ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

Ingeniería Mecatrónica

PROGRAMACIÓN AVANZADA

Enero – Junio 2025

M.C. Osbaldo Aragón Banderas

UNIDAD:

1	2	3	4	5

Actividad número: 2

Nombre de actividad:

Implementación del Perceptrón

Actividad realizada por:

Carlos Diego Dominguez López - 21030006

Guadalupe Victoria, Durango

Fecha de entrega:

18	02	2025

INDICE

Introducción

En el ámbito financiero, la toma de decisiones automatizada juega un papel crucial en la eficiencia y precisión de los procesos de evaluación de solicitudes de préstamo. Actualmente, muchas instituciones buscan optimizar este procedimiento mediante modelos de inteligencia artificial que permitan clasificar solicitudes de manera rápida y precisa, reduciendo la intervención humana y minimizando posibles sesgos.

En este contexto, el presente trabajo tiene como objetivo implementar y entrenar un perceptrón en Python para clasificar solicitudes de préstamo en dos categorías: aprobadas (1) o rechazadas (0). Para ello, se utilizarán cuatro variables clave que influyen en la decisión financiera: puntaje de crédito, ingresos mensuales, monto del préstamo solicitado y relación deuda/ingresos. El perceptrón se entrenará con un conjunto de datos históricos, ajustando sus pesos de manera que pueda aprender a diferenciar patrones en las solicitudes y mejorar su precisión en la clasificación.

Este ejercicio permite reforzar conceptos fundamentales del aprendizaje supervisado, incluyendo el ajuste de pesos en redes neuronales simples y la implementación de modelos de clasificación binaria. A través de este desarrollo, se busca comprender mejor el funcionamiento de los perceptrones y su aplicabilidad en problemas reales, demostrando su capacidad para optimizar procesos de decisión en el sector financiero.

Objetivo

Implementar y entrenar un perceptrón en Python que permita clasificar solicitudes de préstamo en aprobadas (1) o rechazadas (0), en función de variables financieras relevantes. Se busca reforzar el conocimiento en aprendizaje supervisado, el ajuste de pesos en redes neuronales simples y la implementación de modelos de clasificación binaria.

Marco Teórico

Perceptrón y su Funcionamiento

El perceptrón es un modelo de red neuronal artificial desarrollado por Frank Rosenblatt en 1958, diseñado para tareas de clasificación binaria. Se basa en la idea de que una combinación lineal de las entradas, ponderadas por coeficientes llamados pesos, puede determinar la salida de un sistema de decisión. Matemáticamente, el perceptrón se representa como una función de la forma:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{1.1}$$

El entrenamiento del perceptrón consiste en ajustar los pesos mediante la regla de aprendizaje del descenso de gradiente, basada en la comparación de la salida predicha con la salida real. A través de múltiples iteraciones sobre un conjunto de datos etiquetado, el modelo minimiza su error y mejora su precisión en la clasificación de nuevas instancias (Goodfellow, Bengio & Courville, 2016).

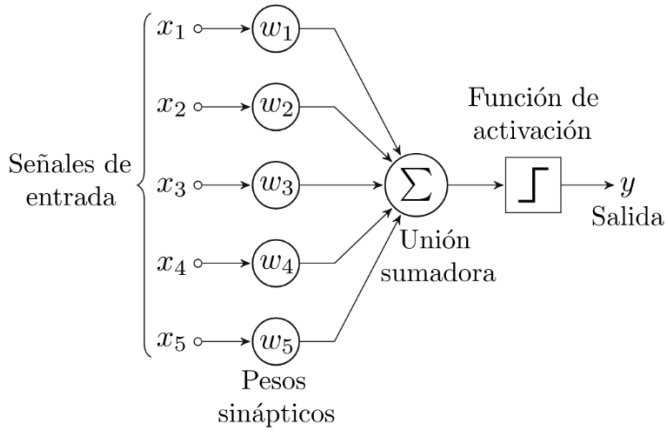

Figura 1.1 Diagrama del funcionamiento del perceptrón

Jupyter Notebook

Jupyter Notebook es un entorno interactivo de código abierto ampliamente utilizado en ciencia de datos y aprendizaje automático. Permite la escritura y ejecución de código en diversos lenguajes de programación, como Python, ofreciendo una

interfaz intuitiva para el desarrollo y la visualización de resultados. Su flexibilidad permite la integración de celdas de código, texto explicativo en formato Markdown, gráficos y visualizaciones interactivas, lo que facilita el análisis y la interpretación de datos (Kluyver et al., 2016).

Figura 1.2 Logo del software Jupyter Notebook

Una de las principales ventajas de Jupyter Notebook es su compatibilidad con bibliotecas especializadas en machine learning, como NumPy, Pandas, Scikit-learn y TensorFlow, lo que lo convierte en una herramienta ideal para la experimentación y desarrollo de modelos de inteligencia artificial. Además, al ser un entorno basado en web, permite compartir código y documentación de manera eficiente, facilitando la colaboración en proyectos de análisis de datos y aprendizaje automático (Perkel, 2018).

