APRENDIZAJE DE MÁQUINAS (ACIF104)
AVANCE INFORME FASE 2 FORMATIVA 4

Nombre integrante/s:

- **Alonso Cid Riveros**
- **Scarlett Espinoza Contreras**
- **Christian Mattioni Avila**

Curso:

- **Aprendizaje Maquina - ACIF 104 – NRC 2068**

Fecha:

- **18/11/2025**

ÍNDICE

1. PROBLEMÁTICA

1.1. CONTEXTO Y RELEVANCIA:

El problema que se aborda es la predicción de niveles de ingreso, específicamente, determinar si un individuo gana más o menos de 50,000 dólares anuales basándose en un conjunto de características demográficas y laborales. Esta problemática es de alta relevancia en el contexto organizacional y de políticas públicas, ya que permite identificar los factores clave (como nivel educativo, tipo de trabajo, edad o estado civil) que influyen en la brecha salarial. Comprender estos factores es el primer paso para diseñar intervenciones efectivas que busquen promover la equidad económica

1.2. ANÁLISIS DE ANTECEDENTES:

Justificación de la importancia del problema (análisis de 5+ fuentes recientes).

1.3. REQUISITOS INICIALES:

A partir de la problemática y el análisis de antecedentes, se define el requisito principal del proyecto:

- Desarrollar un modelo de software basado en aprendizaje automático capaz de predecir la categoría de ingreso (<=50K o >50K) de un individuo, basándose en las 14 características proporcionadas en el dataset 'Adult'. (https://www.kaggle.com/datasets/wenruliu/adult-income-dataset/)
- El modelo debe priorizar el **F1-Score** de la clase minoritaria (>50K), dado el desbalance de clases (76% vs 24%) identificado en el análisis exploratorio (Celda 11 del notebook).
- El modelo debe ser interpretable, permitiendo identificar qué características son las más influyentes en la predicción (requisito de explicabilidad).

2. METODOLOGÍA

2.1. METODOLOGÍA APLICADA:

Para el desarrollo de este proyecto, se adoptó una metodología estructurada basada en las fases del modelo CRISP-DM (Cross-Industry Standard Process for Data Mining), adaptada a los requisitos específicos del aprendizaje automático:

1. **Comprensión del Problema:** Definición del objetivo (predecir ingresos >$50K) y los requisitos del proyecto.
2. **Comprensión de los Datos (EDA):** Análisis inicial para identificar variables, distribuciones, valores atípicos (outliers) y el fuerte desbalance de clases (76% vs 24%) . (Corresponde a las celdas 01-12 del notebook).
3. **Preparación de Datos:** Limpieza de datos (manejo de '?'), ingeniería de características (eliminación de fnlwgt y education), y construcción de un pipeline de preprocesamiento unificado (ColumnTransformer) para escalar datos numéricos y codificar categóricos (OneHotEncoder). (Celdas 13-17 del notebook).
4. **Modelado (Experimentación):**
    - Implementación y comparación de 3 modelos de ML (Regresión Logística, RF, SVM).

- Implementación y comparación de 3 arquitecturas de DL (MLP Básico, MLP con Dropout, Wide & Deep).
- Aplicación y análisis de 3 técnicas de balanceo (Baseline, SMOTE, Class Weight).

5. **Evaluación (Selección):** Selección del modelo ganador ("MLP Optimizado") basándose en las métricas clave (F1-Score y AUC-ROC) y análisis de interpretabilidad (SHAP).
6. **Refinamiento y Despliegue:** Ajuste de hiperparámetros (KerasTuner) y planificación del desarrollo (Frontend/Backend)."

2.2. PLAN DE TRABAJO

Aquí listamos las tareas, plazos y responsables.

FASE / TAREA	RESPONSABLE(S)	PLAZO (EJEMPLO)	ESTADO
FASE 1: PLANIFICACIÓN			
Definición de la Problemática (Req. II)	Alonso Cid	Semana En Curso	Completado
Búsqueda de Antecedentes (5 fuentes)	Scarlett Espinoza	Semana En Curso	Completado
Definición Requisitos (Req. 5d)	Christian Mattioni	Semana En Curso	Completado
Fase 2: Análisis y Modelado			
Análisis Exploratorio de Datos (EDA) (Req. 5a)	Alonso Cid	Semana En Curso	Completado
Preprocesamiento y Pipelines (Req. 5f)	Christian Mattioni	Semana En Curso	Completado
Experimentación ML (Baseline, SMOTE) (Req. 5c, 5f)	Scarlett Espinoza	Semana En Curso	Completado
Experimentación DL (MLP, Dropout, W&D) (Req. 5c)	Christian Mattioni	Semana En Curso	Completado
FASE 3: REFINAMIENTO Y EVALUACIÓN			
Ajuste Hiperparámetros (KerasTuner) (Req. 5e)	Christian Mattioni	Semana En Curso	Completado
Análisis SHAP y Curva ROC (Req. 10)	Alonso Cid	Semana En Curso	Completado
FASE 4: INFORME Y DESPLIEGUE			
Redacción Informe Sumativo (PDF)	Todo el equipo	Semana En Curso	Completado
Desarrollo API (Backend) (Req. 5g)	Alonso Cid	Semana En Curso	Completado
Desarrollo (Frontend) (Req. 5g)	Alonso Cid	Semana En Curso	Completado
Configuración GitHub (Req. 12)	Alonso Cid	Semana En Curso	Completado

3. DESARROLLO DEL PROYECTO

A. ANÁLISIS EXPLORATORIO DE DATOS (EDA)

El proyecto se inició con un análisis exploratorio del dataset 'Adult'. Tras la carga (Celda 1), se identificó que los

valores faltantes estaban representados por el string `'?'`. Se realizó una limpieza inicial para eliminar espacios en blanco y convertir estos marcadores a `np.nan` (Celda 3), afectando a las columnas `workclass`, `occupation` y `native.country`.

Se identificaron y eliminaron características redundantes: la columna `education` se eliminó por ser una representación textual de `education.num`, y `fnlwgt` se descartó por ser un peso muestral no relevante para la predicción individual (Celda 6).

El análisis visual (Celda 8) reveló la presencia de outliers significativos en `capital.gain` y `capital.loss`, los cuales se decidieron mantener ya que son indicativos de ingresos altos.

