**INSTITUTO POLITECNICO NACIONAL**
**ESCUELA SUPERIOR DE CÓMPUTO**
**NOMBRE:** Cerda García Gustavo
**Materia**: Inteligencia Artificial

# **Ejercicio de Laboratorio 8.** Clasificador Simple

## Ejercicios

1. Empleando el mismo enfoque revisado en la última sesión, realiza el entrenamiento de un clasificador sencillo con los datos del archivo `train.csv`
    a. Probar primero con '`petallength`'
    b. Probar con '`petalwidth`'
2. Una vez definido el umbral de separación, probar con los datos del archivo `test.csv`
    a. Clasifica los datos de '`petallength`', comparar con la clase real y contar cuantos se clasificaron correctamente.
    b. Clasifica los datos de '`petalwidth`', comparar con la clase real y contar cuantos se clasificaron correctamente.

## Desarrollo

### Ejercicio 1

#### Probar con '`petallength`'

```python
# Autor: Cerda García Gustavo
# Practica 8: Clasificador Simple (petallength)

# Programa que clasifica una flor en base a la longitud del pétalo
# Se calcula el umbral de separación para las longitudes del pétalo
# Se clasifican las flores de un archivo de pruebas y se calcula el porcentaje de aciertos

# Librerias
import pandas as pd #Libreria para manejo de datos

# Cargar el archivo CSV
datos = pd.read_csv("../Materiales/train.csv")

# Calcular los umbrales de separación para las longitudes del pétalo
umbral_petallength = datos["petallength"].mean() # Se calcula el promedio de la longitud del pétalo
print("-----------------------------------------------------")
print("                  PETALLENGTH                        ")
print("-----------------------------------------------------")
print(f"Umbral de separación para la longitud del pétalo: {umbral_petallength}")
print("-----------------------------------------------------")
```

Como indica la práctica, primero cargamos el archivo `train.csv`, que es donde están los datos de la siguiente forma:

| A | B | C |
|---|---|---|
| **petallength** | **petalwidth** | **class** |
| 1.4 | 0.2 | Iris-setosa |
| 1.4 | 0.2 | Iris-setosa |
| 1.3 | 0.2 | Iris-setosa |
| 1.5 | 0.2 | Iris-setosa |
| 1.4 | 0.2 | Iris-setosa |
| 1.7 | 0.4 | Iris-setosa |
| 1.4 | 0.3 | Iris-setosa |
| 1.5 | 0.2 | Iris-setosa |
| 1.4 | 0.2 | Iris-setosa |
| 1.5 | 0.1 | Iris-setosa |
| 1.5 | 0.2 | Iris-setosa |
| 1.6 | 0.2 | Iris-setosa |

Con los datos ya cargados en el programa, de la columna 'petallength' obtenemos la media para así calcular el umbral de separación de los datos y poder clasificarlos. Esta es la salida de esta primera parte:

```
2-GCG\Practica-8\Ejercicios> python .\E1-petallenght.py
--------------------------------------------------
                  PETALLENGTH
--------------------------------------------------
Umbral de separación para la longitud del pétalo: 3.788095238095239
--------------------------------------------------
```

### Probar con 'petalwidth'

El código de este inciso es prácticamente el mismo, solo que ahora la columna que utilizaremos es 'petalwidth'

```python
# Autor: Cerda García Gustavo
# Practica 8: Clasificador Simple (petalwidth)

# Programa que clasifica una flor en base a la anchura del pétalo
# Se calcula el umbral de separación para las anchuras del pétalo
# Se clasifican las flores de un archivo de pruebas y se calcula el porcentaje de aciertos

# Librerias
import pandas as pd #Libreria para manejo de datos

# Cargar el archivo CSV
datos = pd.read_csv("../Materiales/train.csv")

# Calcular los umbrales de separación para las longitudes del pétalo
umbral_petalwidth = datos["petalwidth"].mean() # Se calcula el promedio de la longitud del pétalo
print("-----------------------------------------------------")
print("                    PETALWIDTH                       ")
print("-----------------------------------------------------")
print(f"Umbral de separación para la anchura del pétalo: {umbral_petalwidth}")
print("-----------------------------------------------------")
```

Su salida es:

```
2-GCG\Practica-8\Ejercicios> python .\E2-petalwidth.py
-----------------------------------------------------
                    PETALWIDTH
-----------------------------------------------------
Umbral de separación para la anchura del pétalo: 1.1904761904761905
-----------------------------------------------------
```

