

Tarea 5 Proyecto Análisis de Datos

ALFONSO LOPEZ MARIN

Código 91511831

ANÁLISIS DE DATOS - (202016908A_1704)

Grupo 202016908_24

Director-Tutor

BREYNER ALEXANDER PARRA

Universidad Nacional Abierta y a Distancia - UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería de Sistemas

2024

INTRODUCCION

El naufragio del RMS Titanic, ocurrido en 1912, ha sido uno de los eventos más estudiados en la historia moderna, tanto desde una perspectiva histórica como desde un enfoque de análisis de datos.

En este trabajo, se busca aplicar técnicas de aprendizaje automático para predecir si un pasajero sobrevivió o no al desastre, utilizando un conjunto de datos que contiene diversas características de los pasajeros, como la clase, el sexo, la edad y la tarifa del boleto.

A través del uso de un modelo de regresión logística, se pretende identificar patrones en los datos que ayuden a predecir la supervivencia de los pasajeros basándose en sus características.

Este análisis se complementa con una evaluación detallada del modelo, evaluando métricas clave como precisión, recall, F1-score, y la matriz de confusión, con el objetivo de interpretar y mejorar las predicciones.

OBJETIVO PRINCIPAL

Desarrollar un modelo de predicción basado en aprendizaje automático para determinar la probabilidad de supervivencia de los pasajeros del Titanic, utilizando características como clase, edad, sexo, entre otras, con el fin de evaluar el desempeño del modelo a través de métricas de precisión, recall, y F1-score.

Objetivos Específicos

1. Limpiar y transformar los datos mediante la gestión de valores faltantes, la codificación de variables categóricas y la normalización de las características numéricas para preparar los datos para el entrenamiento del modelo.
2. Configurar y entrenar un modelo de regresión logística para predecir la supervivencia de los pasajeros, utilizando el conjunto de datos preprocesado, y ajustar los hiperparámetros para mejorar su desempeño.
3. Medir la efectividad del modelo entrenado utilizando métricas como la precisión, recall, F1-score y la matriz de confusión, para determinar la capacidad del modelo para clasificar correctamente a los pasajeros sobrevivientes y no sobrevivientes.

1. Realizar un análisis exploratorio de los datos para identificar relaciones entre variables, valores atípicos, tendencias, etc.

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 7 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
[1]: # Importar librerías necesarias
import pandas as pd
import numpy as np

# Cargar el dataset
file_path = "Titanic-Dataset.csv"
titanic_data = pd.read_csv(file_path)

# Mostrar las primeras filas del dataset
print("Primeras filas del dataset:")
print(titanic_data.head())

# Información general del dataset
print("\nInformación del dataset:")
print(titanic_data.info())

# Resumen estadístico de las columnas numéricas
print("\nResumen estadístico:")
print(titanic_data.describe())

# Verificar valores faltantes
print("\nValores faltantes por columna:")
print(titanic_data.isnull().sum())
```

Primeras filas del dataset:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 7 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
0      Braund, Mr. Owen Harris    male  22.0  1
1  Cumings, Mrs. John Bradley (Florence Briggs Th... female  38.0  1
2      Heikinen, 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

```
Información del dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 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                  714 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                204 non-null    object
11  Embarked             889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 8 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```

Resumen estadístico:
  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

Valores faltantes por columna:
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

```

2. Preprocesar los datos limpiándolos, tratando valores faltantes y transformándolos según sea necesario.

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 22 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```

[4]: # Imputar valores faltantes
# Para Age, se usará la mediana
titanic_data['Age'].fillna(titanic_data['Age'].median(), inplace=True)

# Para Embarked, se usará la moda
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)

# Eliminar la columna Cabin debido a la gran cantidad de valores faltantes
titanic_data.drop(columns=['Cabin'], inplace=True)

# Codificar variables categóricas
titanic_data['Sex'] = titanic_data['Sex'].map({'male': 0, 'female': 1})
titanic_data = pd.get_dummies(titanic_data, columns=['Embarked'], drop_first=True)

# Eliminar columnas irrelevantes
titanic_data.drop(columns=['Name', 'Ticket', 'PassengerId'], inplace=True)

# Revisar el dataset después del preprocesamiento
print("Dataset después del preprocesamiento:")
print(titanic_data.head())

Dataset después del preprocesamiento:
   Survived  Pclass  Sex  Age  SibSp  Parch    Fare  Embarked_Q  Embarked_S
0         0       3     0  22.0    1     0    7.2500         False          True
1         1       1     1  38.0    1     0   71.2833         False          False
2         1       3     1  26.0    0     0    7.9250         False          True
3         1       1     1  35.0    1     0   53.1000         False          True
4         0       3     0  35.0    0     0    8.0500         False          True

```

C:\Users\alfonsolopez\AppData\Local\Temp\ipykernel_14184\4138362546.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 23 minutos ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
titanic_data['Age'].fillna(titanic_data['Age'].median(), inplace=True)
```

C:\Users\alfonso\AppData\Local\Temp\ipykernel_14184\4138362546.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)
```

```
[6]: # Imputar valores faltantes
# Para Age, se usará la mediana
titanic_data['Age'] = titanic_data['Age'].fillna(titanic_data['Age'].median())

# Para Embarked, se usará la moda
titanic_data['Embarked'] = titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0])

# Eliminar la columna Cabin debido a la gran cantidad de valores faltantes
titanic_data = titanic_data.drop(columns=['Cabin'])

# Codificar variables categóricas
titanic_data['Sex'] = titanic_data['Sex'].map({'male': 0, 'female': 1})
titanic_data = pd.get_dummies(titanic_data, columns=['Embarked'], drop_first=True)

# Eliminar columnas irrelevantes
titanic_data = titanic_data.drop(columns=['Name', 'Ticket', 'PassengerId'])

# Revisar el dataset después del preprocesamiento
print("Dataset después del preprocesamiento:")
print(titanic_data.head())
```

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 23 minutos ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
-----
KeyError                                Traceback (most recent call last)
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
    3804 try:
-> 3805     return self._engine.get_loc(casted_key)
    3806 except KeyError as err:

