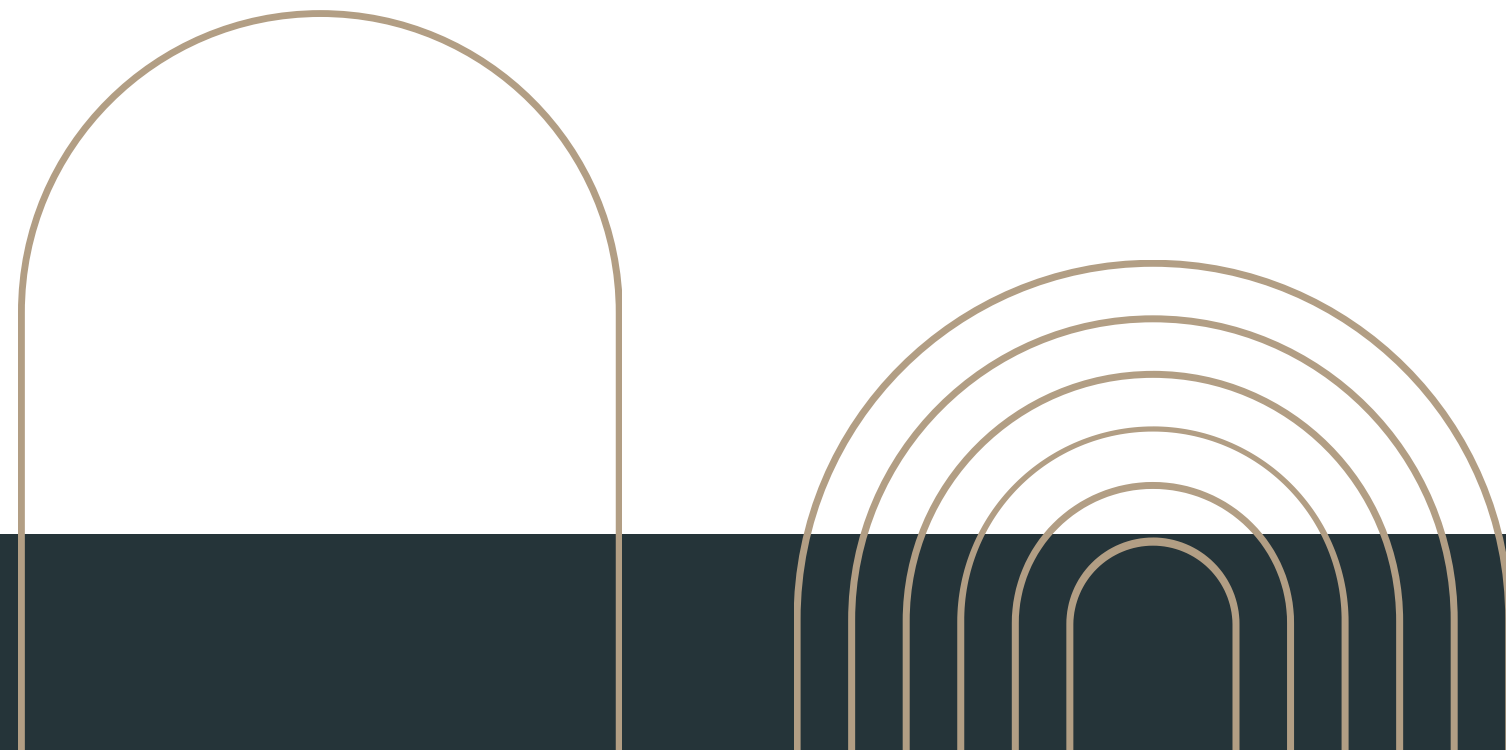


# RETO: TITANIC – MACHINE LEARNING FROM DISASTER

*Equipo 2*



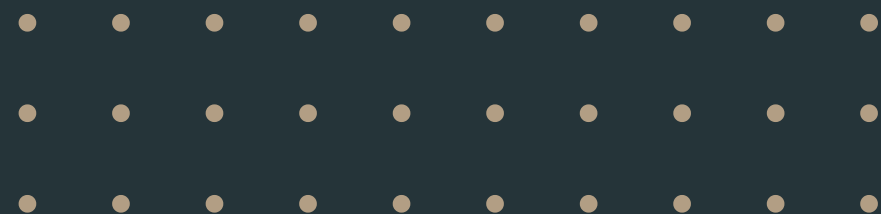
**01.** INTRODUCCIÓN

**02.** PROCESAMIENTO DE DATOS

**03.** ANÁLISIS DE DATOS

**04.** MODELOS PREDICTIVOS

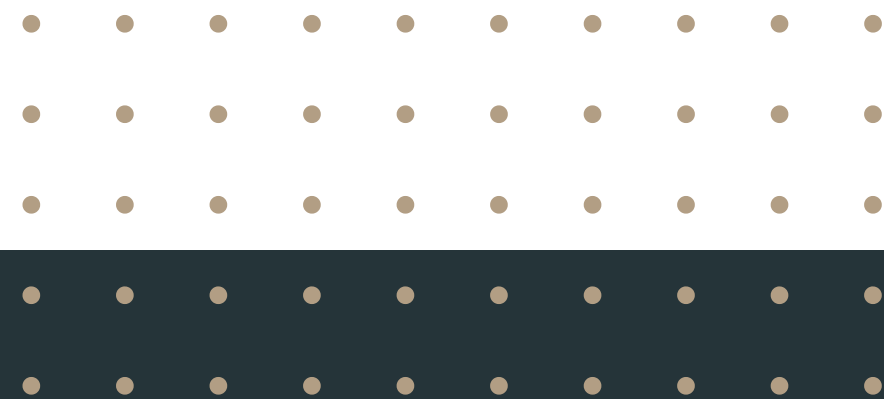
**05.** CONCLUSIONES

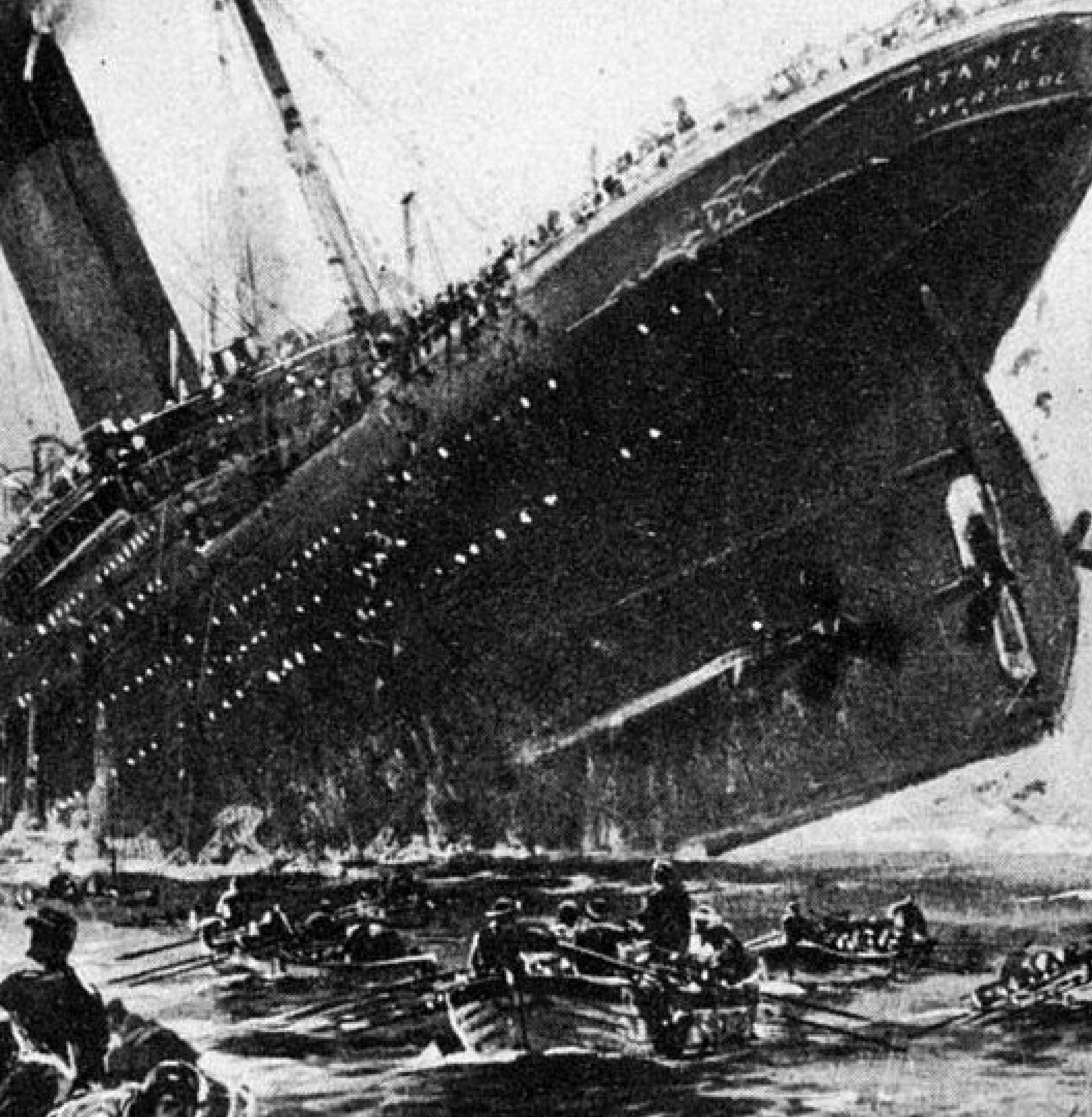


**ÍNDICE**

01.

# INTRODUCCIÓN





El hundimiento del Titanic el 15 de abril de 1912, tras chocar contra un iceberg, resultó en la muerte de aproximadamente 1,500 personas de las 2,224 a bordo.

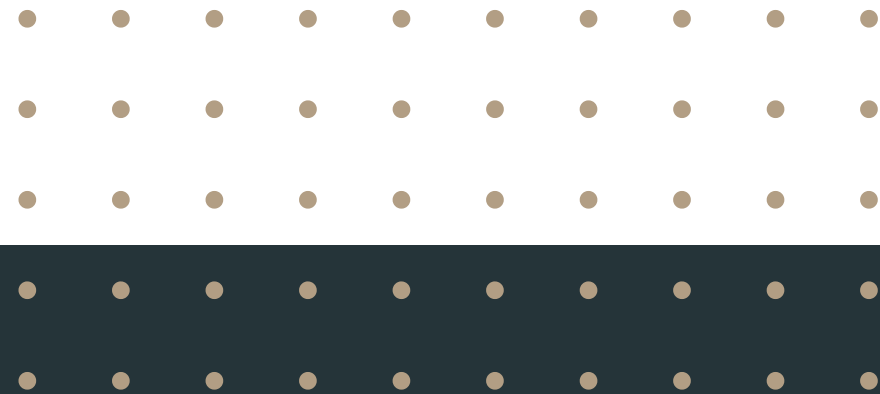
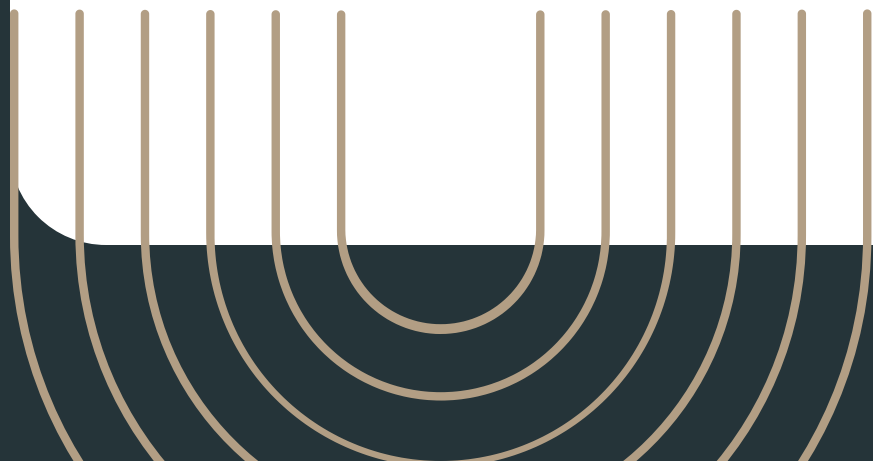