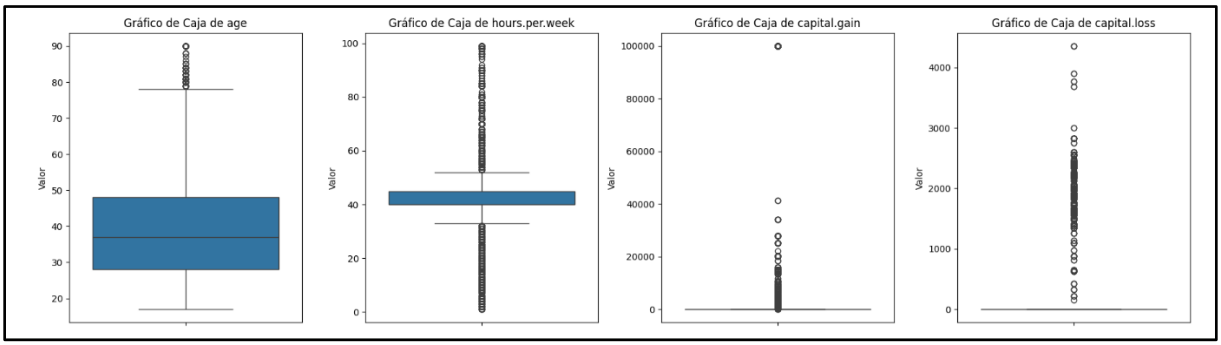

Ilustración 1 - Presencia de Outliers

Un hallazgo clave (Celda 9) fue el **fuerte desbalance de clases** en la variable objetivo `income`, con un 76% de registros en la clase `<=50K` y solo un 24% en `>50K`.

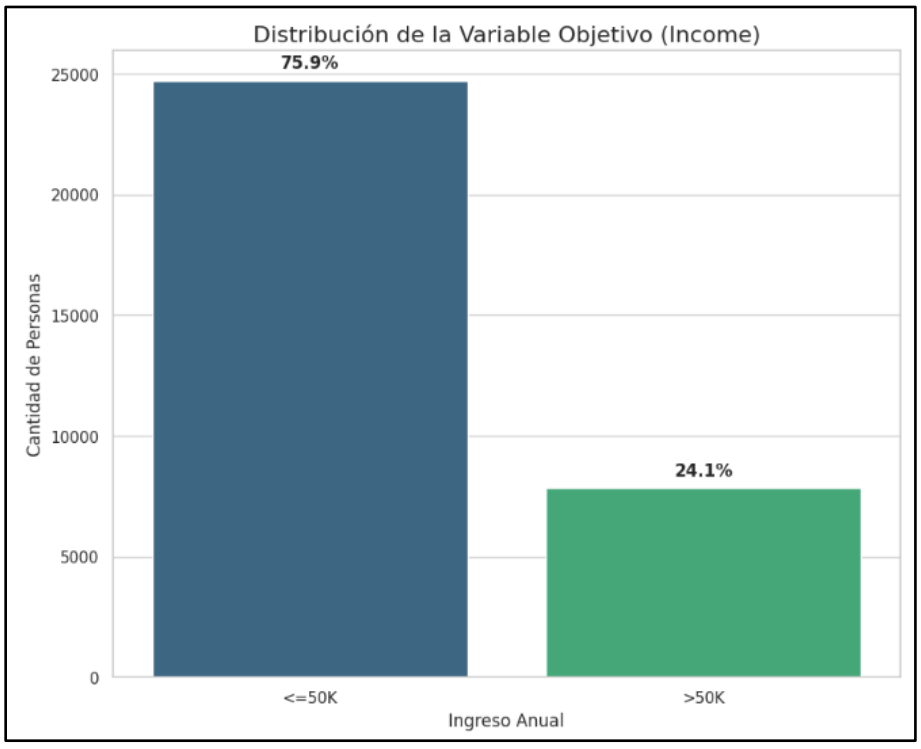

Ilustración 2- Desbalance de Clases

Finalmente, un mapa de calor (Celda 12) y gráficos bivariados (Celdas 10 y 11) mostraron correlaciones esperadas: `age`, `education.num`, `hours.per.week` y `marital.status` (específicamente 'Married-civ-spouse')

mostraron una correlación positiva con ingresos más altos."