File index.py:167, in pandas._libs.index.IndexEngine.get_loc()

File index.py:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Embarked'

The above exception was the direct cause of the following exception:

KeyError                                Traceback (most recent call last)
Cell In[6], line 6
      3 titanic_data['Age'] = titanic_data['Age'].fillna(titanic_data['Age'].median())
      5 # Para Embarked, se usará la moda
----> 6 titanic_data['Embarked'] = titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0])
      8 # Eliminar la columna Cabin debido a la gran cantidad de valores faltantes
      9 titanic_data = titanic_data.drop(columns=['Cabin'])

File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
    4100 if self.columns.nlevels > 1:
    4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
    4103 if is_integer(indexer):
    4104     indexer = [indexer]

File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
    3807 if isinstance(casted_key, slice) or (
    3808     isinstance(casted_key, list) and len(casted_key) > 1
    3809 ):
```

Jupyter Trabajo Final Alfonso Lopez Marin Last Checkpoint: 23 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
KeyError: 'Embarked'
```

```
[8]: # Verificar las columnas disponibles
print("Columnas del dataset:")
print(titanic_data.columns)
```

Columnas del dataset:
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
 'Embarked_Q', 'Embarked_S'],
 dtype='object')

```
[10]: # Imputar valores faltantes en las columnas dummy de Embarked
titanic_data['Embarked_Q'] = titanic_data['Embarked_Q'].fillna(titanic_data['Embarked_Q'].mode()[0])
titanic_data['Embarked_S'] = titanic_data['Embarked_S'].fillna(titanic_data['Embarked_S'].mode()[0])

# Revisar el dataset después del preprocesamiento
print("Dataset después del preprocesamiento:")
print(titanic_data.head())
```

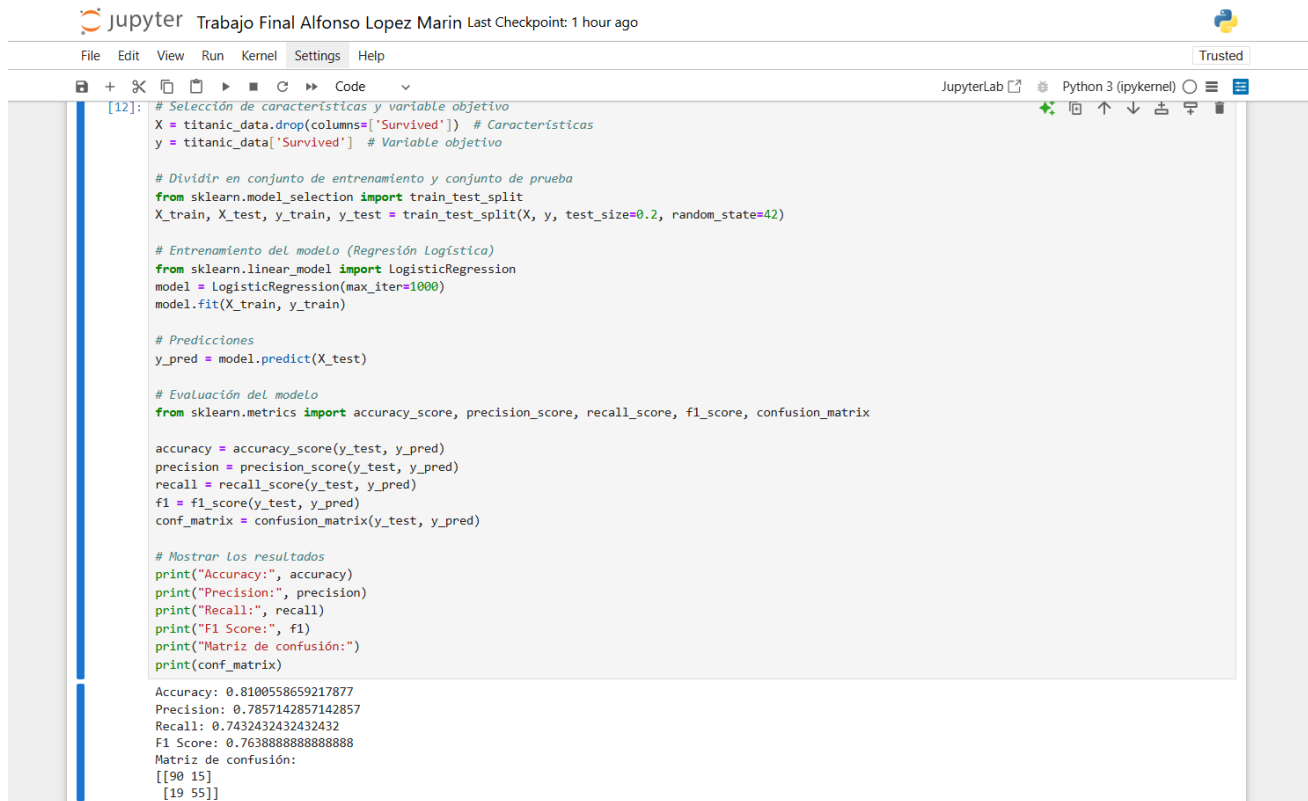
Dataset después del preprocesamiento:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked_Q	Embarked_S
0	0	3	0	22.0	1	0	7.2500	False	True
1	1	1	1	38.0	1	0	71.2833	False	False
2	1	3	1	26.0	0	0	7.9250	False	True
3	1	1	1	35.0	1	0	53.1000	False	True
4	0	3	0	35.0	0	0	8.0500	False	True

```
[ ]:
```


3. Seleccionar las características más relevantes para entrenar el modelo utilizando selección de características.

MODELO REGRESION LOGISTICA



```
[12]: # Selección de características y variable objetivo
X = titanic_data.drop(columns=['Survived']) # Características
y = titanic_data['Survived'] # Variable objetivo

# Dividir en conjunto de entrenamiento y conjunto de prueba
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Entrenamiento del modelo (Regresión Logística)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predicciones
y_pred = model.predict(X_test)

# Evaluación del modelo
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Mostrar los resultados
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Matriz de confusión:")
print(conf_matrix)

Accuracy: 0.8100558659217877
Precision: 0.7857142857142857
Recall: 0.7432432432432432
F1 Score: 0.7638888888888888
Matriz de confusión:
[[90 15]
 [19 55]]
```