Aunque el barco fue promocionado como insumergible, la falta de suficientes botes salvavidas y varios errores humanos contribuyeron al desastre.

Solo unos 700 pasajeros sobrevivieron, y las probabilidades de supervivencia variaron según factores como la clase social, el género, la edad, entre otros.

Este proyecto utilizaremos modelos de predicción para analizar estas variables y determinar qué factores influenciaron la probabilidad de sobrevivir en esta tragedia.

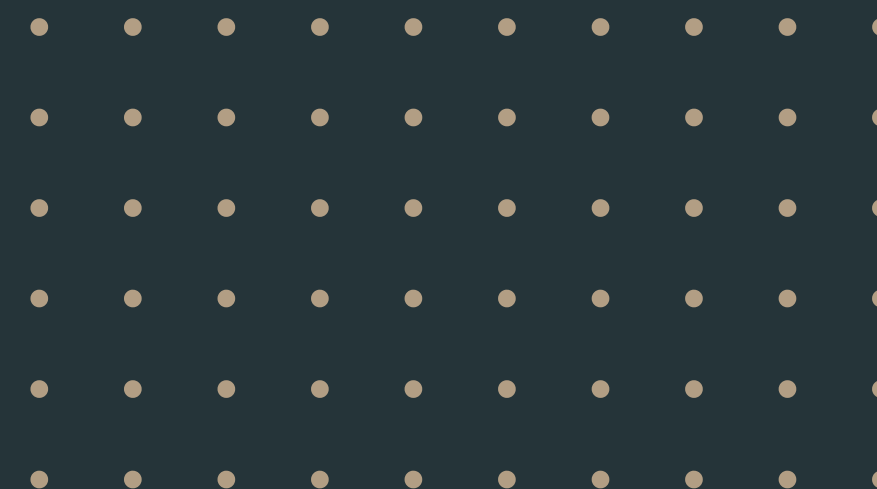
02.

# PROCESAMIENTO DE DATOS



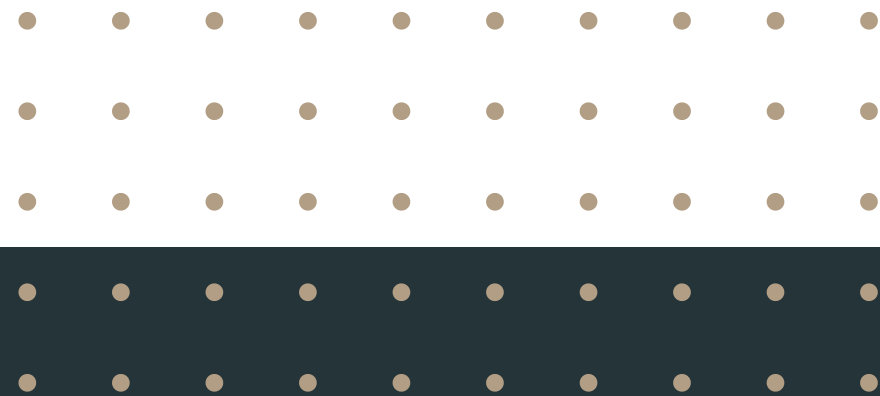
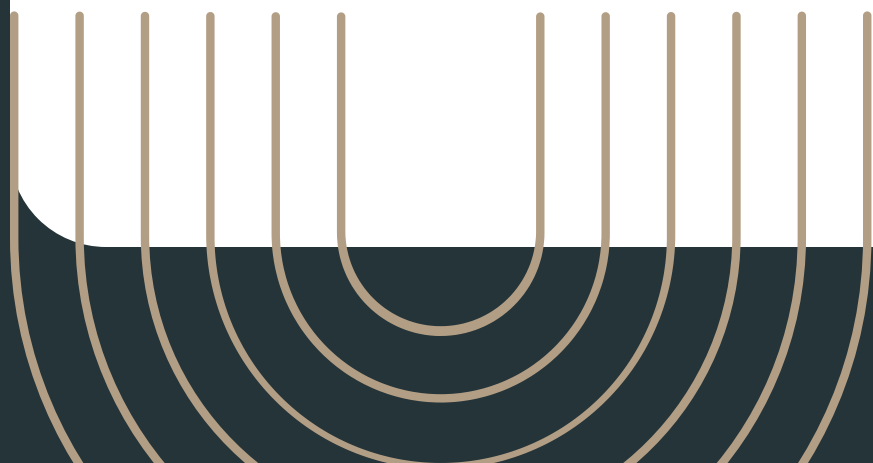
## 2.1 LIMPIEZA DE DATOS

