```
function [accuracy max_cluster_size] = tempfunction()

%Copyright Charles Davi, all rights reserved

%loads dataset

file = "/Users/charlesdavi/Desktop/Datasets/UCI/wine.txt";
dataset = csvread(file);

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

[weight_vector] = mean_normalization(dataset,N);

dataset = dataset.*weight_vector;

%copies classifiers and generates training / testing datasets
dataset(:,N+2) = dataset(:,N+1);
testing_percentage = .15;
num_testing_rows = floor(num_rows*testing_percentage);
testing_rows = randperm(num_rows,num_testing_rows);
dataset(testing_rows,N+1) = -1; %flags testing rows


RH_cluster_size = zeros(1,num_testing_rows);
LH_cluster_size = zeros(1,num_testing_rows);

%sorts dataset

sorted_dataset = sortrows(dataset,1:N);

%finds the testing rows in the sorted dataset
sorted_testing_rows = find(dataset(:,N+1) == -1);

%applies prediction
for i = 1 : num_testing_rows

  testing_row = sorted_testing_rows(i);

  %right hand side--------------------------------------------------

  %if not true, it's the end of the list
  if(testing_row + 1 <= num_rows)

    j = testing_row + 1;
    RH_class(i) = sorted_dataset(j,N+1);

    %if true, then the adjecent entry is a training row
    if(RH_class(i) != -1)

      RH_cluster_size(i) = 1;
      break_loop = 0;

    %otherwise, the adjacent entry is a testing row
    else

      RH_cluster_size(i) = 0;
      break_loop = 1;

    endif
```

```
  %finds the cluster until first error
  while(break_loop == 0 && j <= num_rows)

    test_class = sorted_dataset(j,N+1);

    %if true, then we increment the cluster size
    if(test_class == RH_class(i))

      RH_cluster_size(i) = RH_cluster_size(i) + 1;

    %otherwise, we break the loop
    else

      break_loop = 1;

    endif

  j = j + 1;

  endwhile

  %otherwise, we flag a rejection
  else

    RH_class(i) = -1;

  endif

%left hand side-----------------------------------------------------

%if not true, it's the first entry in the list
if(testing_row - 1 > 0)

  j = testing_row - 1;
  LH_class(i) = sorted_dataset(j,N+1);

  %if true, then the adjecent entry is a training row
  if(LH_class(i) != -1)

    LH_cluster_size(i) = 1;
    break_loop = 0;

  %otherwise, the adjacent entry is a testing row
  else

    LH_cluster_size(i) = 0;
    break_loop = 1;

  endif

  %finds the cluster until first error
  while(break_loop == 0 && j > 0)

    test_class = sorted_dataset(j,N+1);

    %if true, then we increment the cluster size
    if(test_class == LH_class(i))

      LH_cluster_size(i) = LH_cluster_size(i) + 1;
```

```
      %otherwise, we break the loop
      else

        break_loop = 1;

      endif

    j = j - 1;

    endwhile

  %otherwise, we flag a rejection
  else

    LH_class(i) = -1;

  endif

endfor

%tests accuracy-----------------------------------------------------

num_errors = 0;
num_rejections = 0;

error_vector = zeros(1,num_testing_rows);

for i = 1 : num_testing_rows

  testing_row = sorted_testing_rows(i);
  actual_class = sorted_dataset(testing_row,N+2);

  if(RH_class(i) != -1 && LH_class(i) != -1)

    %righthand prediction
    RH_temp_cluster_size = RH_cluster_size(i);
    RH_prediction_vector = RH_class(i)*ones(1,RH_temp_cluster_size); %creates a vector with class
labels

    %lefthand prediction
    LH_temp_cluster_size = LH_cluster_size(i);
    LH_prediction_vector = LH_class(i)*ones(1,LH_temp_cluster_size); %creates a vector with class labels

    prediction_vector = [RH_prediction_vector LH_prediction_vector];

  %otherwise the testing row was either the front or end of the list, rejection
  else

    prediction_vector = [];

  endif
  %tests prediction accuracy

  %if true, then the cluster is empty, or the classes are not equal, a rejection
  if((size(prediction_vector,2)) == 0 || (LH_class(i) != RH_class(i)))

    num_rejections = num_rejections + 1;
    predicted_class_vector(i) = -1;
```

```
  %otherwise, we find the modal class
  else

    predicted_class = mode(prediction_vector);
    predicted_class_vector(i) = predicted_class;

    %if true, the prediction is an error
    if(predicted_class != actual_class)

      num_errors = num_errors + 1;
      error_vector(i) = 1;

    endif

  endif

endfor

accuracy = 1 - num_errors/(num_testing_rows - num_rejections);

%applies confidence-------------------------------------------------

cluster_size_vector = RH_cluster_size .+ LH_cluster_size;
max_cluster_size = max(cluster_size_vector);
actual_class_vector = sorted_dataset(sorted_testing_rows,N+2);

%confidence iterates from zero up to the max cluster size
for i = 0 : max_cluster_size

  x = find(cluster_size_vector >= i);
  num_predictions(i+1) = size(x,2);

  num_errors = sum(error_vector(x));

  y = find(predicted_class_vector(x) == -1);
  num_rejections = size(y,2);

  accuracy(i+1) = 1 - num_errors/(num_predictions(i+1) - num_rejections);

endfor

endfunction
```