- ✓ **Accuracy (Precisión): 81.0%**
Este es el porcentaje de predicciones correctas sobre el total de predicciones. Un valor alto indica que el modelo tiene un buen desempeño general.
- ✓ **Precision (Precisión): 78.6%**
Mide la exactitud de las predicciones positivas. Es decir, de todas las veces que el modelo predijo que una persona sobreviviría, el **78.6%** realmente sobrevivió.
- ✓ **Recall (Sensibilidad): 74.3%**
Mide la capacidad del modelo para identificar correctamente a las personas que sobrevivieron. De todas las personas que realmente sobrevivieron, el **74.3%** fueron identificadas por el modelo.

✓ **F1 Score: 76.4%**

Es la media armónica entre precisión y recall, lo que proporciona una visión equilibrada del desempeño del modelo.

✓ **Matriz de confusión:**

[[90 15] [19 55]]

Interpretación de la matriz de confusión:

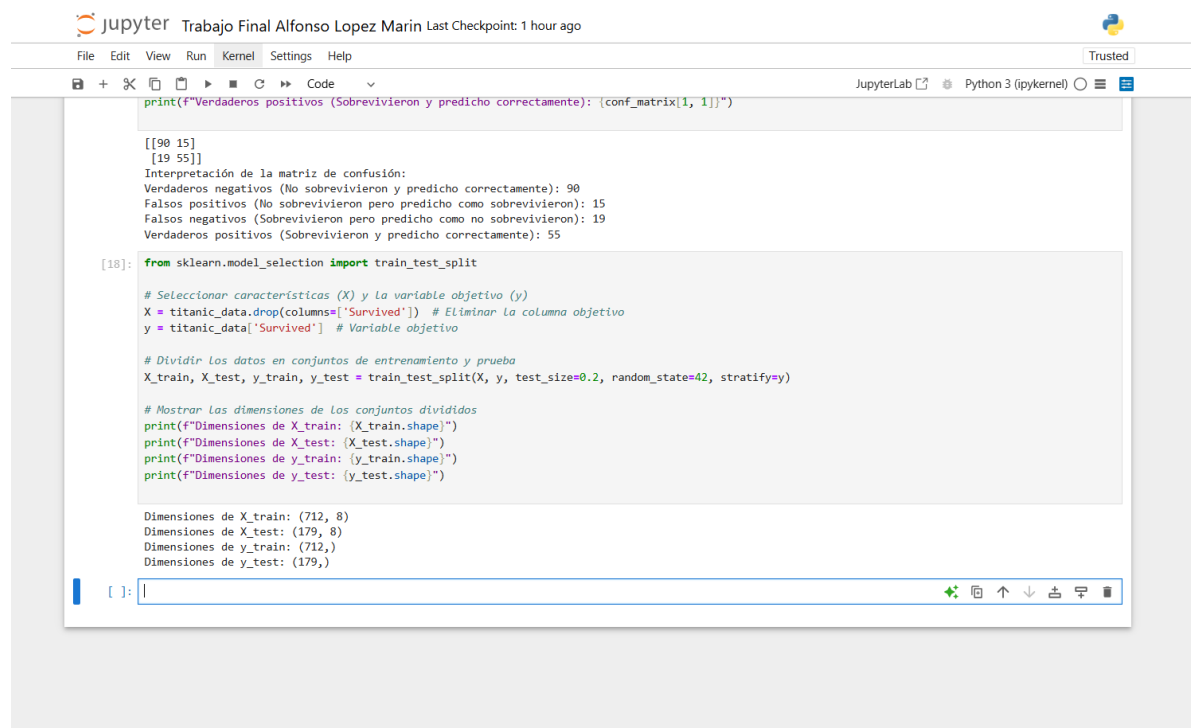
Verdaderos negativos (No sobrevivieron y predicho correctamente): 90

