Trabajo práctico Final:
Red Neuronal

Alumnos:

Fernandez Tadeo

Badino Ivan

Materia:

Matemática III

Docentes:

Prudente Tomas

Bompensieri Josefina

Universidad Nacional de San Martín

Año: 2025

Elección de la Base de Datos:

En esta ocasión elegimos la base de datos “Diabetes prediction dataset” subida a la página de Kaggle hace dos años.

Los motivos que nos llevaron a elegir consisten principalmente en que, en principio, se ve sencilla y corta, pero aún así reúne varios tipos de datos a los cuales se debe transformar y trabajar en conjunto. Además, presenta un objetivo muy claro el cual es predecir si una persona tiene o no diabetes, premisa la cual también está incluida para cada una de las cien mil muestras que este documento contiene.

https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset/data

Análisis General de los Datos:

- Gender: es una variable categórica que nos indica el género de la muestra, con opciones entre male, female y other.

- Age: es una variable cuantitativa continua que nos indica la edad de la muestra (en años).

- Hypertension: es una variable binaria que nos indica si la muestra sufre de hipertensión.

- Heart_disease: es una variable binaria que nos indica si la muestra tiene alguna enfermedad cardíaca.

- Smoking_history: es una variable categórica que nos indica el historial de tabaquismo de la muestra, con sus posibles resultados “never” (nunca), “former” (fumó activamente en un pasado), “current” (fuma activamente), “ever” (fumó alguna vez), “not current” (no fuma actualmente), “no info” o “unknown” (no hay información registrada)

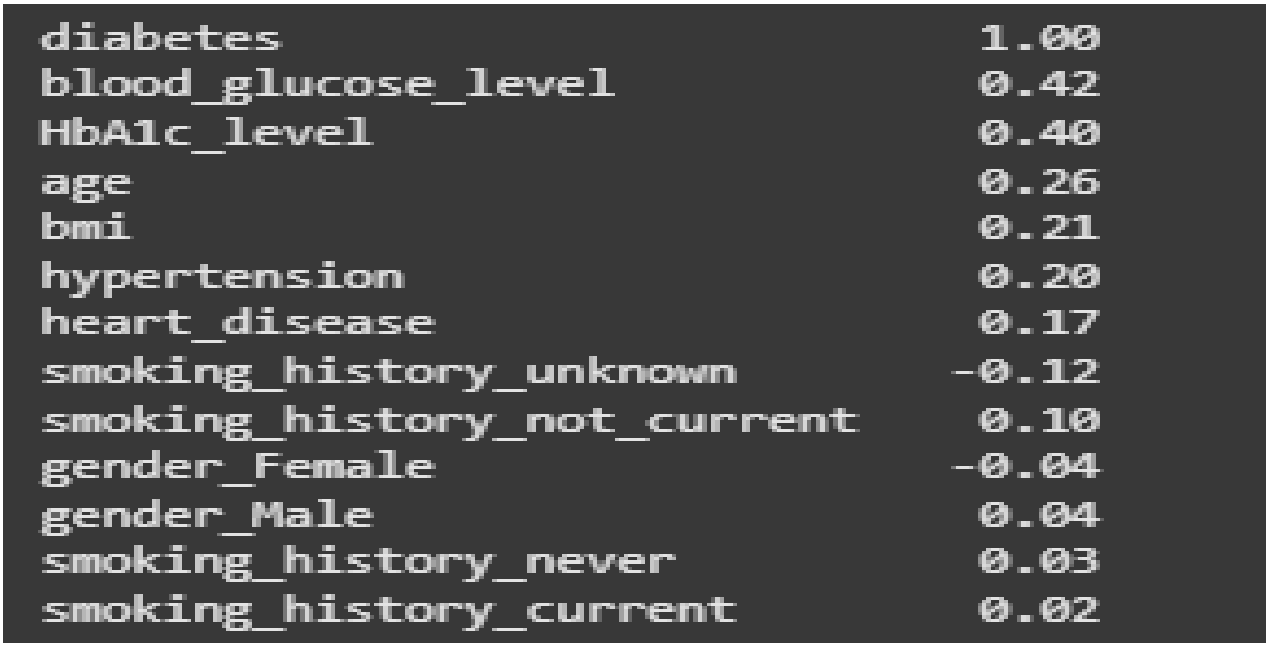
- Bmi: es una variable cuantitativa continua que nos indica la masa corporal de la muestra

