Universidad Nacional de San Martín

Tecnicatura Superior en Programación Informática

MATEMÁTICA III

Trabajo Práctico Final

Segundo Cuatrimestre 2024

Alumno: Juan Ignacio Veloso

Fecha de entrega: 01/11/2024

Indice

Parte 1: Análisis de la Base de Datos

El dataset proporcionado contiene 17000 registros y 7 columnas. A continuación, detallo cada columna y su descripción.

1. Descripción de las columnas:

Edad: Variable continua que indica la edad del paciente en años.

Hipertensión: Variable discreta (binaria) que indica si el paciente tiene hipertensión
(1 = Sí, 0 = No)

Problemas Cardíacos: Variable discreta (binaria) que indica si el paciente tiene problemas cardíacos
(1 = Sí, 0 = No)

IMC: Variable continua que representa el índice de masa corporal del paciente.

HbA1c: Variable continua que indica el nivel promedio de azúcar en sangre en los últimos 3 meses (hemoglobina glicosilada)

Glucosa: Variable continua que indica el nivel de glucosa en sangre en ayunas.

Diagnóstico: Variable objetivo discreta (binaria) que indica si el paciente ha sido diagnosticado con diabetes (1 = Sí, 0 = No)

2. Análisis de Correlaciones:

Las características más influyentes para el diagnóstico de diabetes (de mayor a menor correlación) son:

Glucosa mg/dl: 0.42

HbA1c mmol/mol: 0.40

Edad: 0.26

IMC: 0.21

Hipertensión: 0.20

Problemas Cardíacos: 0.17

La glucosa y la hemoglobina glicosilada (HbA1c) muestran las correlaciones más altas con el diagnóstico de diabetes, lo cual es coherente dado que ambos son indicadores clave en la evaluación de esta enfermedad.

3. Análisis de Factibilidad:

El dataset es adecuado para entrenar una red neuronal de clasificación. El objetivo del modelo será predecir el diagnóstico de diabetes en función de las características biométricas y de salud de los pacientes (columna "Diagnóstico"). Las correlaciones identificadas sugieren que algunas características, como los niveles de glucosa y HbA1c, tienen una relación significativa con el diagnóstico, lo que hace viable la tarea.

4. Datos Atípicos: y Limpieza de Datos:

Tras realizar un breve análisis del dataset, se concluye que no existen datos atípicos, ya que todos los valores se encuentran dentro de rangos clínicos válidos según normativa médica. Con esto, aseguro que la integridad de la información se preserve, lo cual es esencial para un análisis preciso y relevante.

Inicialmente, el dataset contaba con 100,000 registros, pero presentaba salidas desbalanceadas, con 91,500 casos en 0 y solo 8,500 en 1. Para abordar este desbalance y evitar sesgos en el modelo, realicé una limpieza que equilibrara las cantidades, logrando así una representación similar entre ambas salidas y un conjunto de datos adecuado para un análisis confiable.

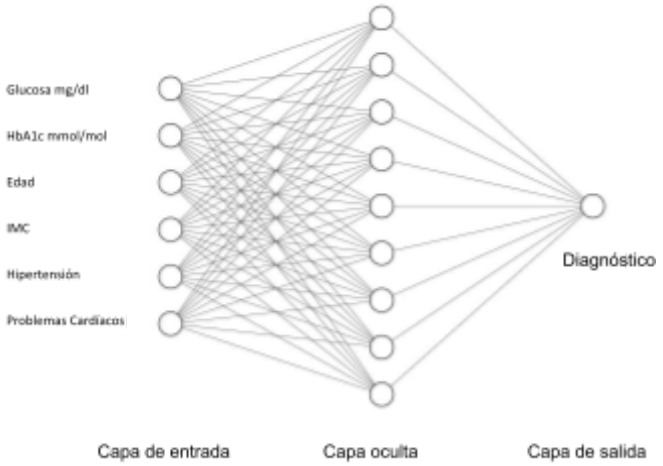
5. Transformaciones Preliminares:

Las variables continuas como IMC, glucosa y HbA1c tienen diferentes rangos. Normalizarlas asegura que no influyan desproporcionadamente en los modelos de aprendizaje automático. Es por eso que considero apropiado realizar la conversión.

