The Neocortex API generates three sequences of numbers, categorizing them as Even, Odd, or decimal numbers, which serve as predicted cells for creating a

comprehensive dataset for model training.

The process begins with the utilization of the 'LoadDataset(Datasetfilepath)' method from class 'Classifierleaning', which enables the model to ingest and

comprehend the dataset from a specified JSON file path. Following this, the dataset is partitioned into a 70-30 ratio using the

'SplitDataset(sequenceDataEntries, out trainingfeatures, out testing features, out traininglabels, out testinglabels)' method from same class

'Classifierleaning'.

The 70% of the data is allocated for training the Classifier model, allowing it to discern intricate patterns and relationships inherent in the dataset.

Meanwhile, the remaining 30% is reserved for assessing the model's performance.

During the testing phase, the K-Nearest Neighbors (KNN) Classifier employs the 'Classifier(testingFeatures, trainingFeatures, trainingLabels, k: 3)'

method from class 'KNNClassifier' to predict the labels of the testing data.

To evaluate the model's accuracy, the predicted labels are compare with the actual labels extracted from the training dataset. This comparison is

accomplished through the 'CalculateAccuracy(predictedLabels, testingLabels)' method from Class 'KNNClassifier', which quantifies the accuracy of the model's

predictions, providing valuable insights into its efficacy and performance.

For an Example:

we have Sample Data in a Dataset which we split in training and testing data

training data = [

{

"SequenceName": "S1",

"SequenceData": [8039, 8738, 9334, 9558, 9604, 9697, 9772, 9841, 9851, 9922, 9963, 10023, 10121, 10197, 10373, 10459, 10594, 10629, 10664, 11124]

},

{

"SequenceName": "S2",

"SequenceData": [9051, 9075, 9133, 9178, 9365, 9448, 9481, 9599, 9635, 9740, 10032, 10224, 10281, 10762, 10778, 10934, 11143, 11306, 11494, 11763]

},

{

"SequenceName": "S3",

"SequenceData": [10808, 10834, 11053, 11085, 11434, 11471, 11479, 11553, 11597, 11634, 11720, 11743, 11766, 11812, 11872, 11897, 11909, 12094, 12332, 12504]

}, ...

]

testing Data = [

{

"SequenceName": "S1",

"SequenceData": [7665, 8260, 8304, 8495, 9285, 9366, 9388, 9603, 9641, 9707, 9774, 9819, 9837, 10020, 10096, 10149, 10263, 10313, 10873, 10914]

}

]

Here's the verdict: The model has predicted the testing data as Class S1, representing sequence S1 SDR's closet to testing data SDR'S.

The output includes the label class of the testing data and the accuracy of the model.