- hbA1c_level: es una variable cuantitativa continua que nos indica el nivel promedio de hemoglobina glicosilada en sangre de la muestra (se utiliza para medir el nivel promedio de glucosa en un periodo de 2 a 3 meses)

- Blood_glucose_level: es una variable cuantitativa discreta que nos indica el nivel de glucosa en sangre en mg/dL de la muestra (nivel de glucosa al momento de tomar la muestra)

- Diabetes: es una variable binaria y nuestro objetivo, indica si la muestra tiene o no diabetes

Análisis de correlaciones:



Para el análisis de correlaciones utilizamos Pearson, centrado en diabetes, para ver qué tanto impacto positivo o negativo tiene cada valor sobre nuestra columna objetivo, siendo la cercanía con 0 su menor relevancia y con 1 o -1 su mayor relevancia.

A simple vista destaca que las características con mayor relevancia de impacto hacia tener diabetes son las relacionadas con el azúcar en sangre, con una diferencia tan pequeña que nos indica que no tiene diferencia el momento en la consumición de la misma. Seguido por la edad, el índice de masa corporal y podemos mencionar también la hipertensión, lo cual acompaña a las investigaciones médicas alrededor de dicha condición. Sobre esa línea, también podemos ver que una persona fume no tiene prácticamente influencia sobre esta enfermedad y, aunque con valores irrisorios, las mujeres tienen menos probabilidad de tener diabetes.

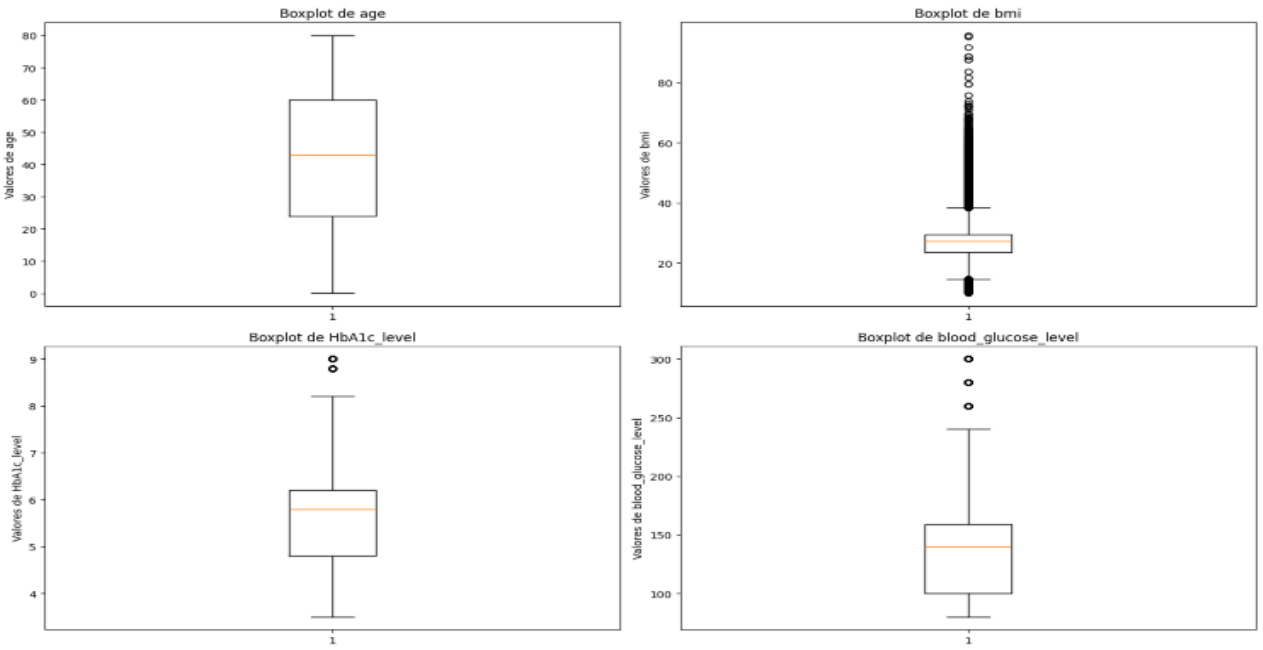
Cabe destacar que para este punto realizamos OneHot Encoding de las columnas smoking_history y gender, no sin antes agrupar los valores redundantes de smoking_history para una mayor limpieza y entendimiento, además de eliminar de gender el valor other, ya que su cantidad era despreciable.

