```
%==========================================================================
====
%LOADS IMAGE
%==========================================================================
====

pkg load image

directory_path = '/Users/charlesdavi/Desktop/grocery_IM_dataset/';

counter = 1;

for k = 1 : 3

num_images = 10;

category_number = k;

directory = [directory_path int2str(k) '/'];

for i = 1 : num_images

I = imread([directory int2str(i) '.jpg']);

IMG_array{counter} = I;
IMG_category{counter} = k; %stores the classifier associated with the image
counter = counter + 1;

endfor

endfor

%==========================================================================
====
%EXTRACTS SHAPE INFORMATION
%==========================================================================
====
tic;
[final_avg_matrix final_indexes] = partition_image_vectorized(IMG_array{1}); %this is to size the
partitions for the entire dataset
t_1 = toc;

N = size(final_avg_matrix,2); %we take the second dimension, because of repeat entries in the first
dimensions

total_num_images = 3*num_images; %there are 3 categories of images

counter = 1; %index for the observations
dataset = [];

scramble = randperm(total_num_images,total_num_images); %this is to permute the dataset, which
would otherwise be sorted by digit
```

```
tic;
%iterates through entire dataset
while(counter <= total_num_images)

I = IMG_array{scramble(counter)};

%creates a matrix for each channel of the image
I_R = I(:,:,1);
I_G = I(:,:,2);
I_B = I(:,:,3);

%this calculates the average luminosity across each channel of the image
[avg_matrix_R] = calc_avg_color_vect(final_indexes, I_R, N);
[avg_matrix_G] = calc_avg_color_vect(final_indexes, I_G, N);
[avg_matrix_B] = calc_avg_color_vect(final_indexes, I_B, N);

input_vector_R = reshape(avg_matrix_R, [1 N^2]);
input_vector_G = reshape(avg_matrix_G, [1 N^2]);
input_vector_B = reshape(avg_matrix_B, [1 N^2]);

input_vector = [input_vector_R input_vector_G input_vector_B]; %assembles a single input vector for
the dataset

input_vector(3*N^2+1) = IMG_category{scramble(counter)}; %the hidden classifier

dataset(counter,:) = input_vector;

counter = counter + 1;

endwhile
t_2 = toc;

%================================================================
%EXTRACTS BACKGROUND
%================================================================

tic;
s_vector = std(dataset(:,1:3*N^2)); %the standard deviation of every dimension
avg = mean(s_vector); %the average standard deviation

bin_vector = s_vector > avg; %binary vector of dimensions greater than the mean

dataset(:,1:3*N^2) = dataset(:,1:3*N^2).*bin_vector; %sets likely background entries to zero
t_3 = toc;

%================================================================
%MAKES PREDICTIONS; TESTS ERROR
%================================================================

num_rows = size(dataset,1);

tic;nearest_neighbors = NN_fully_vectorized(dataset, 3*N^2);toc
```

```
error_vector = dataset(nearest_neighbors, 3*N^2+1) != dataset(:, 3*N^2+1);

num_errors = sum(error_vector);

accuracy = 1 - num_errors / num_rows
```