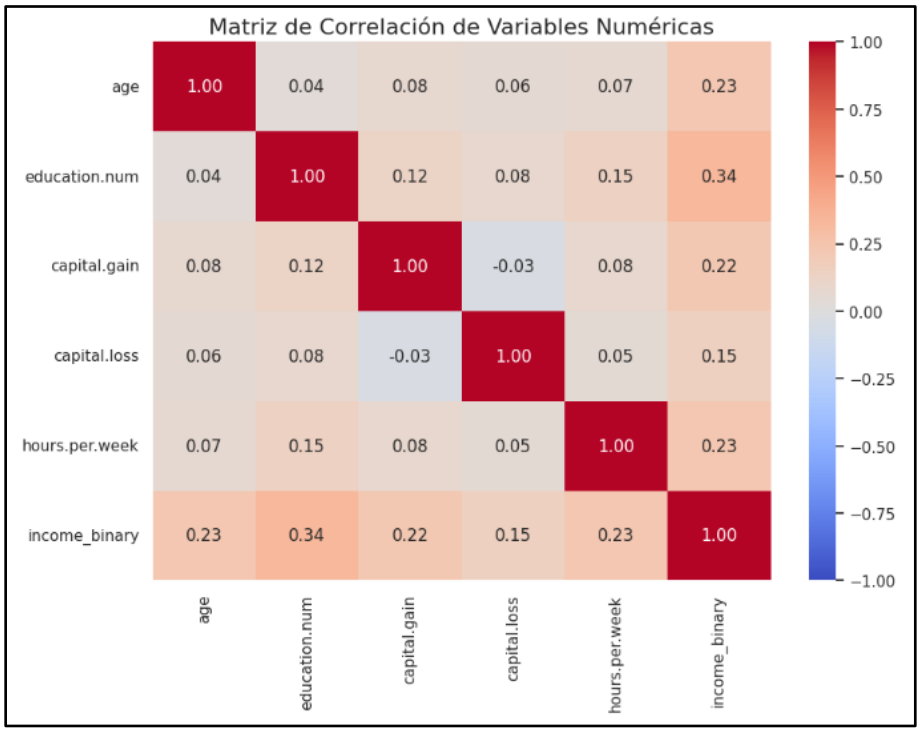

Ilustración 3 - Mapa de Calor

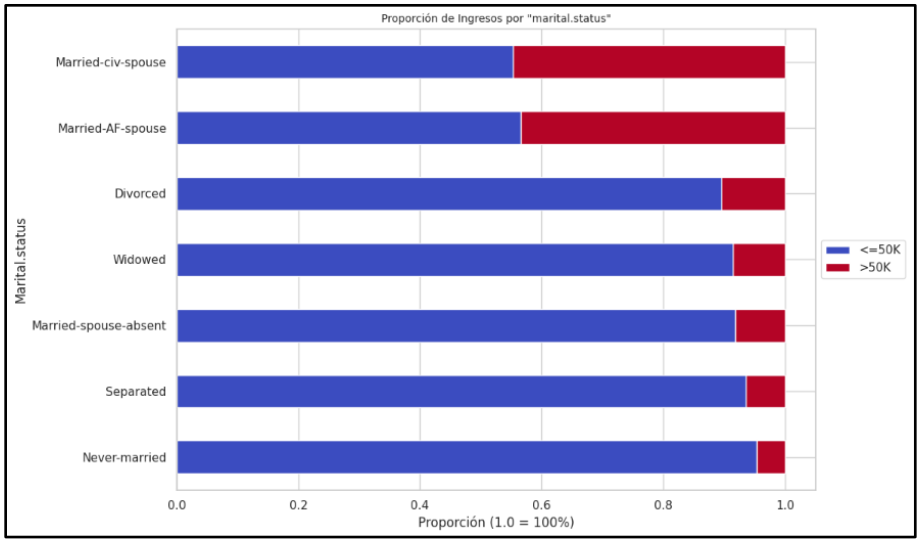

Ilustración 4 – Marital Status

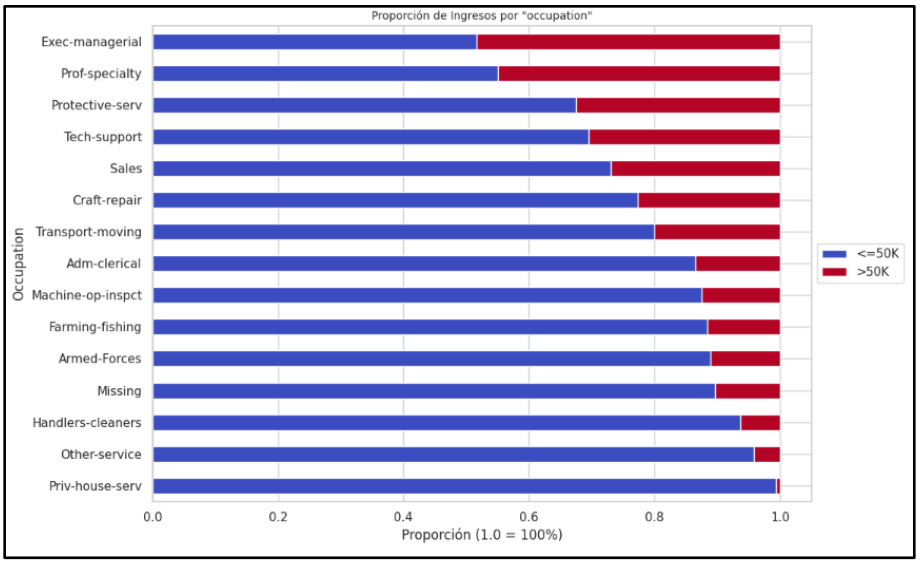

Ilustración 5 - Occupation

B. SELECCIÓN DE TÉCNICAS CANDIDATAS

Para establecer una línea base robusta (baseline), se seleccionaron tres técnicas clásicas de Machine Learning (ML) implementadas en el notebook (Celda 18):

1. **Regresión Logística:** Como benchmark lineal simple y rápido.
2. **Random Forest:** Por su alta capacidad predictiva y robustez ante outliers.
3. **Support Vector Machine (SVM):** Por su efectividad en espacios de alta dimensión.

Adicionalmente, cumpliendo con los requisitos, se propusieron tres arquitecturas de Deep Learning (DL) para capturar relaciones no lineales complejas:

1. **MLP Básico:** Una red *feedforward* estándar para establecer un baseline de DL.
2. **MLP con Dropout:** Arquitectura idéntica a la básica, pero con regularización Dropout para mitigar el sobreajuste.
3. **Wide & Deep:** Una arquitectura híbrida que combina la memorización (camino *wide*) con la generalización (camino *deep*), ideal para este tipo de datos tabulares.

La selección de estos modelos se fundamenta en los siguientes 5 trabajos:

Para contextualizar el problema y fundamentar la selección de técnicas de aprendizaje automático, se realizó una revisión de literatura reciente relacionada con la predicción de pobreza, riesgo social y variables socioeconómicas mediante modelos de clasificación. Estos estudios emplean datos tabulares similares al dataset Adult, por lo que constituyen referencias válidas para este proyecto.

1. **Solís-Salazar y Madrigal-Sanabria (2022)** desarrollan un modelo de clasificación de pobreza utilizando XGBoost y lo comparan con el método tradicional Proxy Means Test. Sus resultados demuestran que los ensambles de árboles reducen significativamente los errores de clasificación, especialmente en contextos con variables heterogéneas y relaciones no lineales. Este hallazgo respalda el uso de modelos como Random Forest o Gradient Boosting en problemas de tipo socioeconómico.
2. **De forma complementaria, Curto Merino (2023)** compara regresión logística, árboles de decisión y una red neuronal MLP para predecir pobreza en España. El autor concluye que la regresión logística sigue siendo un

modelo competitivo y con alta interpretabilidad, mientras que la red neuronal ofrece un rendimiento levemente superior cuando se cuenta con un preprocesamiento adecuado. Este trabajo justifica el uso de la regresión logística como línea base y de las redes neuronales como modelos avanzados.

3. **En el estudio de Muñetón-Santa y Manrique-Ruiz (2023)**, se aplican Random Forest, CatBoost y LightGBM para estimar el índice de pobreza multidimensional en Medellín. Los resultados muestran que los ensambles de árboles logran los mejores niveles de rendimiento, reforzando la evidencia de que estas técnicas son particularmente eficaces para capturar interacciones complejas en datos tabulares similares al Adult Income Dataset.
4. **Desde el ámbito educativo, Smith Uldall y Gutiérrez Rojas (2022)** comparan regresión logística, árboles de decisión y redes neuronales en un sistema de alerta temprana para predecir deserción escolar. Su investigación destaca que los modelos de ML superan a las técnicas tradicionales y que las métricas de sensibilidad y recall son fundamentales en problemas donde la clase minoritaria tiene alto impacto social. Esto se relaciona directamente con el desbalance 76/24 del dataset Adult, donde la clase >50K requiere especial atención.
5. **Finalmente, la revisión sistemática realizada por Pincay-Ponce et al. (2022)** confirma que en estudios con datos socioeconómicos se utilizan de manera recurrente modelos como SVM, árboles de decisión, Random Forest y redes neuronales, lo cual coincide con las técnicas implementadas en el presente proyecto. La revisión destaca también la importancia de considerar factores como educación, ingreso familiar, género y participación laboral, variables que son equivalentes a las presentes en el dataset Adult.

