CSE 343: Machine Learning - Assignment 2

Akshat Gupta

November 5, 2023

Section A

(a) For a CNN with an input of dimensions I_H x I_W x I_C, and kernel of size K_H x K_W x K_{IC} x K_{OC} with padding = P and stride = S where

$$\begin{aligned} I_H &\rightarrow \text{Input height} \\ I_W &\rightarrow \text{Input width} \\ I_C &\rightarrow \text{Input channels} \\ K_H &\rightarrow \text{Kernel height} \\ K_W &\rightarrow \text{Kernel width} \\ K_{IC} &\rightarrow \text{Kernel Input Channels} \\ K_{OC} &\rightarrow \text{Kernel Output Channels} \end{aligned}$$

the output of the kernel is

$$\begin{aligned} O_H &= \frac{(I_H + 2P - K_H)}{S} + 1 \\ O_W &= \frac{(I_W + 2P - K_W)}{S} + 1 \\ O_C &= K_{OC} \end{aligned}$$

(a) For an input of size 15 x 15 x 4, the outputs of each of the layers are

$$\begin{aligned} O_{1H} &= \frac{(15 + 2 \cdot 1 - 5)}{1} + 1 = 13 \\ O_{1W} &= \frac{(15 + 2 \cdot 1 - 5)}{1} + 1 = 13 \\ O_{1C} &= 1 \end{aligned}$$

$$\begin{aligned} O_{2H} &= \frac{(13 - 3)}{2} + 1 = 6 \\ O_{2W} &= \frac{(13 - 3)}{2} + 1 = 6 \\ O_{2C} &= 1 \end{aligned}$$

$$\begin{aligned} O_{3H} &= \frac{(6 + 2 \cdot 2 - 5)}{2} + 1 = 3 \\ O_{3W} &= \frac{(6 + 2 \cdot 2 - 3)}{2} + 1 = 4 \\ O_{3C} &= 1 \end{aligned}$$

(b) Pooling could possible serve the following purposes in a CNN architecture:

- **Dimensionality Reduction** - Pooling helps reduce the dimensions of the input volume. By downsampling the feature maps, the computational requirements in subsequent layers are reduced. This also reduces the number of learnable parameters and can thus prevent overfitting since fewer parameters imply less chance of fitting to the noise in the data.
- **Translational Invariance** - Pooling provides a degree of translation invariance i.e. the exact position of a feature in the input becomes less critical, making the network more robust to variations in the location of the features.
- **Feature Selection** - It captures the best of the salient features by taking the maximum or average value within a neighborhood, thus retaining the most dominant features.

(c) For a convolutional layer:
Number of learnable parameters = K_H x K_W x K_{IC} x K_{OC}
For a pooling layer:
Number of learnable parameters = 0
$\Rightarrow$ For convolutional layer 1:
Number of learnable parameters = 5 x 5 x 4 x 1 = 100
$\Rightarrow$ For convolutional layer 2:
Number of learnable parameters = 0
$\Rightarrow$ For convolutional layer 3:
Number of learnable parameters = 5 x 3 x 4 x 1 = 60

$\Rightarrow$ Number of learnable parameters for the CNN = 100 + 60 = 160

(b) If the value of k is kept constant, the algorithm can converge to the same k-way partition based on the choice of cluster centroids taken initially. This is because if the initial centroids are chosen from within the same cluster that they converged to in the original configuration then the same configuration will be revisited by the k-means algorithm. The algorithm would converge to the same configuration since the nearest points to it would remain the same irrespective of the number of times the algorithm is run. This is because the algorithm is minimizing a cost function, specifically the sum of squared distances between data points and their assigned cluster centroids. Since the number of possible configurations (assignments of points to clusters and corresponding centroids) is finite, and the cost function decreases or remains the same at each iteration, the algorithm cannot continue indefinitely cycling through configurations. The monotonic decrease in the cost function and the finite number of configurations ensure that the k-means algorithm converges in a finite number of steps.

