Práctica 01 – Regresión lineal y clasificación binaria

Semana 01

Realizar al menos 02 predicciones, explicarlas y generar un gráfico.

Conceptos Clave a Recordar:

- **MLPRegressor**: Para problemas de regresión.
- **MLPClassifier**: Para problemas de clasificación.
- **hidden_layer_sizes**: Define la arquitectura de las capas ocultas (número de capas y neuronas por capa). Por ejemplo, (100,) es una capa oculta con 100 neuronas, (50, 20) son dos capas ocultas, la primera con 50 neuronas y la segunda con 20.
- **activation**: La función de activación para las capas ocultas. Opciones comunes son 'relu', 'tanh', 'logistic' (sigmoide).
- **solver**: El algoritmo de optimización (aprendizaje). Opciones comunes son 'adam' (recomendado para la mayoría de los casos), 'sgd' (descenso de gradiente estocástico), 'lbfgs'.

Ejercicios de Regresión Lineal

Objetivo: Predecir un valor numérico continuo.

Ejercicio 1: Predecir el Precio de una Vivienda basado en su Tamaño

- **Contexto:** Quieres estimar el precio de una casa (en miles de dólares) basándote en su tamaño (en metros cuadrados). Imagina que tienes datos históricos de ventas.
- **Tipo de Problema:** Regresión
- **Datos Ficticios:**
    - **X_reg (Tamaño en m²):** np.array([50, 75, 100, 120, 150, 180, 200, 220, 250, 280, 300, 320, 350, 380, 400]).reshape(-1, 1)
    - **y_reg (Precio en miles de $):** np.array([150, 200, 250, 280, 320, 360, 400, 430, 480, 520, 550, 580, 620, 660, 700])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 50
- **Función de Activación:** 'relu'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 2: Predecir el Consumo de Energía de un Electrodoméstico

- **Contexto:** Quieres estimar el consumo de energía (en kWh) de un electrodoméstico basándote en el tiempo que lleva encendido (en horas).
- **Tipo de Problema:** Regresión
- **Datos Ficticios:**
    - **X_reg (Tiempo encendido en horas):** np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]).reshape(-1, 1)
    - **y_reg (Consumo de energía en kWh):** np.array([0.5, 0.8, 1.2, 1.5, 2.0, 2.3, 2.8, 3.0, 3.5, 3.8, 4.2, 4.5, 4.8, 5.2, 5.5])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 100
- **Función de Activación:** 'tanh'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 3: Predecir la Temperatura en un Día Soleado

- **Contexto:** Quieres predecir la temperatura máxima (en °C) de un día basándote en la cantidad de horas de sol que recibió ese día.
- **Tipo de Problema:** Regresión
- **Datos Ficticios:**
    - **X_reg (Horas de Sol):** np.array([4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]).reshape(-1, 1)
    - **y_reg (Temperatura Máxima en °C):** np.array([18, 20, 22, 23, 25, 26, 28, 29, 30, 31, 32, 33])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 2
    - **Neuronas por Capa:** (60, 30)
- **Función de Activación:** 'relu'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 4: Predecir el Nivel de Azúcar en Sangre

- **Contexto:** Quieres predecir el nivel de azúcar en sangre (mg/dL) de una persona basándote en la cantidad de carbohidratos consumidos en la última comida (en gramos).

- **Tipo de Problema:** Regresión
- **Datos Ficticios:**
    - **X_reg (Carbohidratos consumidos en gramos):** np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]).reshape(-1, 1)
    - **y_reg (Nivel de azúcar en sangre mg/dL):** np.array([80, 95, 110, 125, 140, 155, 170, 185, 200, 215, 230, 245])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 70
- **Función de Activación:** 'logistic'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 5: Predecir el Salario de un Empleado

- **Contexto:** Quieres predecir el salario anual (en miles de $) de un empleado basándote en sus años de experiencia laboral.
- **Tipo de Problema:** Regresión
- **Datos Ficticios:**
    - **X_reg (Años de Experiencia):** np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]).reshape(-1, 1)
    - **y_reg (Salario anual en miles de $):** np.array([30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 40
- **Función de Activación:** 'relu'
- **Algoritmo de Aprendizaje (solver):** 'lbfgs'

Ejercicios de Clasificación Binaria con Contexto y Datos Ficticios

Objetivo: Clasificar datos en una de dos categorías (0 o 1).

