = 'plotly_dark',
    plot_bgcolor = '#000000',
    paper_bgcolor = '#000000',
    font = dict(color = 'white'),
    title = 'Number of People per Title',
    xaxis = dict(title = 'Count'),
    yaxis = dict(title = 'Title'),
    height = 550
)

fig.show()
```

In [25]:
```
average_age_per_title = df_train_cleaned.groupby('Título')['Age'].mean().dropna()
average_age_per_title = average_age_per_title.sort_values(ascending = True)

fig = go.Figure()

for i, (title, age) in enumerate(average_age_per_title.items()):
    fig.add_trace(go.Bar(
        y = [title],
        x = [age],
        orientation = 'h',
        name = title,
        marker = dict(color = neon_colors[i % len(neon_colors)]),
```

```
        text = f'{age:.2f}',
        textposition = 'inside'
    ))

fig.update_layout(
    template = 'plotly_dark',
    plot_bgcolor = '#000000',
    paper_bgcolor = '#000000',
    font = dict(color = 'white'),
    title = 'Average Age by Title',
    xaxis = dict(title = 'Age'),
    yaxis = dict(title = 'Title'),
    height = 550
)

fig.show()
```

Dropeamos variables que no necesitamos

In [26]:
```
df_train_cleaned = df_train_cleaned.drop(['Sex', 'Embarked', 'Name', 'Ticket', 'Cabin'
```

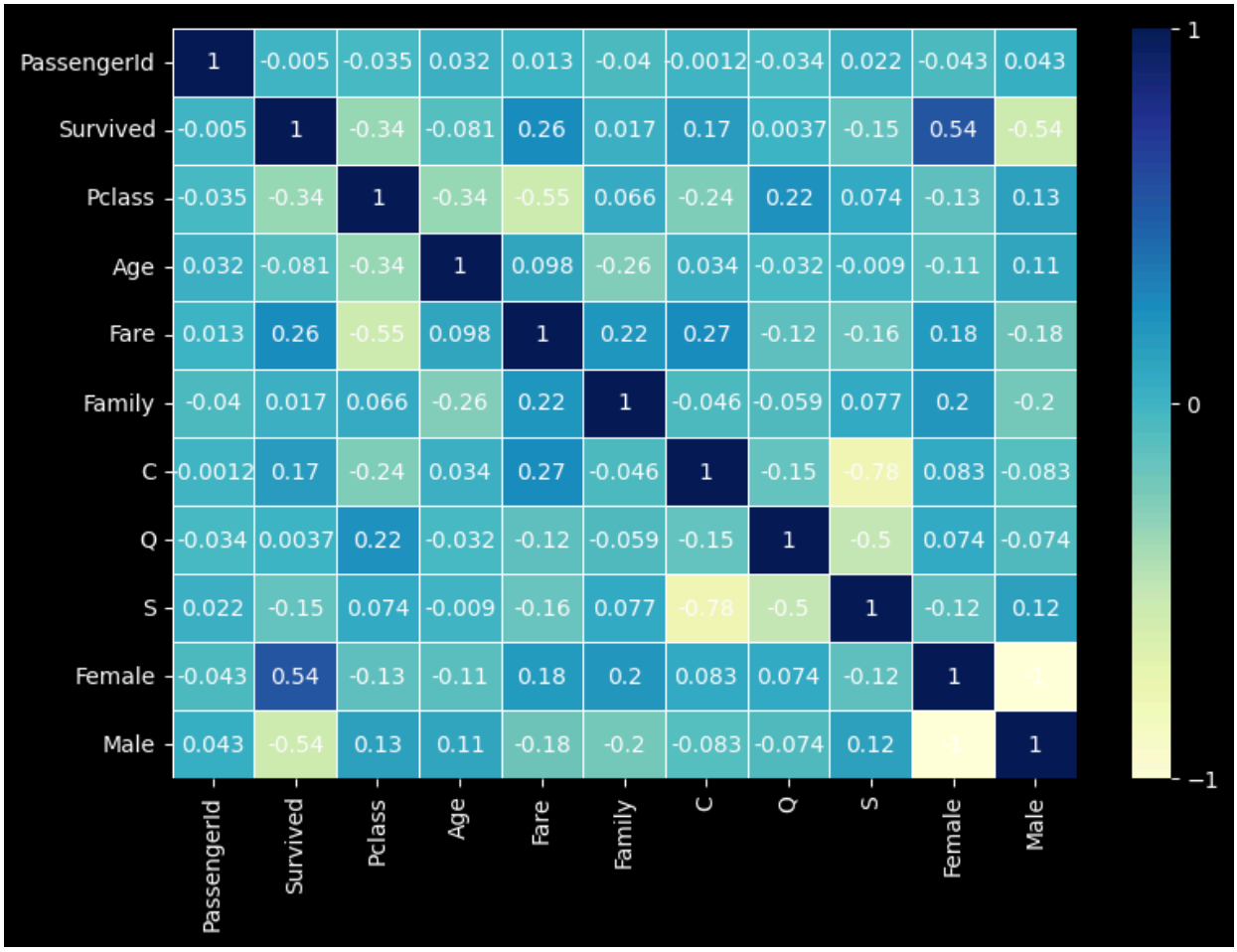
In [27]:
```
df_train_cleaned.head(20)
```

Out[27]:

	PassengerId	Survived	Pclass	Age	Fare	Family	C	Q	S	Female	Male
0	1	0	3	22.0	7.2500	2	0.0	0.0	1.0	0.0	1.0
