

Predicción de Accidentes Cerebrovasculares Usando Perceptron Multicapa

Integrantes:

Diego Antonio Rosario Palomino

Katherine Coralie Figueroa Avalos

Xiomara Mayela Siche Eusebio



Índice

1. INTRODUCCIÓN
2. OBJETIVOS
3. CONJUNTO DE DATOS
4. PREPROCESAMIENTO
5. METODOLOGÍA
6. RESULTADOS
7. INTERPRETABILIDAD
8. DISCUSIÓN
9. CONCLUSIONES Y RECOMENDACIONES

1. Introducción

¿Qué es un ACV?

- También conocido como “stroke”.
- Ocurre cuando se interrumpe el flujo de sangre al cerebro.
- Puede causar discapacidad severa o muerte.



Motivación del proyecto

- El ACV es una de las principales causas de muerte a nivel mundial.
- La detección temprana es clave para prevenirlo.
- Los métodos actuales suelen depender de especialistas y análisis tardíos.
- El Machine Learning puede aportar una herramienta preventiva accesible.



2. OBJETIVOS

OBJETIVOS	
GENERAL	Desarrollar un sistema de Machine Learning para la predicción interpretable de accidentes cerebrovasculares (ACV).
ESPECÍFICOS	<ul style="list-style-type: none">• Preprocesar y analizar el dataset de ACV, abordando el desbalance de clases.• Entrenar y evaluar modelos de clasificación robustos para la predicción de ACV.• Implementar un análisis de interpretabilidad del modelo para explicar las predicciones.

3. CONJUNTO DE DATOS

- Fuente: *Stroke Prediction Dataset* – Kaggle con un tamaño de 316.97 kB
- Registros: 5,110 pacientes adultos.
- Atributos: 12 columnas, incluyendo datos clínicos, demográficos y estilo de vida.
- Total de registros: 5110
- Casos positivos (ACV = 1): 249 (aproximadamente 4.87% del total)
- Casos negativos (ACV = 0): 4861 (aproximadamente 95.1%)

Variable	Tipo	Descripción / Observaciones
id	Númerica	Identificador único del paciente. No aporta valor predictivo.
gender	Catagórica	Género del paciente (Male, Female, Other). Un valor atípico ('Other').
age	Númerica	Edad del paciente. Varía entre 0.08 y 82 años.
hypertension	Binaria	Indica presencia de hipertensión. Desbalanceada.
heart_disease	Binaria	Indica enfermedad cardíaca. También desbalanceada.
ever_married	Catagórica	Si el paciente ha estado casado. Valores: Yes / No.
work_type	Catagórica	Tipo de ocupación: Private, Govt_job, Self-employed, etc.
Residence_type	Catagórica	Zona de residencia: Urbano o Rural. Casi balanceado.
avg_glucose_level	Númerica	Nivel promedio de glucosa. Con presencia de outliers.
bmi	Númerica	Índice de masa corporal. Contiene valores faltantes (N/A).
stroke	Binaria	Variable objetivo. Muy desbalanceada (5% positivos).

TABLE I: Variables presentes en el dataset y principales observaciones

4. METODOLOGÍA

Se utilizó un **Perceptrón Multicapa (MLP)** con dos capas ocultas:

- Capas ocultas: 64 y 32 neuronas
- Activación: **ReLU** en capas ocultas, **Sigmoide** en la capa de salida

Entrenamiento con función de pérdida:

- **Binary Crossentropy** (adecuada para clasificación binaria)

Regularización empleada:

- **Activation Dropout y Excitation Dropout**
- Objetivo: evitar sobreajuste y promover rutas de decisión más distribuidas

```
1  # activation_dropout.py
2  import torch
3  import torch.nn as nn
4
5  class ActivationDropout(nn.Module):
6      def __init__(self, base_retain_prob=0.5):
7          super().__init__()
8          self.P = base_retain_prob
9
10     def set_retain_prob(self, new_P):
11         self.P = new_P
12
13     def forward(self, x):
14         if not self.training:
15             return x
16
17         shape = x.shape
18         x_flat = x.view(x.size(0), -1)
19
20         act_sum = x_flat.sum(dim=1, keepdim=True) + 1e-8
21         p_act = x_flat / act_sum
22
23         N = x_flat.size(1)
24         numerator = self.P
25         denominator = ((1 - self.P) * (N - 1)) * p_act + self.P
26         retain_prob = numerator / (denominator + 1e-6)
27
28         mask = torch.bernoulli(retain_prob).to(x.device)
29         x_dropped = x_flat * mask / retain_prob.clamp(min=1e-6)
30
31         return x_dropped.view(shape)
```

