Simple Neural Network Using Scikit-learn

Scikit-learn is a powerful Python module for machine learning. It contains function for regression, classification, clustering, model selection and dimensionality reduction. Today, I will explore the sklearn.neural_network module. This module implements the MLP algorithm that can train models to solve both classification and regression problems. MLP stands for Multi-layer Perceptron. It is a supervised learning algorithm that learns a function by training on a dataset. Given a set of features and a target, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

Let's have a look on MLPclassifier first.

In [1]:
```
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

In [2]:
```
# Load Titanic dataset
titanic = sns.load_dataset('titanic')

# Preprocessing
# Dropping rows with missing 'age' and 'embarked' values for simplicity
titanic.dropna(subset=['age', 'embarked'], inplace=True)

# Converting categorical variables to dummy variables
titanic = pd.get_dummies(titanic, columns=['sex', 'embarked', 'class', 'who', 'deck'], drop_first=True)

# Selecting features and target
X = titanic.drop(['survived', 'alive', 'embark_town', 'adult_male', 'alone'], axis=1)
y = titanic['survived']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing the data
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Creating a simple neural network
mlp = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000, random_state=42) # 1 hidden layer with 10 neurons

# Training the model
mlp.fit(X_train, y_train)

# Predicting the test set results
y_pred = mlp.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.8041958041958042
```

This code snippet demonstrates how to preprocess the Titanic dataset, create a simple Multilayer Perceptron (MLP) model using MLPClassifier from sklearn, train the model, and evaluate its performance. You can run this code in your local Python environment to experiment with a neural network on the Titanic dataset.

In [ ]: