Descarga de los archivos del Adult Dataset

```
import os
import urllib.request

# Crear la carpeta donde AIF360 busca los archivos
adult_path = '/usr/local/lib/python3.12/dist-packages/aif360/data/raw/adult'
os.makedirs(adult_path, exist_ok=True)

# Descargar los 3 archivos necesarios
print("Descargando archivos del Adult Dataset...")
urllib.request.urlretrieve('https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data',
urllib.request.urlretrieve('https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test',
urllib.request.urlretrieve('https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names

print("¡Archivos descargados correctamente!")
```

```
Descargando archivos del Adult Dataset...
¡Archivos descargados correctamente!
```

Celda 1: Instalación

```
!pip install shap
!pip install aif360
!pip install scikit-learn
print("Instalado SHAP y dependencias")
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.12/dist-packages (0.50.0)

Requirement already satisfied: numpy>=2 in /usr/local/lib/python3.12/dist-packages (from shap) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from shap) (1.16.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (from shap) (1.6.
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.12/dist-packages (from shap) (4.67
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.12/dist-packages (from shap) (25
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.12/dist-packages (from shap) (0.0
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.12/dist-packages (from shap) (0.60.(
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.12/dist-packages (from shap) (3.1.2
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.12/dist-packages (from shap)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pa
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->shaj
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->sl
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lea
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scil
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil
Collecting aif360
  Downloading aif360-0.6.1-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.12/dist-packages (from aif360) (2.0
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from aif360) (1.
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.12/dist-packages (from aif360) (
Requirement already satisfied: scikit-learn>=1.0 in /usr/local/lib/python3.12/dist-packages (from aif360
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from aif360) (3.10
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pa
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.2
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lea
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scil
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplot
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplo
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplo
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotl
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->ai
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplot

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil
Downloading aif360-0.6.1-py3-none-any.whl (259 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 259.7/259.7 kB 19.2 MB/s eta 0:00:00
Installing collected packages: aif360
Successfully installed aif360-0.6.1
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-packages (from scikit-lea
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lear
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lea
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scil
Instalado SHAP y dependencias
```

Celda 2: Importar librerías y cargar dataset + modelo

```
import shap
import numpy as np
import pandas as pd
from aif360.datasets import AdultDataset
from aif360.algorithms.preprocessing import Reweighing
from sklearn.ensemble import RandomForestClassifier

# Cargar dataset (mismo que antes)
dataset_orig = AdultDataset(
    protected_attribute_names=['sex'],
    privileged_classes=[['Male']],
    features_to_drop=['race']
)

privileged_groups = [{'sex': 1.0}]
unprivileged_groups = [{'sex': 0.0}]


# Mitigar
RW = Reweighing(unprivileged_groups=unprivileged_groups, privileged_groups=privileged_groups)
dataset_transf = RW.fit_transform(dataset_orig)

# Dividir train/test (mismo seed para reproducibilidad)
dataset_orig_train, dataset_orig_test = dataset_orig.split([0.7], shuffle=True, seed=42)
dataset_transf_train, dataset_transf_test = dataset_transf.split([0.7], shuffle=True, seed=42)

print("Datasets cargados y divididos")
```

```
WARNING:root:No module named 'fairlearn': ExponentiatedGradientReduction will be unavailable. To install
pip install 'aif360[Reductions]'
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailable. To install, run:
pip install 'aif360[Reductions]'
WARNING:root:No module named 'inFairness': SenSeI and SenSR will be unavailable. To install, run:
pip install 'aif360[inFairness]'
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailable. To install, run:
pip install 'aif360[Reductions]'
WARNING:root:Missing Data: 3620 rows removed from AdultDataset.
Datasets cargados y divididos
Datasets cargados y divididos
```