- Manejo de valores faltantes (Cabin, Age, Fare, Embarked)
- Transformación de nuevas variables (Title, Ticket, Tipo\_fam)
- Codificación de variables categóricas



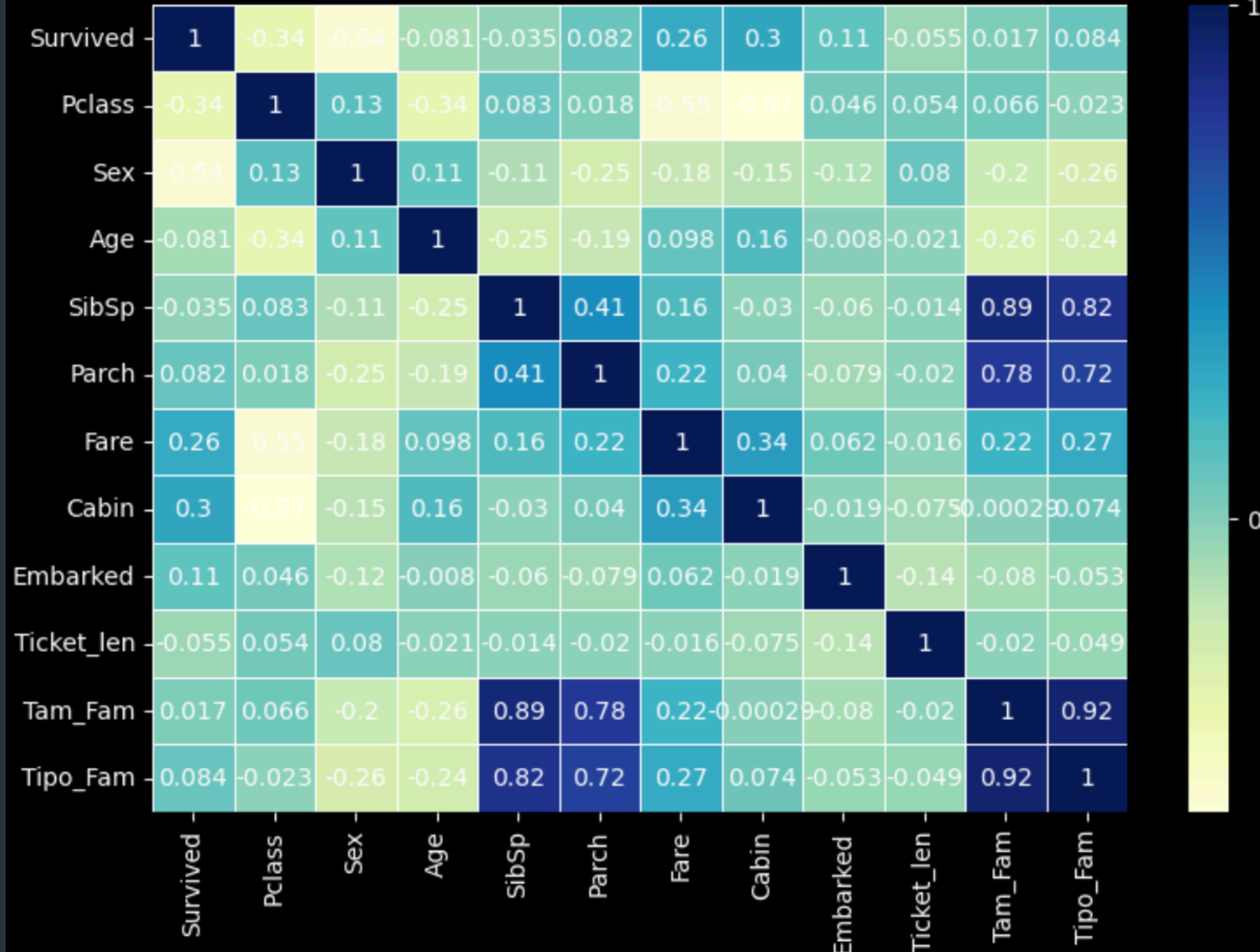
03.

# ANÁLISIS DE DATOS

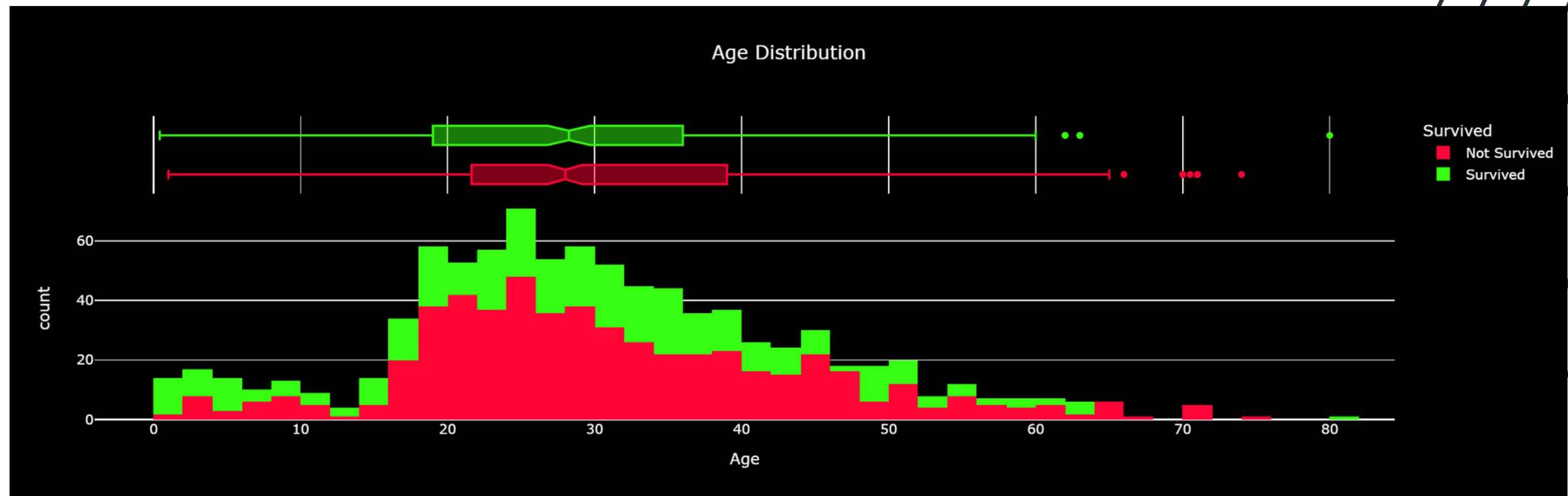
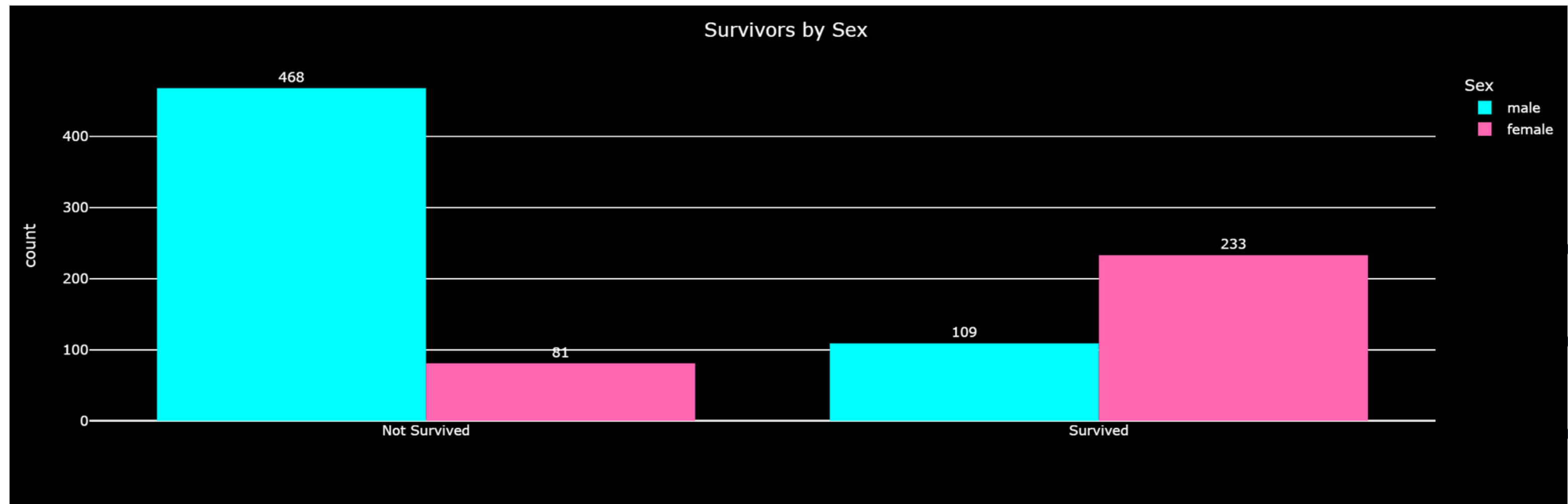




