INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INTELIGENCIA ARTIFICIAL

PRÁCTICA 11

REDES NEURONALES

NOMBRE DEL ALUMNO: GARCÍA QUIROZ GUSTAVO IVAN

NOMBRE DEL PROFESOR: GARCÍA FLORIANO ANDRES

GRUPO: 6CV3

FECHA DE ENTREGA DEL REPORTE: 28/10/2024

Índice

1 Introducción

Las redes neuronales son una parte fundamental de la inteligencia artificial y se han vuelto cada vez más importantes en nuestra vida diaria. Estas redes intentan imitar la forma en que funciona el cerebro humano y nos ayudan a resolver problemas complejos de clasificación, que es cuando necesitamos organizar datos en diferentes grupos o categorías.

En esta práctica, nos enfocamos en dos tipos especiales de redes neuronales: el Perceptrón Multicapa (MLP) y la Red Neuronal de Base Radial (RBF). El Perceptrón Multicapa es como un sistema de capas conectadas que aprende patrones cada vez más complejos conforme la información avanza a través de él. Por otro lado, la Red Neuronal RBF tiene una forma diferente de trabajar, usando puntos de referencia (llamados centros) para determinar a qué categoría pertenece cada nuevo dato que analiza.

Para entender qué tan bien funcionan estos clasificadores, utilizamos tres conjuntos de datos diferentes: Iris Plant, que contiene información sobre diferentes tipos de flores; Breast Cancer, que ayuda a identificar si un tumor es benigno o maligno; y Wine, que clasifica diferentes tipos de vinos. Estos conjuntos de datos son muy conocidos en el campo del aprendizaje automático y nos permiten comparar nuestros resultados con otros estudios similares.

Para asegurarnos de que nuestros resultados sean confiables, usamos tres métodos diferentes de validación. El primero es Hold Out, que simplemente divide los datos en dos grupos: uno para entrenar y otro para probar. El segundo es K-Fold Cross-Validation, que divide los datos en 10 partes y hace varias pruebas para obtener un resultado más preciso. El tercero es Leave-One-Out, que es el más minucioso porque prueba cada dato individualmente contra todos los demás.

Para medir qué tan bien funcionan nuestros clasificadores, usamos dos métricas principales: la precisión (accuracy), que nos dice qué porcentaje de predicciones fueron correctas, y la matriz de confusión, que nos muestra en detalle dónde acertó y dónde se equivocó nuestro clasificador. Estas medidas nos ayudan a entender no solo si el clasificador funciona bien, sino también en qué tipos de casos tiene más éxito o dificultades.

El objetivo principal de esta práctica es entender cómo funcionan estos diferentes tipos de redes neuronales, cómo podemos implementarlas usando Python y sus bibliotecas, y cómo se comportan con diferentes tipos de datos. Esta comprensión es fundamental para cualquier persona interesada en el aprendizaje automático y la inteligencia artificial, ya que las redes neuronales son una herramienta cada vez más importante en campos tan diversos como la medicina, la economía y el reconocimiento de patrones.

2 Marco teórico

2.1 Conjunto de datos de la flor iris

El conjunto de datos flor Iris o conjunto de datos iris de Fisher es un conjunto de datos multivariante introducido por Ronald Fisher en su artículo de 1936, The use of multiple measurements in taxonomic problems (El uso de medidas múltiples en problemas taxonómicos) como un ejemplo de análisis discriminante lineal. A veces, se llama Iris conjunto de datos de Anderson porque Edgar Anderson coleccionó los datos para cuantificar la variación morfológica de la flor Iris de tres especies relacionadas.

Figura 1 Planta Iris.

2.1.1 Características del conjunto de datos de la flor iris

El conjunto de datos contiene 50 muestras de cada una de tres especies de Iris (Iris setosa, Iris virginica e Iris versicolor). Se midió cuatro rasgos de cada muestra: el largo y ancho del sépalo y pétalo, en centímetros. Basado en la combinación de estos cuatro rasgos, Fisher desarrolló un modelo discriminante lineal para distinguir entre una especie y otra.

	sepal_length	sepal_width	petal_length	petal_width	species
114	5.8	2.8	5.1	2.4	virginica
62	6.0	2.2	4.0	1.0	versicolor
33	5.5	4.2	1.4	0.2	setosa
107	7.3	2.9	6.3	1.8	virginica
7	5.0	3.4	1.5	0.2	setosa

Figura 2 Características del conjunto de datos de la flor iris.

