

---

**Homework 3**  
SM2001 : Data-Driven Methods in Engineering  
Kirtan Premprakash Patel

---

## Problem 1

We are given a 1001x2 dataset, which we first use to calculate the derivative using the difference method using the 1001x1 time data. These derivatives are found between two datapoints, hence, we find the midpoints of the given datapoints to construct a smaller 1000x2 dataset which we consider to be the datapoints where the derivatives are calculated.

We then use the 1000x2 midpoints dataset and 1000x2 derivatives calculated to identify the non0linear dynamics using the SINDy methodology.

Given that the order of dynamics is less than or equal to 3, we use it to construct library of functions i.e. Theta.

Command Window :

	x1dot	x2dot
	-----	-----
1	0.49916	0
x	0.99973	0.20003
y	-0.99896	-0.16002
xx	0	0
xy	0	0
yy	0	0
xxx	-0.33318	0
xyy	0	0
xyy	0	0
yyy	0	0

Thus, we have the required coefficients

$$\begin{aligned}c_{0,0} &= 0.49916 \\c_{1,0} &= 0.99973 \\c_{0,1} &= -0.99896 \\c_{3,0} &= -0.33318 \\d_{1,0} &= 0.20003 \\d_{0,1} &= -0.16002\end{aligned}$$

This agrees with the hint provided that we should be expecting to find a total of 6 non-zero coefficients.

## Problem 2

### Multi-Class Classification

Multiclass classification is a type of machine learning task in which the goal is to assign input data points to one of multiple predefined classes or categories. This is a common problem in various fields, including image recognition, natural language processing, and medical diagnostics. Multiclass classification and compare supervised and unsupervised methods used for solving this task, including dendrograms, K-means clustering, and neural networks.

Supervised methods for multiclass classification, like Neural Networks, involve training a model on a labeled dataset, where each data point is associated with a specific class label. The model learns to map input features to class labels based on the training examples. Unsupervised methods for multiclass classification, like K-means Clustering and Hierarchical Clustering, do not rely on labeled data but instead aim to group data points into clusters based on similarities in their features. These clusters can then be interpreted as classes.

## Supervised Methods vs Unsupervised Methods

Supervised Methods possess various advantages such as

1. their ability to capture complex patterns in data (as seen in the executed example as well)
2. ability to effectively handle high-dimensional input features

They also have disadvantages such as

1. require large amount of labeled data for training
2. prone to over-fitting, especially with limited data
3. require high computational resources
4. complex architecture which might be difficult to interpret

Unsupervised Methods on the other hand have advantages like

1. scalability and computational efficiency
2. no need for pre-defined labels
3. Hierarchical clustering further does not need the number of clusters to be pre-defined.

but possesses disadvantages like

1. might not always give optimal classification
2. inefficient classification in high-dimensional non-linear dataset
3. Sensitive to the choice of distance metric, linkage method and initial placement of cluster centroids
4. Assumes spherical and equally sized clusters, which may not reflect the true data distribution

### Labeled Dataset

The labelled dataset used to compare the methods is self-generated. It is an upward spiral with 3 branches.

