```
%==========================================================================
%==========================================================================

%==========================================================================
%UCI CREDIT DATASET
%==========================================================================

%==========================================================================
%==========================================================================

%COPYRIGHT CHARLES DAVI, 2021

%==========================================================================
%LOADS THE DATASET
%==========================================================================

%==========================================================================
%LOADS UCI CREDIT DATASET
%==========================================================================

%clears the memory and the command prompt
clear
clc

pkg load image
%--------------------------------------------------------------------------

file_path =  '/Users/charlesdavi/Desktop/Datasets/UCI/UCI_Credit_Card.csv';
A = csvread(file_path);
A(:,1) = []; %deletes the first column which contains row numbers
total_num_rows = size(A,1); %the total number of rows in the dataset

num_rows = 3500;
selected_rows = randperm(total_num_rows, num_rows); %randomly selects a subset
dataset = A(selected_rows,:);

N = size(dataset,2) - 1; %the size of the dataset less the classifier dimension

%==========================================================================
%NORMALIZES DATASET
%==========================================================================

%calculate the average of each dimension-------------------------------------
digit_vector = log10(mean(abs(dataset(:,1:N))))
max_dim = ceil(max(digit_vector))
min_dim = floor(min(digit_vector))

s = std(digit_vector);

temp_vector = digit_vector > s;
max_dimensions_vector = find(temp_vector == 1);

%normalizes the largest category of dimensions-------------------------------
num_items = size(max_dimensions_vector,2);
weight_vector = ones(1,N+1); %used to normalize the dataset

%sets the original accuracy
[accuracy output_matrix error_vector] = find_NN_dataset(dataset, []);
max_accuracy = accuracy;
```

```
for i = min_dim : max_dim

  for j = 1 : num_items

    current_index = max_dimensions_vector(j);
    current_num_digits = digit_vector(current_index);

    diff = i - current_num_digits;

    weight_vector(current_index) = 10^diff;

  endfor

  weight_vector

  modified_dataset = dataset.*weight_vector;
  [accuracy output_matrix error_vector] = find_NN_dataset(modified_dataset, []);

  if(accuracy > max_accuracy)

    max_accuracy = accuracy;
    final_weight_vector = weight_vector;
    final_dataset = modified_dataset;

  endif

endfor

%================================================================================
%GENERATES CLUSTERS
%================================================================================
tic;
s = std(dataset(:,1:N)); %calculates the standard deviation of the dataset in each dimension
s = mean(s); %takes the average standard deviation
s = s*N;

num_rows = size(dataset,1);

cluster_matrix = zeros(num_rows,num_rows);

delta = s/24; %this is the value of delta based upon experimentation

for i = 1 : num_rows

  input_vector = dataset(i,:);
  [cluster_vector diff_vector] = find_delta_cluster(input_vector, dataset, delta, N);
  cluster_matrix(i,:) = cluster_vector;

endfor
toc

%================================================================================
%GENERATES TRAINING / TESTING DATASET
%================================================================================

num_iterations = 150;

accuracy_p = [];
accuracy_c = [];
accuracy_cp = [];
```

```
num_rows = size(dataset,1);

for i = 1 : num_iterations

%permutes the dataset

%Generates a training and testing dataset----------------------------------------
num_training_rows = floor(.85*num_rows); %selects a portion of the dataset
training_rows = randperm(num_rows,num_training_rows);
testing_dataset = dataset;
testing_dataset(training_rows,:) = [];
training_dataset = dataset(training_rows,:);
num_testing_rows = size(testing_dataset,1);

%================================================================================
%CLUSTER PREDICTION STEP
%================================================================================

[predicted_class_vector confidence_vector probability_vector CP_accuracy error_vector] =
cluster_prediction(dataset, training_dataset, testing_dataset, training_rows, cluster_matrix, N);

%================================================================================
%BENCHMARK PREDICTION STEP - NEAREST NEIGHBOR
%================================================================================

num_errors = 0;

for j = 1 : num_testing_rows

  input_vector = testing_dataset(j,:);

  [predicted_vector predicted_class prediction_row diff_vector] = find_NN(input_vector, training_dataset,N);

  actual_class = input_vector(N+1);

  if(predicted_class != actual_class)

    num_errors = num_errors + 1;

  endif

endfor

NNaccuracy(i) = 1 - num_errors/num_testing_rows;

%================================================================================
%PROBABILITY; CONFIDENCE
%================================================================================

%probability---------------------------------------------------------------
counter = 1;
increment = .01;
num_levels = size(0 : increment : 1,2);

for j = 0 : increment : 1

  x = find(probability_vector >= j);

  num_errors = sum(error_vector(x));
```

```
  num_predictions = size(x,2);

  if(num_predictions > 0)

    accuracy_p(i,counter) = 1 - num_errors/num_predictions;

  endif

  counter = counter + 1;

endfor

%confidence------------------------------------------------------------------
counter = 1;
increment = .01;
num_levels = size(0 : increment : 1,2);

confidence_vector = confidence_vector/max(confidence_vector);

for j = 0 : increment : 1

  x = find(confidence_vector >= j);

  num_errors = sum(error_vector(x));

  num_predictions = size(x,2);

  if(num_predictions > 0)

    accuracy_c(i,counter) = 1 - num_errors/num_predictions;

  endif

  counter = counter + 1;

endfor

%confidence and probability---------------------------------------------------
counter = 1;
increment = .001;
num_levels = size(0 : increment : 1,2);

confidence_vector = confidence_vector/max(confidence_vector);

for j = 0 : increment : 1

  x = find(probability_vector >= j);
  y = find(confidence_vector >= j);

  temp1 = zeros(num_testing_rows,1);
  temp2 = zeros(num_testing_rows,1);

  temp1(x) = 1;
  temp2(y) = 1;

  z = temp1.*temp2;

  z = find(z == 1);
```

```
  z = z';
  num_errors = sum(error_vector(z));
  num_predictions = size(z,2);
  if(num_predictions > 0)
    accuracy_cp(i,counter) = 1 - num_errors/num_predictions;
  endif
  counter = counter + 1;
endfor
endfor %end of outer loop
plot_data_p = mean(accuracy_p);
plot_data_c = mean(accuracy_c);
plot_data_cp = mean(accuracy_cp);
figure, plot(plot_data_p)
figure, plot(plot_data_c)
figure, plot(plot_data_cp) %this is the plot generated by both thresholds
file_path = '/Users/charlesdavi/Desktop/UCI Credit Dataset Accuracy.png';
saveas(gcf,file_path);
```