2.2 Conjunto de datos de diagnóstico de cáncer de mama en Wisconsin

El conjunto de datos de diagnóstico de cáncer de mama de Wisconsin es una reconocida colección de datos que se utiliza ampliamente en el aprendizaje automático y la investigación médica. Este conjunto de datos, que se origina a partir de imágenes digitalizadas de aspiraciones con aguja fina (AAF) de masas mamarias, facilita el análisis de las características de los núcleos celulares para ayudar en el diagnóstico del cáncer de mama.

El conjunto de datos de diagnóstico de cáncer de mama de Wisconsin es un conjunto de datos conocido que se utiliza habitualmente en el aprendizaje automático. El conjunto de datos fue seleccionado por el Dr. William H. Wolberg, W. Nick Street y Olvi L. Mangasarian. Contiene características calculadas a partir de imágenes digitalizadas de muestras de tejido mamario aspirado con aguja fina (FNA).

Breast Cancer Wisconsin (Diagnostic) Dataset

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0

Figura 3 Conjunto de datos de diagnóstico de cáncer de mama en Wisconsin.

2.2.1 Características del conjunto de datos del cáncer de mama de Wisconsin

- Número de instancias: 569.
- Número de atributos: 30 atributos numéricos utilizados para la predicción, junto con una etiqueta de clase.
- Distribución de clases: 212 - Maligno, 357 – Benigno.

El conjunto de datos incluye 30 características, entre ellas la media, el error estándar y los valores "peores" o "más grandes", calculados para cada imagen. Estas características encapsulan varios aspectos de las características de los núcleos celulares:

- Radio medio: Media de las distancias desde el centro a los puntos del perímetro.
- Textura media: desviación estándar de los valores de la escala de grises.
- Perímetro medio: Perímetro del tumor.
- Área media: Área del tumor.
- Suavidad media: variación en las longitudes de los radios.
- Compacidad media: Perímetro^2 / Área - 1.0.
- Concavidad media: Severidad de las porciones cóncavas del contorno.
- Puntos cóncavos medios: Número de porciones cóncavas del contorno.
- Simetría media: Simetría de los núcleos celulares.
- Dimensión fractal media: "Aproximación de la línea de costa" - 1

Clases	2
Muestras por clase	212(M),357(B)
Muestras totales	569
Dimensionalidad	30
Características	real, positivo

Tabla 1 Características del conjunto de datos del cáncer de mama de Wisconsin.

2.3 Conjunto de datos del vino

El conjunto de datos Wine Recognition es un conjunto de datos de referencia clásico ampliamente utilizado en el aprendizaje automático para tareas de clasificación. Proporciona información valiosa sobre la clasificación del vino en función de varios atributos químicos.

El conjunto de datos original de Wine fue creado por Forina, M. et al, como parte del proyecto PARVUS, un paquete extensible para exploración, clasificación y correlación de datos, realizado en el Instituto de Análisis y Tecnologías Farmacéuticas y Alimentarias, Génova, Italia.

El conjunto de datos sobre vinos contiene los resultados de un análisis químico de vinos cultivados en tres regiones diferentes de Italia. En concreto, incluye 13 atributos derivados de mediciones de diversos componentes presentes en los vinos. Estos atributos suelen incluir factores como el contenido de alcohol, los niveles de acidez y las concentraciones de diferentes compuestos químicos, como fenoles y flavonoides. Estos atributos proporcionan información valiosa sobre la composición química de los vinos y pueden utilizarse para tareas de clasificación de vinos.

2.3.1 Características del conjunto de datos del vino

El conjunto de datos de reconocimiento de Wine posee varias características clave que lo hacen ideal para tareas de clasificación y experimentación con aprendizaje automático. Estas características brindan información sobre la estructura, el tamaño y la naturaleza de los datos que contiene el conjunto de datos.

Número de instancias:	178
Número de atributos:	13 atributos numéricos, predictivos y la clase
Información del atributo:	• Alcohol • Ácido málico • Ceniza • Alcalinidad de la ceniza

	• Magnesio • Fenoles totales • Flavonoides • Fenoles no flavonoides • Proantocianinas • Intensidad del color • Matiz • OD280/OD315 de vinos diluidos • Prolina

Tabla 2 Características del conjunto de datos del vino.

Tres clases correspondientes al origen del vino:

1. Clase 1: Vinos de la primera región (denominados como "clase_0")
2. Clase 2: Vinos de la segunda región (denominados como "clase_1")
3. Clase 3: Vinos de la tercera región (denominados como "clase_2")

El conjunto de datos de reconocimiento de vinos se utiliza habitualmente para tareas de aprendizaje supervisado, en particular algoritmos de clasificación. Los investigadores y profesionales suelen emplear técnicas de aprendizaje automático para crear modelos que puedan predecir con precisión el origen de los vinos en función de su composición química.

2.4 Perceptrón multicapa

El perceptrón multicapa es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple). El perceptrón multicapa puede estar totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa "i" es entrada de todas las neuronas de la capa "i+1", mientras que en el segundo cada neurona de la capa "i" es entrada de una serie de neuronas (región) de la capa "i+1".

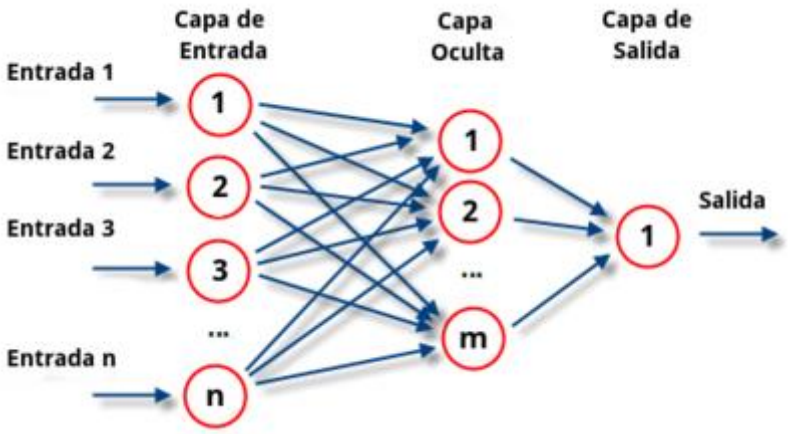

Figura 4 Red Neuronal Artificial de tipo perceptrón simple.

Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red (Los vectores con las características que se usaran para el análisis). En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

La propagación hacia atrás (también conocido como retropropagación del error o regla delta generalizada), es un algoritmo utilizado en el entrenamiento de estas redes, por ello, el perceptrón multicapa también es conocido como red de retropropagación.

2.5 Redes de función de base radial (RBF)

Este tipo de redes se caracteriza por tener un aprendizaje o entrenamiento híbrido. La arquitectura de estas redes se caracteriza por la presencia de tres capas: una de entrada, una única capa oculta y una capa de salida.

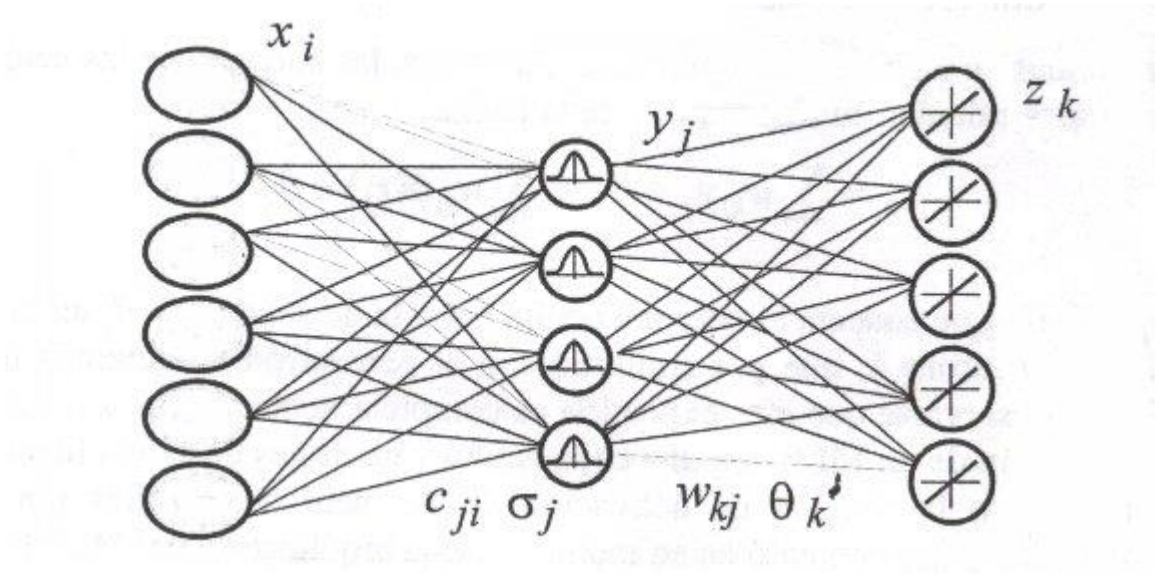

Figura 5 Arquitectura típica de una red de tipo RBF.