Celda 3: Entrenar los dos modelos (original y mitigado)

```
# Modelo original
X_train_orig = dataset_orig_train.features
y_train_orig = dataset_orig_train.labels.ravel()
X_test = dataset_orig_test.features
y_test = dataset_orig_test.labels.ravel()

model_orig = RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
model_orig.fit(X_train_orig, y_train_orig)

# Modelo mitigado (usa pesos del dataset_transf_train)
sample_weights = dataset_transf_train.instance_weights
model_fair = RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
model_fair.fit(X_train_orig, y_train_orig, sample_weight=sample_weights)  # Usa los pesos

print("Modelos entrenados: original y fair (con pesos Reweighing)")
```

```
Modelos entrenados: original y fair (con pesos Reweighing)
```

Celda 4: Elegir ejemplos interesantes y explicar con SHAP

```
# Convertir test a DataFrame para nombres de features
df_test, _ = dataset_orig_test.convert_to_dataframe()
feature_names = df_test.columns[:-1]  # Excluye la label
```

```
# Explainer SHAP para ambos modelos
explainer_orig = shap.TreeExplainer(model_orig)
explainer_fair = shap.TreeExplainer(model_fair)

# Elegir 2 ejemplos interesantes (índices del test set)
idx_woman_low = 110   # Cambia si quieres otro (prueba varios)

idx_man_high = 150

X_sample = X_test[[idx_woman_low, idx_man_high]]

shap_values_orig = explainer_orig.shap_values(X_sample)
shap_values_fair = explainer_fair.shap_values(X_sample)

print("Ejemplos seleccionados:")
print("Mujer (índice 110):", df_test.iloc[idx_woman_low])
print("Hombre (índice 150):", df_test.iloc[idx_man_high])
```

```
Ejemplos seleccionados:
Mujer (índice 110): age                                   24.0
fnlwgt                            209034.0
education-num                         12.0
sex                                    0.0
capital-gain                           0.0
                                     ...
native-country=Trinadad&Tobago         0.0
native-country=United-States           1.0
native-country=Vietnam                 0.0
native-country=Yugoslavia              0.0
income-per-year                        0.0
Name: 41987, Length: 99, dtype: float64
Hombre (índice 150): age                                  35.0
fnlwgt                            53553.0
education-num                        10.0
sex                                   1.0
capital-gain                       7298.0
                                    ...
native-country=Trinadad&Tobago        0.0
native-country=United-States          1.0
native-country=Vietnam                0.0
```

```
native-country=Yugoslavia            0.0
income-per-year                      1.0
Name: 5027, Length: 99, dtype: float64
```

Celda 5: Gráficos SHAP (force plots)

```
# Celda 5: Gráficos SHAP (force plots) que funcionan en Colab
import shap

# Elegir un ejemplo del test set
idx_example = 110  # Prueba varios índices (5, 20, 50, 100...) para encontrar casos interesantes

X_sample = X_test[idx_example:idx_example+1]  # Shape (1, n_features)

print(f"Ejemplo seleccionado (índice {idx_example}):")
display(df_test.iloc[idx_example])

# Calcular SHAP values
shap_values_orig = explainer_orig.shap_values(X_sample)
shap_values_fair = explainer_fair.shap_values(X_sample)

# Forma típica: (1, n_features, 2) → clase positiva es index 1
shap_values_orig_pos = shap_values_orig[0, :, 1]  # (n_features,)
shap_values_fair_pos = shap_values_fair[0, :, 1]

# Base value para clase positiva
base_value_orig = explainer_orig.expected_value[1]
base_value_fair = explainer_fair.expected_value[1]

# Waterfall plot - Modelo ORIGINAL
print("\n=== Explicación LOCAL - Modelo ORIGINAL (sin mitigación) ===")
shap.waterfall_plot(
    shap.Explanation(
        values=shap_values_orig_pos,
        base_values=base_value_orig,
        data=X_sample[0],
        feature_names=feature_names
    ),
    ),
    max_display=15
)

# Waterfall plot - Modelo MITIGADO
print("\n=== Explicación LOCAL - Modelo MITIGADO (con Reweighing) ===")
shap.waterfall_plot(
    shap.Explanation(
        values=shap_values_fair_pos,
        base_values=base_value_fair,
        data=X_sample[0],
        feature_names=feature_names
    ),
    max_display=15
)
```