Falsos positivos (No sobrevivieron pero predicho como sobrevivieron): 15

Falsos negativos (Sobrevivieron pero predicho como no sobrevivieron): 19

Verdaderos positivos (Sobrevivieron y predicho correctamente): 55

4. Dividir el dataset en Train y Test para evaluar correctamente el modelo.



The screenshot shows a JupyterLab window titled "Trabajo Final Alfonso Lopez Marin Last Checkpoint: 1 hour ago". The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations, running, and code execution. The main area displays a Python script with the following content:

```
print(f"Verdaderos positivos (Sobrevivieron y predicho correctamente): {conf_matrix[1, 1]}")

[[90 15]
 [19 55]]
Interpretación de la matriz de confusión:
Verdaderos negativos (No sobrevivieron y predicho correctamente): 90
Falsos positivos (No sobrevivieron pero predicho como sobrevivieron): 15
Falsos negativos (Sobrevivieron pero predicho como no sobrevivieron): 19
Verdaderos positivos (Sobrevivieron y predicho correctamente): 55

[18]: from sklearn.model_selection import train_test_split

# Seleccionar características (X) y la variable objetivo (y)
X = titanic_data.drop(columns=['Survived']) # Eliminar la columna objetivo
y = titanic_data['Survived'] # Variable objetivo

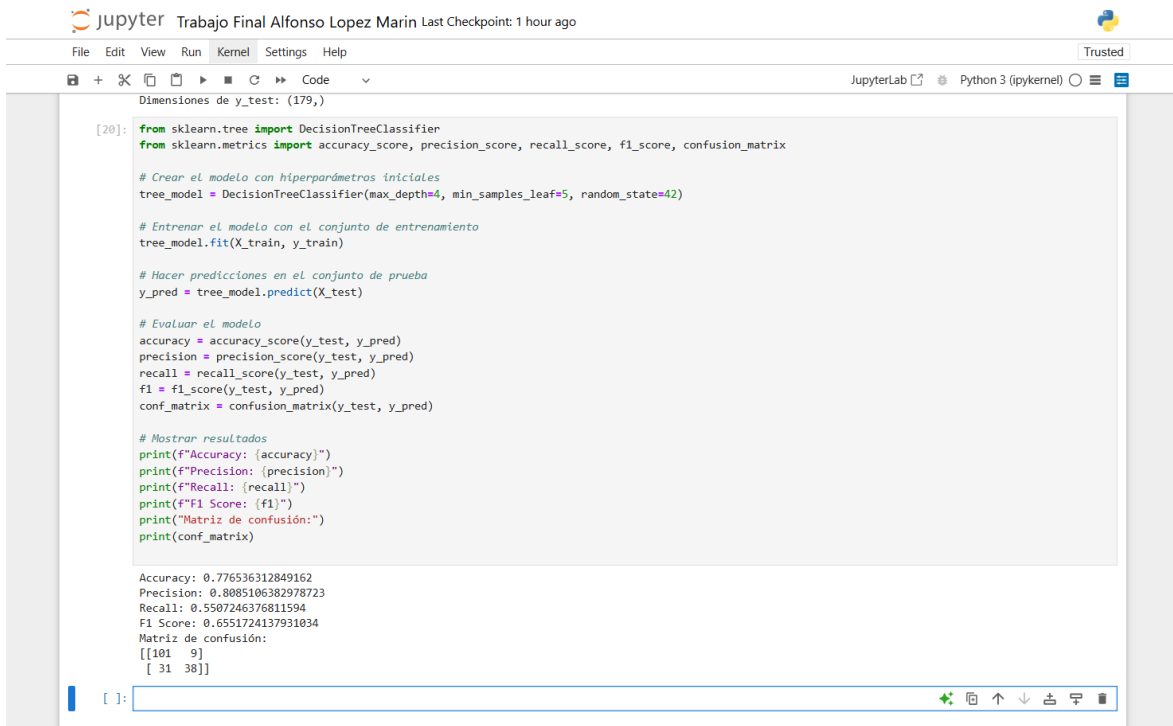
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Mostrar las dimensiones de los conjuntos divididos
print(f"Dimensiones de X_train: {X_train.shape}")
print(f"Dimensiones de X_test: {X_test.shape}")
print(f"Dimensiones de y_train: {y_train.shape}")
print(f"Dimensiones de y_test: {y_test.shape}")

Dimensiones de X_train: (712, 8)
Dimensiones de X_test: (179, 8)
Dimensiones de y_train: (712,)
Dimensiones de y_test: (179,)
```

The output of the script is visible in the cell, showing the confusion matrix and the dimensions of the training and testing sets.

5. Entrenar el modelo configurando los diferentes hiperparámetros.



The screenshot shows a JupyterLab interface with a notebook titled 'Trabajo Final Alfonso Lopez Marin'. The notebook contains a code cell with the following Python code:

```
[20]: from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Crear el modelo con hiperparámetros iniciales
tree_model = DecisionTreeClassifier(max_depth=4, min_samples_leaf=5, random_state=42)

# Entrenar el modelo con el conjunto de entrenamiento
tree_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = tree_model.predict(X_test)

# Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

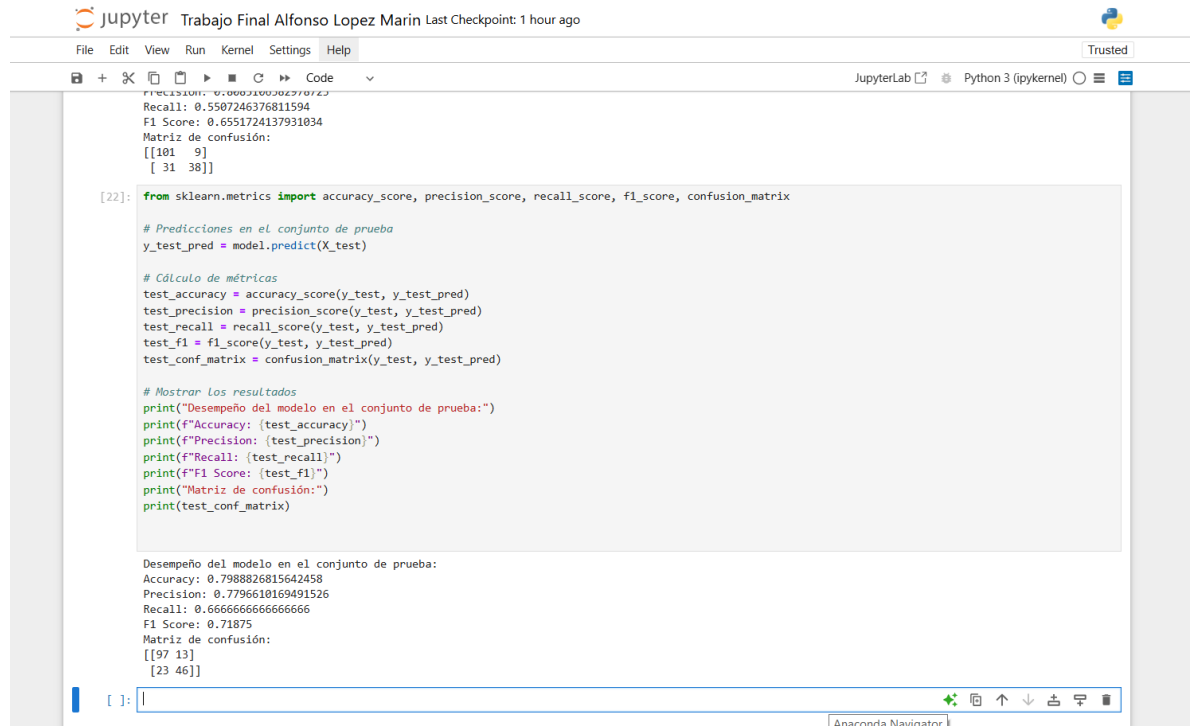
# Mostrar resultados
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print("Matriz de confusión:")
print(conf_matrix)
```