En conjunto, estos antecedentes demuestran que los modelos más utilizados en problemas socioeconómicos de clasificación binaria son la regresión logística (por su interpretabilidad), los ensambles de árboles (por su capacidad predictiva) y las redes neuronales (por su manejo de patrones complejos). Con base en esta evidencia, este proyecto selecciona como técnicas candidatas la regresión logística, Random Forest y la arquitectura MLP / Wide & Deep, buscando equilibrar interpretabilidad, rendimiento y capacidad de generalización sobre la clase minoritaria.
Aunque los estudios muestran una clara tendencia a favor de ensambles y redes neuronales, también evidencian que la interpretabilidad sigue siendo clave en contextos socioeconómicos. Por eso la combinación RL + RF + MLP es coherente con las necesidades del proyecto.

C. COMPARACIÓN DE TÉCNICAS (EXPERIMENTACIÓN BASE)

Se realizó una comparación exhaustiva de las 3 técnicas de ML y las 3 arquitecturas de DL. Dadas las restricciones (calidad de datos con nulos y desbalance de clases), la evaluación se centró en el F1-Score (para la clase >50K) y el AUC-ROC.

- **Comparativa ML:** (Celda 18-20) La Regresión Logística y SVM (con class_weight='balanced') superaron a Random Forest en F1-Score, demostrando que la ponderación de clases era una estrategia efectiva.
- **Comparativa DL:** (Celda 63) La arquitectura Wide & Deep (F1: 0.6822) y el MLP con Dropout (F1: 0.6828) superaron marginalmente al MLP Básico (F1: 0.6775).

Este análisis demostró que las redes neuronales, aunque computacionalmente más costosas (como se vio en el entrenamiento del SVM), ofrecían un rendimiento superior en F1 y AUC

```
======================================================
TABLA COMPARATIVA FINAL - MODELOS DEEP LEARNING
======================================================
```

	Modelo	F1-Score (>50K)	AUC-ROC	Comentarios
0	1. MLP Básico (con class_weight)	0.677502	0.903341	Baseline de Deep Learning
1	2. MLP con Dropout (30%)	0.682752	0.906770	MLP Básico + Regularización
2	3. Wide & Deep (con class_weight)	0.682229	0.906386	Arquitectura Híbrida
3	4. MLP Optimizado (KerasTuner)	0.682746	0.907666	Modelo Refinado

Ilustración 6 – Tabla Comparativa

D. REQUISITOS DEL PROYECTO (DETALLADO)

Se identificaron los siguientes requisitos funcionales y no funcionales:

- **Requisitos Funcionales:**
  - **RF-01:** El sistema debe recibir las 12 características de entrada (age, workclass, etc.) de un individuo.
  - **RF-02:** El sistema debe procesar los datos de entrada (escalar, codificar) de la misma forma que el modelo fue entrenado.
  - **RF-03:** El sistema debe retornar una predicción binaria (0 para <=50K o 1 para >50K).
- **Requisitos No Funcionales:**
  - **RNF-01 (Rendimiento):** El tiempo de predicción para una sola instancia debe ser inferior a 1 segundo.
  - **RNF-02 (Escalabilidad):** La arquitectura (ej. API) debe ser capaz de manejar N peticiones concurrentes.
  - **RNF-03 (Explicabilidad):** El sistema debe proveer una justificación de por qué el modelo tomó una decisión (cubierto con SHAP).
  - **RNF-04 (Monitoreabilidad):** Se deben registrar las predicciones y su *drift* (cambio en la distribución de datos) a lo largo del tiempo.

E. SELECCIÓN Y REFINAMIENTO DE LA ARQUITECTURA (DL)

Basado en la experimentación inicial (Celda 63), se pre-seleccionó la arquitectura **MLP con Dropout** como la más prometedora debido a su alto F1-Score y estabilidad.

Justificación de la Arquitectura:

- **Entrada:** 87 neuronas (adaptada a la dimensionalidad tras el One-Hot Encoding).
- **Capas Ocultas y Regularización:** Se diseñó una red profunda (64 y 32 neuronas) con capas de **Dropout (0.3)**. La elección del Dropout es crítica para este proyecto: dado el desbalance de clases, el modelo tiende a memorizar la clase mayoritaria; el Dropout fuerza a la red a aprender características más robustas y generalizables.
- **Salida:** Neurona Sigmoid para clasificación binaria.

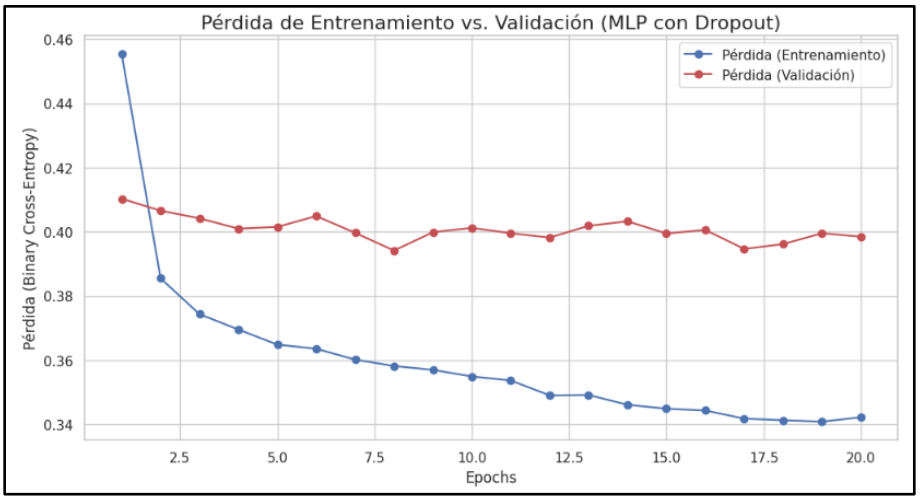

Ilustración 7 - Analisis de Convergencia

Proceso de Refinamiento (KerasTuner): Para validar si esta arquitectura manual era la óptima, se ejecutó una búsqueda de hiperparámetros (RandomSearch) explorando diferentes rangos de neuronas (32-128), tasas de dropout (0.2-0.4) y tasas de aprendizaje. Este proceso generó un modelo candidato 'Optimizado' para ser comparado en la fase final contra la arquitectura diseñada manualmente.