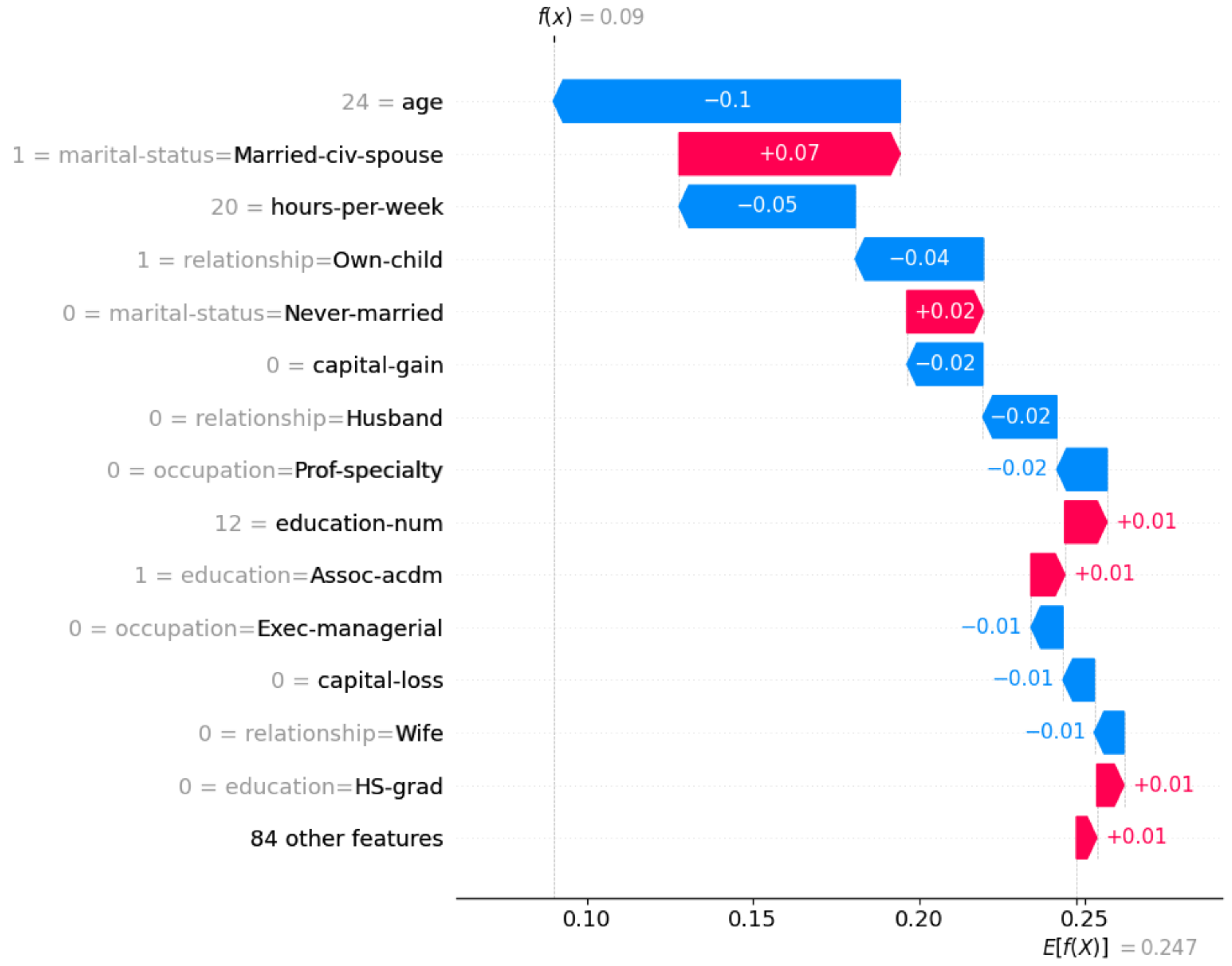
```
Ejemplo seleccionado (índice 110):
```