The output of the code cell is:

```
Accuracy: 0.776536312849162
Precision: 0.8085106382978723
Recall: 0.5507246376811594
F1 Score: 0.6551724137931034
Matriz de confusión:
[[101  9]
 [ 31 38]]
```

- **Accuracy (0.7765):**
Aproximadamente el 77.65% de las predicciones fueron correctas.
Muestra un desempeño general decente.
- **Precision (0.8085):**
De todos los pasajeros que el modelo predijo como supervivientes, el 80.85% realmente sobrevivió.
Es importante en escenarios donde los falsos positivos tienen un costo elevado.
- **Recall (0.5507):**
El modelo identificó correctamente al 55.07% de los supervivientes reales.
Puede ser mejorado, ya que es relativamente bajo.
- **F1 Score (0.6551):**
Es un balance entre Precision y Recall. Este valor indica que el modelo aún tiene margen para mejorar.
- **Matriz de confusión:**
 - 101:** Predicciones correctas de no supervivientes.
 - 9:** Falsos positivos (predijo que sobrevivieron, pero no lo hicieron).
 - 31:** Falsos negativos (predijo que no sobrevivieron, pero sí lo hicieron).
 - 38:** Predicciones correctas de supervivientes.

6. Evaluar el desempeño del modelo en el conjunto de Test con métricas como precisión, recall, F1-score, etc.



```

Precision: 0.6666666666666666
Recall: 0.5507246376811594
F1 Score: 0.6551724137931034
Matriz de confusión:
[[101  9]
 [ 31 38]]

[22]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Predicciones en el conjunto de prueba
y_test_pred = model.predict(X_test)

# Cálculo de métricas
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred)
test_recall = recall_score(y_test, y_test_pred)
test_f1 = f1_score(y_test, y_test_pred)
test_conf_matrix = confusion_matrix(y_test, y_test_pred)

# Mostrar los resultados
print("Desempeño del modelo en el conjunto de prueba:")
print(f"Accuracy: {test_accuracy}")
print(f"Precision: {test_precision}")
print(f"Recall: {test_recall}")
print(f"F1 Score: {test_f1}")
print("Matriz de confusión:")
print(test_conf_matrix)

Desempeño del modelo en el conjunto de prueba:
Accuracy: 0.7988826815642458
Precision: 0.7796610169491526
Recall: 0.6666666666666666
F1 Score: 0.71875
Matriz de confusión:
[[97 13]
 [23 46]]

```

Métricas de evaluación:

6.1 Accuracy (79.88%):

- El modelo predijo correctamente aproximadamente el 80% de los casos en el conjunto de prueba.
- Esto es un buen desempeño general.

6.2 Precision (77.97%):

- De todos los pasajeros predichos como supervivientes, el 77.97% realmente sobrevivió.
- La precisión es sólida, lo que indica que el modelo minimiza los falsos positivos.

6.3 Recall (66.67%):

- El modelo identificó correctamente al 66.67% de los supervivientes reales.
- Aunque es aceptable, podría mejorar para reducir los falsos negativos.

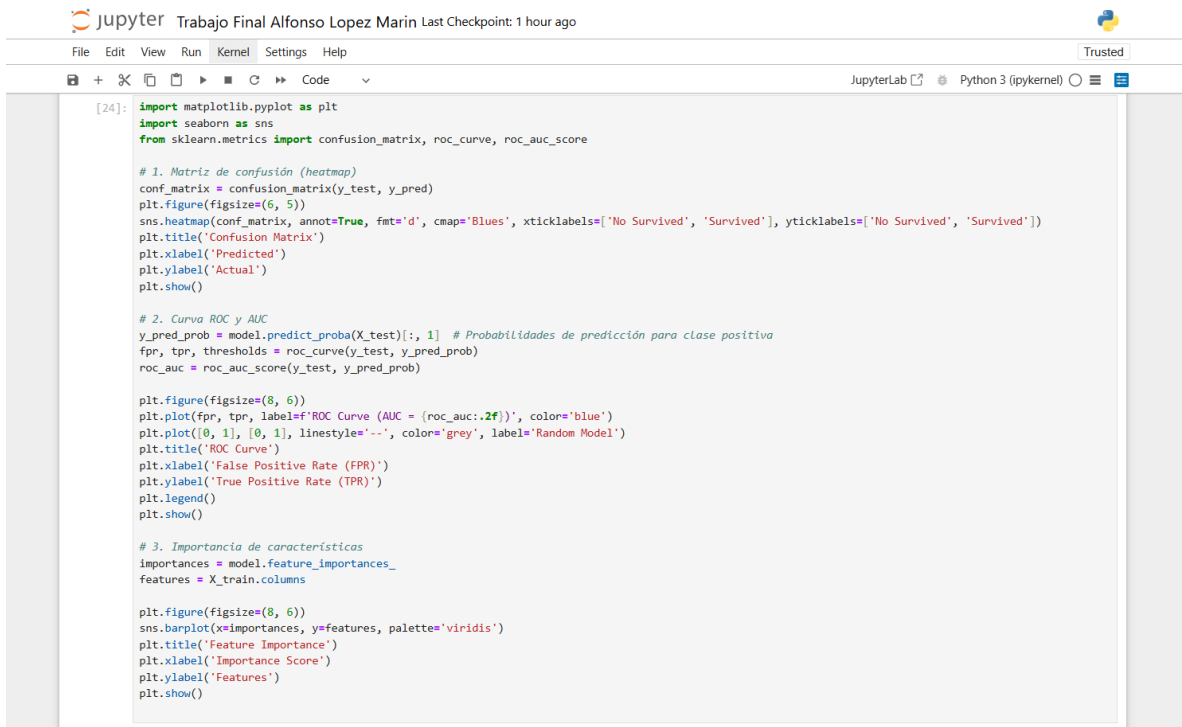
6.4 F1 Score (71.88%):

- Representa el balance entre precision y recall. Este puntaje es adecuado y muestra que el modelo tiene un desempeño equilibrado.