Análisis de factibilidad:

El propósito de esta red se centra en identificar si una persona tiene diabetes con la mayor precisión posible, basándonos en datos relacionados a la salud tales como glucosa en sangre, edad, género, problemas con el corazón, historial de fumador y masa corporal. Gracias a esto, se puede detectar si una persona está en camino de tener diabetes basándonos en las muestras anteriores, así como también tomar ejemplo de cómo puede evitarse.

Consideramos que la base de datos es altamente adecuada para una red neuronal de clasificación. Nuestra opinión se sostiene por el hecho de que tiene una columna objetivo muy clara y viable, como lo es la existencia de diabetes. Además, se sustenta al ver que posee características con una alta relevancia en torno a nuestro objetivo, así como también algunas que a primera vista podrían parecer importantes pero que resultan no serlo como uno esperaría. Aunque sí es cierto que tal vez hayan características que falten para hacerla una red neuronal profesional, creemos que tiene las más relevantes.

Datos Atípicos y Limpieza de Datos:



En cuanto a limpieza de datos, ya mencionamos previamente los cambios a gender y smoking_history. Para revisarlos rápidamente, quitamos las muestras con categoría “other” dentro de gender ya que eran ínfimas, y juntamos valores dentro de smoking_history que consideramos redundantes y entorpecían nuestra red.

En cuanto a datos atípicos, encontramos muchos, muchos de ellos dentro del índice de masa corporal (bmi) y algunos en las mediciones de glucosa, y nuestra resolución es la siguiente:

Decidimos no eliminar los datos atípicos de nuestra base de datos, y la justificación se encuentra en la naturaleza misma de la diabetes. Los valores extremadamente altos de masa corporal, glucosa y hemoglobina no son errores de medición, sino indicadores altamente relevantes al desarrollo de la diabetes, fundamentado en la información que se encuentra disponible en internet, además de los resultados de nuestra red. Consideramos que quitar estos datos sería perder información clave para entrenar al modelo con datos relevantes. Nuestra decisión es conservarlos, pero estandarizar de forma beneficiosa para ellos (explicados a continuación).

Transformaciones Preliminares:

Este apartado ya lo hemos cubierto a medida que desarrollamos los anteriores, pero hagamos un repaso.

Conversión de datos categóricos: Para que la red pueda procesarlos debimos convertirlos a valores numéricos, por eso utilizamos la técnica oneHot-Encoding para separarlos en columnas y otorgarles valores binarios True/False según corresponda, luego reemplazamos estos valores con 1 y 0, aunque somos conscientes que no es realmente necesario ya que el lenguaje los interpreta

Eliminación del valor “other” dentro de la columna de gender: El motivo es la baja, ínfima, pequeña cantidad de muestras que coinciden con este valor, juntando un poco sorprendente total de 18 en 100.000 muestras, sin aportar ninguna relevancia.

Agrupamiento de valores redundantes en columna smoking_history: Consideramos que había valores que no hacían prácticamente diferencia del otro, como “former”, “not_current” y “ever”. Basados en esto, decidimos juntar estos valores en uno solo llamado “not_current”, así los datos no se pierden y dan una mayor simpleza.

Aplicamos normalización: Como nuestros datos tienen escalas muy diferentes, algunos datos podrían comerse totalmente a otros, haciendo que nuestra red neuronal los desprecie de entrada, arruinando su proceso de aprendizaje, pudiendo hacer que tarde mucho más en ajustarse.

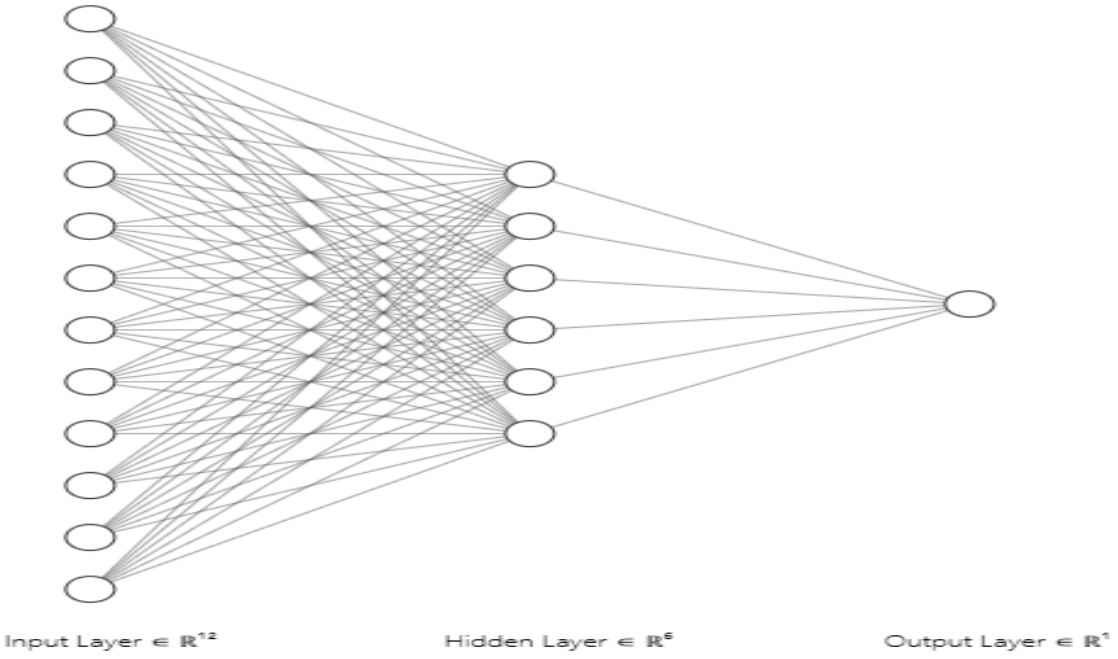
La elección del método no fue sencilla y se necesito mucha busqueda de informacion, pero una vez concluido, elegimos utilizar RobustScaler. Esta decisión surge de darnos cuenta la inmensa cantidad de datos atípicos que teníamos en categorías imperativamente relevantes, junto a nuestra decisión de no eliminarlos con los argumentos previamente presentados.

La forma StandarScaler es altamente sensible a la presencia de valores atípicos debido a su uso de promedio y desviación estándar. De haberla utilizado nos hubiera comprimido demasiado la diferencia entre valores normales, reduciendola desproporcionadamente.

La forma MinMaxScaler, por dar otro ejemplo de formas que consideramos, también es altamente sensible a los valores atípicos, ya que utiliza de forma directa los valores mínimos y máximos de la columna. De haberla utilizado se reduciría drásticamente la variabilidad de los datos no atípicos, convirtiéndolos prácticamente en inútiles.

Al utilizar RobustScaler usamos la mediana y el rango intercuartílico, que no se ven tan afectados por valores atípicos, de forma que la distribución de datos no se ve afectada por estos valores extremos, lo cual genera que la red pueda beneficiarse de estos valores clínicamente relevantes sin perjudicar la escala general de valores, optimizando el proceso de aprendizaje

Arquitectura de la red:



Nuestra red neuronal es bastante simple. Consta de tres capas, una para los inputs con un total de 12 neuronas representando el total de inputs que recibe, una capa oculta formada por 6 neuronas, esta elección fue por proporcionar un estándar que no falle pero nos gustaría probar con incluso menos neuronas debido a la sencillez de los inputs, y una neurona de salida que será la encargada de determinar si efectivamente la muestra tiene o no diabetes.

La capa oculta tendrá la función de activación ReLU para evitar negativos, mientras que la neurona de salida utilizara Logistic, la cual sitúa el valor entre 0 y 1, convirtiéndola en una elección perfecta para una red de salida binaria como en nuestro caso

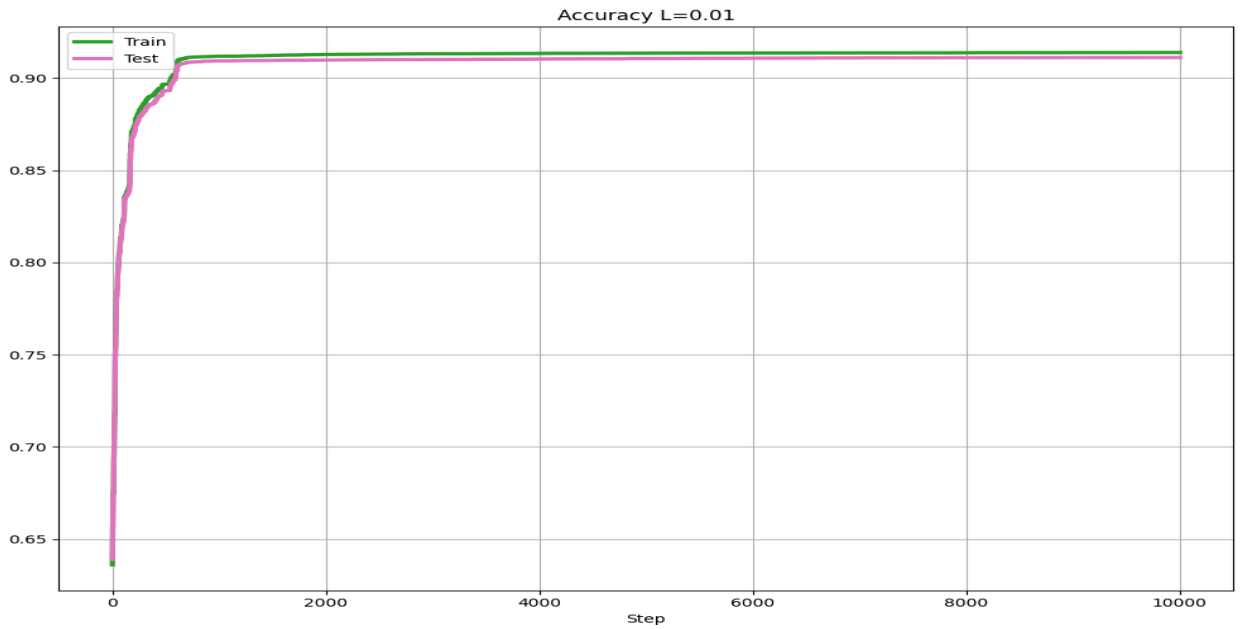
Implementación en numpy:

En el código ya hay comentarios explicando celda a celda que es lo que vamos haciendo, pero en el siguiente escrito complementaremos.

Empezamos haciendo las importaciones y trayendo como DataFrame nuestro CSV con las modificaciones realizadas en la seccion “analisis de datos” de la cual ya hemos hablado, dividimos la cantidad de datos de aprendizaje y la cantidad de datos de testeo y los transponemos, para no tener que hacerlo luego, y procedemos a inicializar las neuronas.

Continuando, creamos las funciones de activación Relu y Logistic y sus derivadas, además de la función forwardProp para crear y activar los datos, a modo de prueba hicimos un testeo de la red sin ajustar los pesos y sesgos para comprobar que funciona hasta ese punto. Sobre el mismo hilo realizamos el backward propagation traído desde la unidad teórica para utilizarlo luego en nuestro descenso de gradiente, el cual, en base a un ritmo de aprendizaje L y una cierta cantidad de iteraciones que optimizamos, entrenará los pesos y sesgos para lograr la mejor precisión posible.