1	5	0	3	35.0	8.0500	1	0.0	0.0	1.0	0.0	1.0
2	6	0	3	44.5	8.4583	1	0.0	1.0	0.0	0.0	1.0
3	7	0	1	54.0	51.8625	1	0.0	0.0	1.0	0.0	1.0
4	13	0	3	20.0	8.0500	1	0.0	0.0	1.0	0.0	1.0
5	14	0	3	39.0	31.2750	7	0.0	0.0	1.0	0.0	1.0
6	18	1	2	37.0	13.0000	1	0.0	0.0	1.0	0.0	1.0
7	21	0	2	35.0	26.0000	1	0.0	0.0	1.0	0.0	1.0
8	22	1	2	34.0	13.0000	1	0.0	0.0	1.0	0.0	1.0
9	24	1	1	28.0	35.5000	1	0.0	0.0	1.0	0.0	1.0
10	27	0	3	23.5	7.2250	1	1.0	0.0	0.0	0.0	1.0
11	28	0	1	19.0	263.0000	6	0.0	0.0	1.0	0.0	1.0
12	30	0	3	42.5	7.8958	1	0.0	0.0	1.0	0.0	1.0
13	34	0	2	66.0	10.5000	1	0.0	0.0	1.0	0.0	1.0
14	35	0	1	28.0	82.1708	2	1.0	0.0	0.0	0.0	1.0
15	36	0	1	42.0	52.0000	2	0.0	0.0	1.0	0.0	1.0
16	37	1	3	31.5	7.2292	1	1.0	0.0	0.0	0.0	1.0
17	38	0	3	21.0	8.0500	1	0.0	0.0	1.0	0.0	1.0
18	43	0	3	21.0	7.8958	1	1.0	0.0	0.0	0.0	1.0
19	46	0	3	21.0	8.0500	1	0.0	0.0	1.0	0.0	1.0

In [28]:

```
df_numeric = df_train_cleaned.select_dtypes(include=['number'])
corr = df_numeric.corr()

plt.figure(figsize = (9, 6))

sns.heatmap(corr, annot = True, cmap = 'YlGnBu', annot_kws = {'color': 'white'},
            cbar_kws = {'ticks': [-1, 0, 1], 'orientation': 'vertical'},
            linewidths = 0.5, linecolor = 'white')

plt.show()
```



In [29]:
```
df_train_cleaned.columns
```

Out[29]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Age', 'Fare', 'Family', 'C', 'Q',
       'S', 'Female', 'Male'],
      dtype='object')
```