Parte 2: Desarrollo de la Red Neuronal sin Sklearn

Arquitectura de la Red Neuronal

La red neuronal que hemos diseñado consta de tres capas:



Capa de Entrada:

Número de Neuronas: 6

Descripción: Esta capa recibe las entradas del modelo, que corresponden a las características del conjunto de datos sobre diabetes, como la edad, hipertensión, problemas cardíacos, IMC, HbA1c y glucosa. Cada neurona en esta capa representa una de estas características.

Capa Oculta:

Número de Neuronas: 9

Descripción: Esta capa se encarga de realizar transformaciones en los datos de entrada. La elección de 9 neuronas permite una representación más compleja de las características combinadas, ayudando a la red a aprender patrones en los datos.

Capa de Salida:

Número de Neuronas: 1

Descripción: Esta capa produce la salida final de la red, que representa la predicción del diagnóstico de diabetes (0 o 1).

Funciones de Activación

Capa Oculta:

Se utiliza la función de activación ReLU (Rectified Linear Unit). La función ReLU es preferida en la capa oculta debido a su capacidad para manejar la no linealidad en los datos. Además, ReLU es computacionalmente eficiente y ayuda a evitar el problema del desvanecimiento del gradiente, lo que permite que la red aprenda de manera más rápida y efectiva.

Capa de Salida:

Se utiliza la función de activación Sigmoide. La función sigmoide es adecuada para la capa de salida en un problema de clasificación binaria, ya que transforma la salida en un rango entre 0 y 1, lo que permite interpretar la salida como una probabilidad. Esto es particularmente útil para el diagnóstico de diabetes, donde se busca predecir la probabilidad de que un paciente tenga la enfermedad.

Overfitting:

Mi modelo no presenta overfitting, ya que el desempeño en el conjunto de validación se mantiene estable y cercano al desempeño en el conjunto de entrenamiento. Esto indica que el modelo no está capturando ruidos o patrones específicos de los datos de entrenamiento que afecten su capacidad de generalización. Las métricas de error o pérdida en ambos conjuntos no muestran una brecha

significativa, lo que sugiere que el modelo es capaz de generalizar correctamente sin ajustarse en exceso a los datos de entrenamiento.

Parte 2: Desarrollo de la Red Neuronal con Sklearn

Curvas de Entrenamiento y Validación:

Similitudes: Ambas redes neuronales muestran curvas de entrenamiento y validación similares, alcanzando un nivel de convergencia parecido en términos de precisión y error. Esto se debe a que se utilizaron arquitecturas y parámetros similares en ambas implementaciones.

Diferencias: La red neuronal en scikit-learn muestra una convergencia más rápida y presenta menos fluctuaciones en las curvas de error y precisión. Esto se debe a que scikit-learn aplica optimizaciones internas y algunas técnicas de regularización que ayudan a estabilizar el entrenamiento, mientras que la implementación en NumPy requiere ajustes adicionales.

Tiempo de Ejecución:

Similitudes: Ambas implementaciones procesan la misma cantidad de datos y siguen una arquitectura similar, por lo que el tiempo de ejecución se ve influido por el tamaño de los datos y el hardware disponible.

Diferencias: La implementación en scikit-learn es significativamente más rápida que la versión en NumPy. Esto se debe a que scikit-learn utiliza librerías de bajo nivel optimizadas, como BLAS y LAPACK, para realizar operaciones de álgebra lineal, lo que reduce considerablemente el tiempo de entrenamiento en comparación con la versión en NumPy.

Conclusión:

Este trabajo práctico me permitió entender en profundidad el funcionamiento de las redes neuronales, especialmente al construir una desde cero en NumPy. Hacer cada paso manualmente, desde la inicialización hasta el ajuste de parámetros, me ayudó a ver los detalles de cada fase y a comprender mejor los desafíos de entrenar una red sin ayuda de optimizaciones automáticas.

Comparando este proceso con el uso de scikit-learn, noté que, aunque construir una red manualmente aporta gran valor educativo, usar una librería es mucho más eficiente y práctico. Scikit-learn permite ajustar modelos rápidamente y manejar los parámetros y la escalabilidad con facilidad, lo que resulta fundamental en aplicaciones prácticas.