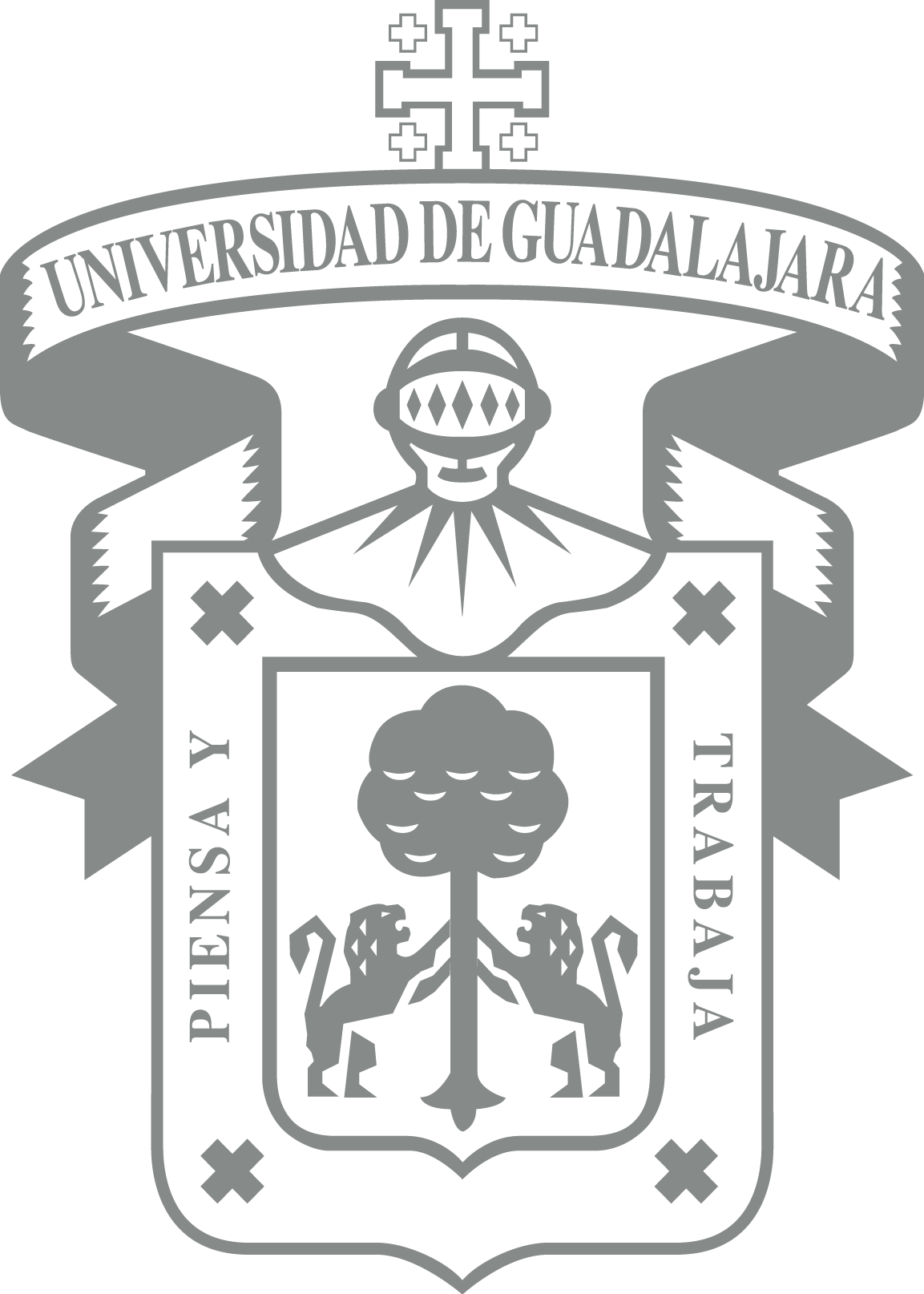
**Universidad de Guadalajara  
Centro Universitario de Ciencias e Ingenierías**



Departamento de Ciencias Computacionales  
Seminario de Solución de Problemas de Inteligencia Artificial ll