Modelos Predictivos

Regresión Logística

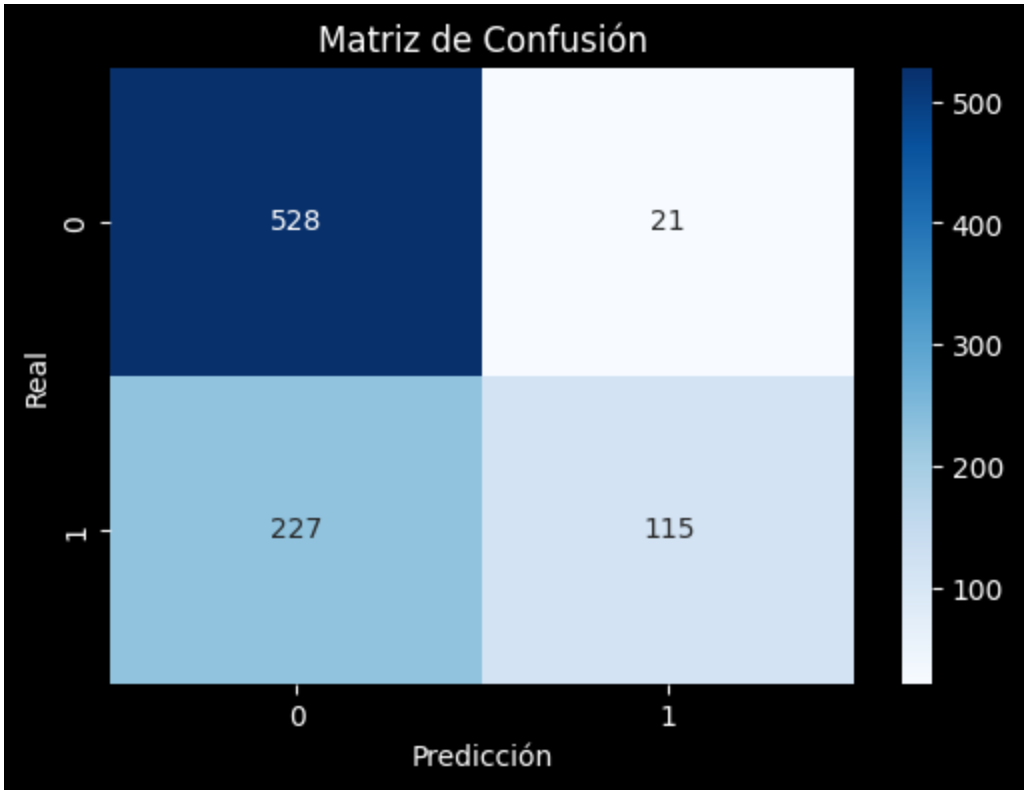
In [30]:
```
X = df_train_cleaned.drop('Survived', axis = 1).values
y = df_train_cleaned['Survived'].values

X = np.hstack((np.ones((X.shape[0], 1)), X))
theta = np.zeros(X.shape[1])

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def compute_cost(X, y, theta):
    m = len(y)
    h = sigmoid(X @ theta)
    cost = (1/m) * (-y @ np.log(h) - (1 - y) @ np.log(1 - h))
    return cost
```

```
def gradient_descent(X, y, theta, alpha, num_iterations):
    m = len(y)
    cost_history = []

    for _ in range(num_iterations):
        h = sigmoid(X @ theta)
        theta -= (alpha/m) * (X.T @ (h - y))
        cost_history.append(compute_cost(X, y, theta))

    return theta, cost_history

alpha = 0.000033
num_iterations = 100000

theta, cost_history = gradient_descent(X, y, theta, alpha, num_iterations)

print("Parámetros finales (theta):", theta)
print("Costo final:", cost_history[-1])
```