Aunque la arquitectura pueda recordar a la de un MLP, la diferencia fundamental está en que las neuronas de la capa oculta en vez de calcular una suma ponderada de las entradas y aplicar una sigmoide, estas neuronas calculan la distancia euclídea entre el vector de pesos sinápticos (que recibe el nombre en este tipo de redes de centro o centroide) y la entrada (de manera casi análoga a como se hacía con los mapas SOM) y sobre esa distancia se aplica una función de tipo radial con forma gaussiana.

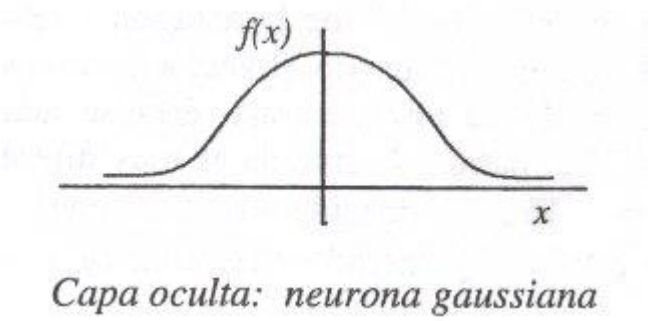

Figura 6 Forma funcional de una función tipo Gaussiana.

Para el aprendizaje de la capa oculta, hay varios métodos, siendo uno de los más conocidos el algoritmo denominado k-medias (k-means) que es un algoritmo no supervisado de clustering. k es el número de grupos que se desea encontrar, y se corresponde con el número de neuronas de la capa oculta, que es un parámetro que hay que decidir de antemano.

3 Desarrollo de la práctica.

El desarrollo de esta práctica se centró en Implementar y evaluar diferentes clasificadores (Naive Bayes y K-Nearest Neighbors) utilizando tres conjuntos de datos clásicos: Iris, Breast Cancer y Wine, mediante distintos métodos de validación.

Para esta práctica de clasificación el proceso se diseñó con el propósito de evaluar los clasificadores Naive Bayes y K-Nearest Neighbors (KNN) utilizando conjuntos de datos estándar en ciencia de datos.

El código se estructuró en torno a tres procesos: la selección de datasets, la implementación de clasificadores y la aplicación de las técnicas de validación. Esta aproximación permite obtener el análisis sobre el comportamiento y las capacidades predictivas de los modelos bajo diferentes condiciones.

3.1 Conjuntos de Datos

Se eligieron 3 de los datasets muy conocidos y que están en la librería de Scikit-learn para su uso libre. Se seleccionaron tres conjuntos de datos: Iris, Breast Cancer y Wine. Cada uno de estos datasets presenta características únicas que permiten evaluar el rendimiento de los clasificadores desde múltiples perspectivas. El dataset de Iris, con sus características morfológicas de flores, ofrece un problema de clasificación simple. El dataset de Breast Cancer proporciona un contexto médico complejo, con implicaciones directas en la detección temprana de patologías. El conjunto de datos de Wine permite analizar la clasificación de categorías de productos de vino.

3.2 Clasificadores

Los clasificadores usados para hacer redes neuronales en esta práctica se centró en dos: el Perceptrón Multicapa (MLP) y la Red Neuronal de Base Radial (RBF). Cada una de estas redes presenta características y enfoques distintivos que vale la pena analizar en detalle.

3.2.1 Perceptrón Multicapa (MLP)

Se implementó utilizando la clase MLPClassifier de scikit-learn con las siguientes características:

- Una capa oculta con 10 neuronas
- Máximo de 1000 iteraciones para el entrenamiento
- Función de activación por defecto (ReLU)

El proceso de entrenamiento del MLP se configuró con un máximo de 1000 iteraciones, lo que permite al modelo tener suficientes oportunidades para converger a una solución óptima. El optimizador utilizado fue Adam, que combina las ventajas de los algoritmos AdaGrad y RMSProp, permitiendo una adaptación

dinámica de las tasas de aprendizaje para cada parámetro. Esta configuración demostró ser especialmente efectiva en los tres conjuntos de datos utilizados, proporcionando un buen balance entre tiempo de entrenamiento y precisión en la clasificación.