(c) KNN is a non-parametric and instance-based algorithm that makes predictions based on the majority class of the K-nearest data points. It relies on measuring distances between data points and doesn't involve learning a set of parameters or weights, which is a fundamental characteristic of neural networks. In kNN, the entire training dataset is used to make predictions, and there is no underlying model that is trained to generalize from the data. Each prediction involves finding the K-nearest neighbors and making a decision based on their labels. Neural networks, on the other hand, are designed for learning complex patterns from data through the adjustment of weights during the training process. They consist of input layers, hidden layers, and output layers, and the weights are optimized using suitable methods. This clearly conflicts with the process of kNN where no weights or parameters are involved and thus, a neural network cannot be used to model the kNN algorithm.

(d) The key difference between linear and non-linear kernels lies in the presence of activation functions. Linear kernels perform a linear transformation on the input data, while non-linear kernels incorporate activation functions to introduce non-linearities, enabling the network to learn and represent more complex patterns. A linear transformation is applied irrespective of the kernel; for non-linear kernels an activation function is used to map this linear transformation to a new value through an activation function f. In CNNs, non-linearities are crucial for the model to capture the hierarchical and abstract features present in images or other structured data, and thus non-linear kernels tend to perform better in most cases.

Section B

(a) `Convolutional` class implements the convolution layer for the CNN and implements the forward and backward pass for the network. It uses windowing with the `kernel_size, stride, num_filters` taken as arguments for the initialization of the network. The forward pass convolves the features from the input to that layer and introduces new salient features from the data with the size of the output of that layer given by `output_dim` = $\frac{\texttt{input\_dim - kernel\_size}}{\texttt{stride}}$ + 1. The backward pass calculates the gradient of the loss with respect to the weights of the current layer and the weights are subsequently updated given the learning rate.

(b) `MaxPooling` class implements the (max) pooling layer for the CNN and implements the forward and backward pass for the network. It uses both forward and backward pooling to evaluate the outputs and the gradients respectively for the network. The pooling layer specifically implements the max-pooling function for the purpose of this problem.

Multiple options for the activation functions were implemented in `utils.py` for the purpose of testing different ones and getting the best possible results. A synthetic dataset was generated using the `np.random.rand()` function of 1000 samples and their corresponding labels (binary classification) generated using `np.random.randint()` to test the CNN. The layer sizes were so set that the final layer produced an output of a single scalar value for each sample. The dataset was pre-processed as standard normalization i.e. each column was reduced to a Standard Normal distribution with mean=0 and std=1.

The data was then split into 70:10:20 sets corresponding to train:val:test sets respectively. The train and val sets, as is conventional, were used to train the model and estimate the error rate with the loss function being Regularized Cross Entropy. The most optimal hyperparameters for this particular data were obtained as

$$\begin{aligned} \texttt{num\_epochs} &= 1000(\text{kept fixed}) \\ \texttt{learning\_rate} &= 5e-3 \\ \texttt{regularization} &= 5e-4 \end{aligned}$$

with the layer sizes kept fixed at

$$\begin{aligned} \text{Conv Layer 1} &: (8 \text{ x } 5 \text{ x } 5) \text{ with Stride} = 1 \\ \text{Pool Layer 1} &: (4 \text{ x } 4 \text{ x } 4) \text{ with Stride} = 2 \\ \text{Conv Layer 2} &: (4 \text{ x } 4 \text{ x } 4) \text{ with Stride} = 1 \\ \text{Pool Layer 2} &: (2 \text{ x } 4 \text{ x } 4) \text{ with Stride} = 2 \\ \text{Conv Layer 3} &: (1 \text{ x } 3 \text{ x } 3) \text{ with Stride} = 1 \\ \text{Pool Layer 3} &: (1 \text{ x } 2 \text{ x } 2) \text{ with Stride} = 2 \end{aligned}$$

The performance metrics are represented in Table 1.

Set	Loss	Accuracy
Train	0.04821574	0.9042857143
Test	0.09360419	0.795

Table 1: Performance of CNN

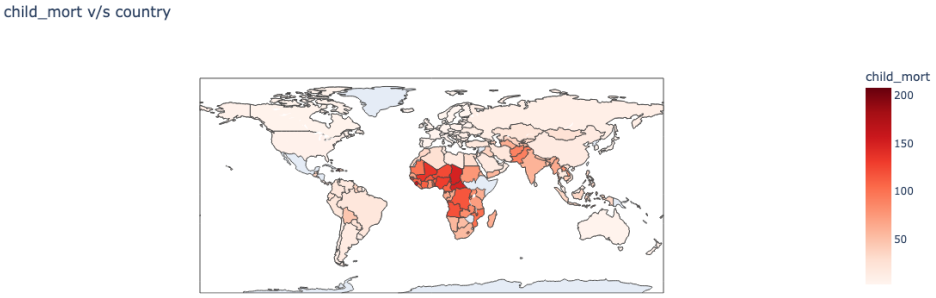

Figure 1: Child Mortality v/s Country

Figure 2: Exports v/s Country

Section C

(a) See Fig.s 1 - 9 for a visualization of the distribution of each of the features for the dataset with respect to the country. The intensity of the color denotes the values - the higher the intensity, the higher the corresponding absolute value of the attribute. The correlation heatmap for the original data is shown in Fig. 10.

The dataset was standardized using the `sklearn.preprocessing.StandardScaler()` module. Each feature was thus, standardized as $X_{new} = \frac{X_{old} - mean(X_{old})}{std(X_{old})}$ reducing it to a Standard Normal Distribution with mean 0 and variance 1.

(b) PCA was implemented on the preprocessed data to reduce the dimensionality. The cumulative explained variance ratio with respect to the number of components is shown in Fig. 11.