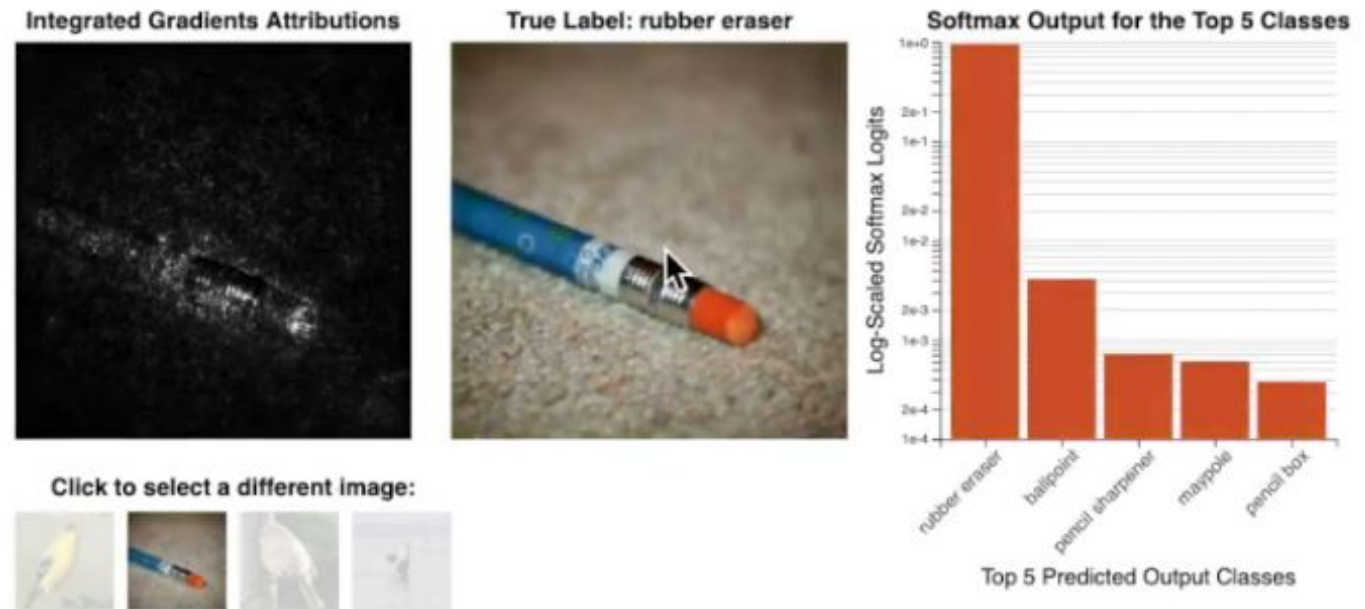
JUSTIFICACIÓN DEL MODELO

¿Por qué usar MLP?

- Capaz de capturar relaciones **no lineales** entre variables clínicas
- Fácil de combinar con técnicas modernas de regularización
- Flexible y expresivo para datos tabulares

Desafíos enfrentados

- Desbalance severo en los datos (solo 5% positivos)
- Rápido sobreajuste sin regularización
- La codificación rígida de datos limitó el impacto del dropout



sh: The predicted logits of the network on the original image. The network correctly classifies all images with

5. Resultados

Métricas del conjunto de prueba:

- **Accuracy:** 95.11% (engañoso por desbalance)
- **Recall:** 0% (no detectó casos positivos)
- **F1-score:** 0