Desarrollo

Problema a Resolver

Una institución financiera desea automatizar la clasificación de solicitudes de préstamo, utilizando un perceptrón que evalúe cuatro factores clave para tomar decisiones:

- Puntaje de crédito: Valor numérico (por ejemplo, entre 300 y 850).
- Ingresos mensuales: Expresado en miles de pesos.
- Monto del préstamo solicitado: Expresado en miles de pesos.
- Relación deuda/ingresos: Valor decimal (por ejemplo, 0.2, 0.5, etc.).
- La institución proporciona un conjunto de datos históricos con ejemplos de solicitudes aprobadas y rechazadas. El perceptrón debe aprender a clasificar correctamente cada solicitud.

Puntaje de Crédito	Ingresos (miles $)	Monto del Préstamo (miles $)	Relación Deuda/Ingresos	Solicitud Aprobada (Salida)
750	5.0	20.0	0.3	1
600	3.0	15.0	0.6	0
680	4.0	10.0	0.4	1
550	2.5	8.0	0.7	0
800	6.0	25.0	0.2	1

Tabla 1 Datos de registro para el perceptrón

Consideraciones Adicionales

- Cuando las entradas son de escalas muy distintas (por ejemplo, puntaje de crédito vs. ingresos), es recomendable normalizar o estandarizar los datos.
- El perceptrón utiliza una función de activación (por ejemplo, la función escalón) para decidir la clase final, aunque en problemas reales se pueden emplear otras funciones y técnicas para mejorar el rendimiento.

Explicación del Código

El código define un conjunto de datos que representa solicitudes de préstamo con cuatro características clave: puntaje de crédito, ingresos mensuales, monto del préstamo solicitado y relación deuda/ingresos. Estos valores se almacenan en un array llamado X, mientras que, y contiene las etiquetas correspondientes, donde 1 indica que el préstamo fue aprobado y 0 que fue rechazado. Estos datos sirven como base para que el perceptrón aprenda a clasificar correctamente nuevas solicitudes.

```python
import numpy as np

## Datos de entrenamiento
X = np.array([
    [750, 5.0, 20.0, 0.3],
    [600, 3.0, 15.0, 0.6],
    [680, 4.0, 10.0, 0.4],
    [550, 2.5, 8.0, 0.7],
    [800, 6.0, 25.0, 0.2]
])

y = np.array([1, 0, 1, 0, 1])
```

Figura 2.1 Código para los datos de entrenamiento del perceptrón

Para mejorar el rendimiento del modelo, los datos se normalizan utilizando la fórmula de normalización min-Max. Esto ajusta los valores de cada característica a un rango entre 0 y 1, evitando que las diferencias de escala entre las variables afecten el entrenamiento. La normalización garantiza que ninguna característica domine a las demás debido a su magnitud y mejora la estabilidad del aprendizaje.

```python
# Normalización de datos
X = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
```

Figura 2.2 Código para la normalización de datos del perceptrón

Se inicializan los pesos (weights) de manera aleatoria para cada característica de entrada, junto con un sesgo (bias). El modelo utiliza una tasa de aprendizaje

(learning_rate = 0.1) para ajustar los pesos en cada iteración, y se define un número de épocas (epochs = 10), que representa la cantidad de veces que el modelo recorrerá los datos para mejorar su desempeño. La función de activación utilizada es una función escalón, que clasifica la salida en 1 o 0 dependiendo de si la combinación lineal de entradas y pesos es mayor o menor que cero.

```
# Parámetros del perceptrón
learning_rate = 0.1
epochs = 10
weights = np.random.rand(X.shape[1])
bias = np.random.rand()

def activation_function(x):
    return 1 if x >= 0 else 0
```

Figura 2.3 Código para los parámetros del perceptrón

Durante cada época, el modelo recorre todas las muestras de datos y calcula una salida a partir de la combinación lineal de las entradas, los pesos y el sesgo. Luego, compara la salida con la etiqueta esperada y ajusta los pesos y el sesgo en función del error obtenido. Este proceso se repite hasta completar todas las épocas. Finalmente, el modelo se evalúa con un nuevo conjunto de datos para determinar si un préstamo hipotético sería aprobado o rechazado.

```
# Entrenamiento del perceptrón
for epoch in range(epochs):
    print(f"Época: {epoch + 1} :")
    for i in range(len(X)):
        linear_output = np.dot(X[i], weights) + bias
        prediction = activation_function(linear_output)
        error = y[i] - prediction

        weights += learning_rate * error * X[i]
        bias += learning_rate * error

        print(f" Muestra {i+1}: Entrada {X[i]}, Salida esperada {y[i]}, predicción {prediction}, error {error}")
    print(f" Pesos actualizados: {weights}, bias actualizada {bias}\n")

print("=========Evaluación del modelo==========")
nuevo_dato = np.array([700, 4.5, 18.0, 0.35])
nuevo_dato = (nuevo_dato - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
Salida = activation_function(np.dot(nuevo_dato, weights) + bias)
print(f"El modelo evaluado para entrada {nuevo_dato} = {Salida}")
if Salida == 1:
    print("El préstamo será aprobado.")
else:
    print("El préstamo será rechazado.")
```

