

Project: Project Extra Credit Step

Name: Priyadarshini Shanmugasundaram Murugan

Pledge: "I have not received unauthorized aid on this assignment. I understand the answers that I have submitted. The answers submitted have not been directly copied from another source, but instead, are written in my own words."

The Python code presented herein exemplifies the implementation of a Multi-layer Perceptron (MLP) model for classification tasks using the scikit-learn library. It constitutes several crucial stages in machine learning model development.

The initial segment of the code entails importing essential libraries, namely pandas, numpy, csv, and various modules from scikit-learn, such as `train_test_split` for data partitioning, `StandardScaler` for feature standardization, `MLPClassifier` for constructing the neural network model, and `accuracy_score` for evaluating the model's performance.

The subsequent section involves loading and preparing data. Specifically, two CSV files, 'trainYX.csv' and 'testYX.csv', are read and converted into respective lists of floats through the `csv.reader` module. These lists are then transformed into NumPy arrays while segregating the target variable (`y_train`, `y_test`) from the features (`X_train`, `X_test`).

The MLP model from scikit-learn is a feedforward neural network designed for supervised learning. It features multiple layers, with each node connected to all nodes in the subsequent layer. The `MLPClassifier` is specifically used for classification, while the `MLPRegressor` is used for regression tasks. These models learn complex patterns in data through the backpropagation algorithm, which entails passing input data forward, computing the output, evaluating the error, and then updating the connection weights backward. Key features of MLP models include `hidden_layer_sizes`, `activation`, `solver`, `batch_size`, and `max_iter`. To prevent overfitting, sufficient data for training and careful hyperparameter tuning are required. Following the data preparation stage, a critical step in the normalization process for neural network models is to standardize the features using `StandardScaler`. This normalization involves adjusting the data to have a mean of 0 and a variance of 1, which greatly improves convergence during model training.

Upon completion of data loading and preprocessing, the script proceeds to standardize the input features using the `StandardScaler()` function from scikit-learn. This process scales the features to produce a mean of zero and a variance of one, thereby enhancing model performance. The next phase involves initializing five instances of the `MLPClassifier`, each with a single hidden layer containing a different number of neurons (50, 100, 150, 200, and 250, respectively). The standardized training data is then used to train the models via the `fit()` method.

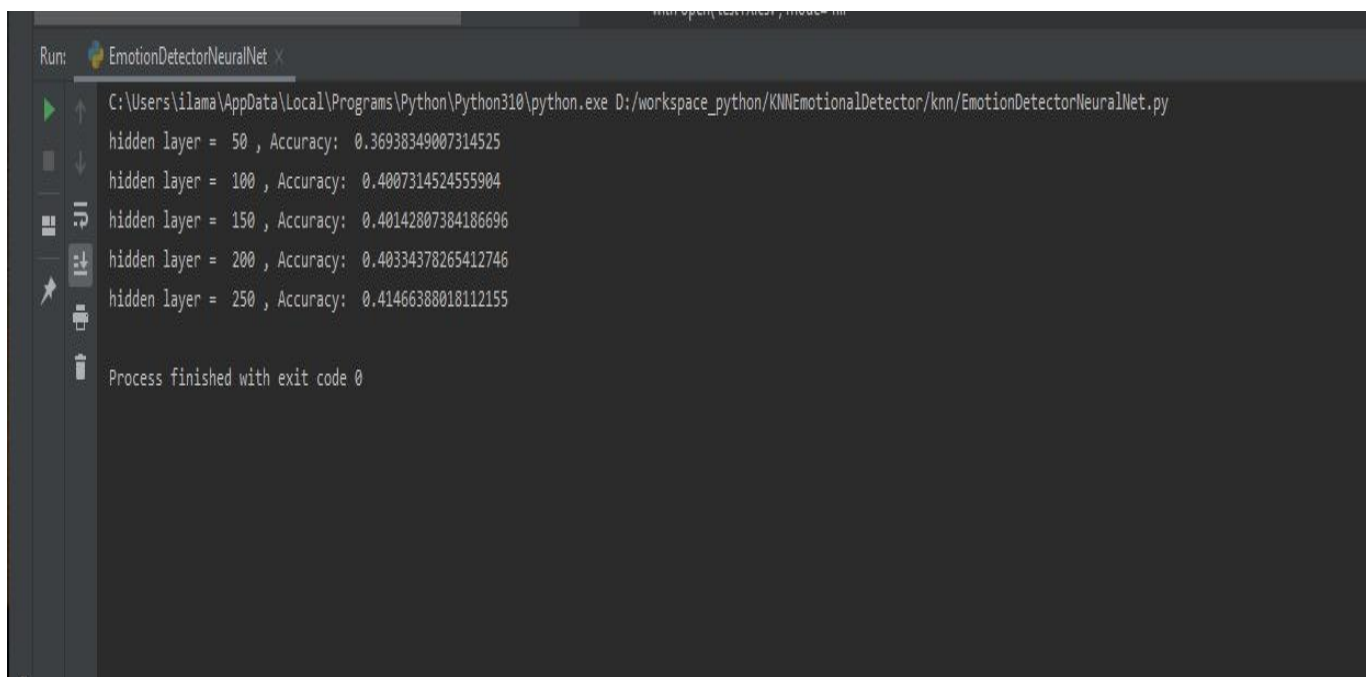
The model training phase commences with fitting the MLP model to the training data (`X_train`, `y_train`) using the `fit` method of the `MLPClassifier`. Once the model is trained, the `predict` method of the `MLPClassifier` is used to make predictions on the test set (`X_test`).

This script demonstrates a workflow for building an MLP classifier using scikit-learn, but it's important to note that the model's performance can vary based on several factors, including dataset characteristics, hyperparameter settings, and problem complexity. Further analysis, hyperparameter tuning, or exploring additional evaluation metrics might offer deeper insights into the model's behavior and performance.

Accuracy table:

Hidden Layer	Accuracy
50	0.369
100	0.400
150	0.401
200	0.433
250	0.4146

Sample Output:



```
Run: EmotionDetectorNeuralNet X
C:\Users\ilama\AppData\Local\Programs\Python\Python310\python.exe D:/workspace_python/KNNEmotionalDetector/knn/EmotionDetectorNeuralNet.py
hidden layer = 50 , Accuracy: 0.36938349007314525
hidden layer = 100 , Accuracy: 0.4007314524555904
hidden layer = 150 , Accuracy: 0.40142807384186696
hidden layer = 200 , Accuracy: 0.40334378265412746
hidden layer = 250 , Accuracy: 0.41466388018112155
Process finished with exit code 0
```