	41987
age	24.0
fnlwgt	209034.0
education-num	12.0
sex	0.0
capital-gain	0.0
...	...
native-country=Trinadad&Tobago	0.0
native-country=United-States	1.0
native-country=Vietnam	0.0
native-country=Yugoslavia	0.0
income-per-year	0.0

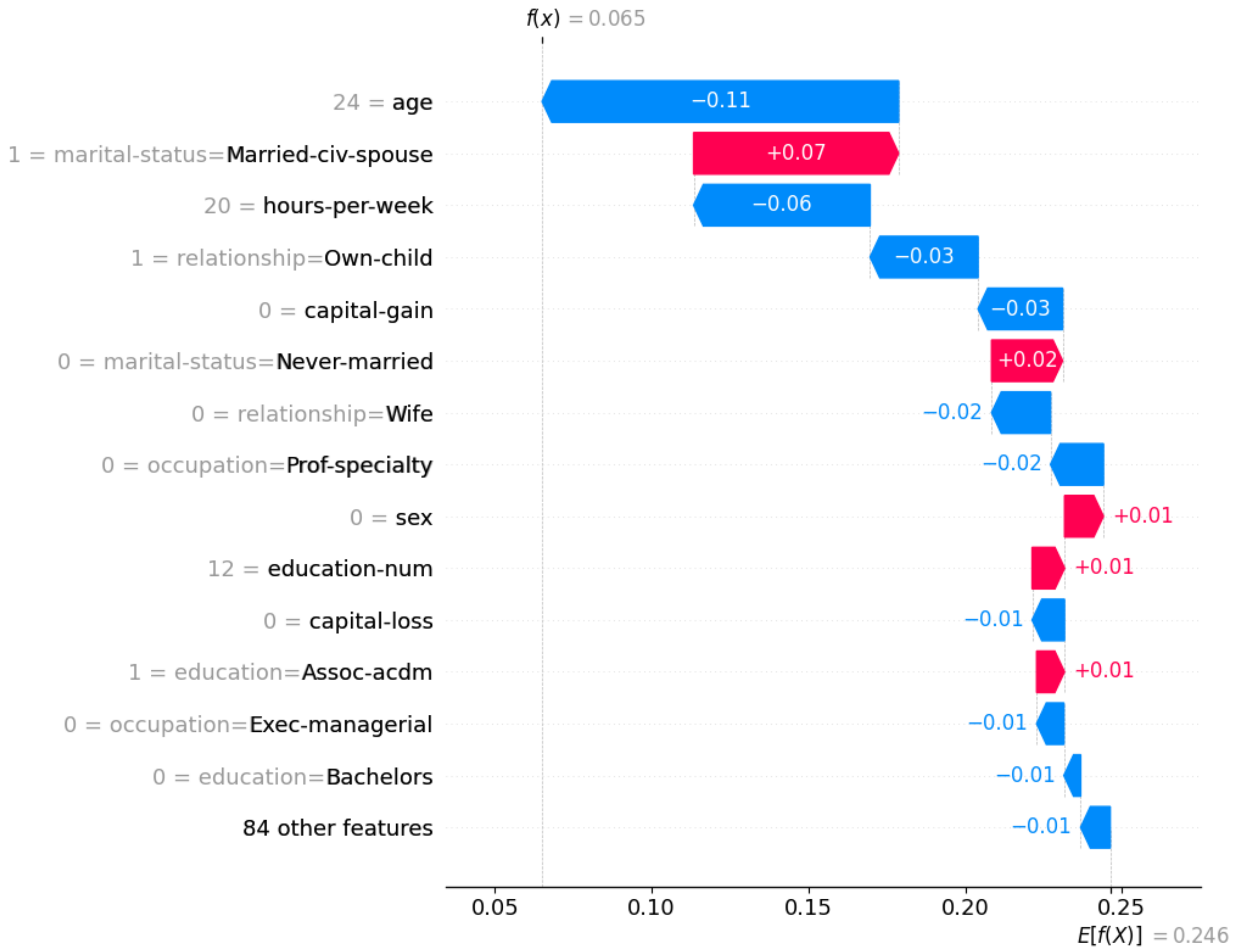
99 rows × 1 columns

dtype: float64

```
=== Explicación LOCAL - Modelo ORIGINAL (sin mitigación) ===
```



```
=== Explicación LOCAL - Modelo MITIGADO (con Reweighing) ===
```



Recuperar X_test y df_test del dataset de test original

```
# Recuperar X_test y df_test del dataset de test original
X_test = dataset_orig_test.features
df_test, _ = dataset_orig_test.convert_to_dataframe()

print(f"X_test recuperado: {X_test.shape[0]} muestras")
```

```
X_test recuperado: 13567 muestras
```

Calcular SHAP values para TODO el test set (necesario para summary plot global)

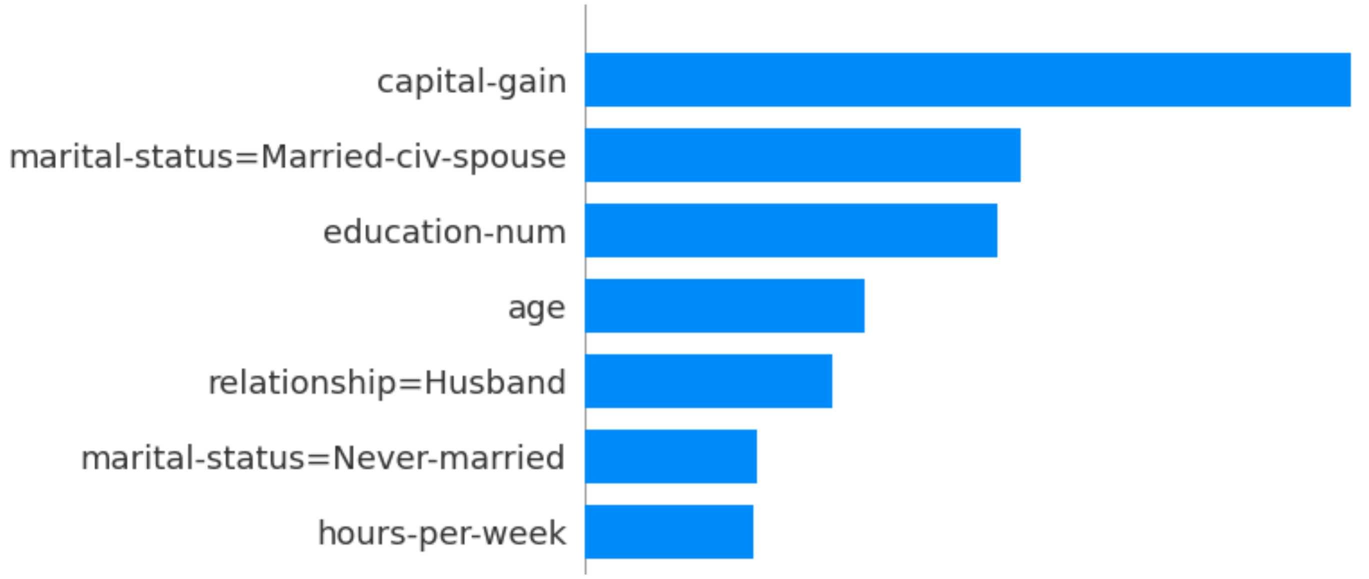
```
import numpy as np

# Muestra aleatoria del test set para summary plot (rápido y representativo)
sample_size = 10  # Suficiente para ver patrones globales sin tardar

indices_sample = np.random.choice(X_test.shape[0], sample_size, replace=False)
X_test_sample = X_test[indices_sample]

print(f"Calculando SHAP values para muestra de {sample_size} ejemplos del test set (rápido)...")
shap_values_test_orig = explainer_orig.shap_values(X_test_sample)
shap_values_test_fair = explainer_fair.shap_values(X_test_sample)
print("¡SHAP values para summary plot calculados!")
```

