In this code, we follow a similar process as before. We generate random data for demonstration purposes, split it into training and testing sets, and scale the input features using StandardScaler.

Then, we create an MLPRegressor model using MLPRegressor from scikit-learn. We specify the number of hidden layers and their sizes using the hidden\_layer\_sizes parameter. The activation parameter determines the activation function used in the hidden layers (in this case, ReLU), and solver parameter specifies the optimization algorithm (in this case, Adam).

We then train the model using the training data by calling the fit method. Once the model is trained, we make predictions on the test set using the predict method.

Finally, we calculate the mean squared error between the predicted values (y\_pred) and the true values (y\_test) using mean\_squared\_error from scikit-learn and print the result.

Using MLPRegressor is more appropriate for regression tasks as it is specifically designed for regression problems.

To optimize the performance of the MLPRegressor model and achieve a good R-squared score, you can tune the hyperparameters. Here's an example code that demonstrates how to pass hyperparameters to the MLPRegressor model using scikit-learn's GridSearchCV for hyperparameter tuning:

In this code, we use the GridSearchCV class from scikit-learn to perform a grid search for hyperparameter tuning. We define a param\_grid dictionary that contains different values for the hyperparameters we want to tune. In this example, we tune the hidden\_layer\_sizes, activation, and solver hyperparameters.

We create an MLPRegressor model with the default hyperparameters and then pass the model and the parameter grid to the GridSearchCV object. We specify the number of cross-validation folds using the cv parameter (in this case, 5-fold cross-validation).

The GridSearchCV object performs an exhaustive search over the provided parameter grid, training and evaluating the model with different combinations of hyperparameters. It returns the best model found during the search, along with the best hyperparameters.

After performing the grid search, we can access the best model using best\_estimator\_ attribute and the best hyperparameters using best\_params\_ attribute of the GridSearchCV object.

We then make predictions on the test set using the best model and calculate the R-squared score using the r2\_score function from scikit-learn.ssss