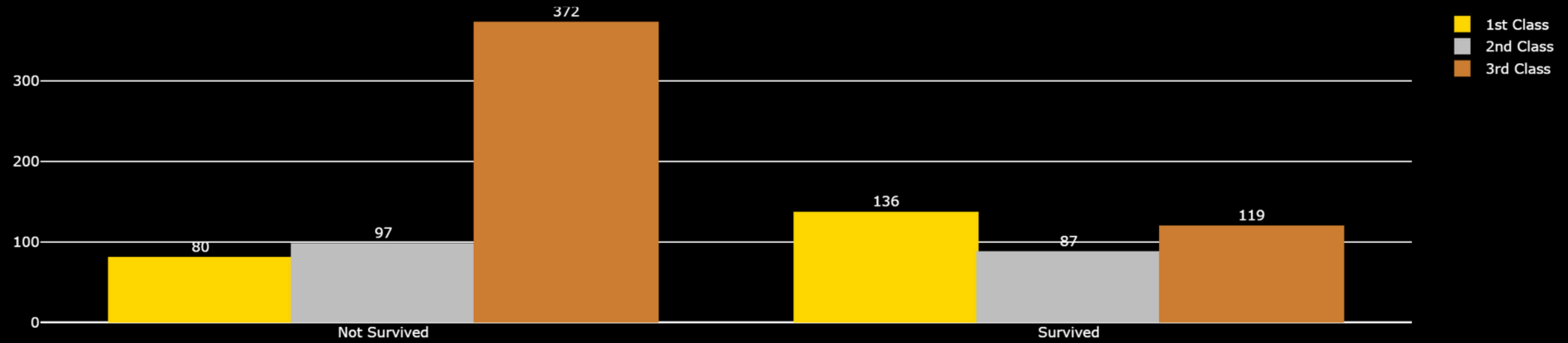
# Matriz de correlación de Train



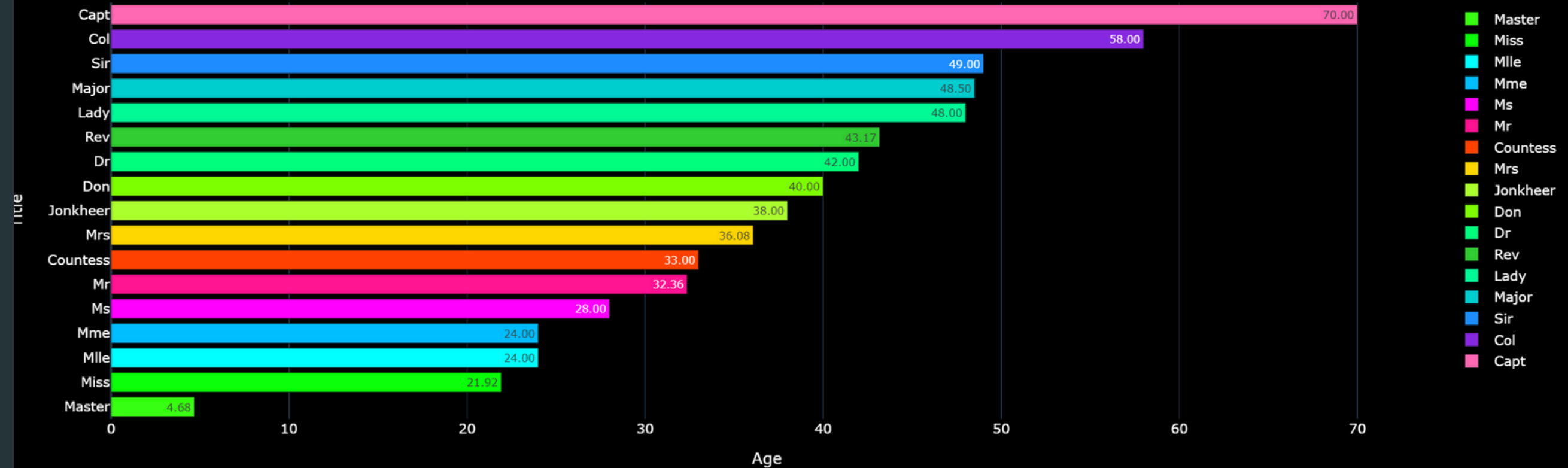




Survivors per Class

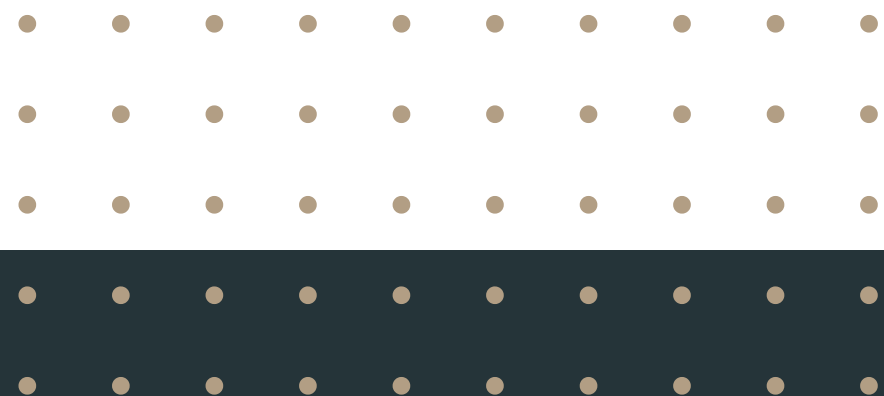


Average Age by Title



04.