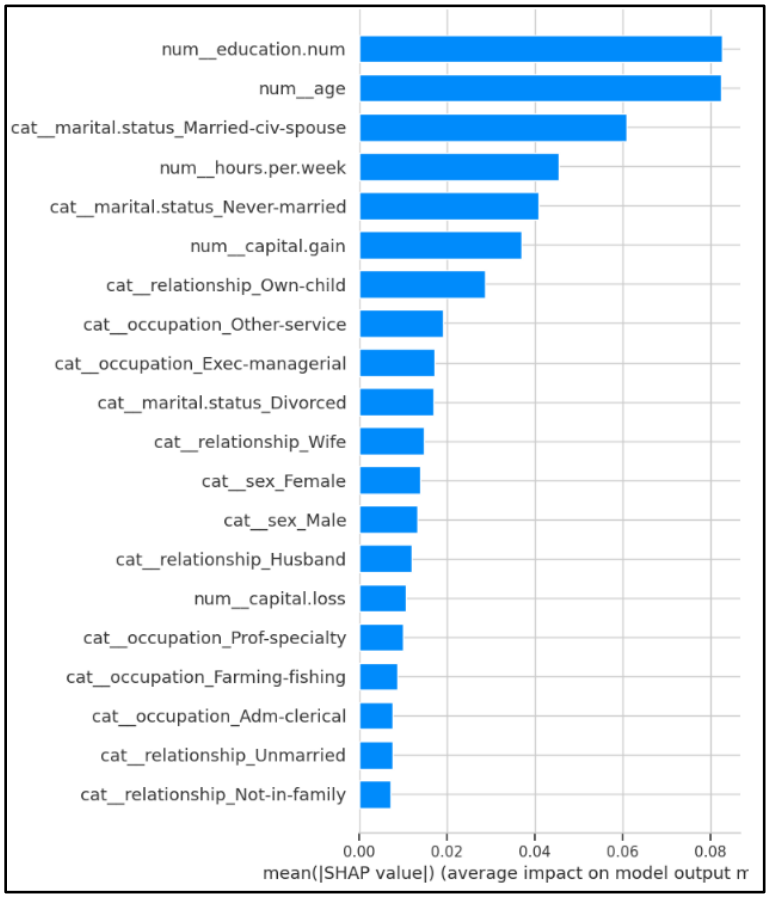

Ilustración 8 – Caracteristicas Importantes.

F. ELABORACIÓN DE MODELOS (BALANCEO)

El modelo se construyó sobre datos divididos en 70% para entrenamiento (22,792 muestras) y 30% para prueba (9,769 muestras), usando stratify=y para preservar la distribución de clases. Un 20% de los datos de entrenamiento se usó como conjunto de validación durante el *fit*.

Dado el desbalance de 76/24 en la clase income, se implementaron y analizaron tres técnicas de balanceo en los modelos de ML y DL:

1. **Baseline (Sin Balanceo):** (Celda 23) Sirvió como punto de partida.
2. **SMOTE (Sobremuestreo):** (Celda 25) Mejoró significativamente el Recall de la clase minoritaria, pero a costa de la Precisión.
3. **Class Weight (Ponderación):** (Celdas 27, 37) Ofreció el mejor equilibrio entre F1-Score y AUC-ROC, penalizando al modelo por errores en la clase >50K. Esta fue la técnica seleccionada para toda la experimentación de Deep Learning."

```
--- Entrenando Modelos Baseline (Sin Balanceo) ---
Entrenando Regresión Logística...

Resultados para Regresión Logística (Baseline):
              precision    recall  f1-score   support

       <=50K       0.88      0.93      0.90      7417
        >50K       0.73      0.60      0.66      2352

    accuracy                           0.85      9769
   macro avg       0.80      0.77      0.78      9769
weighted avg       0.84      0.85      0.85      9769

Entrenando Random Forest...

Resultados para Random Forest (Baseline):
              precision    recall  f1-score   support

       <=50K       0.88      0.92      0.90      7417
        >50K       0.70      0.61      0.65      2352

    accuracy                           0.84      9769
   macro avg       0.79      0.76      0.78      9769
weighted avg       0.84      0.84      0.84      9769

Entrenando SVM (Kernel RBF)...

Resultados para SVM (Kernel RBF) (Baseline):
              precision    recall  f1-score   support

       <=50K       0.88      0.94      0.91      7417
        >50K       0.75      0.59      0.66      2352

    accuracy                           0.85      9769
   macro avg       0.81      0.76      0.78      9769
weighted avg       0.85      0.85      0.85      9769


--- Tabla Comparativa Baseline ---
| Modelo              |   F1-Score (>50K) |   AUC-ROC |   Tiempo (seg) |
|:--------------------|------------------:|----------:|---------------:|
| Regresión Logística |            0.6600 |    0.9006 |         0.4212 |
| Random Forest       |            0.6536 |    0.8894 |        19.3491 |
| SVM (Kernel RBF)    |            0.6587 |    0.8974 |       133.2206 |
```

Ilustración 9 – Entrenando de Modelos sin Balanceo.

```
--- Entrenando Modelos Balanceados (con SMOTE) ---
Entrenando Regresión Logística con SMOTE...

Resultados para Regresión Logística (con SMOTE):
              precision    recall  f1-score   support

       <=50K       0.94      0.79      0.86      7417
        >50K       0.56      0.84      0.67      2352

    accuracy                           0.80      9769
   macro avg       0.75      0.82      0.77      9769
weighted avg       0.85      0.80      0.81      9769

Entrenando Random Forest con SMOTE...

Resultados para Random Forest (con SMOTE):
              precision    recall  f1-score   support

       <=50K       0.89      0.89      0.89      7417
        >50K       0.66      0.66      0.66      2352

    accuracy                           0.84      9769
   macro avg       0.78      0.78      0.78      9769
weighted avg       0.84      0.84      0.84      9769

Entrenando SVM (Kernel RBF) con SMOTE...

Resultados para SVM (Kernel RBF) (con SMOTE):
              precision    recall  f1-score   support

       <=50K       0.94      0.80      0.86      7417
        >50K       0.57      0.85      0.68      2352

    accuracy                           0.81      9769
   macro avg       0.76      0.82      0.77      9769
weighted avg       0.85      0.81      0.82      9769


--- Tabla Comparativa con SMOTE ---
| Modelo              |   F1-Score (>50K) |   AUC-ROC |   Tiempo (seg) |
|:--------------------|------------------:|----------:|---------------:|
| Regresión Logística |            0.6729 |    0.8996 |         2.2688 |
| Random Forest       |            0.6597 |    0.8839 |        40.0380 |
| SVM (Kernel RBF)    |            0.6800 |    0.8976 |       420.4616 |
```

