**Reporte sobre Redes Neuronales Multicapa y su Implementación en Python**

**Introducción**

Este documento presenta una explicación detallada sobre la implementación de una Red Neuronal Multicapa (MLP) utilizando Python y la librería scikit-learn. Se abordan los fundamentos teóricos, la preparación de datos, la construcción del modelo y la evaluación de su desempeño.

**Fundamentos teóricos**

Las Redes Neuronales Multicapa (MLP) son un tipo de red neuronal artificial que consta de al menos tres capas: entrada, una o más capas ocultas y salida. Estas redes se entrenan mediante el algoritmo de Backpropagation, que ajusta los pesos de la red usando el descenso de gradiente.

Los principales pasos del entrenamiento son:

1. **Propagación hacia adelante (Forward Propagation):** Se calcula la salida de la red.
2. **Cálculo del error:** Se mide la diferencia entre la salida predicha y la real.
3. **Retropropagación del error (Backward Propagation):** Se ajustan los pesos mediante derivadas parciales.
4. **Actualización de pesos:** Se utilizan optimizadores como Adam o SGD para minimizar el error.

**Implementación en Python**

A continuación, se detalla el código utilizado y su funcionamiento.

**1. Importación de librerías**

Se importan las librerías necesarias para el análisis de datos, visualización y construcción del modelo.

**2. Carga del dataset Iris**

Se utiliza el dataset Iris de scikit-learn, que contiene características de tres especies de flores.

**3. Visualización de los datos**

Se generan gráficos de dispersión y un mapa de calor para analizar la correlación entre las variables.

**4. Preprocesamiento de datos**

Se separan las características (X) y la etiqueta de clase (y), y se dividen en conjuntos de entrenamiento y prueba.

Se normalizan las características con StandardScaler para mejorar el desempeño del modelo.

**5. Construcción del modelo**

Se crea una red neuronal multicapa con la clase **MLPClassifier** de scikit-learn, con dos capas ocultas de 10 neuronas cada una, activación ReLU y optimizador Adam.

**6. Entrenamiento del modelo**

El modelo se ajusta con los datos de entrenamiento.

**7. Evaluación del modelo**

Se predicen las etiquetas en el conjunto de prueba y se evalúa el modelo usando:

* **Accuracy**: Porcentaje de predicciones correctas.
* **Matriz de confusión**: Muestra los errores de clasificación.
* **Reporte de clasificación**: Indica precision, recall y F1-score.

**Conclusiones**

Se implementó exitosamente una red neuronal multicapa en Python con scikit-learn. Se demostró el uso del dataset Iris, la normalización de datos, la configuración del modelo y la evaluación de su desempeño.

Este tutorial proporciona una guía para el uso de MLP en tareas de clasificación y puede extenderse a problemas más complejos con arquitecturas más profundas y grandes volúmenes de datos.