# MODELOS PREDICTIVOS



# 3.1 REGRESIÓN LOGÍSTICA

```
# Características y variable objetivo
y = df_train_cleaned['Survived']
features = ['Pclass', 'Title', 'Embarked', 'Tipo_Fam', 'Ticket_len', 'Ticket_2letter']
X = df_train_cleaned[features]

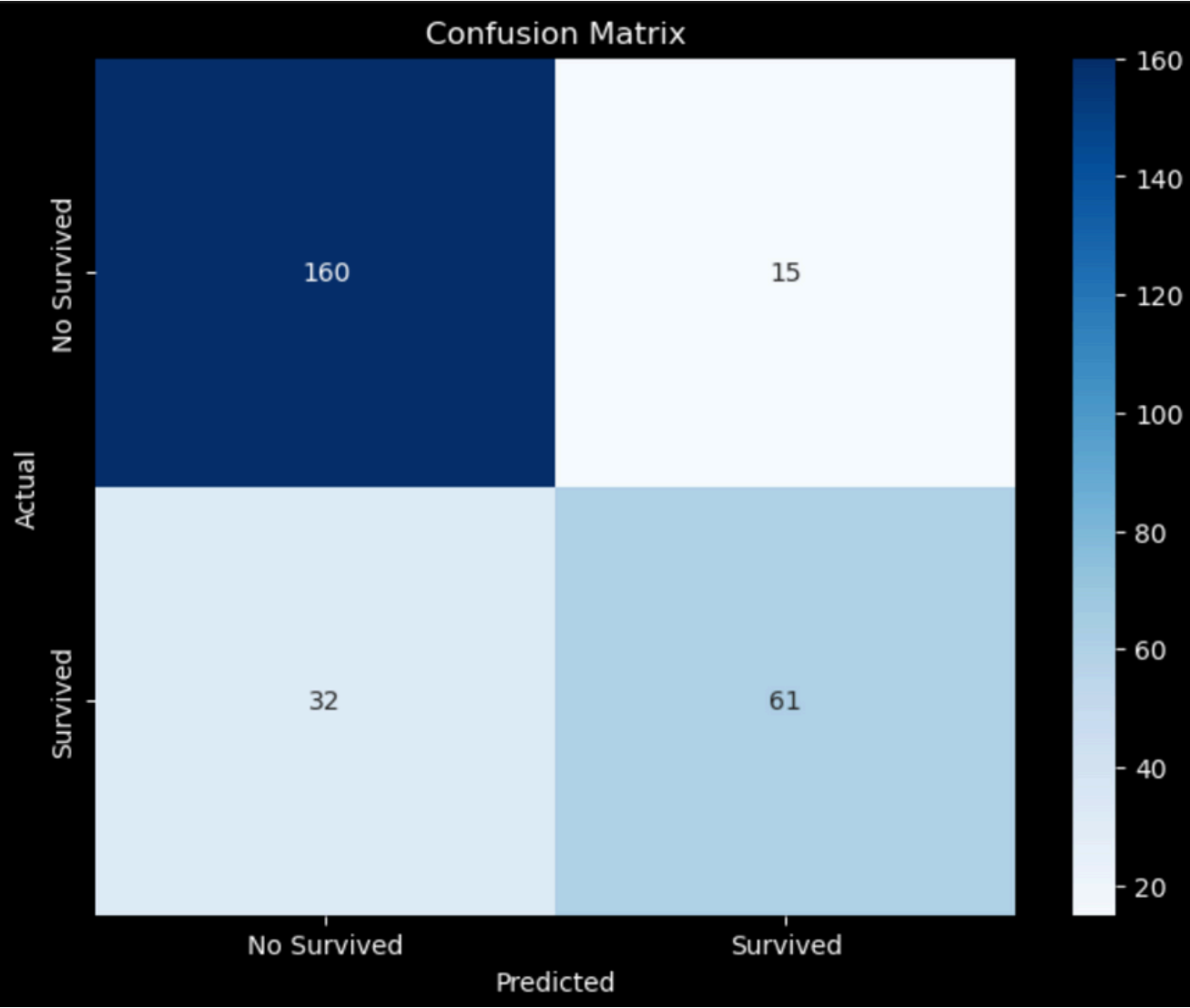
# Dividimos los datos en entrenamiento y validación (70% entrenamiento, 30% validación)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)

# Procesador para transformar las variables
# Aquí agrego un SimpleImputer para valores numéricos (si es necesario) y con OneHotEncoder para categóricas
preprocessor = ColumnTransformer(transformers=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'), features)
])

# Definimos el modelo de regresión logística con los hiperparámetros proporcionados
modelo = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', LogisticRegression(
        random_state=42,
        max_iter=1000,
        C=100,                # Hiperparámetro C ajustado
        class_weight=None,    # Sin balanceo de clases
        penalty='l2',         # Penalización L1
        solver='newton-cg'))])

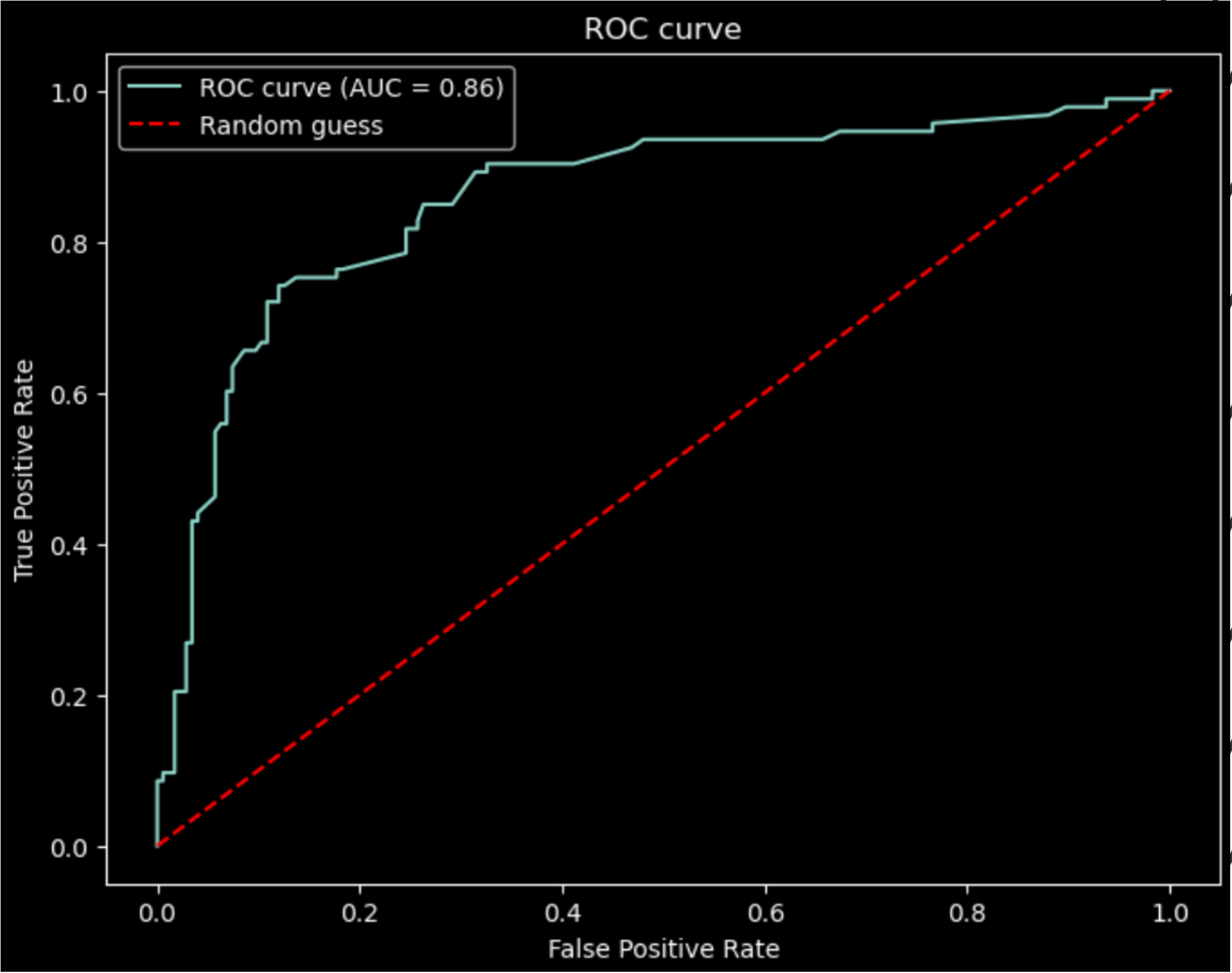
# Entrenamiento del modelo
modelo.fit(X_train, y_train)

# Predicciones en el conjunto de validación
y_pred = modelo.predict(X_val)
```