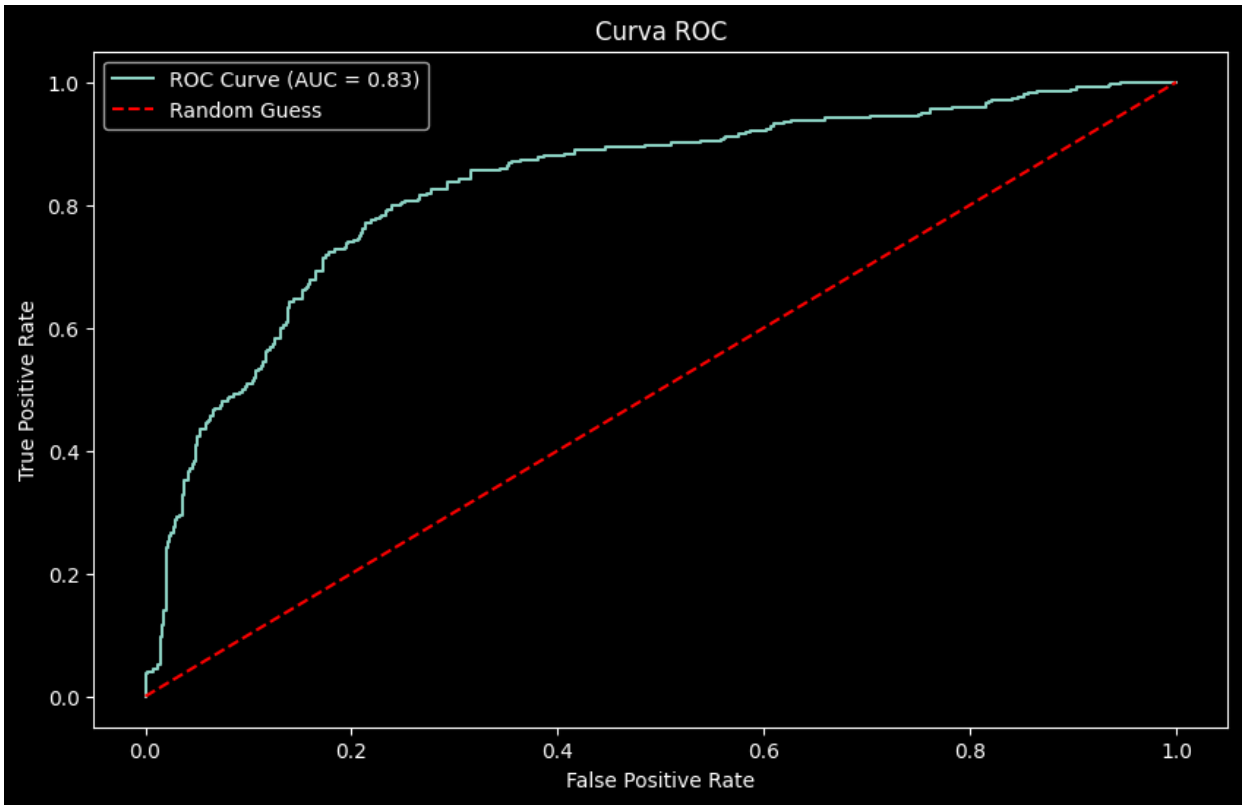
```
Parámetros finales (theta): [ 0.01324333  0.00032775 -0.17431588 -0.01495037  0.01405
764 -0.09055647
  0.0474905   0.01628143 -0.0505286   0.31605913 -0.3028158 ]
Costo final: 0.5458265999750608
```

In [31]:

```
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score

predictions = sigmoid(X @ theta) >= 0.5

cm = confusion_matrix(y, predictions)
plt.figure(figsize = (6, 4))
sns.heatmap(cm, annot = True, fmt = 'd', cmap = 'Blues')
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión')
plt.show()
```



In [32]:
```
fpr, tpr, thresholds = roc_curve(y, sigmoid(X @ theta))
roc_auc = roc_auc_score(y, sigmoid(X @ theta))

plt.figure(figsize = (10, 6))
plt.plot(fpr, tpr, label = f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], linestyle = '--', color = 'r', label = 'Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Curva ROC')
plt.legend()
plt.show()
```



Procesamiento de datos con "test.csv"

In [33]:
```
test_df = pd.read_csv('/content/test.csv')
test_df.head()
```

Out[33]:

	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
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

In [34]:
```
test_df["Title"] = test_df["Name"].apply(lambda x: x.split(",")[1].split(".")[0].strip
df_mr = test_df[test_df["Title"] == "Mr"]
df_mr[df_mr.isnull().any(axis = 1)]
df_mr.loc[:, "Age"] = df_mr["Age"].interpolate(method = "linear")
df_miss = test_df[test_df["Title"] == "Miss"]
df_miss.loc[:, "Age"] = df_miss["Age"].interpolate(method = "linear")
df_mrs = test_df[test_df["Title"] == "Mrs"]
df_mrs.loc[:, "Age"] = df_mrs["Age"].interpolate(method = "linear")
df_master = test_df[test_df["Title"] == "Master"]
df_master.loc[:, "Age"] = df_master["Age"].interpolate(method = "linear")
df_extra = test_df[~test_df["Title"].isin(["Mr", "Miss", "Mrs", "Master"])]
mean_age_dr = df_extra[df_extra["Title"] == "Dr"]["Age"].mean()
df_extra.loc[(test_df["Title"] == "Dr") & (test_df["Age"].isnull()), "Age"] = mean_age
df_list = [df_mr, df_miss, df_mrs, df_master, df_extra]
test_df = pd.concat(df_list, ignore_index = True)
```