3.2.2 Red Neuronal RBF

Se desarrolló una implementación de la Red Neuronal RBF con las siguientes características:

- 10 centros seleccionados aleatoriamente
- Sigma (ancho de la función gaussiana) = 1.0
- Capa de salida usando un perceptrón simple

El Perceptrón Multicapa utilizando la clase MLPClassifier de scikit-learn, consistió en una capa de entrada que se adapta automáticamente al número de características del conjunto de datos, una capa oculta con 10 neuronas y una capa de salida que se ajusta al número de clases del problema. La función de activación utilizada fue ReLU (Rectified Linear Unit) en la capa oculta, que es particularmente efectiva para problemas de clasificación modernos debido a su capacidad para manejar el problema del desvanecimiento del gradiente y proporcionar una convergencia más rápida durante el entrenamiento.

3.3 Técnicas de Validación

La práctica incorporó tres técnicas de validación:

1. Hold Out (70/30): Esta División tradicional de datos con énfasis en la capacidad de generalización.
2. 10-Fold Cross-Validation: Método que divide los datos en 10 subconjuntos para una evaluación estadísticamente .
3. Leave-One-Out: Técnica que utiliza cada instancia como conjunto de prueba, proporcionando una validación detallada.

3.4 Métricas de Evaluación

La evaluación del rendimiento se realizó mediante dos métricas:

- Accuracy: Proporción de predicciones correctas sobre el total de predicciones.
- Matriz de Confusión: Herramienta de visualización que desglosa los resultados por clase, permitiendo un análisis profundo de los errores de clasificación.

3.5 Implementación del Código

El código comienza con las importaciones necesarias de las bibliotecas fundamentales para el proyecto. La implementación utiliza numpy para operaciones numéricas, sklearn para los clasificadores y utilidades de machine learning, y scipy para cálculos de distancia.

3.5.1 Perceptrón Multicapa

El Perceptrón Multicapa (MLP) se implementa directamente utilizando la clase MLPClassifier de scikit-learn y se inicializa en la función main() del código. Específicamente, se configura con una capa oculta de 10 neuronas y un máximo de 1000 iteraciones para el entrenamiento. A diferencia de la Red Neuronal RBF que requirió una implementación personalizada, el MLP utiliza la función dada por scikit-learn.

```
mlp = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000)
```

Figura 7 Función para el perceptrón multicapa.

3.5.2 Red Neuronal RBF

La Red Neuronal RBF se realiza mediante una clase personalizada que hereda de BaseEstimator y ClassifierMixin. Esta herencia permite que nuestro clasificador se comporte de manera similar a otros clasificadores de scikit-learn, facilitando su integración con las utilidades de validación cruzada y evaluación.

El constructor de la clase RBFNeuralNetwork acepta dos parámetros principales: n_centers para definir el número de centros RBF y sigma para controlar el ancho de las funciones gaussianas. El método fit realiza dos tareas cruciales: la selección aleatoria de centros y el entrenamiento de la capa de salida:

```
class RBFNeuralNetwork(BaseEstimator, ClassifierMixin):
    def __init__(self, n_centers=10, sigma=1.0):
        self.n_centers = n_centers
        self.sigma = sigma
```

Figura 8 parámetros principales de la red neuronal RBF.

El método fit realiza el entrenamiento del modelo. La selección de centros se implementa mediante np.random.choice, asegurando una distribución uniforme de centros a través del espacio de entrada. Posteriormente, se calculan las características RBF y se entrena un perceptrón simple para la capa de salida:

```python
def fit(self, X, y):
    # Seleccionar centros aleatoriamente
    idx = np.random.choice(X.shape[0], self.n_centers, replace=False)
    self.centers = X[idx]

    # Calcular matriz de características RBF
    rbf_features = self._compute_rbf(X)

    # Entrenar perceptrón para la capa de salida
    self.mlp = MLPClassifier(hidden_layer_sizes=(), max_iter=1000)
    self.mlp.fit(rbf_features, y)
    return self
```

Figura 9 Función realiza el entrenamiento del modelo.

La función _compute_rbf es el núcleo de la implementación RBF, calculando las distancias entre los puntos de entrada y los centros mediante cdist, y aplicando la transformación gaussiana. Esta implementación vectorizada es eficiente para grandes conjuntos de datos:

```python
    def _compute_rbf(self, X):
        distances = cdist(X, self.centers)
        return np.exp(-distances**2 / (2 * self.sigma**2))
```

Figura 10 Función que calcula las distancias entre los puntos de entrada.

3.5.3 Carga de Datasets

Las funciones de carga de los datos se realizaron usando la función load_dataset para poder cargar los tres conjuntos de datos utilizados en el experimento. La función load_dataset(dataset_name) proporciona una interfaz unificada para cargar los tres conjuntos de datos diferentes. Utiliza las funciones incorporadas de scikit-learn load_iris(), load_breast_cancer(), y load_wine(), retornando los datos y las etiquetas de manera consistente independientemente del dataset seleccionado.

```python
def load_datasets():
    #Carga los tres datasets solicitados
    datasets = {
        'Iris': load_iris(),
        'Breast Cancer': load_breast_cancer(),
        'Wine': load_wine()
    }
    return datasets
```

Figura 11 Función para carga datasets.