Figure 3: Imports v/s Country

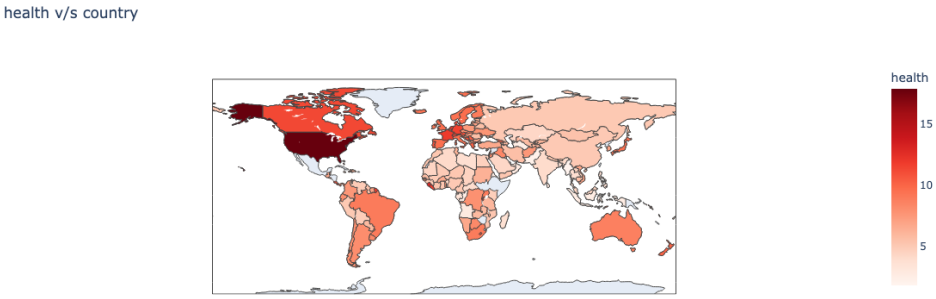

Figure 4: Health v/s Country

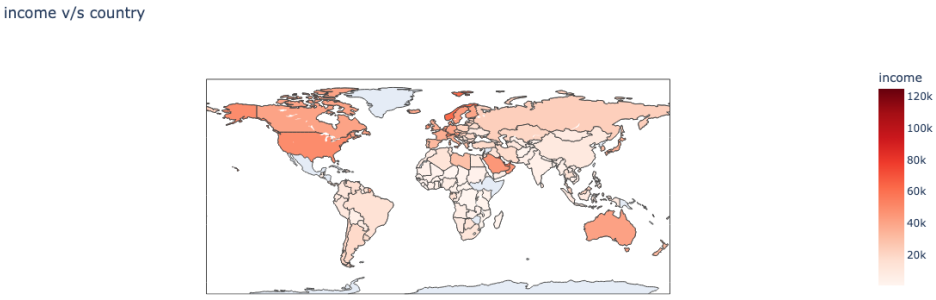

Figure 5: Income v/s Country

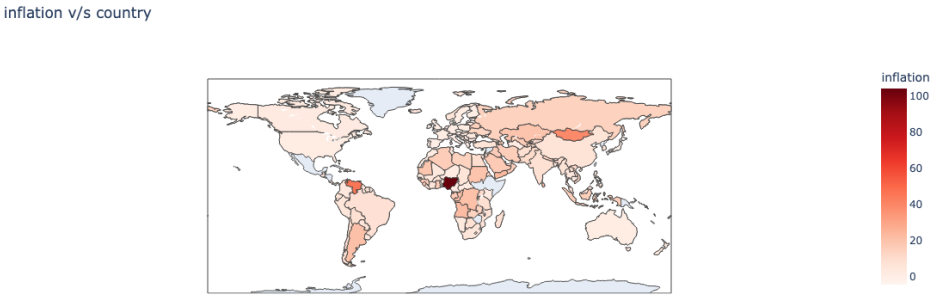

Figure 6: Inflation v/s Country

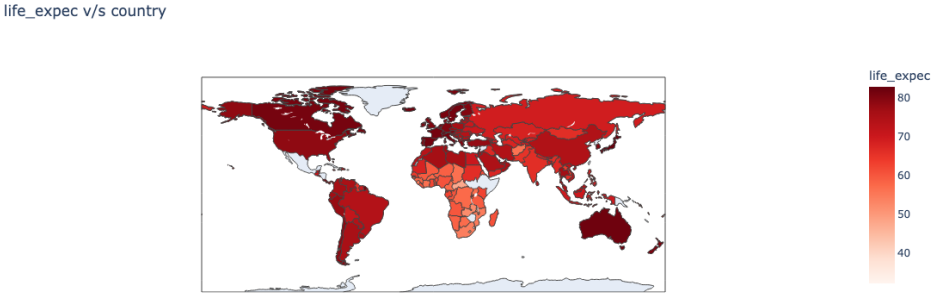

Figure 7: Life Expectancy v/s Country

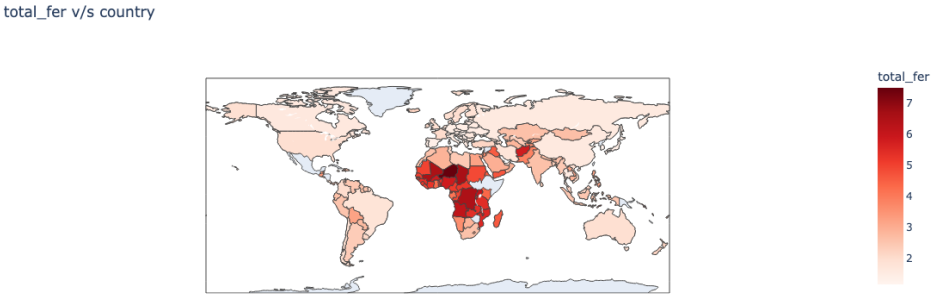

Figure 8: No. of Children Born at Current Fertility Rate v/s Country

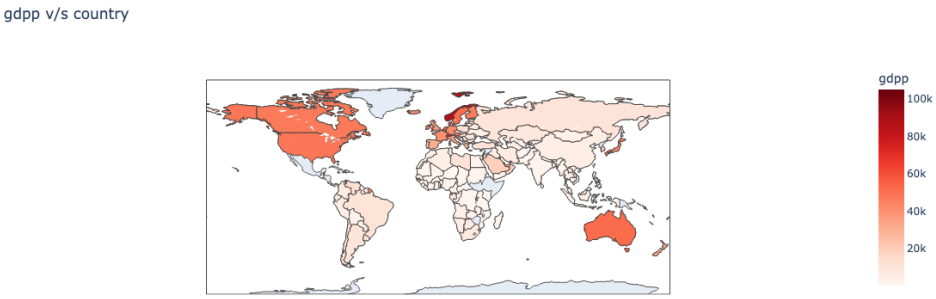

Figure 9: GDP Per Capita v/s Country

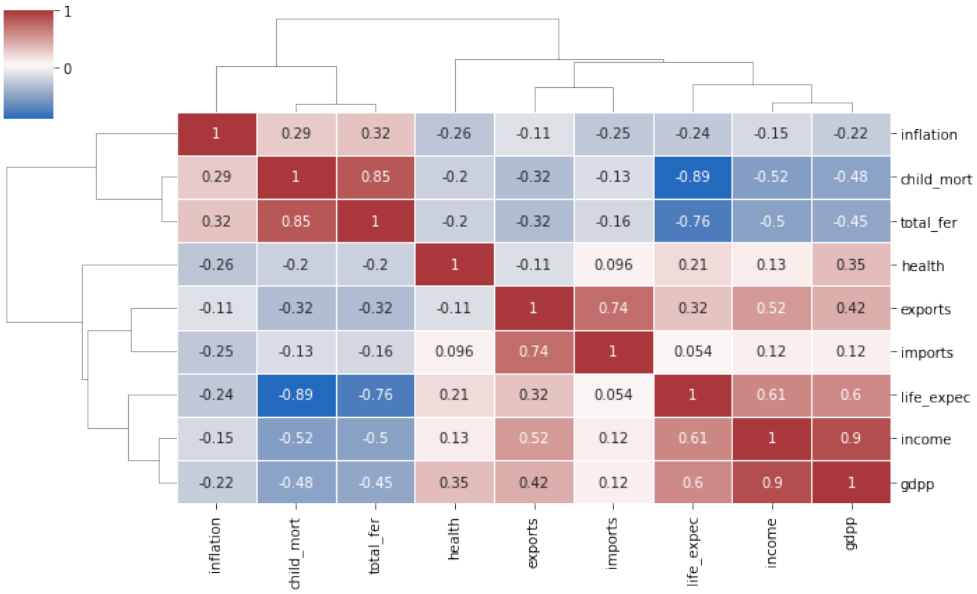

Figure 10: Correlation Heatmap

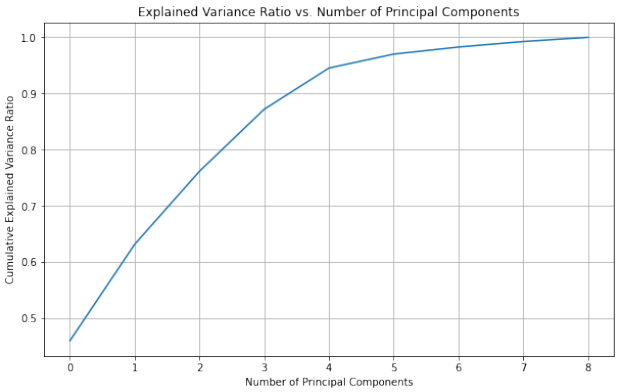

Figure 11: Cumulative Explained Variance Ratio

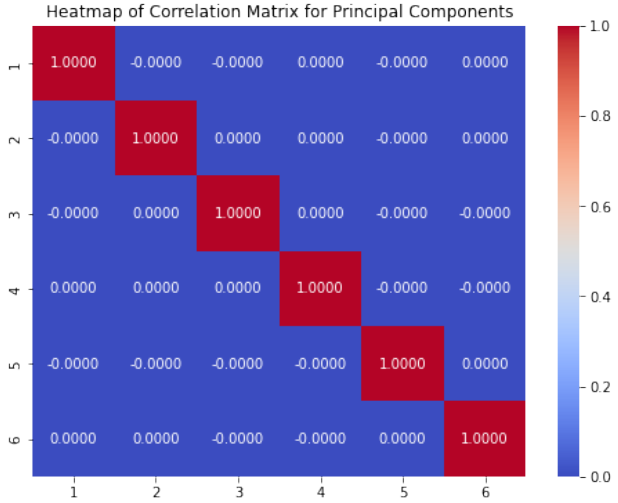

Figure 12: Correlation Heatmap for PCA

The optimal number of components is chosen as 6 looking at the graph in Fig. 11. The threshold value for the cumulative explained variance ratio was kept at 0.95 and `n_components=6` was the first value for which it was ≥ 0.95. See Fig. 12 for the correlation heatmap for the data after PCA was applied with `n_components=6` and `whiten=True`.

(c) k-Means clustering was then applied to the PCA-reduced data to label each of the samples with a cluster number. See Fig. 13 and Fig. 14 for the elbow method visualization and the silhouette score visualization. Since the elbow method did not clearly indicate a value for the number of clusters, only the silhouette score was considered for evaluating the optimal number of clusters. The maximum silhouette score was seen at `n_clusters=6`, and so the data was classified into 6 clusters.

See Fig. 15 for a visualization of the number of points that belong to each cluster.

