1. Take the earlier 3 datasets. Compute the reconstruction error for each of them as discussed in the class defined in the EigenFace paper (https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf). You can use pre-defined python libraries for computing eigen vectors and eigen values.
   Brief Summary of the Process to be Followed:
   Let us assume X is your data, do not take the label or target variable into account. Let $\boldsymbol{\mu}$ be the mean of X, compute column wise mean. Let A = X - $\boldsymbol{\mu}$. Compute the eigenvectors of A^TA. Take the n eigenvectors corresponding to the top n eigenvalues (experiment with n=1, 2, 3, 5, 7, 10, 15 based on the columns you have). Then compute the projection w_j of each data point ($a_i=x_i-\boldsymbol{\mu}$) on the eigen vectors. Then the linear combination of the data point $l_i=\sum w_j e_j$ and the reconstructed point is $r_i=l_i+\boldsymbol{\mu}$. Calculate the reconstruction error for the whole dataset = $\sum(x_i-r_i)^2$. Plot the reconstruction error vs the number of eigenvectors.
2. Implement both the improvements of gradient descent on the earlier 3 datasets and find out which of them is faster in convergence. Compare it with the simple gradient descent algorithm.
   a. Momentum based Gradient Descent
      $v_{t+1} = \gamma\, v_t + \eta\, \nabla L(w_t)$
      $w_{t+1} = w_t - v_{t+1}$
   b. Nesterov Accelerated Gradient Descent (NAG)
      $v_{t+1} = \gamma\, v_t + \eta\, \nabla L(w_t - \gamma\, v_t)$
      $w_{t+1} = w_t - v_{t+1}$