Figura 2.4 Código para el entrenamiento del perceptrón

Resultados del Código

```
Época: 1 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 1, error -1
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 1, error -1
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [0.00705954 0.36497089 0.23308424 0.23682658], bias actualizada 0.44212515214678394

Época: 2 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 1, error -1
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 1, error -1
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [-0.01294046  0.35068517  0.19190777  0.05682658], bias actualizada 0.24212515214678396

Época: 3 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 1, error -1
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 1, error -1
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [-0.03294046  0.33639946  0.15073129 -0.12317342], bias actualizada 0.042125152146783945

Época: 4 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 1, error -1
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 0, error 1
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [-0.00094046  0.36497089  0.12131953 -0.16317342], bias actualizada 0.042125152146783945


Época: 5 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 1, error -1
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 0, error 1
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [ 0.03105954  0.39354232  0.09190777 -0.20317342], bias actualizada 0.042125152146783945

Época: 6 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 0, error 0
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [ 0.03105954  0.39354232  0.09190777 -0.20317342], bias actualizada 0.042125152146783945

Época: 7 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 0, error 0
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [ 0.03105954  0.39354232  0.09190777 -0.20317342], bias actualizada 0.042125152146783945
```

```
Época: 8 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 0, error 0
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [ 0.03105954  0.39354232  0.09190777 -0.20317342], bias actualizada 0.042125152146783945

Época: 9 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 0, error 0
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [ 0.03105954  0.39354232  0.09190777 -0.20317342], bias actualizada 0.042125152146783945

Época: 10 :
 Muestra 1: Entrada [0.8        0.71428571 0.70588235 0.2        ], Salida esperada 1, predicción 1, error 0
 Muestra 2: Entrada [0.2        0.14285714 0.41176471 0.8        ], Salida esperada 0, predicción 0, error 0
 Muestra 3: Entrada [0.52       0.42857143 0.11764706 0.4        ], Salida esperada 1, predicción 1, error 0
 Muestra 4: Entrada [0. 0. 0. 1.], Salida esperada 0, predicción 0, error 0
 Muestra 5: Entrada [1. 1. 1. 0.], Salida esperada 1, predicción 1, error 0
 Pesos actualizados: [ 0.03105954  0.39354232  0.09190777 -0.20317342], bias actualizada 0.042125152146783945

=========Evaluación del modelo==========
El modelo evaluado para entrada [7.0e+02 4.5e+00 1.8e+01 3.5e-01] = 1
El préstamo será aprobado.
```

Figura 3.1 Resultados de la simulación del código para el perceptrón

Conclusión

La implementación de un perceptrón en Python para la clasificación de solicitudes de préstamo representa una aplicación fundamental del aprendizaje supervisado y de los modelos de clasificación binaria. Este enfoque permite a las instituciones financieras optimizar la toma de decisiones mediante la automatización del proceso de evaluación, basándose en factores clave como el puntaje de crédito, los ingresos mensuales, el monto del préstamo y la relación deuda/ingresos. A través del ajuste de pesos y la iteración sobre un conjunto de datos históricos, el perceptrón puede aprender a clasificar nuevas solicitudes con una mayor precisión, reduciendo errores y mejorando la eficiencia operativa.

El uso de Jupyter Notebook como entorno de desarrollo ha sido crucial en este proceso, ya que proporciona una plataforma interactiva que facilita la implementación, visualización y análisis de los resultados obtenidos. Su compatibilidad con bibliotecas de machine learning y su capacidad para integrar

código con documentación lo convierten en una herramienta ideal para la experimentación y el desarrollo de modelos de inteligencia artificial.

En conclusión, el entrenamiento de un perceptrón no solo refuerza los conocimientos en redes neuronales simples y ajuste de pesos, sino que también demuestra la aplicabilidad de estas técnicas en problemas reales del sector financiero. La combinación de algoritmos de machine learning y herramientas computacionales adecuadas permite mejorar la eficiencia en la clasificación de solicitudes de préstamo, contribuyendo a una toma de decisiones más objetiva y fundamentada en datos.

Bibliografía

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Kluyver, T. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (pp. 87-90). IOS Press.
- Perkel, J. M. (2018). Why Jupyter is data scientists' computational notebook of choice. *Nature*, *563*(7732), 145-147.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.