	Precision	Recall	F1-Score	Support
Class 0	0.83	0.91	0.87	175
Class 1	0.80	0.66	0.72	93
Accuracy			0.82	268
Macro Avg	0.82	0.79	0.80	268
Weighted Avg	0.82	0.82	0.82	268

Cuadro 1: Informe de clasificación con puntuación de precisión de 0.8246



## 3.2 RANDOM FOREST

```
# Características y variable objetivo
y = df_train_cleaned["Survived"]
features = ["Pclass", "Title", "Embarked", "Tipo_Fam", "Ticket_len", "Ticket_2letter"]
X = df_train_cleaned[features]

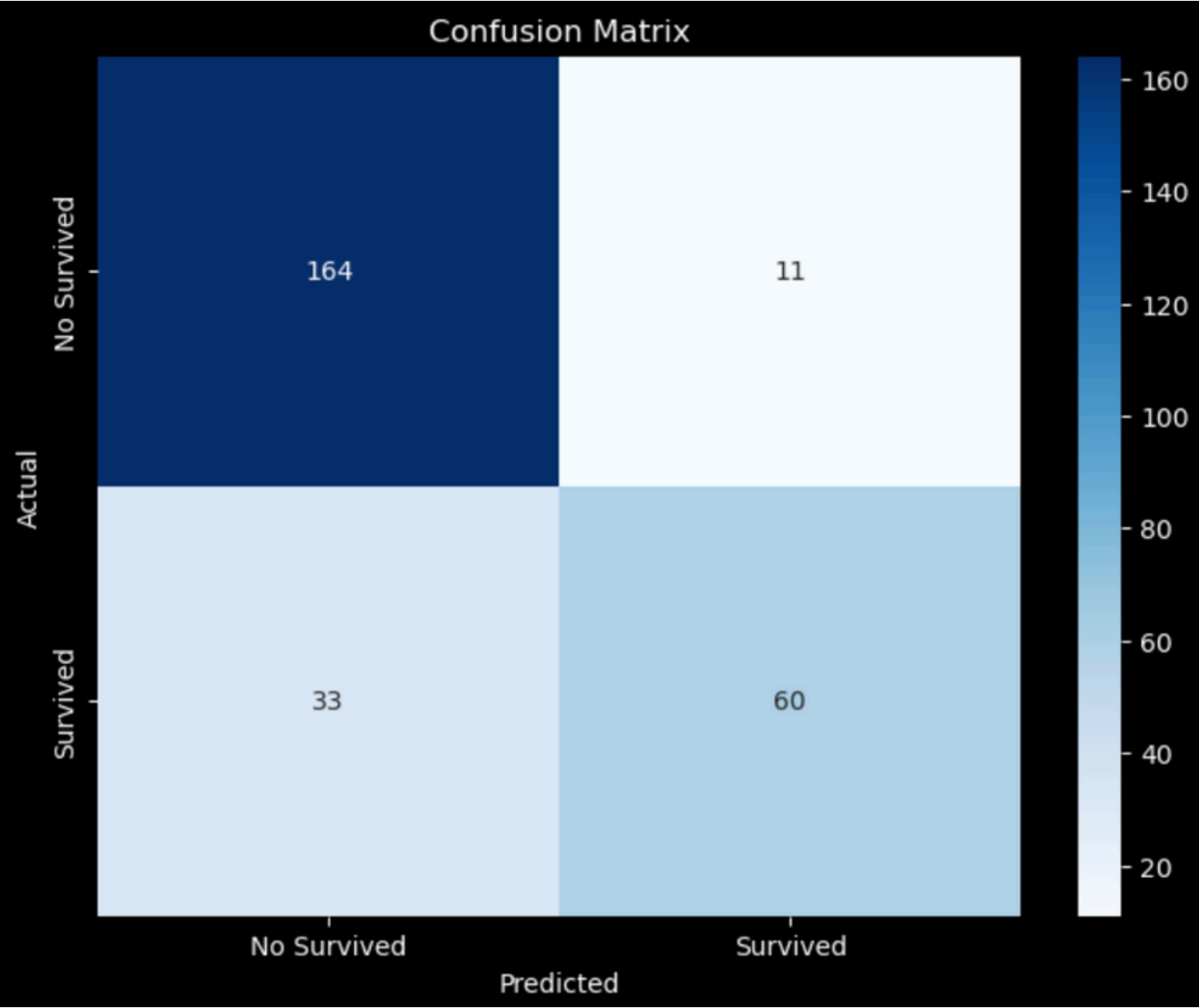
# Dividimos los datos en entrenamiento y validación (70% entrenamiento, 20% validación)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.3, random_state = 42)

# Procesador para transformar a variables dummy
preprocessor = ColumnTransformer(transformers=[('onehot', OneHotEncoder(handle_unknown = "ignore"), features), ])

# Definimos el modelo y hacemos la transformación a variables dummy
modelo = Pipeline(steps=[('preprocessor', preprocessor),
|   ('model', RandomForestClassifier(random_state = 42, n_estimators = 500, max_depth = 5))])

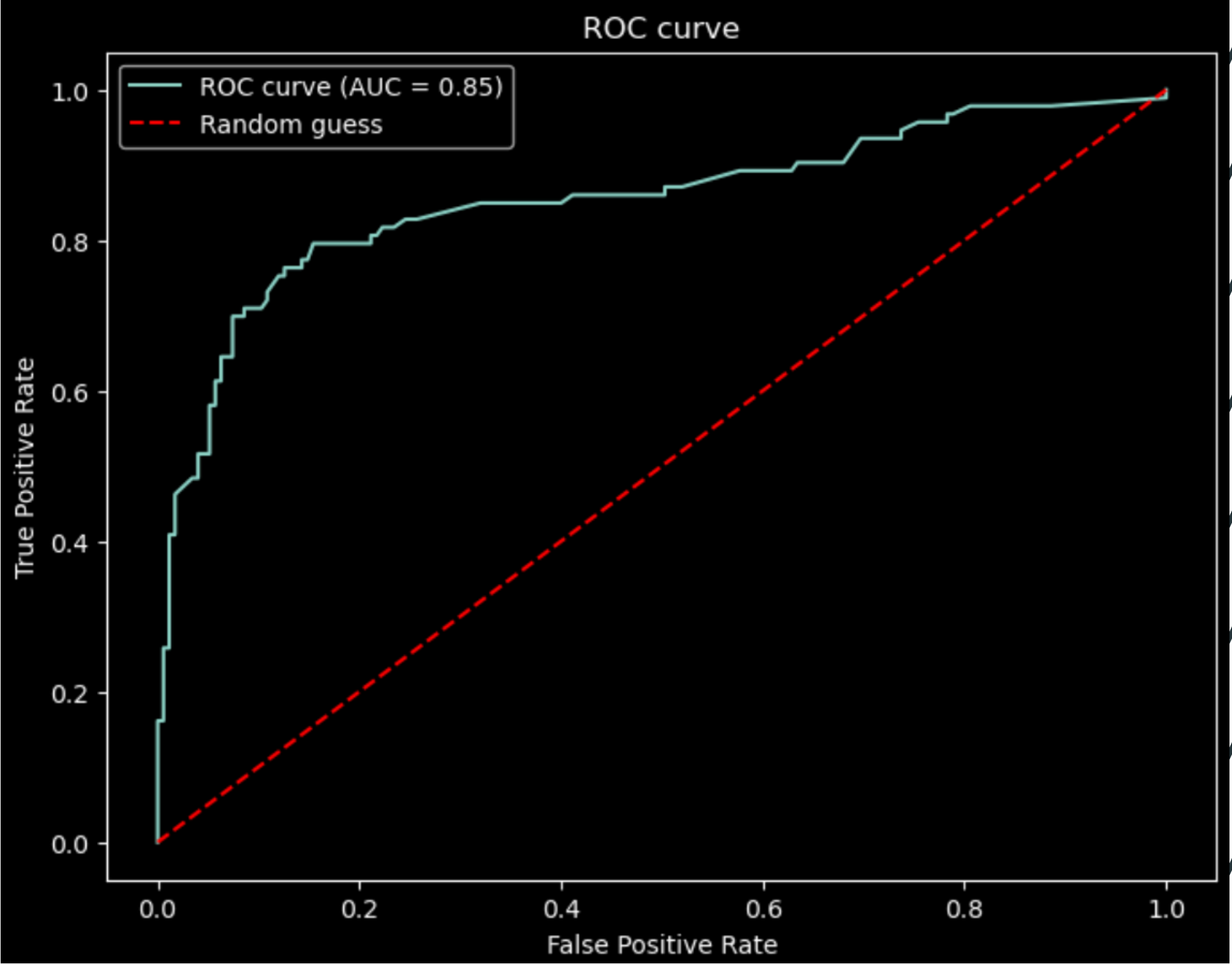
# Entrenamiento del modelo
modelo.fit(X_train, y_train)

# Predicciones en el conjunto de validación
y_pred = modelo.predict(X_val)
```



