DOCUMENTATION

## Slurry Deformation Prediction using a Radial Basis Function (RBF) Neural Network

This documentation provides a detailed explanation of the provided Python code, which implements a Radial Basis Function (RBF) Neural Network for slurry deformation prediction. The code utilizes libraries like Pandas, NumPy, scikit-learn, pyswarm, TensorFlow, and Matplotlib for data processing, model building, optimization, and visualization.

1. Data Preparation

The code begins by loading the training and validation data from CSV files "training.csv" and "validation.csv" using Pandas. The "training.csv" file contains the features used for model training, while "validation.csv" contains the target variable representing the slurry deformation.

2. Data Preprocessing

* Handling Missing Values: The code replaces missing values (NaN) in the training data with the mean value of the respective feature using np.nan\_to\_num().
* Feature Scaling: To improve the model's performance, the features are scaled using StandardScaler from scikit-learn. This standardizes the features to have zero mean and unit variance.

3. Train-Test Split

The preprocessed training data is split into training and testing sets using train\_test\_split() from scikit-learn. This ensures that the model is evaluated on unseen data and avoids overfitting.

4. Defining the RBF Layer

A custom RBF layer is defined using layers.Layer from TensorFlow. This layer utilizes the RBF kernel, which computes the Euclidean distance between input data points and the layer's centers (mu). The resulting distances are transformed using an exponential function, producing a non-linear activation.

* units: This parameter determines the number of RBF units in the layer, which corresponds to the number of centers.
* gamma: This parameter controls the width of the RBF kernel, determining the smoothness of the activation function.

5. Creating the RBF Model

The create\_rbf\_model() function defines the complete RBF neural network architecture. It consists of:

* Input Layer: An input layer with the shape determined by the number of features in the training data.
* RBF Layer: A custom RBF layer as defined above, with the specified number of units and gamma value.
* Output Layer: A dense layer with one neuron for predicting the slurry deformation.

6. Model Optimization using Particle Swarm Optimization (PSO)

To find the optimal values for the RBF network's hyperparameters (units and gamma), Particle Swarm Optimization (PSO) is employed.

* pso\_objective(): This function defines the objective function for PSO, which calculates the mean squared error (MSE) between the predicted and actual deformation values on a validation set.
* lb and ub: These arrays define the lower and upper bounds for the hyperparameters.
* pso(): This function from the pyswarm library executes the PSO algorithm with the given objective function, bounds, swarm size, and maximum iterations. The algorithm returns the best hyperparameter values found.

7. Training the Optimized Model

* The best hyperparameter values are extracted from the PSO results.
* A new RBF model is created using the optimized hyperparameters.
* The model is compiled with the Adam optimizer and MSE loss function.
* The model is trained on the training data using the fit() method, with specified epochs, batch size, and validation split.

8. Evaluation

After training, the model is evaluated on the testing data using various metrics:

* Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values.
* Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values.
* R² Score: Measures the goodness of fit, indicating how well the model explains the variance in the data.

9. Saving the Model and Scaler

The trained RBF model is saved using optimized\_model.save() for future use. The StandardScaler used for feature scaling is also saved using joblib.dump() to ensure consistent data preprocessing during future predictions.

10. Visualization

The code generates two plots:

* Learning Curves: Plots the training and validation losses over the epochs, providing insights into the model's learning progress.
* Predicted vs. Actual Values: Creates a scatter plot comparing the predicted slurry deformation values against the actual values from the testing set, providing a visual evaluation of the model's performance.

Conclusion:

This code provides a framework for building and optimizing an RBF neural network for slurry deformation prediction. By utilizing PSO to find the optimal hyperparameter values, the code demonstrates a powerful approach to improve model accuracy and performance. The visualization of learning curves and prediction results provides useful insights into the model's behavior and effectiveness. This framework can be further extended and customized to address specific requirements and datasets in slurry deformation prediction and other related applications.