```
%clears the memory and the command prompt
clear
clc
pkg load image
%-------------------------------------------------------------------

file_path =  '/Users/charlesdavi/Desktop/Datasets/UCI/UCICredit.csv';
orig_dataset = csvread(file_path);
orig_num_rows = size(orig_dataset,1);
N = size(orig_dataset,2) - 1;

row_percentage = 1

num_rows = floor(orig_num_rows*row_percentage)
selected_rows = randperm(orig_num_rows,num_rows);

dataset = orig_dataset(selected_rows,:);

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

%calculate the number of digits in each dimension-----------------------------
temp_vector = mean(abs(dataset(:,1:N)));

temp_vector = temp_vector + rand()*.00001; %to prevent log(0) errors
digit_vector = log10(temp_vector)
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;
final_dataset = dataset;

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

  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

max_accuracy
dataset = final_dataset;

%==========================================================================
%GENERATES TRAINING / TESTING ROWS
%==========================================================================

%Generates a training and testing dataset----------------------------------
num_training_rows = floor(.9*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);

%==========================================================================
%GENERATES CLUSTERS
%==========================================================================

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

cluster_matrix = zeros(num_testing_rows,num_training_rows);

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

for i = 1 : num_testing_rows

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

endfor
toc

%==========================================================================
%MODAL PREDICTION
%==========================================================================

num_classes = size(unique(dataset(:,N+1)),1)

[predicted_class_vector probability_vector confidence_vector] = modal_prediction_train_test(training_dataset,
cluster_matrix, num_classes, N);

actual_class_vector = testing_dataset(:,N+1)';

error_vector = predicted_class_vector != actual_class_vector;

raw_accuracy = 1 - sum(error_vector)/num_testing_rows
```

```
%================================================================================
%CONFIDENCE FILTERING
%================================================================================

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

%probability-------------------------------------------------------------------
counter = 1;
increment = .0001;
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(counter) = 1 - num_errors/num_predictions;

  endif

  counter = counter + 1;

endfor

%confidence-------------------------------------------------------------------
counter = 1;
increment = .0001;
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(counter) = 1 - num_errors/num_predictions;

  endif

  counter = counter + 1;

endfor

%confidence and probability---------------------------------------------------
counter = 1;
increment = .0001;
```

```
num_levels = size(0 : increment : 1,2);

num_predictions_vector = []

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_rows,1);
  temp2 = zeros(num_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(counter) = 1 - num_errors/num_predictions;
    num_predictions_vector(counter) = num_predictions; %num predictions in scope

  endif

  counter = counter + 1;

endfor

figure, plot(accuracy_p)

figure, plot(accuracy_c)

figure, plot(accuracy_cp) %this is the plot generated by both thresholds

hold

plot(num_predictions_vector/num_rows)

file_name = '/Users/charlesdavi/Desktop/Datasets/UCI_Credit.png';

saveas(gcf,'Barchart.png')

%================================================================================
%MAGIC BUTTON
%================================================================================

%finds accuracy within a .025 threshold of the desired accuracy
%tie-breaker is number of rows returned at a given accuracy

target_accuracy = .94
```

```
num_entries = size(num_predictions_vector,2)

best_accuracy = -1
best_num_rows = -1
best_index = -1; %flag for no matching accuracy
threshold = .025

for i = 1 : num_entries

 current_accuracy = accuracy_cp(i);
 current_num_rows = num_predictions_vector(i);

 %if true, then the accuracy is acceptable
 if(abs(current_accuracy - target_accuracy) <= threshold)

  %this is the tie breaker
  if(current_num_rows > best_num_rows)

    best_index = i;
    best_accuracy = current_accuracy;
    best_num_rows = current_num_rows;

  endif

 endif

endfor

%displays the returned accuracy and num rows, -1 indicates no match
best_accuracy
best_num_rows
```