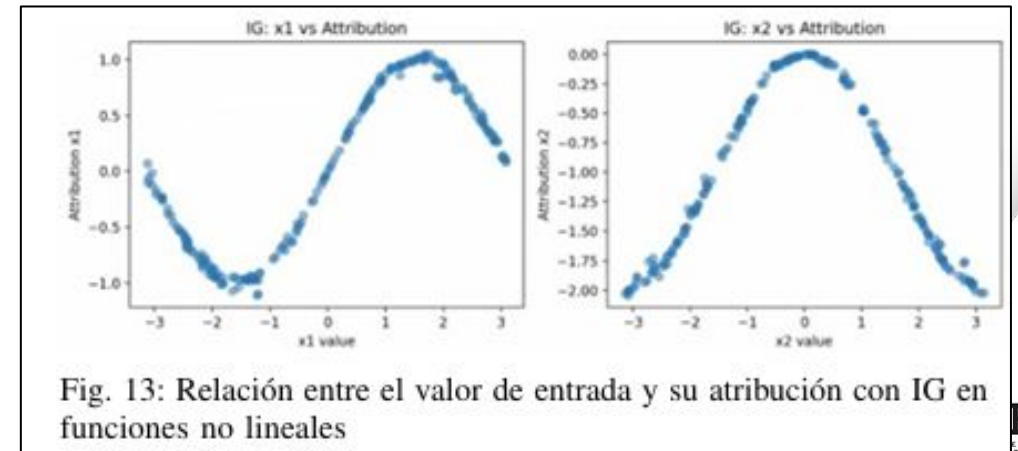
El modelo **falló en todos los positivos (25 casos)** → Todos fueron falsos negativos

Conclusión clave:

El modelo predijo bien la clase negativa, pero **ignoró la positiva**. Esto lo hace **inútil clínicamente sin ajustes**.

TABLE II: Matriz de confusión del modelo MLP en test

Clase real \ Predicción	Negativo (0)	Positivo (1)
Negativo (0)	486	0
Positivo (1)	25	0



Interpretabilidad del Modelo

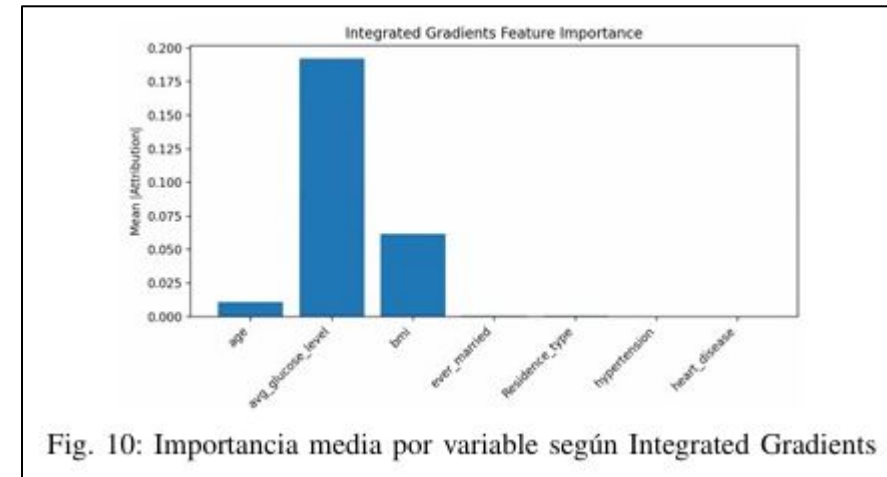
-Se utilizó **Integrated Gradients** para entender la lógica del MLP

-Variables más influyentes: **edad, glucosa, presión arterial**

-Aplicar dropout **no cambió significativamente** las rutas internas del modelo

Conclusión:

La regularización no modificó la forma en que el modelo asigna importancia a las variables → efecto limitado.



6. DISCUSIÓN

Observaciones principales:

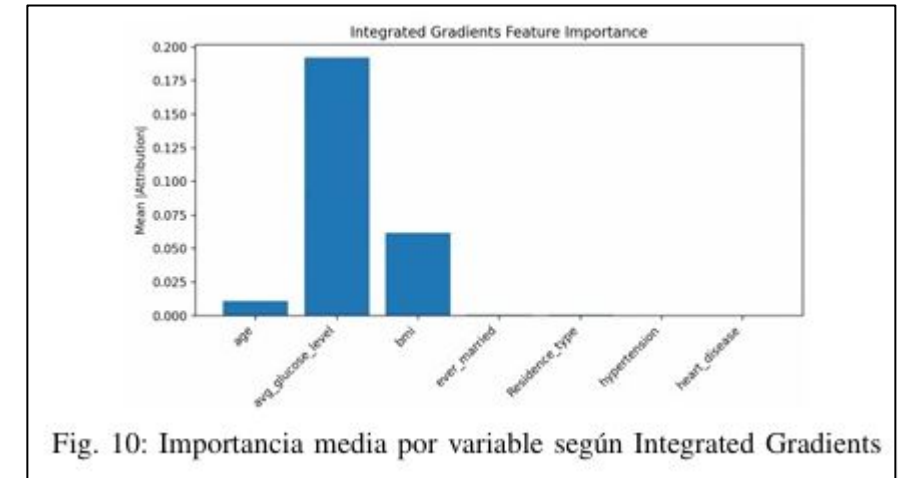
- Se usó **Integrated Gradients** para interpretar cómo el modelo (MLP) toma decisiones.
- **Activation Dropout** no generó cambios significativos en las variables más relevantes (edad, glucosa, presión arterial).
- La lógica interna del modelo se mantuvo prácticamente igual con o sin dropout.

Posibles razones:

- La codificación de variables pudo haber limitado el efecto de la regularización.
- La arquitectura del modelo y los hiper parámetros quizás no fueron óptimos.
- La estructura del dataset (datos desbalanceados) pudo restringir el beneficio esperado de la técnica.

Reflexión:

- Es crucial ajustar tanto el **preprocesamiento de datos** como los **hiper parámetros** a evaluar técnicas avanzadas de regularización.



1 ; 2 ; 3

[1 , 0 , 0] ; [0 , 1 , 0] ; [0 , 0 , 1]

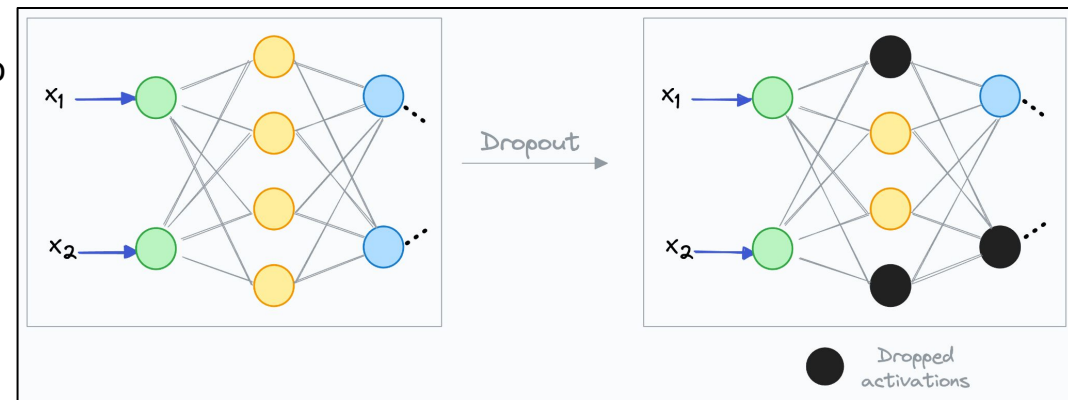
7. CONCLUSIONES

Hallazgos clave:

- **Activation Dropout** no mostró una ventaja clara sobre el entrenamiento estándar en este caso.
- La lógica del modelo, evaluada con atribuciones, se mantuvo **estable** al aplicar dropout.
- No se redistribuyó la importancia de variables ni se observó mayor generalización.

Conclusión general:






- Aunque Activation Dropout es útil en otros contextos, **su impacto aquí fue limitado**, posiblemente por decisiones metodológicas (codificación, arquitectura, dataset).



8. Trabajo Futuro

- **Probar diferentes tasas de dropout:** 0.1, 0.3, 0.5, y observar impacto.
- **Cambiar codificación** de variables categóricas a **embeddings** para permitir mayor flexibilidad al modelo.
- **Varía el random seed y partición de datos** para descartar que los resultados sean producto del azar.
- **Comparar interpretaciones con SHAP o LIME**, para verificar si otras herramientas revelan diferencias más evidentes entre modelos.

Stroke Prediction Form

Age:
Avg Glucose Level:
BMI:
Gender: 
Hypertension (0 or 1):
Heart Disease (0 or 1):
Ever Married: 
Work Type: 
Residence Type: 
Smoking Status: 

Prediction: **Stroke**
Probability: 0.7621

Gracias