	Precision	Recall	F1-Score	Support
Class 0	0.83	0.94	0.88	175
Class 1	0.85	0.65	0.73	93
Accuracy			0.84	268
Macro Avg	0.84	0.79	0.81	268
Weighted Avg	0.84	0.84	0.83	268

Cuadro 2: Informe de clasificación con puntuación de precisión de 0.8358





# 3.3 RED NEURONAL

```
# Características y variable objetivo
y = df_train_cleaned['Survived']
features = ['Pclass', 'Title', 'Embarked', 'Tipo_Fam', 'Ticket_len', 'Ticket_2letter']
X = df_train_cleaned[features]

# Dividimos los datos en entrenamiento y validación (70% entrenamiento, 30% validación)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.3, random_state = 42)

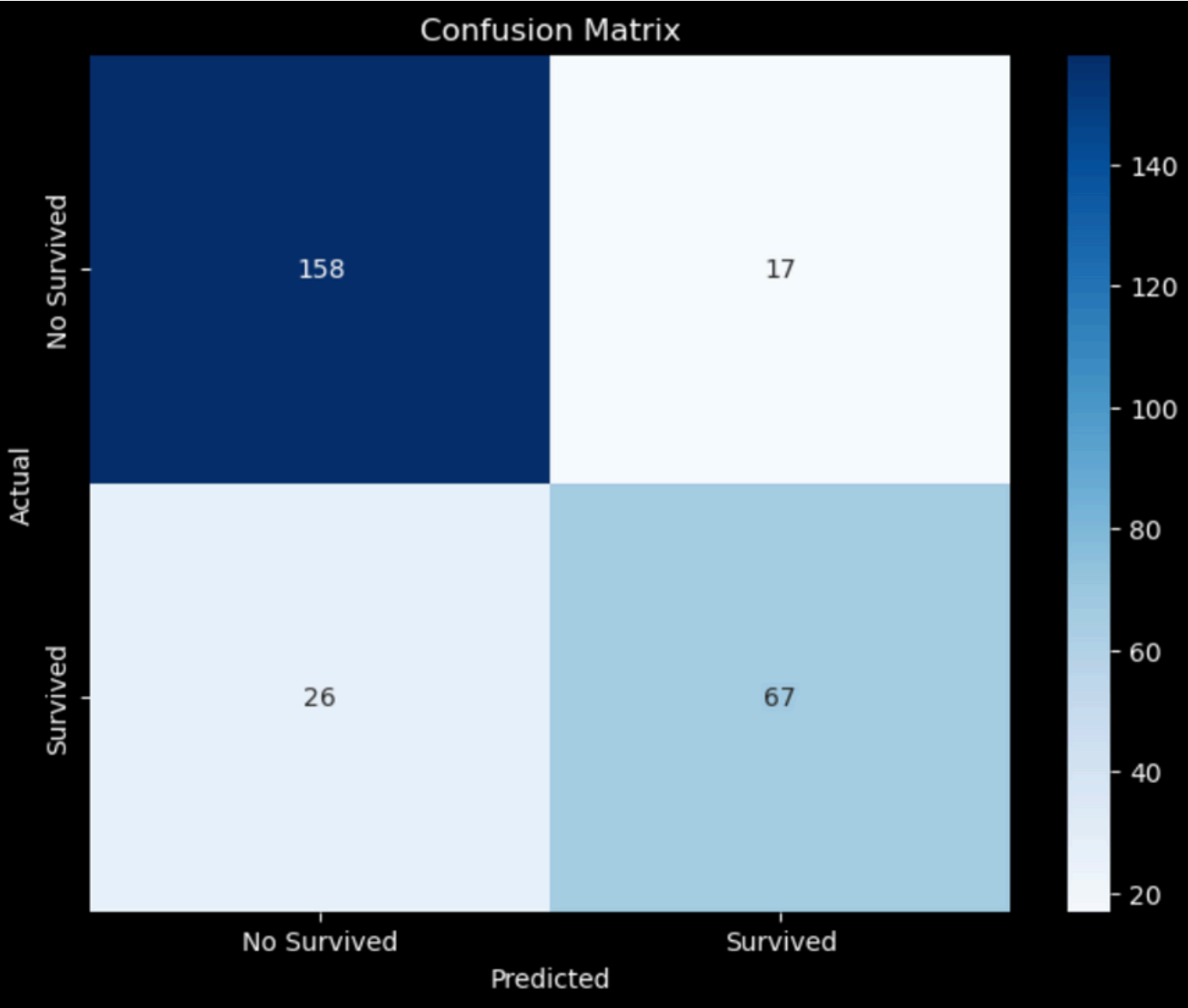
# Procesador para transformar a variables dummy y escalar características numéricas
preprocessor = ColumnTransformer(transformers = [
    ('onehot', OneHotEncoder(handle_unknown = 'ignore'), ['Pclass', 'Title', 'Embarked', 'Tipo_Fam', 'Ticket_2letter']),
    ('scaler', StandardScaler(), ['Ticket_len'])])

# Definimos el modelo utilizando una red neuronal (MLPClassifier) con la función de activación 'tanh'
modelo = Pipeline(steps = [
    ('preprocessor', preprocessor),
    ('model', MLPClassifier(random_state = 42,
                           max_iter = 1000, # Incrementamos el número de iteraciones
                           hidden_layer_sizes = (350, 150, 100), # Aumentamos el tamaño de las capas
                           activation = 'tanh',
                           alpha = 0.01, # Regularización
                           learning_rate_init = 0.01)))])

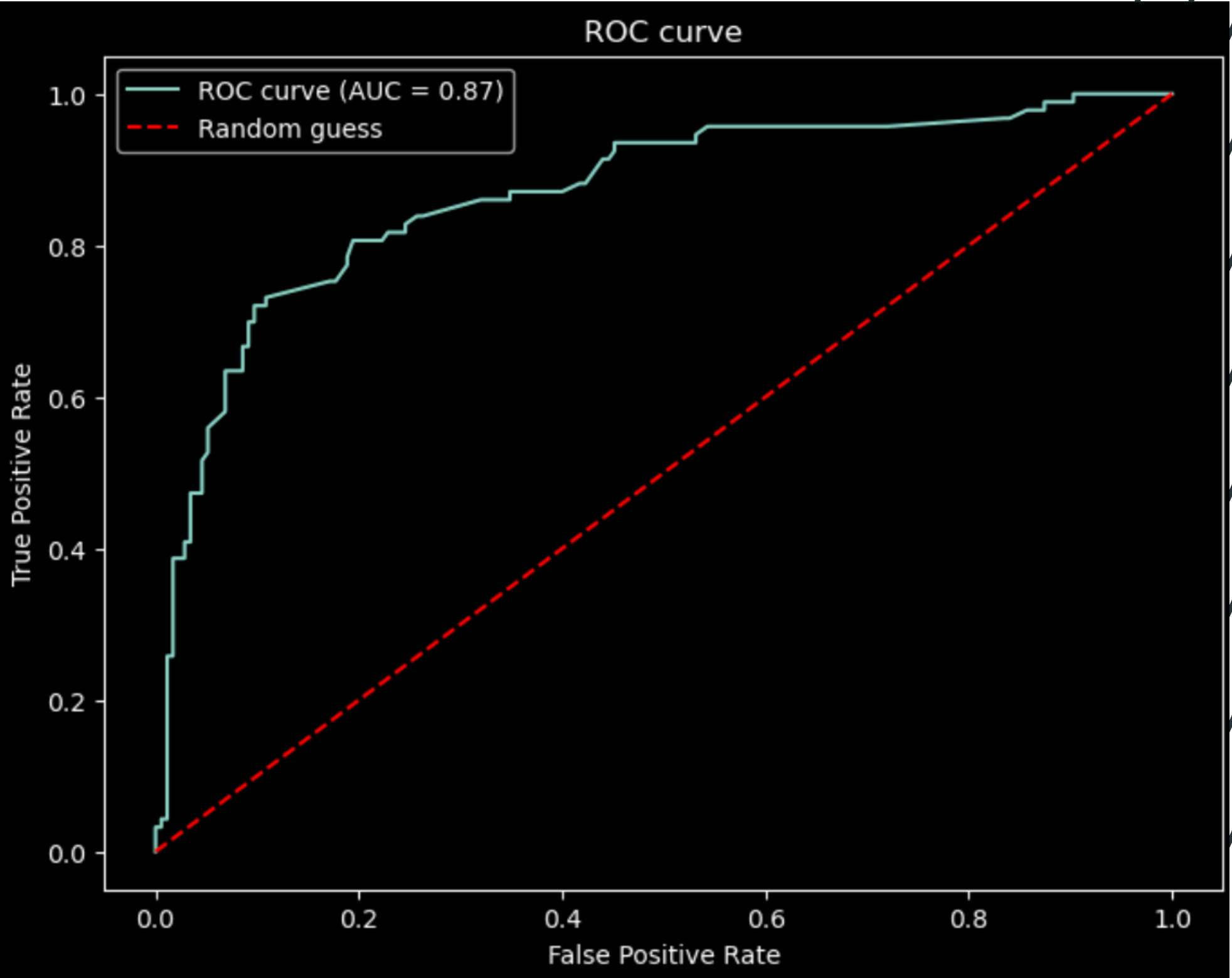
# Entrenamiento del modelo
modelo.fit(X_train, y_train)

# Predicciones en el conjunto de validación
y_pred = modelo.predict(X_val)
```