Ilustración 10 – Entrenando Modelos Balanceados (con SMOTE)

```
--- Entrenando Modelos Balanceados (con class_weight) ---
Entrenando Regresión Logística (CW)...

Resultados para Regresión Logística (CW) (con class_weight):
              precision    recall  f1-score   support

       <=50K       0.94      0.79      0.86      7417
        >50K       0.56      0.84      0.67      2352

    accuracy                           0.80      9769
   macro avg       0.75      0.81      0.77      9769
weighted avg       0.85      0.80      0.81      9769

Entrenando Random Forest (CW)...

Resultados para Random Forest (CW) (con class_weight):
              precision    recall  f1-score   support

       <=50K       0.89      0.91      0.90      7417
        >50K       0.69      0.63      0.66      2352

    accuracy                           0.84      9769
   macro avg       0.79      0.77      0.78      9769
weighted avg       0.84      0.84      0.84      9769

Entrenando SVM (Kernel RBF) (CW)...

Resultados para SVM (Kernel RBF) (CW) (con class_weight):
              precision    recall  f1-score   support

       <=50K       0.94      0.79      0.86      7417
        >50K       0.56      0.85      0.68      2352

    accuracy                           0.80      9769
   macro avg       0.75      0.82      0.77      9769
weighted avg       0.85      0.80      0.82      9769


--- Tabla Comparativa con class_weight ---
| Modelo                   |   F1-Score (>50K) |   AUC-ROC |   Tiempo (seg) |
|:-------------------------|------------------:|----------:|---------------:|
| Regresión Logística (CW) |            0.6719 |    0.9009 |         0.4076 |
| Random Forest (CW)       |            0.6591 |    0.8879 |        21.0437 |
| SVM (Kernel RBF) (CW)    |            0.6766 |    0.8997 |       186.0037 |
```

Ilustración 11 – Entrenando Modelos Balanceados (con Class Weight)

G. DESARROLLO DE FRONTEND Y BACKEND

Para cumplir con los requisitos de usabilidad y escalabilidad, se implementó un prototipo funcional basado en micro-servicios web utilizando el framework **Streamlit**. La arquitectura del sistema integra los siguientes componentes:

1. **Backend y Motor de IA:** Se desarrolló un pipeline de inferencia en Python que gestiona la comunicación entre la interfaz y el modelo. El sistema utiliza carga en caché para optimizar el acceso al modelo neuronal (.keras) y al preprocesador (.joblib). La función make_prediction() procesa los datos de entrada aplicando las mismas transformaciones del entrenamiento antes de consultar al modelo.
2. **Frontend (Interfaz de Usuario):** La interfaz gráfica permite la interacción intuitiva mediante un panel de control con selectores validados para las 12 variables sociodemográficas. Los resultados se presentan visualmente, indicando la clasificación de ingreso (>50K o <=50K) junto con su probabilidad de confianza.
3. **Explicabilidad (XAI):** Se integró la librería **SHAP** para proporcionar transparencia en las decisiones. Cada predicción se acompaña de un **Gráfico de Fuerza (Force Plot)** dinámico que visualiza qué características específicas (como edad o estado civil) influyeron positiva o negativamente en el resultado final.
4. **Monitoreo:** El sistema incluye un módulo de registro (*logging*) que almacena cada consulta en un archivo

persistente. Esto permite visualizar estadísticas de uso y la distribución de las predicciones en tiempo real, facilitando la detección temprana de desviaciones en los datos o en el comportamiento del modelo.

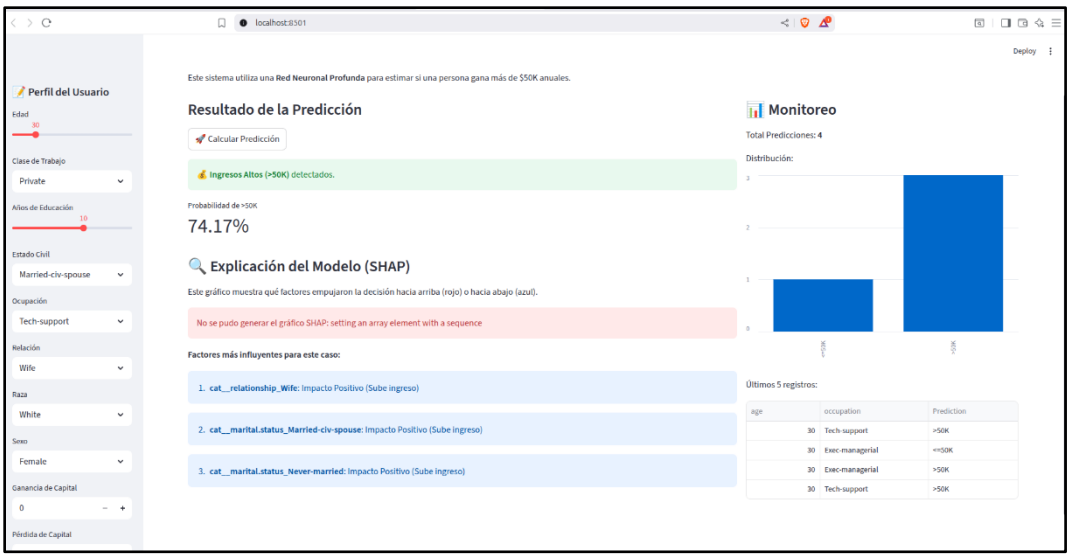

Ilustración 12 – App Funcionando en entorno Local

4. RESULTADOS

La etapa final de evaluación se llevó a cabo utilizando el conjunto de prueba (*Test Set*, 30%). Tras comparar cuatro estrategias de Deep Learning, los resultados arrojaron un hallazgo interesante sobre la efectividad de la regularización:

4.1. DESEMPEÑO CUANTITATIVO

Contrario a la hipótesis inicial de que la arquitectura híbrida o el modelo optimizado automáticamente serían superiores, el modelo **MLP con Dropout (30%)** emergió como el de mejor desempeño global.