Profesor: OLIVA NAVARRO, DIEGO ALBERTO

Alumnos: Torres Hernández David  
Sandoval Gil Sunem

Código: 215428899 Carrera: INCO Sección: D05 Fecha: 22/04/2024

**Practica 1. Ejercicio 3**

# **Introducción**

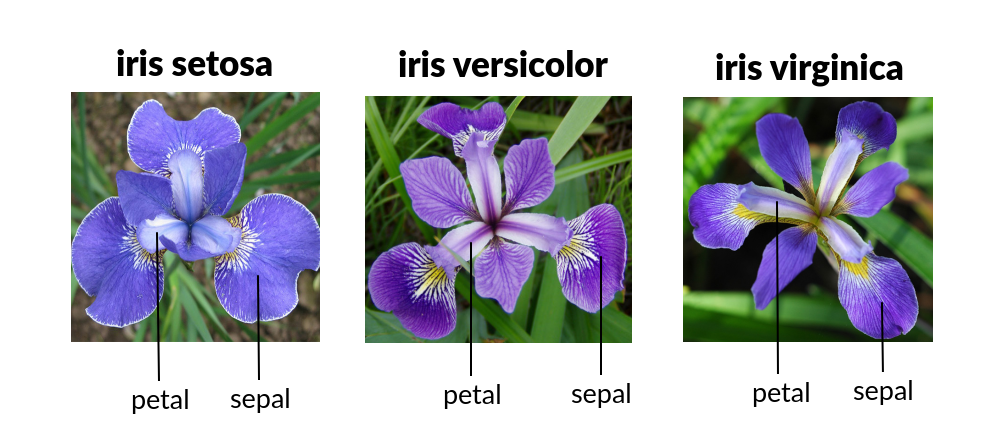
Iris es el género de una planta herbácea con flores que se utilizan en decoración. Dentro de este género existen muy diversas especies entre las que se han estudiado la Iris setosa, la Iris versicolor y la Iris virginica (ver Figura 3).

Flor de color morado

Descripción generada automáticamente

Las tres especies se pueden diferenciar en base a las dimensiones de sus pétalos y sépalos. Se ha recopilado la información de 50 plantas de cada especie y se han almacenado en el archivo irisbin.csv.

Dichas mediciones están en centímetros junto con un código binario que indica la especie a la que pertenece [-1, -1, 1] = setosa, [-1, 1, -1] = versicolor, [1, -1, -1] = virginica, la Figura 4 muestra la distribución de los datos contenidos en el archivo. Se debe crear un programa capaz de clasificar automáticamente los datos de 150 patrones usando un perceptrón multicapa. Es recomendable considerar 80% de los datos para entrenamiento y 20% para generalización.



Con la estructura optima de la red, se deben validar los resultados usando lo métodos leave-k-out y leave-one-out con un perceptrón multicapa como clasificador. Se debe estimar el error esperado de clasificación, el promedio y la desviación estándar de ambos métodos

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

# **Código**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import KFold, LeaveOneOut

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score

from sklearn.decomposition import PCA

class PerceptronMulticapa:

    def \_\_init\_\_(self, hidden\_layer\_sizes=(100,), activation='relu', solver='adam', max\_iter=2000, random\_state=None):

        self.model = MLPClassifier(hidden\_layer\_sizes=hidden\_layer\_sizes, activation=activation,

                                   solver=solver, max\_iter=max\_iter, random\_state=random\_state)

    def train(self, X\_train, y\_train):

        self.model.fit(X\_train, y\_train)

    def evaluate(self, X\_test, y\_test):

        y\_pred = self.model.predict(X\_test)

        return accuracy\_score(y\_test, y\_pred)

def leave\_k\_out(X, y, k):

    kf = KFold(n\_splits=k)

    errors = []

    for train\_index, test\_index in kf.split(X):

        X\_train, X\_test = X[train\_index], X[test\_index]

        y\_train, y\_test = y[train\_index], y[test\_index]

        perceptron\_multicapa = PerceptronMulticapa(hidden\_layer\_sizes=(10,), activation='relu', solver='adam', max\_iter=2000, random\_state=42)

        perceptron\_multicapa.train(X\_train, y\_train)

        errors.append(1 - perceptron\_multicapa.evaluate(X\_test, y\_test))

    return errors

def leave\_one\_out(X, y):

    loo = LeaveOneOut()

    errors = []

    for train\_index, test\_index in loo.split(X):