## Ejercicio 2

### Clasificar con 'petallength'

**Función** `clasificador_petallength (petallength)`

```python
# Nombre: clasificador_petallenght
# Descripción: Clasifica una flor en base a la longitud del pétalo
# Entrada:
#   - petallenght: longitud del pétalo
# Salida:
#   - Clase a la que pertenece la flor
def clasificador_petallength(petallength):
    if petallength < umbral_petallength:
        return "Iris-setosa"
    elif petallength < umbral_petallength + 1.5:
        return "Iris-versicolor"
    else:
        return "Iris-virginica"
```

Esta función clasifica una flor en una de las clases '`Iris-setosa`', '`Iris-versicolor`' o '`Iris-virginica`' basándose en la longitud del pétalo.

- Si la longitud del pétalo es menor al umbral, se clasifica como '`Iris-setosa`'.
- Si es mayor o igual que el umbral y menor que el umbral mas 1.5, se clasifica como '`Iris-versicolor`'.
- Si es mayor o igual al umbral más 1.5, se clasifica como '`Iris-virginica`'.

Utilizando la primera parte y esta función, ya podemos clasificar los datos del archivo '`test.csv`', y así comparar la clase predicha con la clase real y obtener el número de aciertos y el porcentaje de aciertos.

```python
# Cargar archivo de pruebas
pruebas = pd.read_csv("../Materiales/test.csv")

# Imprimir tabla con las predicciones (petallength - clase real - clase predicha) y contar
aciertos
aciertos = 0
print(f"#PRUEBA | PETALLENGTH | CLASE REAL | CLASE PREDICHA")
print("-----------------------------------------------------")
for i in range(len(pruebas)):
    petallength = pruebas["petallength"][i] # Se obtiene la longitud del pétalo
    clase_real = pruebas["class"][i] # Se obtiene la clase real
    clase_predicha = clasificador_petallength(petallength) # Se obtiene la clase predicha
    if clase_real == clase_predicha:
        aciertos += 1
    print(f"  {i}    | {petallength} | {clase_real} | {clase_predicha}")

# Calcular el porcentaje de aciertos
porcentaje_aciertos = aciertos / len(pruebas) * 100
print("-----------------------------------------------------")
print(f"Aciertos: {aciertos}")
print(f"Porcentaje de aciertos: {porcentaje_aciertos}%")
print("-----------------------------------------------------")
```

Primero se carga el archivo '`test.csv`', inicializamos una variable para contar los aciertos y diseñamos la salida, en esta tabla se muestran el numero de prueba, la longitud del pétalo, su clase real y la clase que nuestra función `clasificador_petallength` predijo que seria. Al final se calcula el porcentaje de aciertos y se muestran tanto los aciertos como su porcentaje de efectividad. Esta es la salida del código:

```
2-GCG\Practica-8\Ejercicios> python .\E1-petallenght.py
-----------------------------------------------------
                   PETALLENGTH
-----------------------------------------------------
Umbral de separación para la longitud del pétalo: 3.788095238095239
-----------------------------------------------------
#PRUEBA | PETALLENGTH | CLASE REAL | CLASE PREDICHA
-----------------------------------------------------
  0    | 1.1 | Iris-setosa | Iris-setosa
  1    | 1.2 | Iris-setosa | Iris-setosa
  2    | 1.5 | Iris-setosa | Iris-setosa
  3    | 1.3 | Iris-setosa | Iris-setosa
  4    | 1.4 | Iris-setosa | Iris-setosa
  5    | 1.7 | Iris-setosa | Iris-setosa
  6    | 3.6 | Iris-versicolor | Iris-setosa
  7    | 4.4 | Iris-versicolor | Iris-versicolor
  8    | 4.5 | Iris-versicolor | Iris-versicolor
  9    | 4.1 | Iris-versicolor | Iris-versicolor
  10    | 4.5 | Iris-versicolor | Iris-versicolor
  11    | 3.9 | Iris-versicolor | Iris-versicolor
  12    | 5.1 | Iris-virginica | Iris-versicolor
  14    | 5.5 | Iris-virginica | Iris-virginica
  15    | 6.7 | Iris-virginica | Iris-virginica
  16    | 6.9 | Iris-virginica | Iris-virginica
  17    | 5.0 | Iris-virginica | Iris-versicolor
-----------------------------------------------------
Aciertos: 15
Porcentaje de aciertos: 83.33333333333334%
-----------------------------------------------------
```

