```
function [accuracy, conf_accuracy, size_accuracy] = TempFunction52422()

%MASSIVE UNSUPERVISED MODAL CLASSIFICATION

%===============================================================================
%LOADS AND NORMALIZES DATASET
%===============================================================================

  clear all

  dataset_file = "/Users/charlesdavi/Desktop/Datasets/UCI/abalone.txt";

  dataset = csvread(dataset_file);
  num_rows = size(dataset,1);
  N = size(dataset,2) - 1;

  testing_percentage = .15;
  num_testing_rows = floor(testing_percentage*num_rows);

  %normalizes dataset
  [weight_vector] = mean_normalization(dataset,N);
  dataset = dataset.*weight_vector;

  %finds unique classes
  class_vector = dataset(:,N+1);
  class_vector = unique(class_vector);
  num_classes = size(class_vector,1);

  for k = 1 : 100

  %generates training / testing datasets
  testing_rows = randperm(num_rows,num_testing_rows);

  dataset(:,N+2) = 0; %dummy value
  dataset(testing_rows,N+2) = -1; %flags testing rows

%===============================================================================
%RUNS CLUSTERING
%===============================================================================

tic;
dataset = sortrows(dataset,1:N);
[class_distribution_matrix, delta_vector] = MASS_Unsup_Modal_BlackTree(dataset, class_vector, N);
toc

%===============================================================================
%RUNS PREDICTION
%===============================================================================

  %finds the modal class for each testing row cluster
  for i = 1 : num_testing_rows

    [modal_freq index] = max(class_distribution_matrix(i,:));

    %if true, then the cluster is not empty, prediction accepted
    if(delta_vector(i) != 0)

      predicted_class_matrix(i,k) = class_vector(index); %modal class
      cluster_class_vector = class_distribution_matrix(i,:)
```

```
    l = sum(cluster_class_vector) %the total number of elements
    I = l*spec_log_BlackTree(num_classes); %information
    U = vector_entropy(cluster_class_vector/l); %entropy as distribution
    V = delta_vector(i)^N; %approximately the volume of the cluster

    knowledge_matrix(i,k) = ((I - U)/V); %knowledge per unit volume
    modal_prob_matrix(i,k) = modal_freq/l; %prob. of mode
    cluster_size_matrix(i,k) = l; %cluster size

   %otherwise, the prediction is rejected
   else

    predicted_class_matrix(i,k) = -1;
    knowledge_matrix(i,k) = 0;
    modal_prob_matrix(i,k) = 0;
    cluster_size_matrix(i,k) = 0;

   endif

  endfor

%=============================================================================
%CALCULATES ACCURACY (USING REJECTIONS)
%=============================================================================
   testing_rows = find(dataset(:,N+2) == -1); %redefine due to sorting
   actual_class_matrix(:,k) = dataset(testing_rows,N+1);
   x = find(predicted_class_matrix != -1);
   num_predictions = size(x,1); %number of accepted predictions

   error_vector = (predicted_class_matrix(x) != actual_class_matrix(x));
   num_errors = sum(error_vector);

   accuracy(k) = 1 - num_errors/num_predictions;

endfor %end of outer for loop
%=============================================================================
%CALCULATES ACCURACY (USING CONFIDENCE)
%=============================================================================

  %normalizes knowledge to [0,1]
  norm_knowledge_matrix = knowledge_matrix / max(knowledge_matrix(:));
  confidence_matrix = min(norm_knowledge_matrix, modal_prob_matrix);

  interval_set = unique(confidence_matrix);
  num_intervals = size(interval_set,1);

  %calculates accuracy as a function of confidence
  for i = 1 : num_intervals

   x = find(confidence_matrix >= interval_set(i));

   num_predictions = size(x,1); %number of accepted predictions

   error_vector = (predicted_class_matrix(x) != actual_class_matrix(x));
   num_errors = sum(error_vector);

   conf_accuracy(i) = 1 - num_errors/num_predictions;

  endfor
```

```
 max_cluster_size = max(cluster_size_matrix(:))
%calculates accuracy as a function of cluster size
 for i = 1 : max_cluster_size

   x = find(cluster_size_matrix >= i);

   num_predictions = size(x,1); %number of accepted predictions

   error_vector = (predicted_class_matrix(x) != actual_class_matrix(x));
   num_errors = sum(error_vector);

   size_accuracy(i) = 1 - num_errors/num_predictions;

 endfor


endfunction
```