Entrenamiento y evaluación:



Al realizar los entrenamientos y probando distintos ritmos de aprendizaje e iteraciones, concluimos que lo mejor es un ritmo de L=0.01 y 10000 iteraciones.

En este apartado hay mucho que debe comentarse, empezando por los gráficos que todos tienden a aprender muy rápido y con poco ruido. Luego de analizarlo, creemos que esto puede surgir derivado a los datos de nuestro DF, que son en su extensa mayoría datos binarios, por lo que no hay mucha variación en cada uno, así que la red neuronal puede identificar rápidamente la importancia de unos u otros.

Por lo antes mencionado, nos encontramos atrapados en una situación donde tenemos muchas opciones de combinaciones para elegir, las cuales también probamos a mano utilizando la red neuronal y no solo las gráficas. Eso nos llevó a tomar la polémica decisión de utilizar ese ritmo de aprendizaje con toda la extensa cantidad de iteraciones.

Análisis de overfitting:

Para continuar con lo que nos quedó pendiente de explicación, desarrollaremos en este punto.

Al ver los gráficos podría parecer una locura elegir para un ritmo de entrenamiento avanzado esta extensa cantidad de iteraciones, y somos conscientes de ello. Sin embargo, aquí es donde nuestras pruebas manuales entran en juego, ya que al probar con iteraciones bajas, más bajas y mucho más bajas nos dimos cuenta que no hay cambios reales en el overfitting, siempre se mantiene en el umbral de los 0.0020 a 0.0030 de mayor precisión de los testeos con respecto a las pruebas.

No obstante, lo que sí pudimos observar es un decaimiento realizado por parte de la precisión de los testeos, además de las pruebas, al incluir menos cantidad de iteraciones aunque el overfitting se mantenga constante.

Fue por estas observaciones que decidimos elegir una cantidad elevada de iteraciones, a su vez tuvimos en cuenta la simpleza de nuestra red neuronal que realiza las cosas a una velocidad galopante, lo que nos llevó a priorizar la maximización de resultados por sobre el procesamiento excesivo.

Comparación de rendimiento con scikit-learn:

La red neuronal creada con scikit consiguió una increíble cantidad de 0.96 de precisión y sin overfitting.

A pesar de tener misma cantidad de neuronas, mismas entradas, misma semilla de generación aleatoria y ser prácticamente idéntica, scikit-learn ha demostrado un rendimiento superior, lo cual nos llevó a preguntarnos el por qué y realizar pruebas.

Decidimos reducir drásticamente la cantidad de iteraciones en nuestro modelo para ver si es que en algún momento decrecía, sin darle importancia a los gráficos. Esto no funciono, asi que lo devolvimos a sus valores originales, lo cual nos dejó pensando que tal vez

scikit-learn tiene una optimización absurda de sus cuentas o elecciones, lo cual nosotros tal vez no tuvimos al hacerlas “a mano”, lo único que puede comentarse a favor de nuestra red es que ejecuta más rápido que la de scikit. En definitiva, la suya es superior.

Conclusión:

Hacer una red neuronal, por más simple que resultara, fue un proceso tanto divertido como estresante cuando nos quedamos sin dirección o intentamos optimizar de diversas maneras, y creemos que la parte de analizar los datos del CSV para depurarlo fue una pieza clave en el armado de la red y prácticamente tan fundamental como la construcción de la misma.

Quedamos derrotados ante scikit-learn. Evidentemente está mucho más optimizado de lo que nosotros pudimos hacer, lo cual atrae mejores resultados, aunque de haber tenido más tiempo disponible nos hubiese encantado ir probando con más o menos neuronas y realizar pruebas sobre las mismas para buscar un resultado similar a scikit. Sin embargo, el experimentar hacer la red manualmente fue altamente pedagógico y nos ayudó a cimentar los conocimientos que recibimos de las clases teóricas, además de entender como es que funciona una red por dentro.