In [35]:
```
test_df['Cabin'] = test_df['Cabin'].fillna(test_df['Cabin'].mode()[0])
test_df['Embarked'] = test_df['Embarked'].fillna(test_df['Embarked'].mode()[0])
test_df["Family"] = test_df["SibSp"] + test_df["Parch"] + 1
test_df.isnull().sum()
```

Out[35]:

	0
PassengerId	0
Pclass	0
Name	0
Sex	0
Age	1
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	0
Embarked	0
Title	0
Family	0

dtype: int64

In [36]:
```
pd.reset_option('display.max_rows')
transform_dataframe(test_df)
```

Out[36]:

	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	B57 B59 B63 B66	Q
1	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	B57 B59 B63 B66	Q
2	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	B57 B59 B63 B66	S
3	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	B57 B59 B63 B66	S
4	899	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	B57 B59 B63 B66	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1041	2	Lahtinen, Rev. William	male	30.0	1	1	250651	26.0000	B57 B59 B63 B66	S
414	1056	2	Peruschitz, Rev. Joseph Maria	male	41.0	0	0	237393	13.0000	B57 B59 B63 B66	S
415	1094	1	Astor, Col. John Jacob	male	47.0	1	0	PC 17757	227.5250	C62 C64	C
416	1185	1	Dodge, Dr. Washington	male	53.0	1	1	33638	81.8583	A34	S
417	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C

418 rows × 18 columns

Guardamos el Dataset limpio de "test.csv"

In [37]:
```
test_df.to_csv('/content/test_cleaned_INTER.csv', index = False)
test_df_cleaned = pd.read_csv('/content/test_cleaned_INTER.csv')
```

In [38]:
```
test_df_cleaned = test_df_cleaned.drop(['Sex', 'Embarked', 'Name', 'Ticket', 'Cabin',
```

In [39]:
```
test_df_cleaned.columns
```

Out[39]:
```
Index(['PassengerId', 'Pclass', 'Age', 'Fare', 'Family', 'C', 'Q', 'S',
       'Female', 'Male'],
      dtype='object')
```

Corremos el modelo pero ahora con "test.csv"

In [40]:
```
X_test = test_df_cleaned.values
X_test = np.hstack((np.ones((X_test.shape[0], 1)), X_test))

def predict(X, theta):
    probabilities = sigmoid(X @ theta)
    return (probabilities >= 0.5).astype(int)

predictions = predict(X_test, theta)

test_df_cleaned['Survived'] = predictions
```

Comparamos resultados con gender_submission.csv

In [42]:
```
gender_submission_df = pd.read_csv('/content/gender_submission.csv')
results_df = test_df_cleaned[['PassengerId', 'Survived']].merge(gender_submission_df,

accuracy = (results_df['Survived_pred'] == results_df['Survived_real']).mean()
print(f"Precisión del modelo en el conjunto de test: {accuracy:.4f}")
```

```
Precisión del modelo en el conjunto de test: 0.7584
```

In [43]:
```
correct_predictions = (results_df['Survived_pred'] == results_df['Survived_real']).sum
total_predictions = len(results_df)

print(f"Predicciones correctas: {correct_predictions}/{total_predictions}")
```

```
Predicciones correctas: 317/418
```

In [44]:
```
results_df.head(10)
```

Out[44]:

	PassengerId	Survived_pred	Survived_real
0	892	0	0
1	894	0	0
2	895	0	0
3	897	0	0
4	899	0	0
5	901	0	0
6	902	0	0
7	903	0	0
8	905	0	0
9	908	0	0

Referencias

---

https://www.kaggle.com/code/abhinavatall/titanic-dataset-top-3-accuracy-with-ensambling

https://www.kaggle.com/code/vinothan/titanic-model-with-90-accuracy

In [ ]:
```
%%shell
jupyter nbconvert --to html /content/Port_Imple_WFRAMEWORK_A01571214_Lautaro_Coteja.ip
```