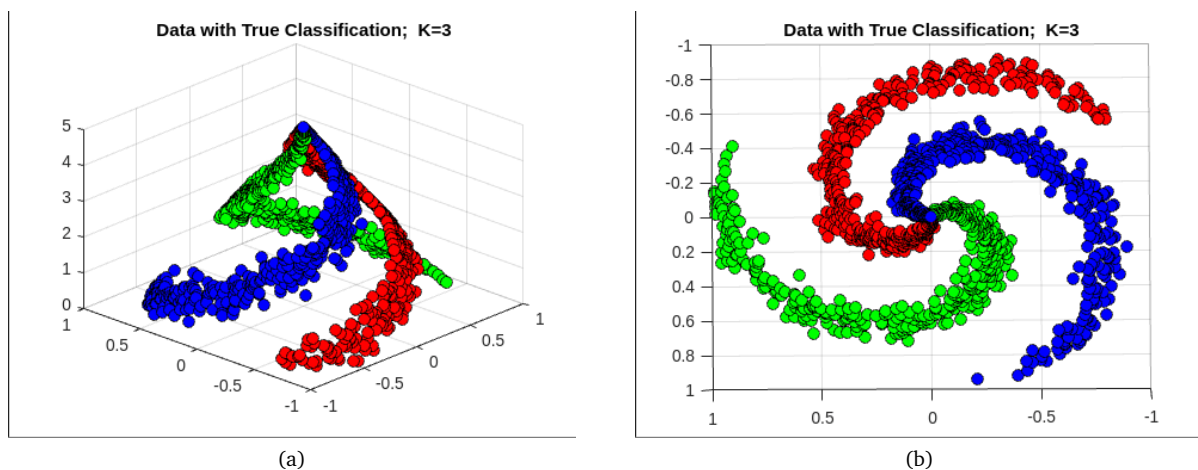


Figure 1: Self-generated Labeled Dataset

The data contains the label in the first column, and the X, Y, and Z co-ordinated in the following columns. Analysis using Supervised methods has been done in the Kaggle Notebook itself, in which the data is generated. This data has then been imported to MATLAB for further analysis using Unsupervised Methods.

## Unsupervised Learning

### K-means Clustering and Hierarchical Clustering

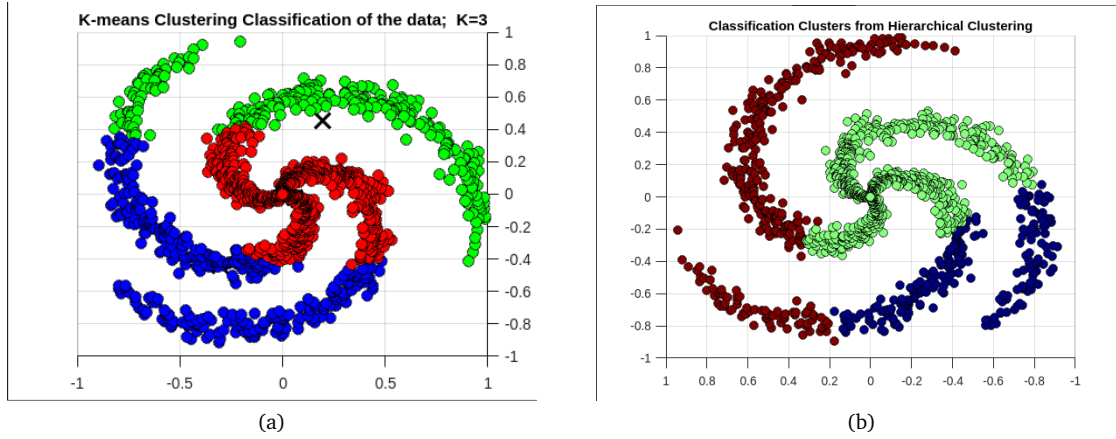


Figure 2: Classification using Unsupervised Clustering

### Evaluation of Clustering Classification

As seen, there is significant lack of accuracy in unsupervised clustering models when used for classification.

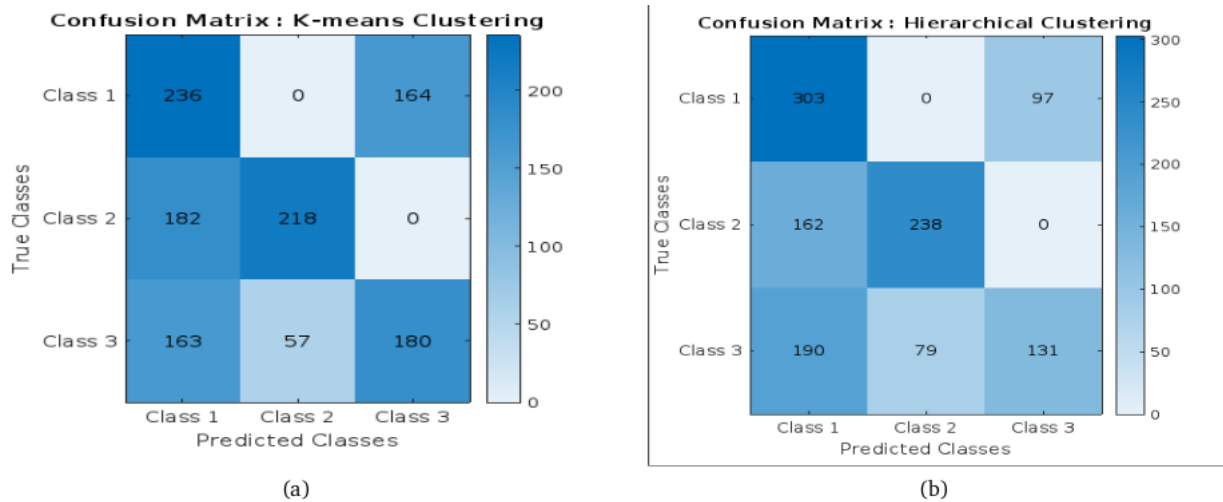


Figure 3: Evaluation of Clustering Classification using Confusion Matrix

Unsupervised learning tends not to detect complex non-linear patterns in the data and provides no visual clarity for the reason behind the clustering in this case. Furthermore, the accuracy can not be improved with more computation.