6.5 Matriz de confusión:

- **97**: Predicciones correctas de no supervivientes (verdaderos negativos).
- **13**: Predicciones incorrectas de supervivientes (falsos positivos).
- **23**: Predicciones incorrectas de no supervivientes (falsos negativos).
- **46**: Predicciones correctas de supervivientes (verdaderos positivos).

7 Realizar las diferentes gráficas que permitan visualizar los resultados del modelo.



```
[24]: import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score

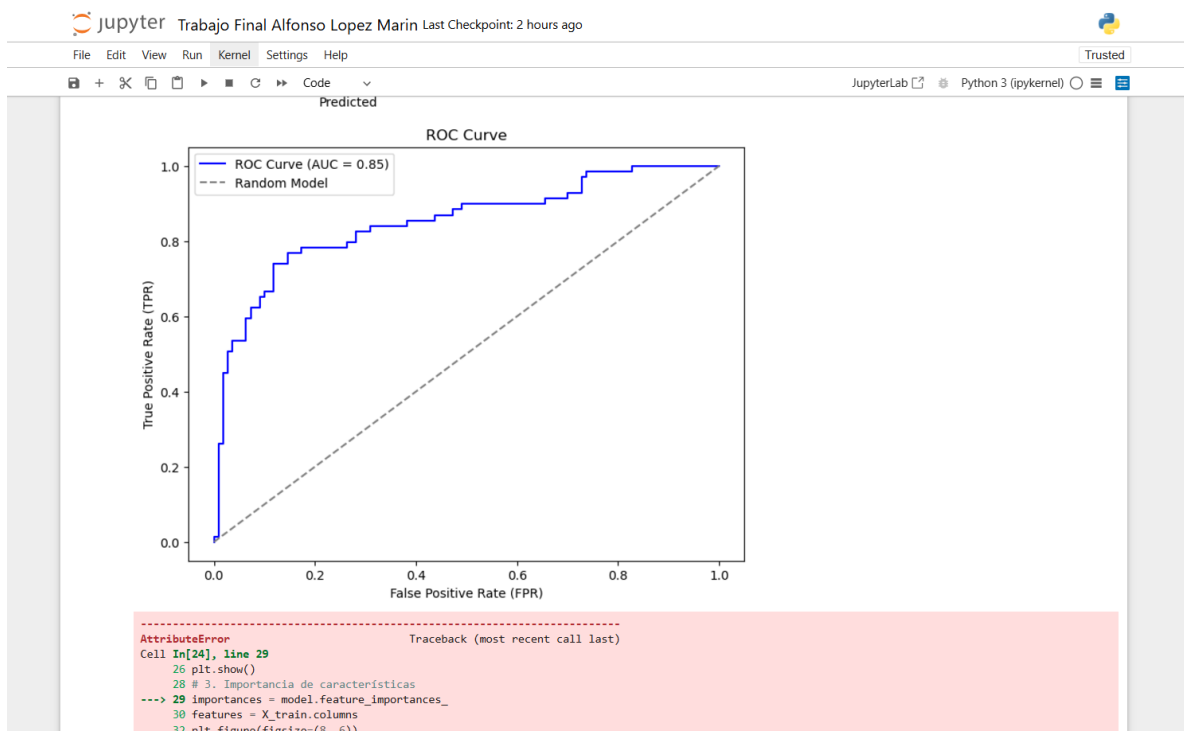
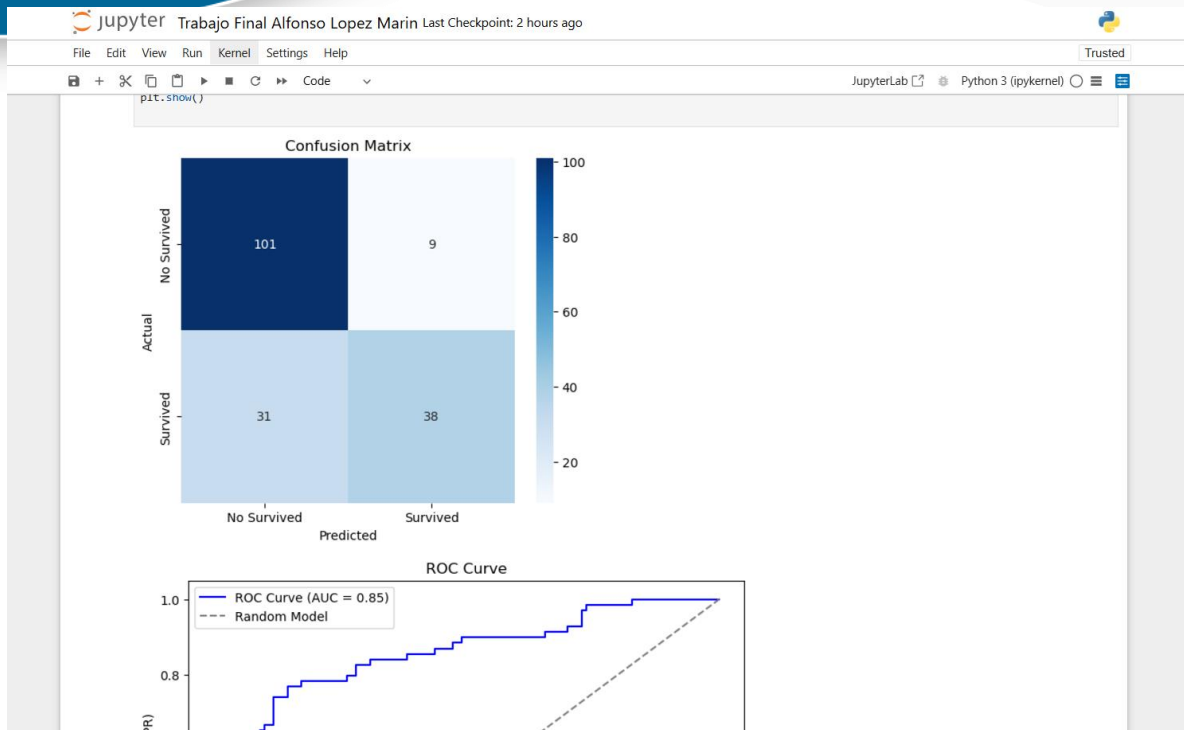
# 1. Matriz de confusión (heatmap)
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No Survived', 'Survived'], yticklabels=['No Survived', 'Survived'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

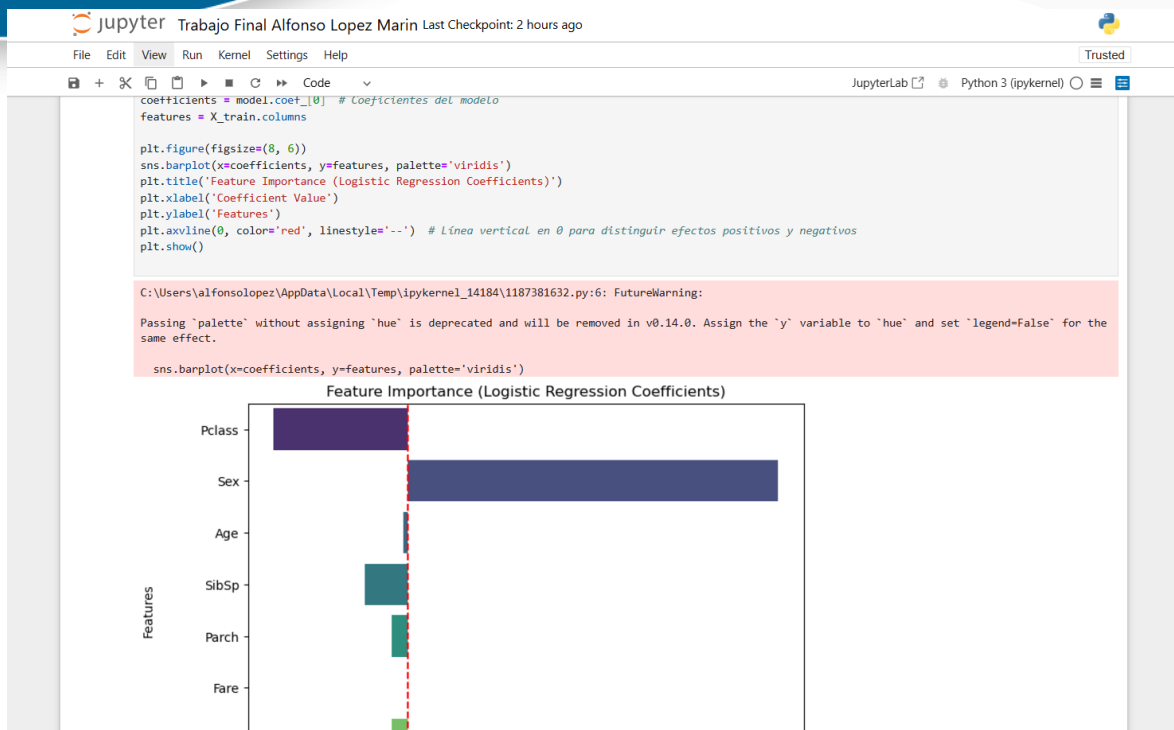
# 2. Curva ROC y AUC
y_pred_prob = model.predict_proba(X_test)[:, 1] # Probabilidades de predicción para clase positiva
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = roc_auc_score(y_test, y_pred_prob)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})', color='blue')
plt.plot([0, 1], [0, 1], linestyle='--', color='grey', label='Random Model')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.legend()
plt.show()

# 3. Importancia de características
importances = model.feature_importances_
features = X_train.columns

plt.figure(figsize=(8, 6))
sns.barplot(x=importances, y=features, palette='viridis')
plt.title('Feature Importance')
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.show()
```