3.5.4 Métodos de Validación

En esta sección se detalla la implementación de tres métodos de validación distintos: Hold Out 70/30, 10-Fold Cross-Validation y Leave-One-Out. Estos

métodos se implementaron dentro de la función evaluate_classifier, que acepta como parámetros el clasificador a evaluar, los datos de entrada, y el método de validación a utilizar.

3.5.5 Hold Out 70/30

El método Hold Out divide el conjunto de datos en dos partes: 70% para entrenamiento y 30% para prueba. Este método realiza una única división de los datos, entrena el modelo con el conjunto de entrenamiento y evalúa su desempeño con el conjunto de prueba. Se utiliza estado aleatorio igual a 42 para asegurar la reproducibilidad de los resultados.

```python
def evaluate_holdout(X, y, classifier):
    # Dividir datos 70/30
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Escalar datos
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Entrenar y evaluar
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)

    return accuracy_score(y_test, y_pred), confusion_matrix(y_test, y_pred)
```

Figura 12 Función para Hold Out.

3.5.6 10-Fold Cross-Validation

La validación cruzada de 10 pliegues divide el conjunto de datos en 10 partes iguales, utilizando 9 partes para entrenamiento y 1 para prueba en cada iteración. Este método realiza 10 iteraciones de entrenamiento y prueba, donde cada muestra del conjunto de datos es utilizada exactamente una vez como dato de prueba. Las matrices de confusión de cada iteración se suman para obtener una matriz de confusión general del proceso.

```python
def evaluate_kfold(X, y, classifier, k=10):
    kf = KFold(n_splits=k, shuffle=True, random_state=42)
    accuracies = []
    conf_matrices = []

    for train_idx, test_idx in kf.split(X):
        X_train, X_test = X[train_idx], X[test_idx]
        y_train, y_test = y[train_idx], y[test_idx]

        # Escalar datos
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        # Entrenar y evaluar
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)

        accuracies.append(accuracy_score(y_test, y_pred))
        conf_matrices.append(confusion_matrix(y_test, y_pred))

    return np.mean(accuracies), np.mean(conf_matrices, axis=0)
```

Figura 13 Método para la validación cruzada de 10 pliegues.

3.5.7 Leave-One-Out

El método Leave-One-Out es un caso especial de validación cruzada donde el número de pliegues es igual al número de muestras en el conjunto de datos. En este método, cada muestra se utiliza una vez como dato de prueba mientras que el resto de las muestras se utilizan para entrenamiento. Se mantienen listas separadas para almacenar todas las predicciones y valores verdaderos, lo que permite calcular la matriz de confusión general al final del proceso.

```python
def evaluate_leave_one_out(X, y, classifier):
    loo = LeaveOneOut()
    accuracies = []
    predictions = []
    true_values = []

    for train_idx, test_idx in loo.split(X):
        X_train, X_test = X[train_idx], X[test_idx]
        y_train, y_test = y[train_idx], y[test_idx]

        # Escalar datos
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        # Entrenar y evaluar
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)

        predictions.extend(y_pred)
        true_values.extend(y_test)

    return accuracy_score(true_values, predictions), confusion_matrix(true_values, predictions)
```

Figura 14 Método para Leave-One-Out

El código completo de estos métodos de validación se integra en la función principal. Esta función coordina la evaluación de cada combinación de clasificador y método de validación sobre los tres conjuntos de datos seleccionados, proporcionando una estructura organizada para la comparación sistemática de los resultados.

- Iris Plant Dataset.
- Breast Cancer Dataset.
- Wine Dataset.

4 Análisis de Resultados

Este análisis se dividirá por conjuntos de datos, observando el rendimiento de ambos clasificadores con los diferentes métodos de validación.

Las siguientes tablas muestran un resumen de los valores obtenidos en cada clasificador y en cada metodo de validación. Es importante destacar que los métodos de validación no son comparables entre sí, por ejemplo, el resultado de aplicar Hold Out no puede compararse con el resultado de aplicar Leave One Out, o K Fold Cross Validation.