## Unsupervised Learning

On the other hand Unsupervised Methods like Neural Network can identify the underlying complex patterns. Furthermore since it uses the provided label on the data, it can be trained to ensure that it achieves 100% accuracy by increasing the number of epochs (or the number of hidden layers). Selecting an optimal training size is crucial to ensure there is no overfitting. Table 1 illustrates both these characteristics.

Number of Epochs	Accuracy		
	Test Size = 0.2	Test Size = 0.1	Test Size = 0.05
5	0.7917	0.8750	0.750
10	0.8917	0.9167	0.8667
15	0.9458	0.9667	0.9500
25	0.9917	1.0000	0.9833
30	1.0000	1.0000	1.0000

Table 1: Effect of Number of Epochs and Size of Training Set on Accuracy

# APPENDIX A

## MATLAB Codes

### Problem 1

```
1 clear all
2 close all
3 clc
4
5 %% Prepare the data
6 X = load('DataHw3Q1.mat').x;
7 t = load('DataHw3Q1.mat').t;
8 n = size(X,2);          % state dimension i.e. number of states
9
10 dt = t(2) - t(1);
11 dx_dt = diff(X) / dt;   % slope between 2 points, NOT at
12 x = (X(1:end-1,:) + X(2:end,:))/2; % datapoints at midpoint
13 % we will form SINDy using x and dx_dt
14
15
16 %% Build library and compute sparse regression
17 order = 3;
18 Theta = poolData(x,n,order,0); % up to third order polynomials
19 % Theta for order 3 gives the poolData as (x = x1, y = x2)
20 % [1, x, y, x*x, x*y, y*y, x*x*x, x*x*y, x*y*y, y*y*y]
21 % column 1 contains variables c, and column 2 contain variables d
22 % [c0,0 c1,0 c0,1 c2,0 c1,1 c0,2 c3,0 c2,1 c1,2 c0,3]
23 % [d0,0 d1,0 d0,1 d2,0 d1,1 d0,2 d3,0 d2,1 d1,2 d0,3]
24 lambda = 0.025;
25 % lambda is our sparsification knob.
26 Xi = sparsifyDynamics(Theta,dx_dt,lambda,n);
27 %disp(Xi);
28
29 rowNames={'1','x','y','xx','xy','yy','xxx','xxy','xyy','yyy'};
30 colNames = {'x1dot','x2dot'};
31 sTable = array2table(Xi,'RowNames',rowNames,'VariableNames',colNames);
32 disp(sTable);
```