- 
- 
- 
- 
- 
- 
- 
-






	Precision	Recall	F1-Score	Support
Class 0	0.86	0.90	0.88	175
Class 1	0.80	0.72	0.76	93
Accuracy			0.84	268
Macro Avg	0.83	0.81	0.82	268
Weighted Avg	0.84	0.84	0.84	268



Cuadro 3: Informe de clasificación con puntuación de precisión de 0.8396

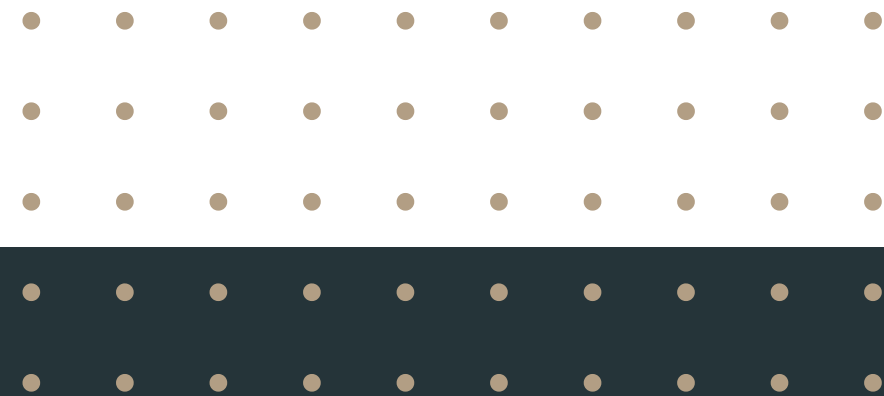
# TABLA DE COMPARACIÓN

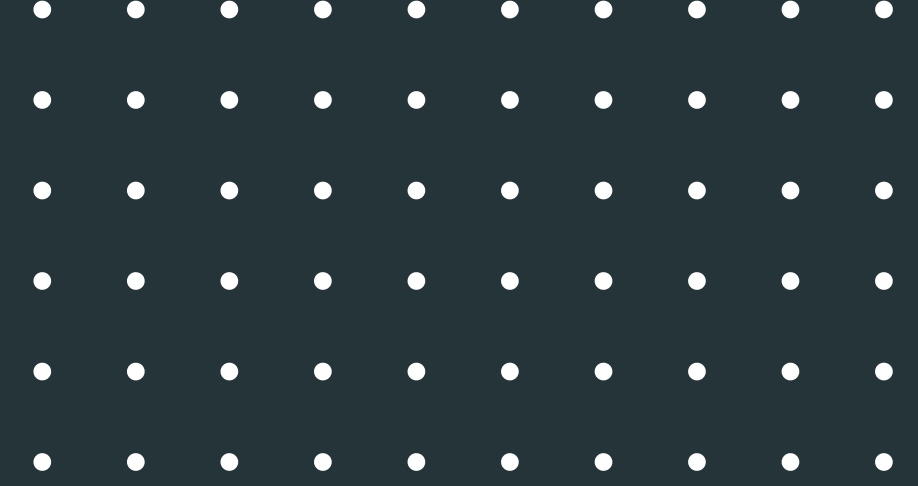
Modelo	Precisión ( %)
Regresión Logística	$\frac{326}{418} = 77,99 \%$
Random Forest	$\frac{335}{418} = 80,14 \%$
Red Neuronal	$\frac{336}{418} = 80,38 \%$

	<b>my_submission.csv</b> Complete · now	<b>0.77990</b>
	<b>my_submission.csv</b> Complete · 37s ago	<b>0.80143</b>
	<b>my_submission.csv</b> Complete · 1m ago	<b>0.80382</b>

05.

# CONCLUSIONES





A lo largo de este proyecto, se exploraron diferentes enfoques de modelado predictivo para estimar la probabilidad de supervivencia de los pasajeros del Titanic, empleando técnicas como Regresión Logística, Random Forest y Redes Neuronales. Cada modelo mostró sus propias fortalezas, destacando que la elección del método adecuado depende de la estructura de los datos y del propósito del análisis.

Mientras que la Regresión Logística permitió una interpretación clara de las relaciones entre las variables, Random Forest y Redes Neuronales destacaron por su capacidad para manejar interacciones no lineales y patrones complejos, logrando una mayor precisión. Este análisis no solo permitió mejorar los resultados predictivos, sino que subrayó la importancia de un procesamiento de datos adecuado y la evaluación de distintas estrategias de modelado.

A decorative grid of small, light brown dots arranged in five rows of ten, located in the top left corner of the white rounded rectangle.

**¡GRACIAS!**

A decorative line art element at the bottom of the white rounded rectangle. It consists of a large, thin, light brown arch on the left and a series of five concentric, smaller arches on the right, all rendered in the same light brown color.