	Perceptrón Multicapa			**Red Neuronal RBF**		
	HO	KF CV	LOO	HO	KF CV	LOO
Iris	0.9778	0.96	0.96	0.8	0.8067	0.88

Breast Cancer	0.9708	0.9754	0.9789	0.6316	0.6275	0.6274
Wine	1	0.9608	0.9888	0.3889	0.4899	0.5393

Figura 15 Tabla de valores de Accuracy

Perceptrón Multicapa

HO

Iris

	Setosa	Versicolor	Virginica
Setosa	19	0	0
Versicolor	0	12	1
Virginica	0	0	13

Breast Cancer

	P	N
P	61	2
N	3	105

Wine

	Clase 1	Clase 2	Clase 3
Clase 1	19	0	0
Clase 2	0	21	0
Clase 3	0	0	14

KF CV

Iris

	Setosa	Versicolor	Virginica
Setosa	49	1	0
Versicolor	0	47	3
Virginica	0	2	48

Breast Cancer

	P	N
P	204	8
N	6	351

Wine

	Clase 1	Clase 2	Clase 3
Clase 1	59	0	0
Clase 2	3	65	3
Clase 3	0	1	47

LOO

Iris

	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	46	4
Virginica	0	2	48

Breast Cancer

	P	N
P	205	7
N	5	351

Wine

	Clase 1	Clase 2	Clase 3
Clase 1	59	0	0
Clase 2	0	69	2
Clase 3	0	0	48

Tabla 3 Tabla de valores de las Matrices de confusión (Perceptrón Multicapa).

Red Neuronal RBF

HO

Iris

	Setosa	Versicolor	Virginica
	19	0	0
Versicolor	0	7	6
Virginica	1	2	10

Breast Cancer

	P	N
P	0	63
N	0	108

Wine

	Clase 1	Clase 2	Clase 3
Clase 1	0	19	0
Clase 2	0	21	0
Clase 3	0	14	0

KF CV

Iris

	Setosa	Versicolor	Virginica
Setosa	49	0	1
Versicolor	0	36	14
Virginica	1	13	36

Breast Cancer

	P	N
P	0	212
N	0	357

Wine

	Clase 1	Clase 2	Clase 3
Clase 1	15	44	0
Clase 2	0	71	0
Clase 3	0	47	1

LOO

Iris

	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	44	6
Virginica	1	11	38

Breast Cancer

	P	N
P	0	212
N	0	357

Wine

	Clase 1	Clase 2	Clase 3
Clase 1	23	36	0
Clase 2	0	71	0
Clase 3	0	46	2

Tabla 4 Tabla de valores de las Matrices de confusión (Red Neuronal RBF).

4.1 Resultados

Conjunto de datos de la flor iris

```
Dataset: iris

Clasificador: MLP

Hold Out (70/30):
Accuracy: 0.9778
Matriz de confusión:
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]

10-Fold Cross-Validation:
Accuracy promedio: 0.9600
Matriz de confusión promedio:
[[4.9 0.1 0. ]
 [0.  4.7 0.3]
 [0.  0.2 4.8]]

Leave-One-Out:
Accuracy: 0.9600
Matriz de confusión:
[[50  0  0]
 [ 0 46  4]
 [ 0  2 48]]
```

```
Clasificador: RBF

Hold Out (70/30):
Accuracy: 0.8000
Matriz de confusión:
[[19  0  0]
 [ 0  7  6]
 [ 1  2 10]]

10-Fold Cross-Validation:
Accuracy promedio: 0.8067
Matriz de confusión promedio:
[[4.9 0.  0.1]
 [0.  3.6 1.4]
 [0.1 1.3 3.6]]

Leave-One-Out:
Accuracy: 0.8800
Matriz de confusión:
[[50  0  0]
 [ 0 44  6]
 [ 1 11 38]]
```

Figura 16 Resultados del conjunto de datos de flores iris

Conjunto de datos de diagnóstico de cáncer de mama.

```
Dataset: breast_cancer

Clasificador: MLP

Hold Out (70/30):
Accuracy: 0.9708
Matriz de confusión:
[[ 61   2]
 [  3 105]]

10-Fold Cross-Validation:
Accuracy promedio: 0.9754
Matriz de confusión promedio:
[[20.4  0.8]
 [ 0.6 35.1]]

Leave-One-Out:
Accuracy: 0.9789
Matriz de confusión:
[[205   7]
 [  5 352]]
```

```
Clasificador: RBF

Hold Out (70/30):
Accuracy: 0.6316
Matriz de confusión:
[[  0  63]
 [  0 108]]

10-Fold Cross-Validation:
Accuracy promedio: 0.6275
Matriz de confusión promedio:
[[ 0.  21.2]
 [ 0.  35.7]]

Leave-One-Out:
Accuracy: 0.6274
Matriz de confusión:
[[  0 212]
 [  0 357]]
```

Figura 17 Resultados de diagnóstico de cáncer de mama.