## Problem 2 : Unsupervised Learning

```
1 clear all
2 close all
3 clc
4
5 %% Load the data
6 labelledData = load('labelledNonLinearDataset.mat').data;
7 % 1200x4 data matrix
8 % Column1 : labels
9 % Column2 : X coordinates
10 % Column3 : Y coordinates
11 % Column4 : Z coordinates
12
13 label = labelledData(:,1); % this is the true classification
14 X = labelledData(:,2:4);
15
16 % number of true classes in the data
17 K = 3;
18
19 % Plot the data with tru label
20 colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k'];
21 figure;
22 hold on;
23 for i = 1:K
24     cluster_points = X(label == i, :);
25     scatter3(cluster_points(:, 1), cluster_points(:, 2),
26             cluster_points(:, 3), 50, colors(i), 'filled', 'DisplayName', ['
27             Cluster ', num2str(i, '%d') ], 'MarkerEdgeColor', 'k');
28 end
29 colormap('jet'); % Set the colormap as needed
30 title(['Data with True Classification; K=', num2str(K, '%d')]);
31 %legend('Location', 'Best');
32 grid on;
33 hold off;
34
35 % Set the perspective view
36 view(-30, -30); % Azimuth: 30 degrees, Elevation: 30 degree
37
38 %% Normalize dataset
39 mean = mean(X);
40 stdDev = std(X);
41 zNormalData = normalize(X);
42 %X_reconstructed = zNormalData.*stdDev + mean;
43
44 %% Part (a) : K-means clustering
45 K = 3;
46 numInitializations = 1000; % define the number of random
47     initializations
48 choiceMultiple = 10;
49 % random centroids generated are numInitializations*choiceMultiple
50 % this is done to further increase randomness by using datasample
51 paddedNumInits = choiceMultiple*numInitializations;
52
53 % Variables to store the results
54 bestIdx = [];
55 bestCentroids = [];
56 bestWCSS = Inf;
57 randomCentroids = [randn(paddedNumInits, 1), randn(paddedNumInits, 1),
58     randn(paddedNumInits, 1)];
```

```

56
57 for i = 1:numInitializations
58     % Randomly initialize cluster centroids
59     initialCentroids = datasample(randomCentroids, K, 'Replace', false
        );
60
61     % Perform K-means clustering with these initial centroids
62     [idx, centroids, sumd] = kmeans(zNormalData, K, 'Start',
        initialCentroids);
63
64     % Calculate the total within-cluster sum of squares (WCSS)
65     wcss = sum(sumd);
66
67     % Check if this initialization results in a lower WCSS
68     if wcss < bestWCSS
69         bestIdx = idx;
70         bestCentroids = centroids;
71         bestWCSS = wcss;
72         %disp(bestWCSS);
73     end
74 end
75
76
77 % Plot the best clustered data
78 upscaledCentroids = bestCentroids.*stdDev + mean;
79 colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k'];
80 figure;
81 hold on;
82 for i = 1:K
83     cluster_points = X(bestIdx == i, :);
84
85     scatter3(cluster_points(:, 1), cluster_points(:, 2),
        cluster_points(:, 3), 50, colors(i), 'filled', 'DisplayName', ['
        Cluster ', num2str(i, '%d') ], 'MarkerEdgeColor', 'k');
86 end
87 colormap('jet'); % Set the colormap as needed
88 scatter3(upscaledCentroids(:, 1), upscaledCentroids(:, 2),
        upscaledCentroids(:, 3), 200, 'k', 'Marker', 'x', 'LineWidth', 2, '
        DisplayName', 'Centroid');
89 title(['K-means Clustering Classification of the data; K=', num2str(K,
        '%d')]);
90 %legend('Location', 'Best');
91 grid on;
92 hold off;
93
94 % Set the perspective view
95 view(-30, -30); % Azimuth: -30 degrees, Elevation: -30 degrees
96
97
98 % Create Confusion Matrix to check Classification of Non-Linear
    Dataset
99 trueClassifications = label;
100 predictedClassifications = bestIdx;
101
102 % Create a confusion matrix
103 C = confusionmat(trueClassifications, predictedClassifications);
104
105 % Display the confusion matrix
106 disp('Confusion Matrix:');
107 disp(C);
108

```

```

109 initial_diagonal_sum = sum(diag(C));
110
111 % Generate all possible permutations of row indices and column indices
112 n = size(C, 1); % Assuming a square matrix
113 row_permutations = perms(1:n);
114 col_permutations = perms(1:n);
115
116 max_diagonal_sum = initial_diagonal_sum;
117 best_row_permutation = 1:n;
118 best_col_permutation = 1:n;
119
120 for i = 1:size(row_permutations, 1)
121     for j = 1:size(col_permutations, 1)
122         B = C(row_permutations(i, :), col_permutations(j, :));
123         diagonal_sum = sum(diag(B));
124         if diagonal_sum > max_diagonal_sum && B(1,1) > B(2,2) && B
            (2,2) > B(3,3)
125             max_diagonal_sum = diagonal_sum;
126             best_row_permutation = row_permutations(i, :);
127             best_col_permutation = col_permutations(j, :);
128         end
129     end
130 end
131
132 % Reorder the best row and column permutation
133 C_mod = C(best_row_permutation, best_col_permutation);
134 disp('Matrix with Maximized Diagonal:');
135 disp(C_mod);
136
137 % Define the minimum and maximum values for the color scale
138 min_val = min(C_mod(:)); % Set the minimum value
139 max_val = max(C_mod(:)); % Set the maximum value
140
141 % Create a figure
142 figure(3);
143 set(gcf, 'Position', [100, 100, 400, 400]);
144
145 % Create the heatmap
146 imagesc(C_mod, [min_val, max_val]);
147 colormap('sky');
148
149 % Add labels and colorbar
150 title('Confusion Matrix : K-means Clustering');
151 xlabel('Predicted Classes');
152 ylabel('True Classes');
153 set(gca, 'XTick', 1:size(C_mod, 2));
154 set(gca, 'YTick', 1:size(C_mod, 1));
155
156 % Manually set the color scale limits
157 clim([min_val, max_val]);
158
159 % Display the values in each cell
160 textStrings = num2str(C_mod(:), '%d');
161 textStrings = strtrim(cellstr(textStrings));
162 [x, y] = meshgrid(1:size(C_mod, 2), 1:size(C_mod, 1));
163 hStrings = text(x(:), y(:), textStrings(:), 'HorizontalAlignment', '
    center');
164 colorbar;
165
166 set(gca, 'XTickLabel', {'Class 1', 'Class 2', 'Class 3'});
167 set(gca, 'YTickLabel', {'Class 1', 'Class 2', 'Class 3'});