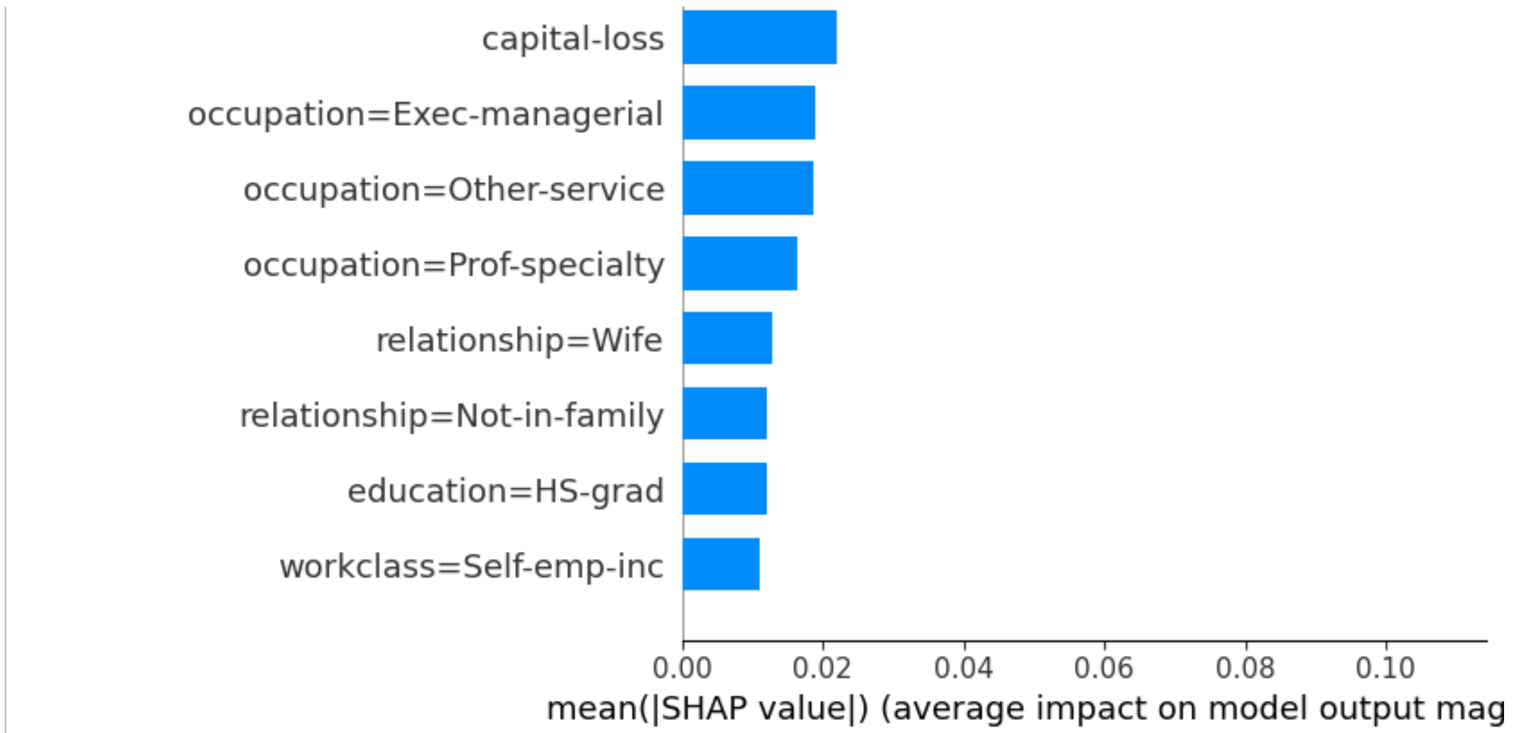
```
Calculando SHAP values para muestra de 10 ejemplos del test set (rápido)...
¡SHAP values para summary plot calculados!
```

