Project Report

Horatiu Luci

5 December 2020

1 Data Extraction

For the data extraction part the given function was used. Normalisation was done by calculating the scaler using **preprocessing.StandardScaler().fit(X_train)** and **preprocessing.scale(X_train)** . After this step, features are standardized by removing the mean and scaling to unit variance.

2 Data Clustering with K-Means Algorithm

The K-Means algorithm is used in searching for a predetermined number of clusters within an unlabeled multidimensional data set. This is accomplished using the following conception of what the optimal clustering looks like:

- The "cluster center" is the arithmetic mean of all the points belonging to the cluster.
- Each point is closer to its own cluster center than to other cluster centers.

The idea is to identify data points with similarities and cluster them together while trying to distance each cluster as far as possible. The shorter the Euclidean distance the more similar the points are. In practice, when running this algorithm, the number of clusters K-Means needs to find is predetermined by some sort of user input. The number of clusters cannot exceed the number of features in the data set (in our case this number is 45).

Next, the algorithm will select a random point for each centroid. Then, it averages Euclidean distance (between each point and its centroid) for each cluster and this point becomes the new centroid. This process repeats for 1000 iterations on default.

However, this behaviour can be changed in SKlearn module. The "n_int" parameter determines the number of times K-Means will randomly select different centroids. "Max_iter" determines how many iterations will run. For instance if we set our parameters at n_int=10 and max_iter=250 K-Means will randomly select 10 initial centroids and run each centroid up to 250 iterations. The best out of those 10 centroids will be the final cluster.

In this project, data clustering has been done with the K-Means algorithm, the number of clusters has been varied increasingly from **2** to **10**. For each cluster that was obtained, the country that was the closest to the cluster centroid was computed using the given function **find_closest_instances_to_kmeans()**

3 Data Visualisation

Data dimensions begin to bear a burden on the ability to interpret the output of K-Means algorithm. Therefore, a visualisation method is needed so that the dimensionality of the data is reduced so it can be seen on the laptop screen.

For each data point (Xi) we'll center a Gaussian distribution over that point. Then we measure the density of all points (Xj) under that Gaussian distribution. Then re-normalize for all points. This gives us a set of probabilities (Pij) for all points. Those probabilities are proportional to the similarities. This means if data points X1 and X2 have equal values under this Gaussian circle then their proportions and similarities are equal and hence you have local similarities in the structure of this high-dimensional space. The Gaussian distribution or circle can be manipulated using what's called perplexity, which influences the variance of the distribution (circle size) and essentially the number of nearest neighbors. Normal range for perplexity is between 5 and 50 (1) and for this project, **a perplexity of 33 was used**.

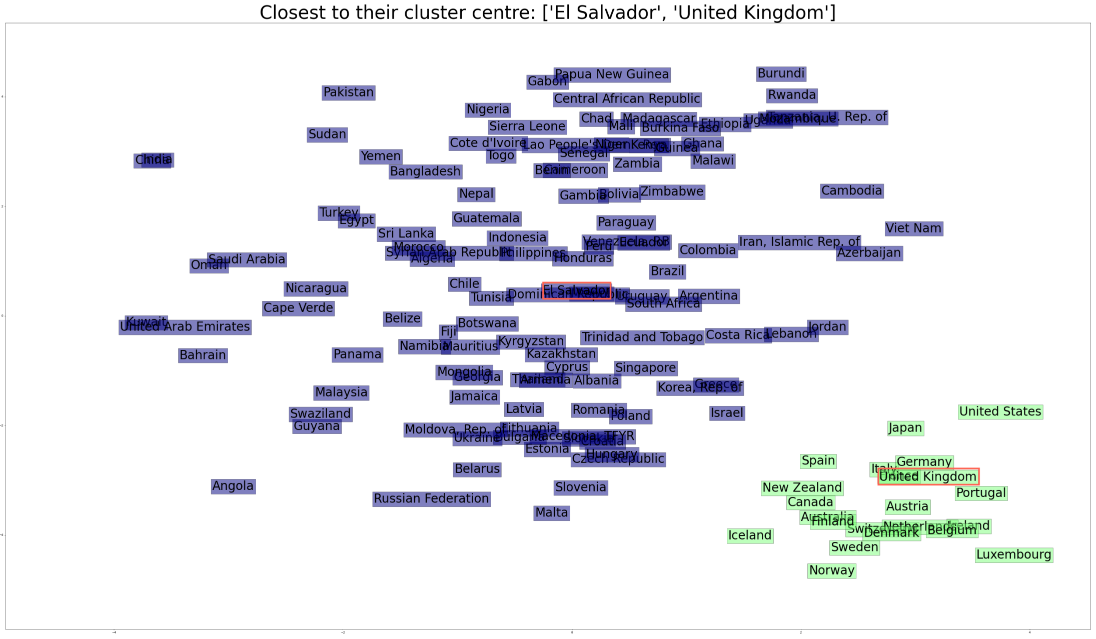
4 Clustering choice

For our project, for each number of clusters (from 2 to 10) the HDR data was clustered with the K-Means algorithm and visualised using the t-distributed stochastic neighbor embedding algorithm with a perplexity of 33. Therefore a number of 9 visualisations has been outputted with an increasing number of clusters.