```

```

168
169
170 %% Part (b) : Hierarchical clustering
171
172 % Perform hierarchical clustering
173 Z = linkage(zNormalData, 'ward'); % You can choose different linkage
    methods
174
175 %{
176 % Visualize the hierarchical clustering using a dendrogram
177 figure;
178 dendrogram(Z);
179
180 title('Hierarchical Clustering Dendrogram');
181 xlabel('Data Points');
182 ylabel('Distance');
183 %}
184
185 % Use Hierarchical Clustering to form Clusters
186 T = cluster(Z,"maxclust",3);
187
188 % Display the clusters using different colors in a scatter plot
189 figure;
190 scatter3(X(:, 1), X(:, 2), X(:, 3), 50, T, 'filled', 'MarkerEdgeColor'
    , 'k');
191 title('Classification Clusters from Hierarchical Clustering');
192 colormap('jet');
193
194 % Set the perspective view
195 view(-30, -30); % Azimuth: 30 degrees, Elevation: 45 degrees
196
197
198 % Create Confusion Matrix to check Classification of Non-Linear
    Dataset
199 trueClassifications = label;
200 predictedClassifications = T;
201
202 % Create a confusion matrix
203 C2 = confusionmat(trueClassifications, predictedClassifications);
204
205 % Display the confusion matrix
206 disp('Confusion Matrix:');
207 disp(C2);
208
209 % But this might lead to false inference since the numbering of the
210 % newly formed clusters can be changed to ensure that the number
211 % assigned to the cluster is such that it maximizes diagonal elements.
212 % i.e. we can get better inference by row and column swapping
213
214 % Calculate the sum of diagonal elements
215 initial_diagonal_sum = sum(diag(C2));
216
217 % Generate all possible permutations of row indices and column indices
218 n = size(C2, 1); % Assuming a square matrix
219 row_permutations = perms(1:n);
220 col_permutations = perms(1:n);
221
222 max_diagonal_sum = initial_diagonal_sum;
223 best_row_permutation = 1:n;
224 best_col_permutation = 1:n;
225

```



```

226 % Iterate through all row and column permutations to find the best
      diagonal sum
227 for i = 1:size(row_permutations, 1)
228     for j = 1:size(col_permutations, 1)
229         B = C2(row_permutations(i, :), col_permutations(j, :));
230         diagonal_sum = sum(diag(B));
231         if diagonal_sum > max_diagonal_sum && B(1,1) > B(2,2) && B
            (2,2) > B(3,3)
232             max_diagonal_sum = diagonal_sum;
233             best_row_permutation = row_permutations(i, :);
234             best_col_permutation = col_permutations(j, :);
235         end
236     end
237 end
238
239 % Reorder the best row and column permutation
240 C_mod = C2(best_row_permutation, best_col_permutation);
241 disp('Matrix with Maximized Diagonal:');
242 disp(C_mod);
243
244
245 % Define the minimum and maximum values for the color scale
246 min_val = min(C_mod(:)); % Set the minimum value
247 max_val = max(C_mod(:)); % Set the maximum value
248
249 % Create a figure
250 figure(5);
251 set(gcf, 'Position', [100, 100, 400, 400]);
252
253 % Create the heatmap
254 imagesc(C_mod, [min_val, max_val]);
255 colormap('sky');
256
257 % Add labels and colorbar
258 title('Confusion Matrix : Hierarchical Clustering');
259 xlabel('Predicted Classes');
260 ylabel('True Classes');
261 set(gca, 'XTick', 1:size(C_mod, 2));
262 set(gca, 'YTick', 1:size(C_mod, 1));
263
264 % Manually set the color scale limits
265 clim([min_val, max_val]);
266
267 % Display the values in each cell
268 textStrings = num2str(C_mod(:), '%d');
269 textStrings = strtrim(cellstr(textStrings));
270 [x, y] = meshgrid(1:size(C_mod, 2), 1:size(C_mod, 1));
271 hStrings = text(x(:), y(:), textStrings(:), 'HorizontalAlignment', '
        center');
272 colorbar;
273
274 % Adjust axis settings for better display
275 set(gca, 'XTickLabel', {'Class 1', 'Class 2', 'Class 3'});
276 set(gca, 'YTickLabel', {'Class 1', 'Class 2', 'Class 3'});

