```
%==============================
%GENERATES DATASET
%==============================

%Declarations

num_points = 10000;
num_sequences = 150;
num_observations = 15;

%DECLARES DATASET1----------------

rate_of_expansion = 1;
initial_boundary = 1;

for i = 1 : num_sequences

  boundary = initial_boundary;

  for j = 1 : num_observations

    dataset{i,j} = rand(num_points,3)*boundary;

    boundary = boundary + rate_of_expansion;

  endfor

endfor

%DECLARES DATASET2----------------

rate_of_expansion = 1.5;
initial_boundary = 1;

for i = num_sequences + 1 : 2*num_sequences

  boundary = initial_boundary;

  for j = 1 : num_observations

    dataset{i,j} = rand(num_points,3)*boundary;

    boundary = boundary + rate_of_expansion;

  endfor

endfor

classifier_vector = [zeros(num_sequences,1); ones(num_sequences,1)];

%==============================
%FLATTEN DATASET
%==============================
```

```
counter = 1;

for i = 1 : 2*num_sequences

  for j = 1 : num_observations

    dataset_FLAT{counter} = dataset{i,j};
    counter = counter + 1;

  endfor

endfor

%================================
%SORT DATASET
%================================

num_items = size(dataset_FLAT,2);

list = 1 : num_items;

tic; sorted_list = Qsort_EMC(list, dataset_FLAT); toc

for i = 1 : num_items

  dataset_SORTED{i} = dataset_FLAT{sorted_list(i)};

endfor

%=========================================================================
%EMBEDS DATASET
%=========================================================================

measure_space = calculate_embedding_EMC(sorted_list,dataset_SORTED);

%=========================================================================
%ASSEMBLES EMBEDDED DATASET
%=========================================================================

clear temp_sequence
clear embedded_dataset

counter = 1;

for i = 1 : 2*num_sequences

for j = 1 : num_observations

temp_index = find(sorted_list == counter);

temp_sequence(j) = measure_space(temp_index);
counter = counter + 1;
```

```
endfor

temp_sequence(num_observations + 1) = classifier_vector(i); %classifier

embedded_dataset(i,:) = temp_sequence;
embedded_dataset_array{i} = temp_sequence;

endfor

%==========================================================================
%CLUSTERS DATA AND CALCULATES DELTA
%==========================================================================

tic; [final_data_categories_array final_delta] =
optimize_categories_fast_N(embedded_dataset,num_observations); toc

%==========================================================================
%GENERATES PREDICTIONS
%==========================================================================

obs_percentage = .3;
num_errors = 0;
dimensions = 1 : ceil(obs_percentage*num_observations);

tic;
for i = 1 : 2*num_sequences

  input_row = i;
  input_vector = embedded_dataset(input_row,:);
  input_class = input_vector(num_observations + 1);

 [cluster] = NM_pred_spot_cluster(input_vector, final_delta, dimensions, embedded_dataset); %returns
a cluster based upon the input vector

  predicted_classes = cluster(:,num_observations + 1);

  error_vector = predicted_classes != input_class;

  num_errors = num_errors + sum(error_vector);

endfor

toc

num_errors

%==============================
%GENERATE STATE DICTIONARY
%==============================

dictionary{1} = dataset_SORTED{1};
```

```
state_count_vector = [1];

tic;

for i = 2 : num_items

  insert_item = dataset_SORTED{i};

  [dictionary, state_count_vector, insert_index] = state_dictionary_insert(dictionary, state_count_vector,
insert_item);

endfor

toc

%==============================
%DISPLAY A SEQUENCE
%==============================

sequence_index = 10;

for i = 1 : num_observations

  temp = dataset{sequence_index,i};

  X = temp(:,1);
  Y = temp(:,2);
  Z = temp(:,3);

  figure, scatter3(X,Y,Z)


  xlim([0 20]);
  ylim([0 20]);
  zlim([0 20]);

endfor

%write a simple function to test that the dictionary is ordered - you don't need Qsort
```