See also Fig.s 16 to 19 for a visualization of how the points in each cluster are distributed with respect to each of the show features.

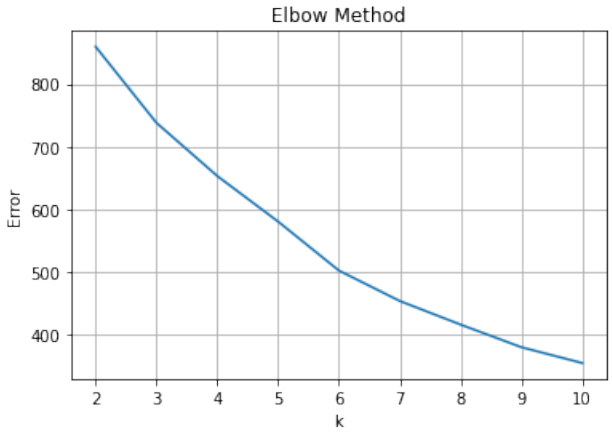

Figure 13: Elbow Method

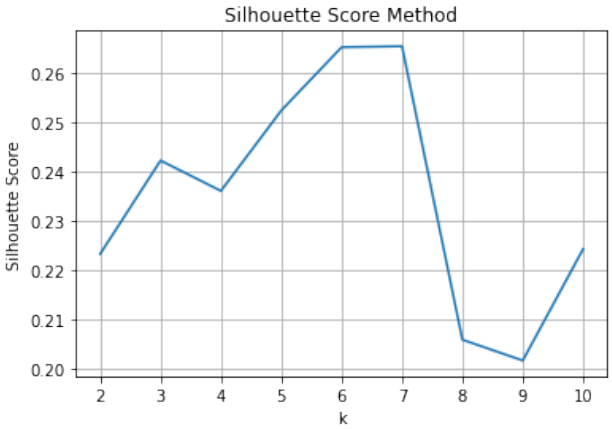

Figure 14: Silhouette Score Method

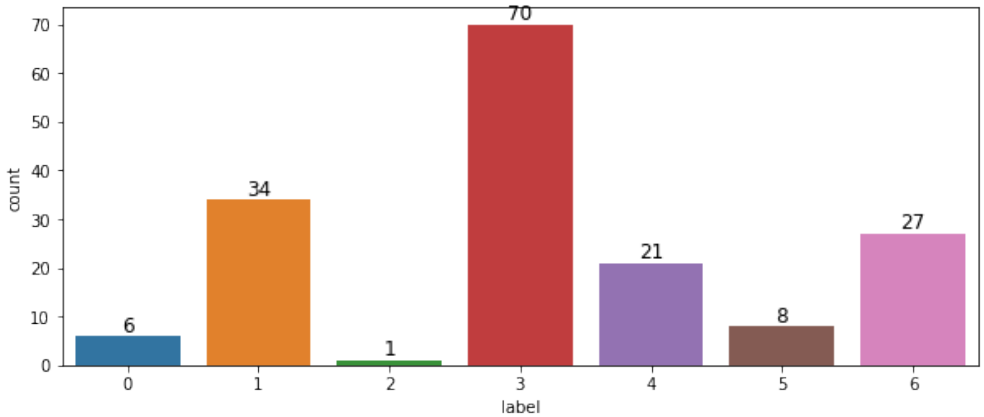