Summary plot global (para ver bias general)

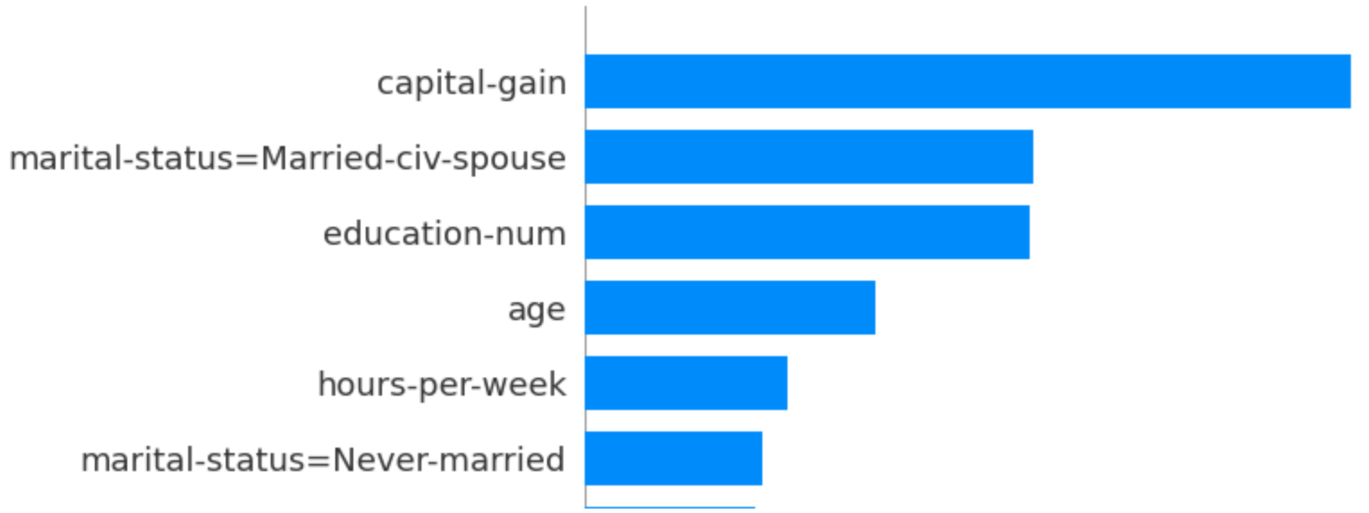
```
# Summary Plot GLOBAL - Modelo ORIGINAL (muestra de 200 ejemplos)
print("=== Summary Plot GLOBAL - Modelo ORIGINAL (sin mitigación) ===")
#shap.summary_plot(shap_values_test_orig[:, :, 1], X_test_sample, feature_names=feature_names, max_disp

# Summary Plot GLOBAL - Modelo MITIGADO
print("=== Summary Plot GLOBAL - Modelo MITIGADO (con Reweighing) ===")
#shap.summary_plot(shap_values_test_fair[:, :, 1], X_test_sample, feature_names=feature_names, max_disp

shap.summary_plot(shap_values_test_orig[:, :, 1], X_test_sample, feature_names=feature_names, plot_type=
shap.summary_plot(shap_values_test_fair[:, :, 1], X_test_sample, feature_names=feature_names, plot_type=
```

