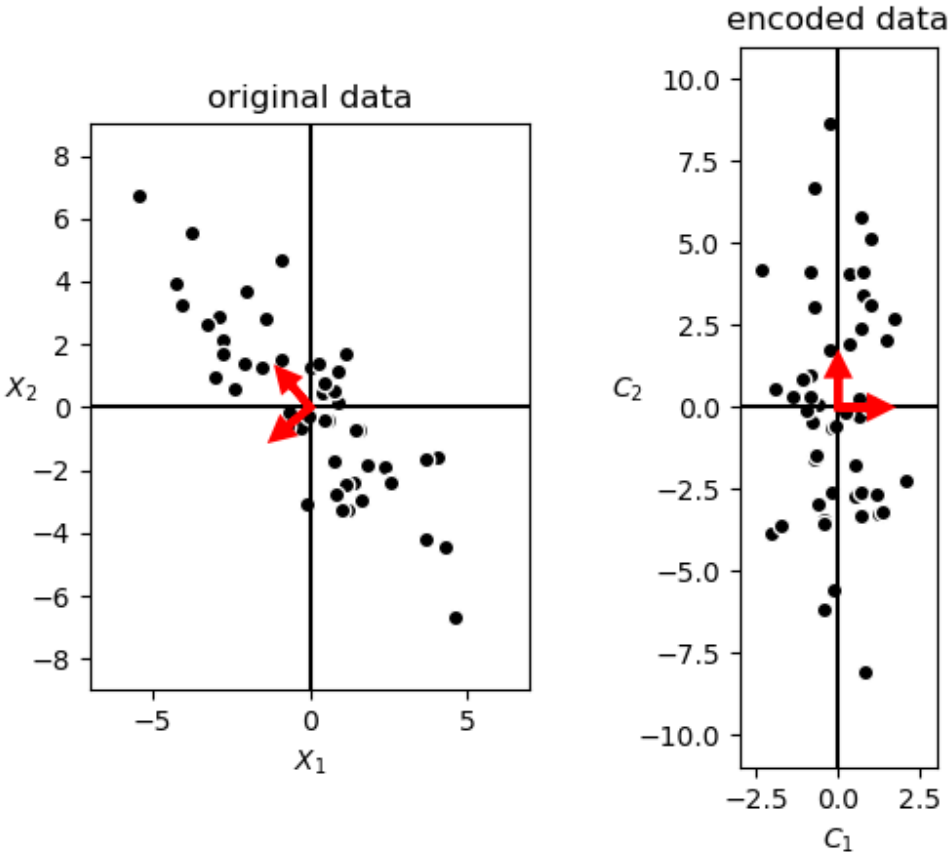
Andrew Hall
CS589 HW6

Task 1:

My implementation of PCA is outlined in the following steps which correspond with comments in my code:

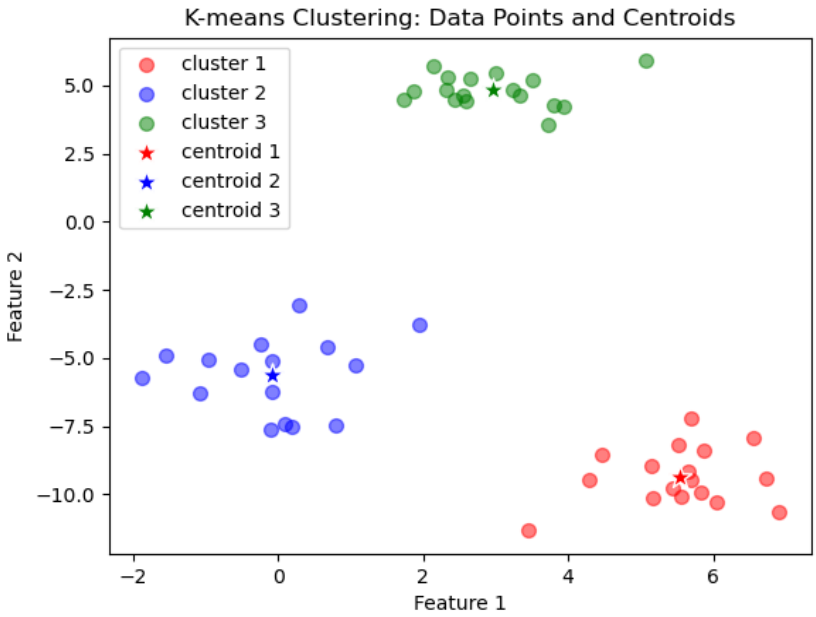
1. Center the data by subtracting the mean of the data from all points.
2. Create the correlation matrix by dotting x with it's transpose and adding a lambda scaled identity matrix for numerical stability, with lambda equal to 10^{-7}. I do this stability trick because I was following the example code in the slides.
3. Compute the eigen values and eigen vectors of the correlation matrix.
4. Extract the principal component vectors from the eigen vectors in descending order based on the magnitude of the eigenvalues.

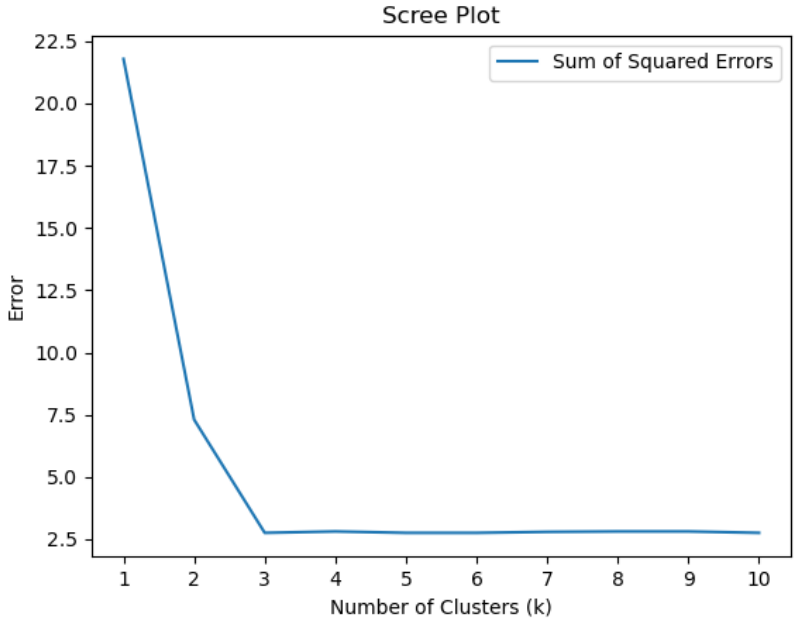
The following plots show the principal component vectors in red. On the left is the original mean-centered data, and on the right is the encoded data which has been transformed using the principal component vectors.



Task 2:

I initialized the centroids randomly with numpy's random.randn function which samples from a normal distribution. I chose this method for its simplicity because the dataset we are clustering is well defined, and by that I mean it's very obvious to which cluster each data point should belong. For a more complex dataset I would be more inclined to try a more more involved initialization approach, along with data preprocessing such as scaling the data but I don't see the need to do those things here. The stopping criteria for my custom K-means algorithm is when the assignments of datapoints to clusters has ceased to change between iterations, and with my initialization method, each value of k from 1 to 10 stopped in at most 6 iterations.





3 is clearly the optimal number of clusters as this is where the scree plot stops decreasing substantially. I believe the reason it doesn't decrease more towards the later iterations is due to my random initialization technique.