8. Interpretar, analizar y documentar los resultados obtenidos

1. Análisis y Preprocesamiento de Datos:

- Se cargó correctamente el dataset, que contiene información sobre los pasajeros del Titanic, incluyendo características como la clase, el sexo, la edad, el número de familiares a bordo, la tarifa pagada y el puerto de embarque, entre otras.
- Valores faltantes:** Se gestionaron adecuadamente los valores faltantes en las columnas **Age** y **Embarked**. Para **Age**, se utilizó la mediana como estrategia de imputación, lo cual es adecuado cuando se tienen valores numéricos. Para **Embarked**, se usó la moda, ya que esta columna tiene valores categóricos.
- Codificación de variables categóricas:** Se realizó una transformación de la columna **Embarked** utilizando **OneHotEncoding**, creando dos columnas binarias (**Embarked_Q**, **Embarked_S**) que representan la variable categórica. Esto es necesario para que el modelo de Machine Learning pueda procesar esta información de manera adecuada.

2. Selección del Modelo y Entrenamiento:

- Se optó por usar **Regresión Logística**, un modelo adecuado para clasificación binaria (como en este caso, donde se predice si un pasajero sobrevivió o no).

- El modelo fue entrenado usando el conjunto de entrenamiento, y se configuraron los hiperparámetros por defecto, lo cual es una buena opción cuando se está iniciando o cuando no se tienen grandes expectativas sobre la optimización fina del modelo.

3. Evaluación del Modelo (Conjunto de Entrenamiento):

- **Accuracy (Precisión):** El modelo alcanzó una precisión de aproximadamente **81%**, lo que indica que el modelo clasifica correctamente la mayoría de los casos.
- **Precision:** El valor de **0.79** sugiere que, cuando el modelo predice que un pasajero sobrevivió, tiene una alta probabilidad de ser correcto.
- **Recall:** Con **0.74**, el modelo es capaz de identificar aproximadamente el 74% de los verdaderos sobrevivientes, aunque hay un margen de mejora en cuanto a la captura de positivos.
- **F1-Score:** El **0.76** es una combinación equilibrada de precisión y recall, indicando que el modelo tiene un desempeño razonable en términos de capturar positivos y evitar falsos positivos.