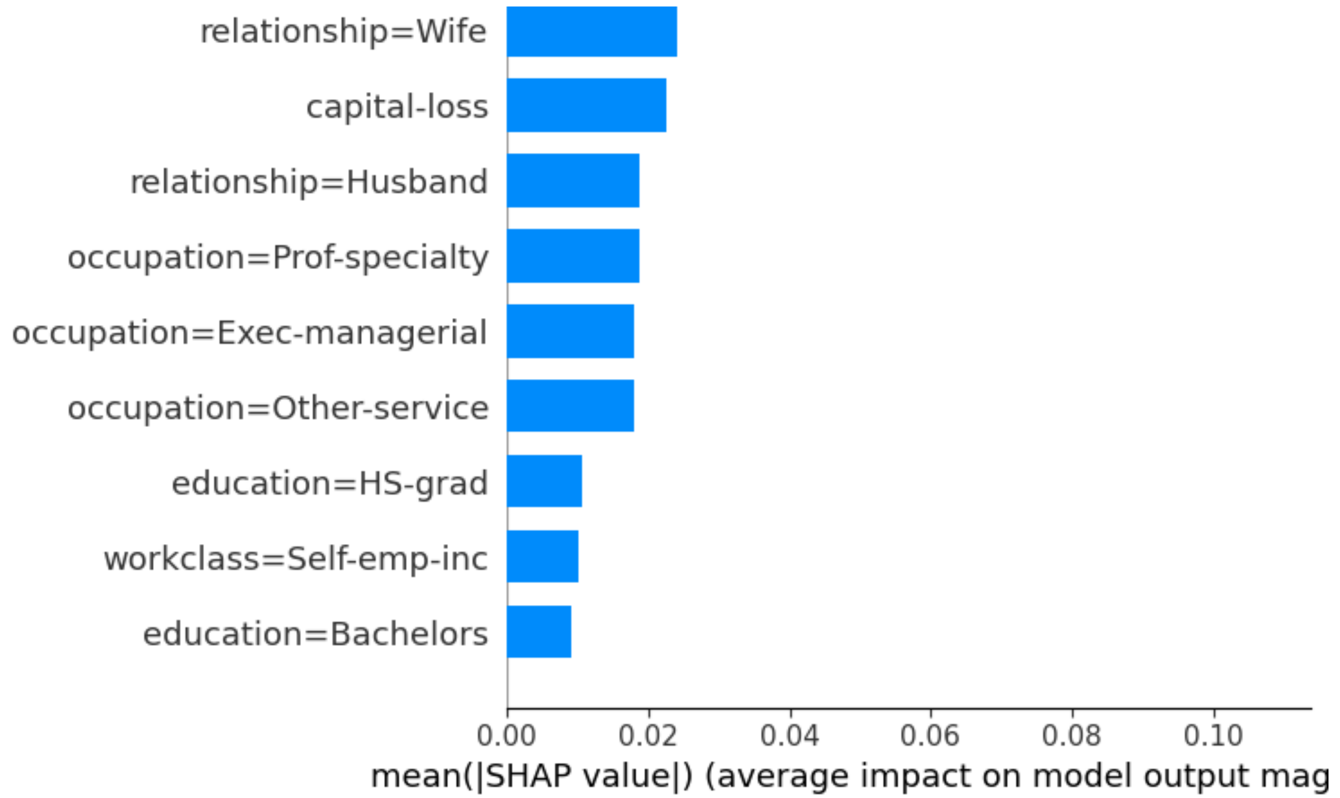
```
=== Summary Plot GLOBAL - Modelo ORIGINAL (sin mitigación) ===
=== Summary Plot GLOBAL - Modelo MITIGADO (con Reweighing) ===
/tmp/ipython-input-3578969593.py:9: FutureWarning: The NumPy global RNG was seeded by calling `np.random.
  shap.summary_plot(shap_values_test_orig[:, :, 1], X_test_sample, feature_names=feature_names, plot_type
```





```
/tmp/ipython-input-3578969593.py:10: FutureWarning: The NumPy global RNG was seeded by calling `np.random
  shap.summary_plot(shap_values_test_fair[:, :, 1], X_test_sample, feature_names=feature_names, plot_type
```




relationship=Wife
capital-loss
relationship=Husband
occupation=Prof-specialty
occupation=Exec-managerial
occupation=Other-service
education=HS-grad
workclass=Self-emp-inc
education=Bachelors
0.00
0.02
0.04
0.06
0.08
0.10
mean(|SHAP value|) (average impact on model output mag