## Modelación estadística

Vamos a elaborar un tutorial más detallado sobre la creación de un modelo logístico para predecir la supervivencia en el Titanic.

### Tutorial: Crear un Modelo Logístico para el Titanic con pandas, seaborn y scikit-learn

#### 1. **Preparación del Entorno**

Primero, instala las bibliotecas necesarias:

pip install pandas seaborn scikit-learn

#### 2. **Carga y Exploración Inicial de Datos**

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Carga los datos  
data = pd.read\_csv('train.csv')  
print(data.head())

**Explicación:** Aquí simplemente cargamos el conjunto de datos y visualizamos las primeras filas para tener una idea general de la estructura de los datos.

#### 3. **Análisis Exploratorio de Datos (EDA)**

# Ver la distribución de los que sobrevivieron vs los que no  
sns.countplot(data['Survived'])  
plt.show()  
  
# Ver la distribución de supervivencia por género  
sns.countplot(data['Sex'], hue=data['Survived'])  
plt.show()  
  
# Distribución de la edad de los pasajeros  
sns.histplot(data['Age'], bins=30, kde=True)  
plt.show()  
  
# ... Y así sucesivamente con otras variables ...

**Explicación:** El EDA nos permite visualizar y comprender mejor los datos. Al observar cómo se distribuyen las variables, podemos tomar decisiones informadas sobre cómo tratar las características y qué técnicas aplicar.

#### 4. **Preprocesamiento de Datos**

# Manejo de datos faltantes  
from sklearn.impute import KNNImputer  
  
imputer = KNNImputer(n\_neighbors=5)  
data[['Age']] = imputer.fit\_transform(data[['Age']])  
  
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)  
  
# Convertir datos categóricos a numéricos  
data = pd.get\_dummies(data, columns=['Embarked', 'Sex'], drop\_first=True)  
  
# Selección de características  
features = ['Pclass', 'Sex\_male', 'Age', 'SibSp', 'Parch', 'Embarked\_Q', 'Embarked\_S']  
X = data[features]  
y = data['Survived']

**Explicación:** Aquí tratamos los datos faltantes usando un imputador basado en KNN. Luego, convertimos las variables categóricas a un formato numérico utilizando one-hot encoding.

#### 5. **División de Datos**

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)

**Explicación:** Es fundamental dividir los datos en conjuntos de entrenamiento y prueba para validar el desempeño del modelo en datos no vistos previamente.

#### 6. **Creación y Entrenamiento del Modelo**

from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score, classification\_report  
  
# Inicializa el modelo  
model = LogisticRegression(max\_iter=500)  
  
# Entrena el modelo  
model.fit(X\_train, y\_train)  
  
# Predicciones  
y\_pred = model.predict(X\_test)  
  
# Evalúa el modelo  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Accuracy: {accuracy \* 100:.2f}%")  
print(classification\_report(y\_test, y\_pred))

**Explicación:** Aquí entrenamos un modelo de regresión logística y luego lo evaluamos utilizando el conjunto de prueba. classification\_report proporciona métricas adicionales como precisión, recall y F1-score.

#### 7. **Optimización del Modelo**

Para mejorar el desempeño, puedes considerar:

* **Ingeniería de características**: Crea nuevas características o transforma las existentes.
* **Selección de características**: Usa técnicas como RFE para reducir el número de características.
* **Afinación de hiperparámetros**: Usa GridSearchCV o RandomizedSearchCV para encontrar los mejores hiperparámetros para tu modelo.

Este tutorial detallado proporciona un enfoque paso a paso para construir un modelo de regresión logística para el conjunto de datos del Titanic. Es solo un punto de partida y siempre hay espacio para mejorar y experimentar con diferentes técnicas y enfoques.