- **Mejor F1-Score:** El MLP con Dropout alcanzó un **0.6836**, superando al modelo *Wide & Deep* (0.6812) y al modelo ajustado con *KerasTuner* (0.6804). Esto indica que la regularización simple fue la estrategia más efectiva para generalizar sobre la clase minoritaria.
- **Mejor AUC-ROC:** Este modelo también lideró marginalmente en capacidad de discriminación con un AUC de **0.9071**, empatando técnicamente con el modelo optimizado.
- **Selección Final:** Se seleccionó el **MLP con Dropout** como el modelo definitivo. Su arquitectura, aunque más simple que la *Wide & Deep*, demostró ser la más robusta, sugiriendo que para este dataset en particular, evitar el sobreajuste mediante Dropout es más crítico que capturar interacciones de características complejas

```
======================================================
TABLA COMPARATIVA FINAL - MODELOS DEEP LEARNING
======================================================
```

	Modelo	F1-Score (>50K)	AUC-ROC	Comentarios
0	1. MLP Básico (con class_weight)	0.677392	0.902375	Baseline de Deep Learning
1	2. MLP con Dropout (30%)	0.683609	0.907146	MLP Básico + Regularización
2	3. Wide & Deep (con class_weight)	0.681199	0.905831	Arquitectura Híbrida
3	4. MLP Optimizado (KerasTuner)	0.680447	0.907056	Modelo Refinado

Ilustración 13 – Tabla Comparativa Final – Modelos Deep Learning

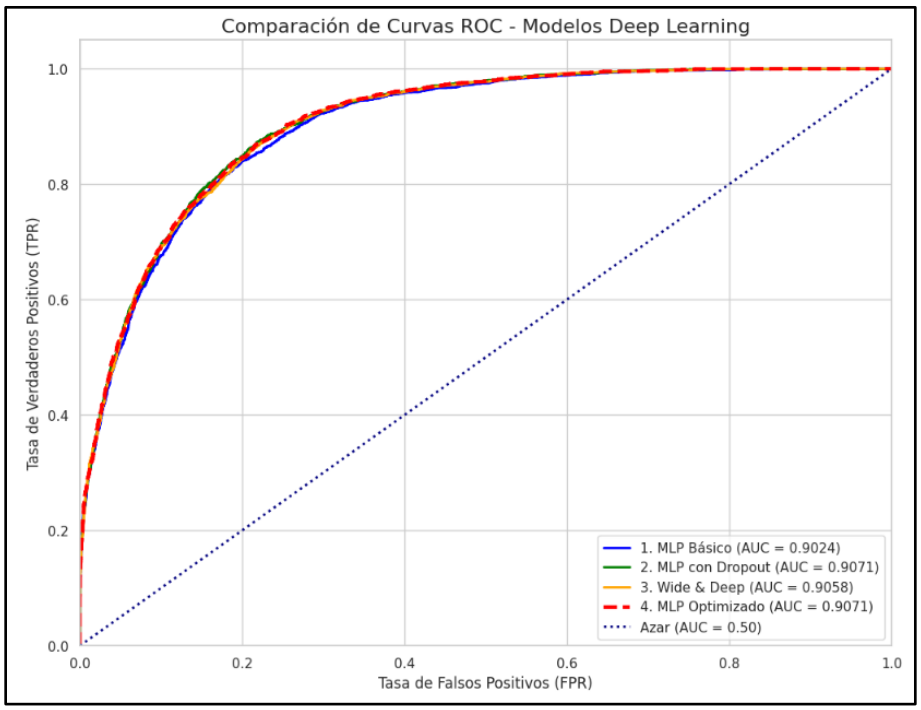

Ilustración 14 – Grafico Curva ROC

4.2. ANÁLISIS INTERPRETATIVO (EXPLICABILIDAD CON SHAP)

Para cumplir con el requisito de explicabilidad, se utilizó SHAP (SHapley Additive exPlanations). El análisis reveló qué factores influyen más en la predicción de altos ingresos:

- **Estado Civil (marital.status / relationship):** El gráfico de resumen (Beeswarm) muestra que la característica cat_relationship_Husband es el predictor más fuerte. Estar casado (True) impulsa fuertemente la probabilidad hacia >50K.
- **Edad (age) y Educación (education.num):** Ambas variables muestran una relación lineal positiva: a mayor edad y mayor nivel educativo, mayor es la probabilidad de ingresos altos.
- **Ganancias de Capital (capital.gain):** Aunque muchos individuos tienen valor 0, aquellos con ganancias de capital altas (puntos rojos en el gráfico) tienen un impacto positivo extremadamente alto en la predicción."