Ejercicio 6: Predecir si un Cliente Comprará un Producto

- **Contexto:** Quieres predecir si un cliente (0 = no compra, 1 = compra) realizará una compra basada en su edad y su gasto promedio mensual en la tienda.
- **Tipo de Problema:** Clasificación Binaria
- **Datos Ficticios:**
    - **X_clf (Edad, Gasto Mensual):**

Python

```
np.array([
  [25, 50], [30, 120], [22, 30], [45, 180], [35, 70],
  [50, 200], [28, 40], [40, 150], [20, 20], [60, 250],
  [33, 90], [27, 60], [55, 220], [19, 10], [48, 170]
])
```

    - **y_clf (Compró: 0 o 1):** np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 50
- **Función de Activación:** 'relu'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 7: Determinar si un Estudiante Aprobará un Examen

- **Contexto:** Quieres predecir si un estudiante aprobará (1) o reprobará (0) un examen basándote en las horas de estudio y su calificación en un examen previo.
- **Tipo de Problema:** Clasificación Binaria
- **Datos Ficticios:**
    - **X_clf (Horas de Estudio, Calificación Previa):**

Python

```
np.array([
  [2, 60], [5, 85], [1, 55], [7, 90], [3, 70],
  [6, 88], [1.5, 50], [4, 75], [0.5, 45], [8, 95],
```

[2.5, 65], [3.5, 72], [7.5, 92], [1.8, 58], [4.5, 80]

])

    - **y_clf (Aprobó: 0 o 1):** np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1])

- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 80
- **Función de Activación:** 'tanh'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 8: Clasificar la Calidad de un Producto (Bueno/Malo)

- **Contexto:** Quieres clasificar si un producto es "bueno" (1) o "malo" (0) basándote en dos características de control de calidad: la solidez del material y la precisión de ensamblaje.
- **Tipo de Problema:** Clasificación Binaria
- **Datos Ficticios:**
    - **X_clf (Solidez del Material, Precisión de Ensamblaje):**

Python

```
np.array([
  [0.7, 0.8], [0.9, 0.95], [0.5, 0.6], [0.85, 0.9], [0.6, 0.7],
  [0.92, 0.98], [0.45, 0.55], [0.8, 0.88], [0.3, 0.4], [0.95, 0.99],
  [0.65, 0.75], [0.58, 0.68], [0.91, 0.96], [0.4, 0.5], [0.83, 0.87]
])
```

    - **y_clf (Calidad: 0 = Malo, 1 = Bueno):** np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 2
    - **Neuronas por Capa:** (40, 20)
- **Función de Activación:** 'relu'
- **Algoritmo de Aprendizaje (solver):** 'adam'

Ejercicio 9: Detectar Fraude en Transacciones Financieras

- **Contexto:** Quieres identificar si una transacción financiera es fraudulenta (1) o legítima (0) basándote en el monto de la transacción y el número de transacciones recientes del usuario.
- **Tipo de Problema:** Clasificación Binaria
- **Datos Ficticios:**
    - **X_clf (Monto, Transacciones Recientes):**

Python

```
np.array([
  [100, 5], [1000, 1], [50, 10], [5000, 2], [200, 7],
  [10000, 0], [70, 8], [2000, 3], [30, 12], [8000, 1],
  [150, 6], [80, 9], [6000, 0], [40, 11], [3000, 2]
])
```

    - **y_clf (Fraude: 0 o 1):** np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 60
- **Función de Activación:** 'logistic'

Ejercicio 9: Detectar Fraude en Transacciones Financieras

- **Contexto:** Quieres identificar si una transacción financiera es fraudulenta (1) o legítima (0) basándote en el monto de la transacción y el número de transacciones recientes del usuario.
- **Tipo de Problema:** Clasificación Binaria
- **Datos Ficticios:**
    - **X_clf (Monto, Transacciones Recientes):**

Python

```
np.array([
  [100, 5], [1000, 1], [50, 10], [5000, 2], [200, 7],
  [10000, 0], [70, 8], [2000, 3], [30, 12], [8000, 1],
  [150, 6], [80, 9], [6000, 0], [40, 11], [3000, 2]
])
```

    - **y_clf (Fraude: 0 o 1):** np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 60
- **Función de Activación:** 'logistic'

Ejercicio 10: Predecir la Suscripción a un Servicio

- **Contexto:** Quieres predecir si una persona se suscribirá (1) o no (0) a un nuevo servicio, basándote en su edad y el tiempo que pasa en línea al día (en horas).
- **Tipo de Problema:** Clasificación Binaria
- **Datos Ficticios:**
    - **X_clf (Edad, Horas en línea):**

Python

```
np.array([
  [20, 8], [35, 3], [25, 10], [40, 2], [30, 6],
  [50, 1], [22, 9], [45, 4], [18, 12], [60, 0.5],
  [38, 5], [28, 7], [55, 1.5], [21, 11], [42, 3.5]
])
```

    - **y_clf (Suscrito: 0 o 1):** np.array([1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0])
- **Arquitectura de la Red:**
    - **Capas Ocultas:** 1
    - **Neuronas por Capa:** 120
- **Función de Activación:** 'relu'
- **Algoritmo de Aprendizaje (solver):** 'lbfgs'