        X\_train, X\_test = X[train\_index], X[test\_index]

        y\_train, y\_test = y[train\_index], y[test\_index]

        perceptron\_multicapa = PerceptronMulticapa(hidden\_layer\_sizes=(10,), activation='relu', solver='adam', max\_iter=2000, random\_state=42)

        perceptron\_multicapa.train(X\_train, y\_train)

        errors.append(1 - perceptron\_multicapa.evaluate(X\_test, y\_test))

    return errors

# Función para cargar y preprocesar los datos del archivo irisbin.csv

def load\_iris\_data(file\_path):

    data = pd.read\_csv(file\_path)

    species = []

    for i in range(len(data)):

        if (data.iloc[i][-3:] == [-1, -1, 1]).all():

            species.append('setosa')

        elif (data.iloc[i][-3:] == [-1, 1, -1]).all():

            species.append('versicolor')

        elif (data.iloc[i][-3:] == [1, -1, -1]).all():

            species.append('virginica')

    X = data.iloc[:, :-3].values

    y = np.array(species)

    return X, y

# Cargar datos desde el archivo irisbin.csv

X\_iris, y\_iris = load\_iris\_data('irisbin.csv')

# Leave-k-out

k\_out\_errors = leave\_k\_out(X\_iris, y\_iris, k=10)

mean\_k\_out = np.mean(k\_out\_errors)

std\_k\_out = np.std(k\_out\_errors)

# Leave-one-out

one\_out\_errors = leave\_one\_out(X\_iris, y\_iris)

mean\_one\_out = np.mean(one\_out\_errors)

std\_one\_out = np.std(one\_out\_errors)

print("Leave-k-out: Mean Error =", mean\_k\_out, "Standard Deviation =", std\_k\_out)

print("Leave-one-out: Mean Error =", mean\_one\_out, "Standard Deviation =", std\_one\_out)

# Reducción de dimensionalidad con PCA para la proyección en dos dimensiones

pca = PCA(n\_components=2)

X\_2d = pca.fit\_transform(X\_iris)

# Dividir los datos en entrenamiento y prueba (80% entrenamiento, 20% prueba)

X\_train\_iris, X\_test\_iris, y\_train\_iris, y\_test\_iris = train\_test\_split(X\_2d, y\_iris, test\_size=0.2, random\_state=42)

# Entrenar el perceptrón multicapa

perceptron\_multicapa = PerceptronMulticapa(hidden\_layer\_sizes=(10,), activation='relu', solver='adam', max\_iter=2000, random\_state=42)

perceptron\_multicapa.train(X\_train\_iris, y\_train\_iris)

# Evaluar el modelo

accuracy\_iris = perceptron\_multicapa.evaluate(X\_test\_iris, y\_test\_iris)

print(f'Accuracy (irisbin.csv): {accuracy\_iris}')

# Graficar la proyección en dos dimensiones de la distribución de clases

plt.figure(figsize=(8, 6))

colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue'}

for species in colors:

    subset = X\_2d[y\_iris == species]

    plt.scatter(subset[:, 0], subset[:, 1], color=colors[species], label=species)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('Proyección en Dos Dimensiones de la Distribución de Clases para el Dataset Iris')

plt.legend()

plt.grid(True)

plt.show()

# Graficar la distribución de clases en dos dimensiones

plt.figure(figsize=(8, 6))

for species in colors:

    subset = X\_2d[y\_iris == species]

    plt.scatter(subset[:, 0], subset[:, 1], color=colors[species], label=species)

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

plt.title('Distribución de Clases para el Dataset Iris')

plt.legend()

plt.grid(True)

# Agregar también los datos de "petal"

for species in colors:

    subset = X\_iris[y\_iris == species]

    plt.scatter(subset[:, 2], subset[:, 3], color=colors[species], marker='x')

plt.xlabel('Length (cm)')

plt.ylabel('Width (cm)')

plt.title('Distribución de Clases para el Dataset Iris')

plt.legend()

plt.grid(True)

plt.show()

Principio del formulario

# **Capturas:**

Texto

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

**Conclusiones**

En este ejercicio podemos abordar la clasificación automática de especies de plantas Iris utilizando un perceptrón multicapa, lo cual demuestra la aplicación de técnicas de inteligencia artificial en la botánica y la biología empleando métodos de validación como leave-k-out y leave-one-out para estimar el error de clasificación, promedio y desviación estándar, lo que resalta la importancia de evaluar y validar los modelos de manera rigurosa en el campo de la inteligencia artificial.

El uso de técnicas de reducción de dimensionalidad, como PCA para la proyección en dos dimensiones de los datos de plantas Iris, resalta la importancia de la visualización de datos en la comprensión y análisis de conjuntos de datos complejos, facilitando la interpretación de patrones y la toma de decisiones informadas en el campo de la inteligencia artificial y la ciencia de datos**.**