```

## Problem 2 : Supervised Learning

In [1]:

```
1 import pandas
2 import numpy as np
3 from tqdm import tqdm
4 import matplotlib.pyplot as plt
5 from scipy.io import savemat, loadmat
6 from keras.models import Sequential
7 from keras.layers import Dense
8 from keras.utils import to_categorical
9 from sklearn.model_selection import train_test_split
```

In [2]:

```
1 def generate_spirals():
2     N = 400 # number of points per class
3     D = 3 # dimensionality
4     K = 3 # number of classes
5     Z_max = 5 # max height
6     data = np.zeros((N*K,D)) # data matrix (each row = single example)
7     label = np.zeros(N*K, dtype='uint8') # class labels
8     for j in range(K):
9         ix = range(N*j,N*(j+1))
10        r = np.linspace(0.0,1,N) # radius
11        t = np.linspace(j*4,(j+1)*4,N) + np.random.randn(N)*0.2 # theta
12        z = np.linspace(Z_max,0.0,N) # radius
13        data[ix] = np.c_[r*np.sin(t), r*np.cos(t), z]
14        label[ix] = j # so that classification is from 1 onwards.
15    # Create a 3D scatter plot
16    fig = plt.figure()
17    ax = fig.add_subplot(111, projection='3d')
18    ax.scatter(data[:, 0], data[:, 1], data[:, 2], c=label, s=40)
19
20    ax.legend()
21
22    plt.show()
23    return [data, label]
```

In [3]:

```
1 class MultiClassNN:
2     def __init__(self, input_dim, num_classes, hidden_units=(64, 32),
3         activation='relu', optimizer='adam'):
4         self.input_dim = input_dim
5         self.num_classes = num_classes
6         self.hidden_units = hidden_units
7         self.activation = activation
8         self.optimizer = optimizer
9
10        self.model = self._build_model()
11
12    def _build_model(self):
13        model = Sequential()
14        model.add(Dense(self.hidden_units[0], input_dim=self.input_dim,
15            activation=self.activation))
16        for units in self.hidden_units[1:]:
17            model.add(Dense(units, activation=self.activation))
18        model.add(Dense(self.num_classes, activation='softmax'))
19        model.compile(loss='categorical_crossentropy',
20            optimizer=self.optimizer, metrics=['accuracy'])
21        return model
22
23    def predict(self, X):
24        return np.argmax(self.model.predict(X), axis=1)
25
26    def evaluate(self, X_test, y_test):
27        y_pred = self.predict(X_test)
28        accuracy = np.mean(y_pred == y_test)
29        return accuracy
```

In [4]:

```
1 X = dataset
2 y = labels
3 input_dim = X.shape[1]
4 num_classes = len(np.unique(y))
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05,
6             random_state=42)
7
8 # Initialize the neural network classifier
9 Classifier = MultiClassNN(input_dim, num_classes)
```

In [5]:

```
1 epochs = 30
2
3 # Train the Classification NN Model
4 print("Training the MultiClass Classification NN Model")
5 progress = tqdm(range(epochs))
6 accuracyRecords = np.zeros(epochs)
7
8 y_categorical = to_categorical(y_train, num_classes=Classifier.num_classes)
9
10 for epoch in progress:
11     Classifier.model.fit(X_train, y_categorical, epochs=1, verbose = 0)
12
13 # Evaluate the classifier
14 print("\nEvaluating Classification Model with Test Data")
15 accuracy = Classifier.evaluate(X_test, y_test)
16 print("Result after training is completed")
17 print(f"Accuracy: {accuracy}")
```