Visualising not enough clusters (2 clusters)

Due to the nature of data being related to the development of countries' population, the scatter plot with 2 clusters will tend to be split into 'less developed' and 'more developed' countries. In the case we can only see the very developed countries from around the world such as **UK** (the closest to cluster centre) in one cluster and less developed countries in the other such as **El Salvador**(closest to its cluster centre). Even though the split is accurate, we still needs more clusters in order to identify those countries in the course of development, since representing this metric in such a binary way is not helpful.

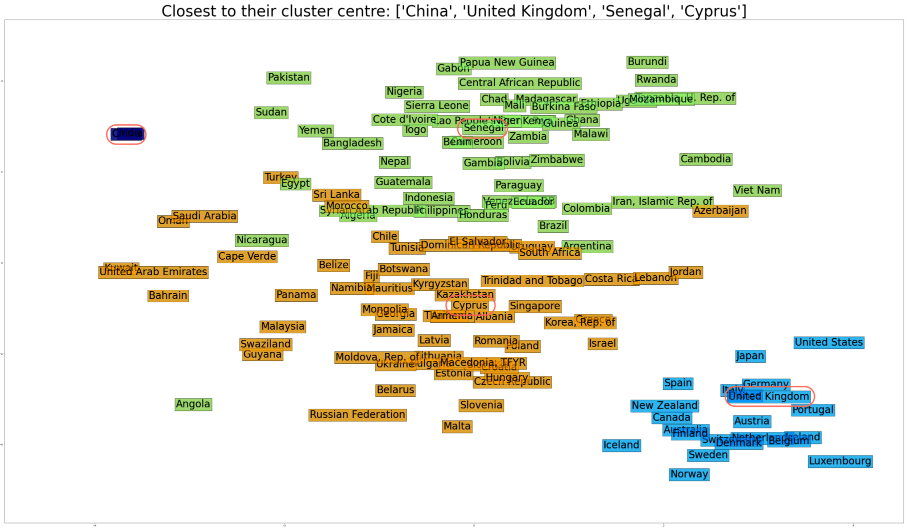
Closest to their cluster centre: ['El Salvador', 'United Kingdom']



Visualising proper clustering (4 clusters)

With 4 clusters it is enough to see the world split into superpowers such as the **UK (light blue)**, countries in course of developing such as **Cyprus (orange)** and undeveloped countries such as **Senegal (green)**. **India and China (dark blue)** are out-liars due to their enormous population.

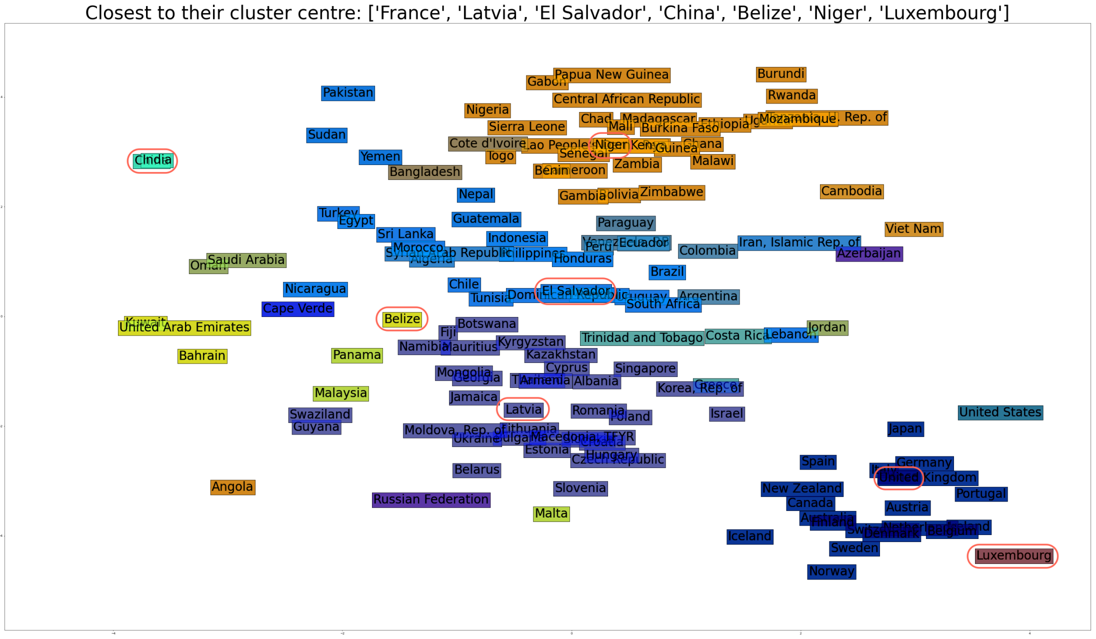
Closest to their cluster centre: ['China', 'United Kingdom', 'Senegal', 'Cyprus']



Visualising too many clusters (7 clusters)

When splitting into 7 clusters, even though some parts of the 4-way split can still be seen, some new clusters are seen that don't represent anything. For example **Luxembourg (lower right Plum color)** is in a cluster by itself and the semi-developed countries cluster (blue and dark blue) is now spit into Europe and the rest of the world. Cluster centres can be seen below.

Closest to their cluster centre: ['France', 'Latvia', 'El Salvador', 'China', 'Belize', 'Niger', 'Luxembourg']



References

[1] https://en.wikipedia.org/wiki/Relative_entropy