# Instituto Tecnológico y de Estudios Superiores de Monterrey

## Escuela de Ingeniería y Ciencias

Inteligencia Artificial Avanzada para la Ciencia de Datos I

Ingeniería en Ciencias de Datos y Matemáticas

## Módulo 2: Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución
## Reto


**Profesores**
Profe Adrián

**Equipo 4**

Renata Vargas A01025281


Monterrey, Nuevo León. 10 de septiembre de 2023

# 1. Introducción

En este reporte técnico, se presenta un análisis comparativo entre dos modelos para la clasificación de cáncer de mama: un modelo de perceptrón y una red neuronal. El objetivo principal es evaluar y comparar el rendimiento de ambos modelos en la tarea de clasificación binaria de tumores malignos y benignos. Para abordar esta problemática, utilicé un conjunto de datos reconocido en la comunidad de aprendizaje automático: el Conjunto de Datos de Cáncer de Mama de Wisconsin (Breast Cancer Wisconsin dataset), que contiene características derivadas de imágenes digitalizadas de biopsias de tejido mamario y está etiquetado con dos clases: maligno (1) y benigno (0).

# 2. Modelos

Implementé dos modelos de aprendizaje: un perceptrón simple y una red neuronal. A continuación sus desarrollos y la comparación entre estos dos modelos para entender como funcionan y la diferencia de complejidad entre ellos. Con el objetivo de comparar estos modelos, ajustaré los modelos para que tengan la mayor cantidad de parámetros iguales: los dos usarán 100 epochs de entrenamiento y contarán con el mismo porcentaje de la base de datos para entrenarse.

## 2.1. Perceptrón

El modelo del perceptrón es una forma simple de red neuronal que se utiliza para problemas de clasificación binaria. En su forma más básica, consiste en una sola neurona artificial que toma entradas, realiza una suma ponderada de esas entradas y luego aplica una función de activación para determinar la salida del modelo.

Dividí los datos de manera aleatoria en dos grupos: el de entrenamiento que contiene el 90 % de los datos y el de prueba que contiene el 10 % de los datos. Utilizaré una función de activación específica para clasificación binaria que es la de escalón que asigna 1 si la entrada es mayor o igual a cero y 0 si es menor que cero. Después se inicializan los pesos y el sesgo de manera aleatoria

#### Entrenamiento

El modelo del perceptrón se entrena de la siguiente manera: para cada epoch de entrenamiento, se pasa a través de los ejemplos de entrenamiento uno por uno y se realiza el siguiente proceso:

- Se calcula la suma ponderada de las características de entrada multiplicadas por los pesos y se le suma el sesgo. Esto se hace mediante la siguiente fórmula:

$$z = \sum_{i=1}^{n} (X_i \cdot W_i) + b$$

Donde:

$z$ es la entrada neta

$X_i$ es el valor de la característica de entrada $i$

$W_i$ es el peso asociado a la característica de entrada $i$

$b$ es el bias

- Luego, se aplica la función de activación (función escalón) a $z$ para obtener la predicción ($y_{\text{pred}}$) del perceptrón.
- Se compara la predicción ($y_{\text{pred}}$) con la etiqueta de clase verdadera ($y_{\text{true}}$) del ejemplo y se actualizan los pesos y el sesgo utilizando la regla de aprendizaje del perceptrón.
- Actualización de pesos y sesgo: La regla de aprendizaje del perceptrón se utiliza para actualizar los pesos y el sesgo en cada iteración de entrenamiento. La regla de aprendizaje se basa en el error entre la predicción y la etiqueta verdadera:

$$W_i^{(nuevo)} = W_i^{(viejo)} + \text{Tasa de Aprendizaje} \cdot (y_{\text{true}} - y_{\text{pred}}) \cdot X_i$$

$$b^{(nuevo)} = b^{(viejo)} + \text{Tasa de Aprendizaje} \cdot (y_{\text{true}} - y_{\text{pred}})$$

Donde:

$W_i^{(nuevo)}$ es el nuevo peso para la característica $i$

$W_i^{(viejo)}$ es el peso anterior para la característica $i$

$b^{(nuevo)}$ es el nuevo sesgo

$b^{(viejo)}$ es el sesgo anterior

La **Tasa de Aprendizaje** es un hiperparámetro que controla cuánto se ajustan los pesos y el sesgo en cada iteración.

Durante el entrenamiento, se calcula la precisión del perceptrón en el conjunto de entrenamiento para registrar cómo mejora el modelo a lo largo de las épocas.

## 2.2. Red Neuronal

Para la red neuronal completa, igual que para el perceptrón, dividí los datos de manera aleatoria en dos grupos: el de entrenamiento que contiene el 90 % de los datos y el de prueba que contiene el 10 % de los datos.