Figure 15: Cluster Distribution

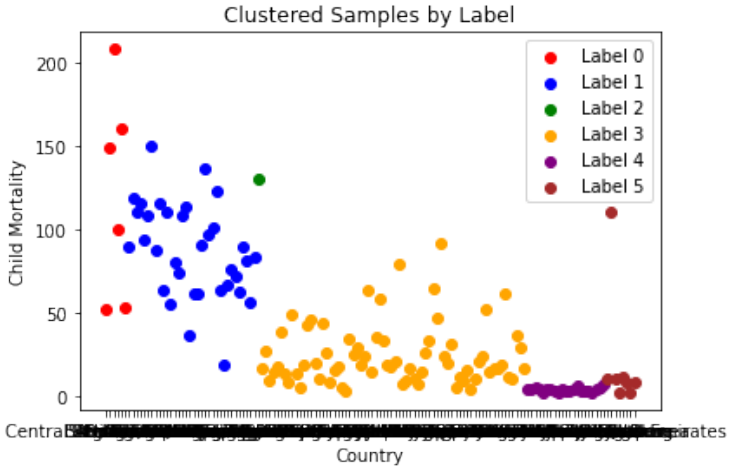

Figure 16: Clustered Samples by Label

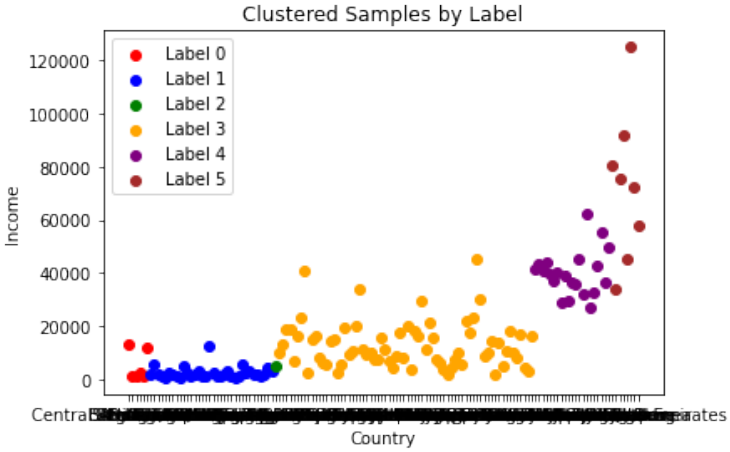

Figure 17: Clustered Samples by Label

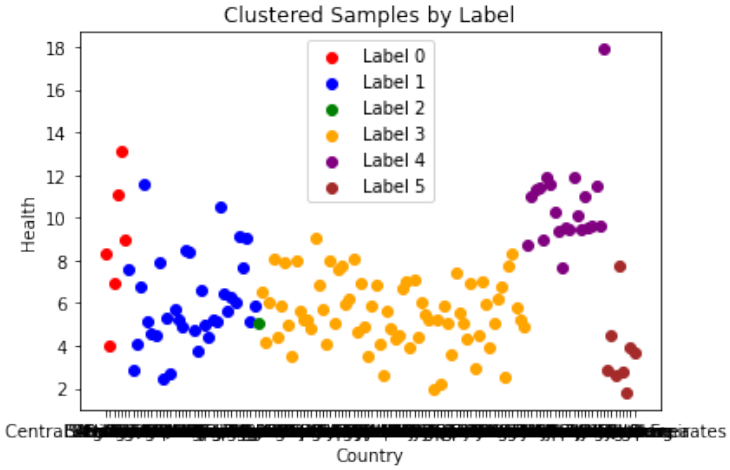

Figure 18: Clustered Samples by Label

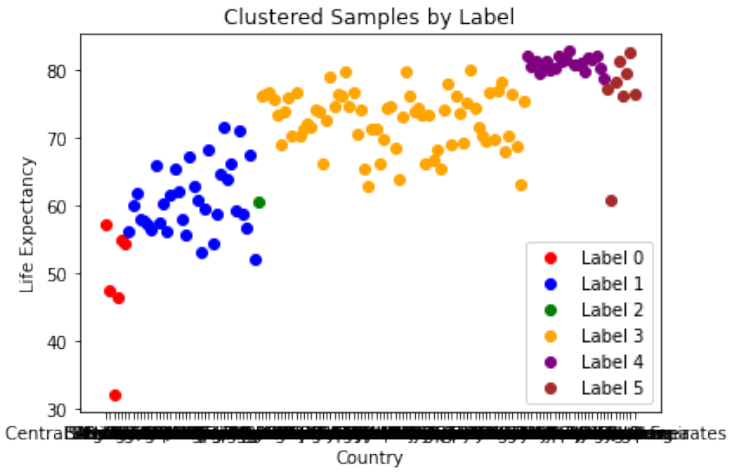

Figure 19: Clustered Samples by Label