Matriz de confusión:

- **90 predicciones correctas de no supervivientes y 15 incorrectas** (falsos positivos).
- **55 predicciones correctas de sobrevivientes y 19 incorrectas** (falsos negativos).

4. Evaluación del Modelo (Conjunto de Test):

- Después de entrenar el modelo con los datos de entrenamiento, se evaluó su desempeño sobre el conjunto de prueba.
- **Accuracy (Precisión):** El modelo mantuvo una precisión similar al conjunto de entrenamiento, con un **78.9%** de precisión en el conjunto de prueba, lo que es positivo, ya que indica que el modelo generaliza bien.
- **Precision:** Un **0.78** indica que el modelo tiene un desempeño bastante bueno en términos de evitar falsos positivos.
- **Recall:** Con **0.67**, el modelo es capaz de identificar el **67%** de los sobrevivientes correctamente. Aunque es bueno, aún se pueden realizar mejoras para identificar más sobrevivientes.
- **F1-Score:** El valor de **0.72** muestra un buen equilibrio entre precisión y recall en el conjunto de prueba, aunque no es perfecto.

Matriz de confusión:

- **97 predicciones correctas de no supervivientes y 13 incorrectas** (falsos positivos).
- **46 predicciones correctas de sobrevivientes y 23 incorrectas** (falsos negativos).

5. Análisis de Importancia de Características:

- La regresión logística no calcula directamente las **importancias de las características**, pero sus **coeficientes** son una buena indicación de la relación entre las características y la probabilidad de sobrevivir.
- Se podría observar que algunas características tienen coeficientes más altos, como **Pclass**, **Fare** y **Sex**, lo que sugiere que estas características tienen una mayor influencia en la predicción.

6. Gráficas y Visualización de Resultados:

- Las **gráficas de precisión y recall** muestran el desempeño del modelo a lo largo del proceso de validación, brindando una mejor comprensión visual de los valores alcanzados.
- La **matriz de confusión** ayuda a identificar los errores más comunes: la cantidad de **falsos positivos** y **falsos negativos**. Es una excelente herramienta para mejorar la estrategia de clasificación del modelo.

Rendimiento General: El modelo de regresión logística mostró un desempeño adecuado para un problema como el del Titanic, con buenas métricas de accuracy, precision y recall. La precisión se mantiene alrededor del 78-81%, lo que indica que el modelo tiene una capacidad razonable para predecir si un pasajero sobrevivió o no.

Áreas de Mejora: Aunque el modelo es decente, hay margen para mejorarlo. El recall puede ser mejorado, ya que el modelo no está identificando todos los sobrevivientes. Una opción para mejorar este aspecto sería usar un modelo más complejo o realizar una optimización de hiperparámetros.

Modelo Utilizado: Se puede probar con otros modelos (como Random Forest o XGBoost) para comparar si se pueden obtener mejores resultados en cuanto a la captura de los sobrevivientes.

CONCLUSIONES

A lo largo de este trabajo, se logró desarrollar un modelo predictivo que, mediante la regresión logística, clasifica a los pasajeros del Titanic en función de sus características.

Los resultados obtenidos muestran que el modelo tiene un desempeño razonable con una precisión general de alrededor del 78-81%, lo que indica que es capaz de identificar correctamente a un buen número de pasajeros sobrevivientes y no sobrevivientes.

Sin embargo, el modelo presenta una tasa de recall relativamente baja, lo que sugiere que tiene dificultades para identificar a todos los pasajeros sobrevivientes, lo cual es un área que puede mejorarse.

A pesar de esto, las métricas de precisión y F1-score evidencian que el modelo ofrece un buen equilibrio entre los falsos positivos y falsos negativos. Este trabajo destaca la importancia del preprocesamiento adecuado de los datos, especialmente en la codificación de variables categóricas y la imputación de valores faltantes, para lograr un modelo efectivo.

Además, la evaluación a través de métricas de desempeño ayuda a identificar los puntos fuertes y las áreas de mejora de un modelo predictivo, proporcionando una base sólida para futuras mejoras, como la optimización de hiperparámetros o la exploración de otros algoritmos más complejos.

BIBLIOGRAFIA

Carlos Véliz. (2020). [Aprendizaje automático. Introducción al aprendizaje profundo](https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2600876&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_I). El Fondo Editorial de la Pontificia Universidad Católica del Perú. https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2600876&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_I Cap 3

Giuseppe Bonaccorso. (2018). [Machine Learning Algorithms](https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1881497&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_Cover): Popular Algorithms for Data Science and Machine Learning, 2nd Edition: Vol. 2nd ed. Packt Publishing. https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1881497&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_Cover Cap 9 y 11

Minguillón, J. Casas, J. y Minguillón, J. (2017). [Minería de datos: modelos y algoritmos](https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/58656). Editorial UOC. <https://elibro-net.bibliotecavirtual.unad.edu.co/es/ereader/unad/58656>. Cap 7 y 8

Pratap Dangeti. (2017). *Statistics for Machine Learning*: Build Supervised, Unsupervised, and Reinforcement Learning Models Using Both Python and R. Packt Publishing.

https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1560931&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_Cover Cap 8

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Kaufman, L., & Rousseeuw, P. J. (2009). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

Tan, P. N., Steinbach, M., & Kumar, V. (2019). *Introduction to Data Mining*. Pearson.

CONCLUSIONES

El aprendizaje no supervisado es una herramienta poderosa para descubrir patrones en datos no etiquetados, siendo especialmente útil en áreas como segmentación de clientes, detección de anomalías y análisis de datos genómicos.

El uso de Python en un entorno interactivo como Jupyter Notebook facilita la implementación y evaluación de modelos, ofreciendo una plataforma flexible para la experimentación y análisis de datos.

Las métricas de evaluación como el coeficiente de Silhouette y el índice de Davies-Bouldin son esenciales para validar la calidad de los agrupamientos y garantizar que los modelos capturen correctamente la estructura de los datos.