**Ejemplo aplicación regulación ISO 42001 código fuente**

**Step1**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/titanic/train.csv"

data = pd.read\_csv(url)

data = data.drop(["Name", "Ticket", "Cabin"], axis=1).dropna()

data = pd.get\_dummies(data, drop\_first=True)

X = data.drop("Survived", axis=1)

y = data["Survived"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step2**

from sklearn.ensemble import RandomForestClassifier

modelo\_sin\_regulacion = RandomForestClassifier(n\_estimators=100, random\_state=42)

modelo\_sin\_regulacion.fit(X\_train, y\_train)

**Step 3 Without regulation**

import shap

# Crear el explicador

explainer = shap.TreeExplainer(modelo\_sin\_regulacion)

# Obtener los valores SHAP

shap\_values = explainer.shap\_values(X\_test)

# Convertir X\_test a DataFrame con nombres de columnas

X\_test\_df = pd.DataFrame(X\_test, columns=X.columns)

# Validar si shap\_values es una lista (clasificación binaria)

if isinstance(shap\_values, list):

shap\_for\_plot = shap\_values[1] # clase positiva

else:

shap\_for\_plot = shap\_values

# Graficar con SHAP

shap.summary\_plot(shap\_for\_plot, X\_test\_df)

import numpy as np

# Importancia media de cada variable

importancia\_sin\_regulacion = np.abs(shap\_values[1]).mean(axis=0)

importancia\_df\_sin = pd.DataFrame({

"variable": X.columns,

"importancia": importancia\_sin\_regulacion

}).sort\_values(by="importancia", ascending=False)

print("Importancia del modelo sin regulación:")

print(importancia\_df\_sin)

***Step 4. With regulation***

# Importar librerías necesarias

import pandas as pd

import shap

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

# Cargar y preparar datos (eliminando variables irrelevantes como PassengerId)

url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/titanic/train.csv"

data = pd.read\_csv(url)

# Eliminar columnas irrelevantes o que generan ruido

data = data.drop(["PassengerId", "Name", "Ticket", "Cabin"], axis=1).dropna()

# Convertir variables categóricas a numéricas

data = pd.get\_dummies(data, drop\_first=True)

# Separar variables predictoras y variable objetivo

X = data.drop("Survived", axis=1)

y = data["Survived"]

# Dividir datos en entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Escalado de datos (buena práctica en IA regulada)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Entrenar el modelo con buenas prácticas

modelo\_regulado = RandomForestClassifier(n\_estimators=100, random\_state=42)

modelo\_regulado.fit(X\_train\_scaled, y\_train)

# Restaurar nombres originales en DataFrame

X\_test\_scaled\_df = pd.DataFrame(X\_test\_scaled, columns=X.columns)

# Crear explicador SHAP para el modelo regulado

explainer = shap.TreeExplainer(modelo\_regulado)

# Obtener valores SHAP (desactivando check de aditividad para prevenir errores)

shap\_values = explainer.shap\_values(X\_test\_scaled\_df, check\_additivity=False)

# Seleccionar clase positiva en caso de clasificación binaria

shap\_for\_plot = shap\_values[1] if isinstance(shap\_values, list) else shap\_values

# Visualizar resultados

shap.summary\_plot(shap\_for\_plot, X\_test\_scaled\_df)