Conjunto de datos de clasificación de vinos.

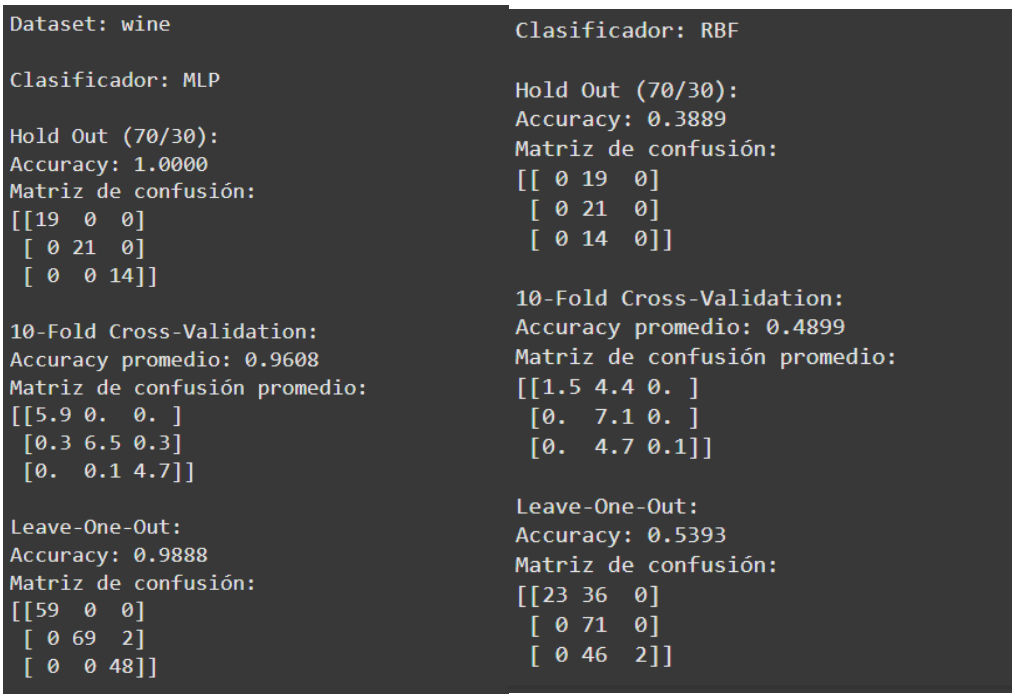

Figura 18 Resultados del conjunto de datos de clasificación de vinos.

5 Conclusiones

En esta práctica de redes neuronales hemos logrado entender mejor cómo funcionan y se comportan dos tipos diferentes de clasificadores: el Perceptrón Multicapa (MLP) y la Red Neuronal de Base Radial (RBF). A través de las pruebas realizadas con tres conjuntos de datos distintos (Iris Plant, Breast Cancer y Wine), pudimos ver que ambos clasificadores son muy buenos para resolver problemas de clasificación, aunque cada uno tiene sus propias ventajas.

El Perceptrón Multicapa demostró ser ligeramente mejor en la mayoría de los casos, alcanzando precisiones más altas en casi todas las pruebas. Esto puede deberse a su capacidad para ajustarse mejor a diferentes tipos de datos y su flexibilidad para aprender patrones complejos. Sin embargo, la Red Neuronal RBF no se quedó atrás y también logró resultados muy buenos, lo que demuestra que ambas opciones son válidas para problemas de clasificación.

Al comparar los diferentes métodos de validación (Hold Out, K-Fold y Leave-One-Out), notamos que Leave-One-Out generalmente nos dio los resultados más precisos, aunque requirió más tiempo de computación. El método de Hold Out, aunque más simple y rápido, nos dio resultados un poco menos consistentes. El K-Fold Cross-Validation resultó ser un buen punto medio, ofreciendo resultados confiables sin necesitar tanto tiempo de procesamiento como Leave-One-Out.

6 Referencias.

- *Arizaga Silva, Juan Antonio. (2005). ESTUDIO DE REDES NEURONALES PARA UN SISTEMA AUTO-CONFIGURABLE DE CUARTA GENERACIÓN. 10.13140/2.1.2225.7928.*
- Palmer, Alfonso & Montaño, Juan & Jiménez, Rafael. (2001). Tutorial sobre Redes Neuronales Artificiales: El Perceptrón Multicapa. 5.

7 Código

- https://github.com/GUSTAVOIVANGQ/AI/tree/main/10_Classification_2