La arquitectura de la red neuronal es la siguiente:

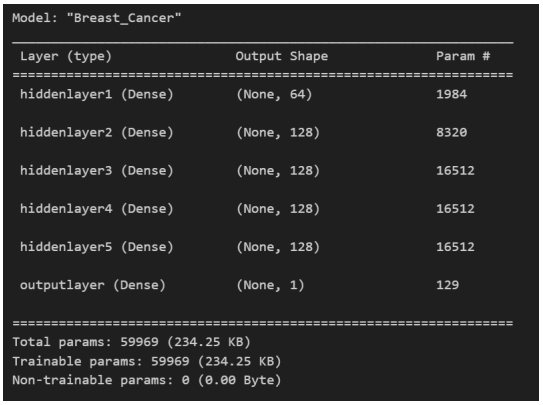


Figura 1: Arquitectura de red neuronal

Está constituida por

- 1 capa de entrada con 64 neuronas. La función de activación para la primera capa es Rectified Linear Unit (ReLU), inicialización de pesos HeUniform (que sirve para controlar la aleatoriedad de los pesos iniciales aplicando una formula con base en la cantidad de unidades de entrada y salida de la capa. Se calcula de la siguiente manera: $limite = sqrt(6/(unidadesdeentrada + unidadesdelasalida)))$
- 4 capas ocultas con 128 neuronas. Por fines de experimentación decidí darle una función de activación diferente a cada capa (en lugar de darle la misma o no variar tanto entre capas). Sin embargo por ser un problema relativamente sencillo, no tuvo un cambio significativo. Las funciones de activación que utilicé fueron: Sigmoide (sigmoid), Tangente Hiperbólica (tanh), Unidad Lineal Exponencial (elu) y Swish (swish).
- 1 capa de salida con 1 neurona. La función de activación para esta última capa es Sigmoide debido a que estamos trabajando con clasificación binaria. Es decir, buscamos un 1 o un 0.

**Entrenamiento**

Utilicé dos optimizadores para la red neuronal (nada más para experimentar) el optimizador Adam y el optimizador Stochastic Gradient Descent (SGD) con una tasa de aprendizaje de 0.0001. Estos fueron los resultados de cada uno. Por lo cual me quedé con el Adam.

| | **adam** | **sgd** |
|---|---|---|
| Accuracy | 0.9473 | 0.7368 |
| Loss | 0.1589 | 0.6345 |

La función de pérdida es la misma del perceptrón: binary crossentropy, por ser un problema de clasificación binaria.

# 3. Evaluación de los modelos

Posteriormente a la implementación de los dos modelos estos fueron los resultados que obtuve para el set de entrenamiento:

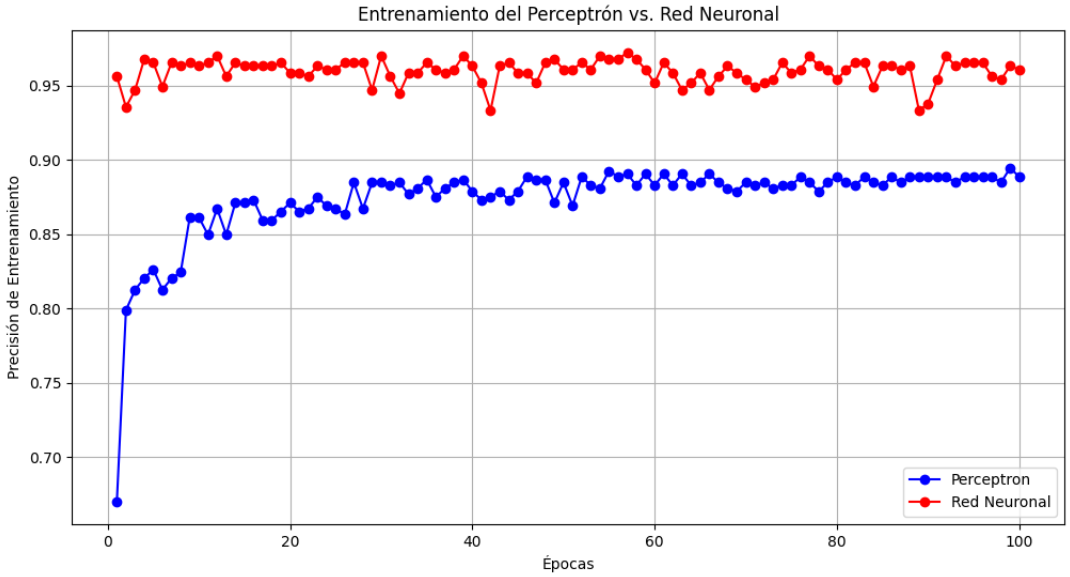


Figura 2: Perceptrón vs. RN

Podemos observar que la Red Neuronal tuvo un mejor desempeño y el perceptrón a pesar de no equiparar la precisión de la Red Neuronal, tuvo muy buenos resultados. Esta diferencia se puede atribuir a la diferencia de complejidad de cada modelo. El perceptrón es un modelo que funciona muy bien con datos linealmente separables. De igual manera si comparamos las dos en cuestiones de arquitectura, la red neuronal cuenta con varias capas que funcionan de manera simplificada como filtros de información. Esos filtros evalúan la importancia de la información que pasa para al final tener la información más valiosa; mientras que el perceptrón funciona con solamente una capa, lo cual traduciéndolo en el Perceptrón, solamente cuenta con uno de esos filtros. Lo que podemos observar es que ese único filtro funciona bastante bien.