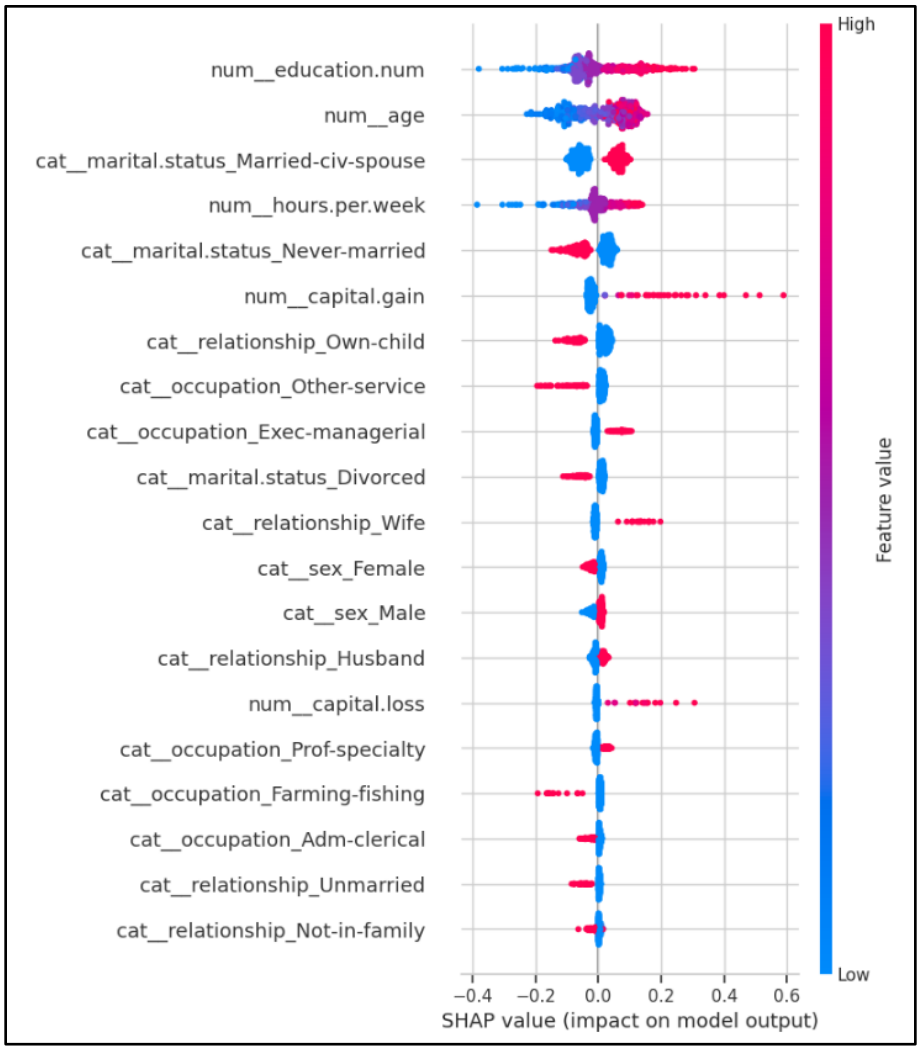

Ilustración 15 - Grafico Beeswarm

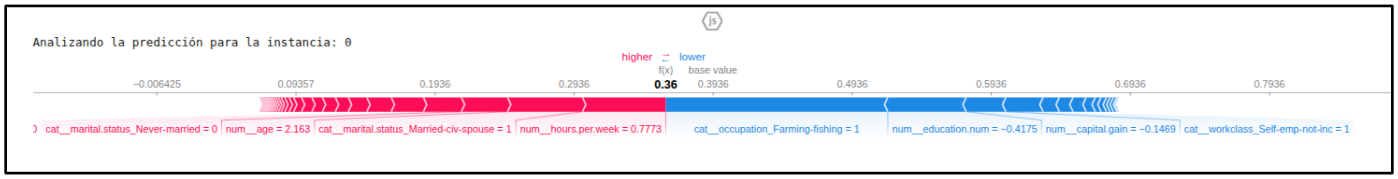

Ilustración 16 – Grafico Force Plot

5. PROPUESTA DE MEJORAS

5.1. IDENTIFICACIÓN DE LIMITACIONES

Analizamos por qué no llegamos a un F1-Score de 0.80 o más

- **Techo de Rendimiento por "Ruido" en los Datos:** A pesar de probar múltiples arquitecturas (MLP, Wide & Deep) y técnicas de balanceo, el F1-Score para la clase >50K se estancó alrededor de 0.68. Esto sugiere que existe un límite inherente en la capacidad predictiva de las variables actuales, es decir, hay personas con las

mismas características demográficas (misma edad, educación, ocupación) que tienen ingresos diferentes, lo que hace imposible una clasificación perfecta sin más datos.

- **Ineficacia de la Búsqueda Aleatoria (Random Search):** Una limitación técnica importante fue que el proceso de *refinamiento* automático (usando KerasTuner con RandomSearch y 10 intentos) no logró superar al diseño manual con Dropout. Esto indica que el espacio de búsqueda definido fue insuficiente o que la búsqueda aleatoria no logró converger al óptimo global en el tiempo asignado.

2. Mejoras Propuestas

Proponemos soluciones técnicas concretas para la Fase 3

- **Mejora 1: Optimización Bayesiana de Hiperparámetros:** En lugar de usar una búsqueda aleatoria (RandomSearch), se propone implementar **Optimización Bayesiana**. Esta técnica utiliza los resultados de los intentos previos para decidir cuál es la mejor combinación de hiperparámetros a probar después. Esto permitiría explorar el espacio de búsqueda de manera mucho más eficiente, ajustando no solo neuronas y dropout, sino también la tasa de aprendizaje y el *batch size* con mayor precisión, lo que podría desbloquear el rendimiento extra que el modelo manual con Dropout ya ha insinuado.
- **Mejora 2: Estrategia de Ensamble (Stacking o Voting):** Dado que el modelo **MLP con Dropout** y el modelo **Wide & Deep** obtuvieron resultados extremadamente similares (F1: 0.6836 vs 0.6812) pero utilizan arquitecturas internas diferentes (una profunda vs. una híbrida), es muy probable que cometan errores en instancias distintas. Se propone implementar un **Ensamble de Modelos (Stacking)**. Esta técnica entrenaría un "meta-modelo" (como una Regresión Logística simple) que tome las predicciones de ambas redes neuronales como entrada para emitir el veredicto final, lo que teóricamente reduciría la varianza y aumentaría el F1-Score final.

6. CÓDIGO FUENTE EN GITHUB

El desarrollo técnico completo de este proyecto, incluyendo los notebooks de análisis (.ipynb), los scripts de preprocesamiento y la documentación técnica asociada, se encuentra alojado y versionado en el siguiente repositorio público de GitHub:

- **Enlace al Repositorio:** https://github.com/MaidoniaN/ACIF104-Sumativa1-Grupo1

El repositorio está organizado siguiendo buenas prácticas de desarrollo y cuenta con un archivo README.md en la raíz que detalla la estructura del proyecto, lista las librerías necesarias (en requirements.txt) y proporciona instrucciones paso a paso para la instalación del entorno y la ejecución de los modelos."

7. BIBLIOGRAFÍA

- Cid R., A., Espinoza C, S., & Mattioni A., C. (2025). *ACIF 104 - Aprendizaje Maquina - Informe Formativa 3.* Santiago.
- Cid R., A., Espinoza C., S., & Mattioni A., C. (2025). *ACIF 104 - Aprendizaje Maquina - Informe Formativa 4.* Santiago.
- Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O' REILLY.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* London: The MIT Press.
- Merino, C. (2023). *Comparación de diferentes técnicas para predecir la pobreza (Trabajo de fin de grado). Universidad de Valladolid.* Obtenido de https://uvadoc.uva.es/handle/10324/63022
- Muñetón-Santa, & Manrique-Ruiz. (2023). *Predicting Multidimensional Poverty with Machine Learning Algorithms: An Open Data Source Approach Using Spatial Data. Social Sciences, 12(5), 296.* Obtenido de https://intellectum.unisabana.edu.co/handle/10818/62548
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* London: The MIT Press.
- Pincay-Ponce et al. (2022). *Analítica de datos de factores socioeconómicos que inciden en el rendimiento escolar: Revisión sistemática*. Obtenido de https://www.researchgate.net/publication/370802446
- Silva, O. (2025). *Consolidado Fase 1 y Fase 2 (Material de Clase).* Santiago: Universidad Andres Bello.
- Silva, O. (2025). *Libros Jupyter Notebook vistos en Clase.* Universidad Andres Bello.
- Smith Uldall, & Gutiérrez Rojas. (2022). *Una aplicación de aprendizaje automático en políticas públicas: Predicción de alerta temprana de deserción escolar en el sistema de educación pública de Chile. Multidisciplinary Business Review,*. Obtenido de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-39922022000100020
- Solís-Salazar, & Madrigal-Sanabria. (2022). *Una propuesta de aprendizaje automático para predecir la pobreza. Revista Tecnología en Marcha, 35(4), 84–94.* Obtenido de https://revistas.tec.ac.cr/index.php/tec_marcha/article/view/5766