### Clasificar con 'petallength'

#### Función `clasificador_petalwidth (petalwidth)`

```python
# Nombre: clasificador_petalwidth
# Descripción: Clasifica una flor en base a la anchura del pétalo
# Entrada:
#   - petalwidth: anchura del pétalo
# Salida:
#   - Clase a la que pertenece la flor
def clasificador_petalwidth(petalwidth):
    if petalwidth < umbral_petalwidth:
        return "Iris-setosa"
    elif petalwidth < umbral_petalwidth + 0.75:
        return "Iris-versicolor"
    else:
        return "Iris-virginica"
```

Esta función clasifica una flor en una de las tres clases: "`Iris-setosa`", "`Iris-versicolor`" o "`Iris-virginica`" basándose en la anchura del pétalo.

- Si la anchura del pétalo es menor que el umbral, la flor se clasifica como '`Iris-setosa`'.
- Si la anchura del pétalo es mayor o igual que el umbral y menor que el umbral más 0.75, la flor se clasifica como '`Iris-versicolor`'.
- Si la anchura del pétalo es mayor o igual que el umbral más 0.75, la flor se clasifica como '`Iris-virginica`'.

```python
# Cargar archivo de pruebas
pruebas = pd.read_csv("../Materiales/test.csv")

# Imprimir tabla con las predicciones (petalwidth - clase real - clase predicha) y contar aciertos
aciertos = 0
print(f"#PRUEBA | PETALWIDTH | CLASE REAL | CLASE PREDICHA")
print("-----------------------------------------------------")
for i in range(len(pruebas)):
    petalwidth = pruebas["petalwidth"][i]
    clase_real = pruebas["class"][i]
    clase_predicha = clasificador_petalwidth(petalwidth)
    if clase_real == clase_predicha:
        aciertos += 1
    print(f"  {i}     | {petalwidth} | {clase_real} | {clase_predicha}")

# Calcular el porcentaje de aciertos
porcentaje_aciertos = aciertos / len(pruebas) * 100
print("-----------------------------------------------------")
print(f"Aciertos: {aciertos}")
print(f"Porcentaje de aciertos: {porcentaje_aciertos}%")
print("-----------------------------------------------------")
```

Al igual que con la longitud del pétalo, ya se puede cargar el archivo '`test.csv`' para hacer las clasificaciones correspondientes y calcular el número de aciertos y su porcentaje de efectividad. Esta es la salida:

```
2-GCG\Practica-8\Ejercicios> python .\E2-petalwidth.py
-----------------------------------------------------
                    PETALWIDTH
-----------------------------------------------------
Umbral de separación para la anchura del pétalo: 1.1904761904761905
-----------------------------------------------------
#PRUEBA | PETALWIDTH | CLASE REAL | CLASE PREDICHA
-----------------------------------------------------
  0     | 0.1 | Iris-setosa | Iris-setosa
  1     | 0.2 | Iris-setosa | Iris-setosa
  2     | 0.4 | Iris-setosa | Iris-setosa
  3     | 0.4 | Iris-setosa | Iris-setosa
  4     | 0.3 | Iris-setosa | Iris-setosa
  5     | 0.3 | Iris-setosa | Iris-setosa
  6     | 1.3 | Iris-versicolor | Iris-versicolor
  7     | 1.4 | Iris-versicolor | Iris-versicolor
  8     | 1.5 | Iris-versicolor | Iris-versicolor
  9     | 1.0 | Iris-versicolor | Iris-setosa
  10     | 1.5 | Iris-versicolor | Iris-versicolor
  11     | 1.1 | Iris-versicolor | Iris-setosa
  12     | 2.4 | Iris-virginica | Iris-virginica
  13     | 2.3 | Iris-virginica | Iris-virginica
  14     | 1.8 | Iris-virginica | Iris-versicolor
  15     | 2.2 | Iris-virginica | Iris-virginica
  16     | 2.3 | Iris-virginica | Iris-virginica
  17     | 1.5 | Iris-virginica | Iris-versicolor
-----------------------------------------------------
Aciertos: 14
Porcentaje de aciertos: 77.77777777777779%
-----------------------------------------------------
```