## 3.1. Predicciones

Para cada modelo, obtuve un set de predicciones utilizando el set de prueba, lo que resultó en una lista de unos y ceros. Para evaluar esas predicciones, saqué las métricas básicas de evaluación: la precisión (accuracy), la matriz de confusión, la precisión, la exhaustividad (recall), la puntuación F1 y la curva ROC. Todas estas métricas las obtuve al comparar la predicción con el valor real. Estos fueron los resultados obtenidos:

| | Precisión global | Precisión | Recall | F1 score | Matriz de confusión |
|---|---|---|---|---|---|
| **Perceptrón** | 92.98% | 91.30% | 100.00% | 95.45% | [[11 4]<br>[0 42]} |
| **Red Neuronal** | 91.23% | 97.44% | 90.48% | 93.83% | [[14 1]<br>[4 38]] |

### 3.2. Bias y Varianza

Evaluar el sesgo y la varianza de un modelo de aprendizaje automático es importante para comprender su capacidad de generalización y su rendimiento con los datos. Para hacerlo, se lleva a cabo una evaluación y un cálculo de precisión en el conjunto de entrenamiento y en el de prueba. Con base en los resultados podemos categorizar estas dos mediciones. Los resultados obtenidos fueron los siguientes:

- Para el perceptrón, obtuve un sesgo alto con overfitting hacia los datos de prueba. Esto significa implica que se está ajustando demasiado a los datos de entrenamiento y no generaliza bien a datos no vistos (los datos de prueba). En otras palabras, el modelo ha aprendido a memorizar los datos de entrenamiento en lugar de aprender patrones subyacentes que puedan aplicarse a nuevos datos. De igual manera, obtuvimos una varianza baja (bien generalizada) lo que significa que las diferencias en el rendimiento entre los conjuntos de entrenamiento y prueba son pequeñas. A pesar de que el modelo se ajusta demasiado a los datos de entrenamiento, todavía es capaz de realizar predicciones bastante consistentes en el conjunto de prueba. Esto podría indicar que el modelo ha capturado algunos patrones generales en los datos.
- Para la Red Neuronal salió que tenía sesgo alto con overfitting hacia los datos de entrenamiento (a diferencia del perceptrón que tenía overfitting hacia los datos de prueba). Esto puede significar que no está capturando bien los patrones presentes en los datos de entrenamiento, lo que puede deberse a que el modelo es demasiado simple o no tiene suficientes características para representar los datos de manera efectiva. Básicamente el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a datos no vistos. Por otro lado, la varianza salió baja, lo cual significa que las predicciones del modelo son consistentes en diferentes conjuntos de datos, incluido el conjunto de prueba. Aunque el modelo no se ajusta bien a los datos de entrenamiento, las predicciones que hace son bastante estables.

## 4. Conclusión

En este reporte, llevé a cabo una comparación entre un modelo de perceptrón y una red neuronal para la clasificación de cáncer de mama. El Perceptrón, a pesar de su simplicidad, logró un rendimiento sorprendentemente bueno, lo que indica su utilidad en problemas de clasificación linealmente separables. Por

otro lado, la Red Neuronal, con su capacidad para aprender representaciones más complejas, mostró un rendimiento aún mejor, superando al perceptrón en términos de precisión y capacidad de adaptación a datos más complejos.

Me encontré con algunas dificultades en el procedimiento, entre ellas la evaluación de varianza y sesgo a través de validación cruzada. Me hubiera gustado implementar la validación cruzada en los dos modelos para poder evaluarlos, pero no logré implementarla en el Perceptrón como me hubiera gustado y fue por lo cual recurrí a usar las condiciones que usé.

Otra dificultad con la que me topé fue la interpretación del sesgo y la varianza de la Red Neuronal. Antes de evaluarlos con código, yo había asumido con base en la complejidad de los modelos, que la RN iba a tener un sesgo practicamente inexistente por el hecho de que tiene una capacidad "superior.ª la del Perceptrón y como consecuencia iba a ser mejor. Sin embargo, al evaluarla con código, salía con un sesgo alto con overfitting a los datos de entrenamiento. Me parece que si hubiera cambiado el tamaño del set de entrenamiento, pude haber tenido mejores resultados y un sesgo más bajo. Otra solución que me hubiera gustado implementar es algún tipo de regularización para los parámetros y que funcionen como cierto limitante para evitar el overfit. O bien la implementación correcta de la validación cruzada hubiera ayudado.

A comparación del reporte anterior donde implementamos modelos de aprendizaje sin frameworks, esta experiencia fue mucho más amena y divertida porque nos permite utilizar herramientas sumamente útiles y fáciles de intuir.