CSC2042S 2025 Assignment 1: Unsupervised Learning

In this assignment, you will apply K-means clustering to analyze global development patterns using the World Development Indicators (WDI) dataset. The assignment will focus on the practical challenges of applying clustering to real-world data, model optimization strategies, and interpretation of results. In addition to submitting the code for reproducing your analysis, you need to submit a report that documents your model design decisions and analyses the results.

Learning Objectives

Upon completion of this assignment, you should be able to:

- Handle real-world data preprocessing challenges in clustering applications
- Evaluate and optimize clustering algorithms through various initialization and convergence strategies
- Assess cluster quality using multiple validation approaches
- Apply dimensionality reduction techniques to improve clustering performance
- Critically analyze the trade-offs in clustering design decisions

Dataset

The World Development Indicators database from the World Bank contains over 1,400 time series indicators for 220 countries and territories from 1960 to present. The dataset contains key development indicators such as:

- Economic indicators (GDP per capita, inflation, trade balance)
- Social indicators (life expectancy, literacy rate, population growth)
- Environmental indicators (CO2 emissions, forest area, renewable energy consumption)
- Infrastructure indicators (internet users, mobile subscriptions, electricity access)

The dataset is available on the course Amathuba site under Content / Resources / Datasets. The source of the dataset is: https://datacatalog.worldbank.org/dataset/world-development-indicators. Your code should load the dataset from a specified folder in the given format: Do not submit the dataset with your assignment or let the code download the dataset. In the main file in the dataset, WDICSV.csv, each row represents an indicator for one country, with columns for the value of that indicator at different years. However, the indicator values are not always available for every year. In your data analysis, treat each country-year combination as one multi-dimensional data point, with development indicators serving as features. This will enable clustering countries with similar indicators and to track the development indicators over time (e.g. how similar is South Africa in 2024 to South Korea in 1990). Additionally, the Human Development Index scores/classifications of countries are also included – this should only be used as a reference classification for analyzing the final clustering.

Assignment Tasks

1. Data Preprocessing (6 marks)

Load the WDI dataset and prepare it for clustering. You will see that the data on some indicators are missing for some countries or not available for each of the years covered by the dataset as a whole. You must implement an approach to handle missing values: For each country, only include years where data is available for a sufficiently high proportion of the features. Also remove features that have missing values for too many data points (remember that each data point represents all the features for each country/year). You will need to test different thresholds for these two different strategies to remove data points to find a good balance between coverage and completeness of the remaining data. For the remaining missing values, apply and justify an appropriate data imputation strategy (i.e. estimate the missing values), e.g. based on the mean of related available values. Also consider normalizing the feature values, e.g. between 0 and 1 or -1 and 1, as that may lead to better clustering performance. You may use a library such as Pandas for data loading and preprocessing.

Conduct exploratory data analysis by visualizing the data. To do this it is recommended to use t-SNE, a dimensionality reduction method that is particularly suitable for visualizing high-dimensional data. You may use an existing implementation of t-SNE, for example using scikit-learn. Document the impact of each preprocessing decision on the final dataset size and characteristics, providing clear justification for your choices.

2. K-means Clustering and Initialization (6 marks)

Implement K-means clustering in Python using NumPy data structures – you may *not* use an implementation in an existing package such as scikit learn for this.

Implement and compare multiple initialization strategies to understand their effects on the resulting clustering. Start with random initialization, running the algorithm multiple times with different random seeds to assess the variability in results. Then implement *K-means++* initialization, which selects initial centers that are far apart, and analyze how this affects convergence speed and final cluster quality.

For each initialization method, track and visualize the convergence behavior across multiple runs, recording metrics such as the number of iterations to convergence, final loss function value, and the stability of cluster assignments across runs. Use this information to make a data-driven recommendation for the best initialization strategy for this dataset.

Optional: Design and implement a custom initialization strategy based on the specific characteristics of the data - for instance, you might select initial centers based on countries representing different development stages or geographic regions.

3. Convergence Criteria (3 marks)

Effective convergence criteria are important for balancing computational efficiency with clustering quality. Implement and evaluate multiple different convergence criteria: maximum iterations, centroid change threshold (when cluster centroids move less than a specified distance), and inertia change threshold (when the loss function improvement falls below a minimum value).

Analyze how each criterion affects the trade-off between computational cost and solution quality. Plot the trajectory of the objective function and centroid movements under different stopping conditions. Document which threshold values are most appropriate for this dataset, based on your empirical findings.

Optional: Also implement an early stopping mechanism which stops training if no improvement (or below threshold improvement) is seen for a specified number of iterations (rather than just comparing the last two iterations).

4. Determining the Optimal K (3 marks)

Selecting the optimal number of clusters is a fundamental challenge in unsupervised learning. Test your K-means implementation with a range of values of K (ranging from 2 to at least 15). Then use the elbow method to determine the optimal value of K, both by manually inspecting the loss function graph and by implementing automatic detection based on the change in the loss function.

Visualize the optimal clusters found at different values of K. Synthesize your findings to provide a recommendation for the optimal K, discussing any disagreements between approaches.

Optional: Another way to compare cluster quality is to use the Sillouette score, which considers both intra-cluster cohesion and inter-cluster separation (while the K-means loss only considers cluster cohesion). You may use the scikit-learn implementation of the Sillouette score.

5. Dimensionality Reduction with PCA (5 marks)

Applying dimensionality reducing before clustering has the potential to improve the performance of clustering high-dimensional data. Apply principal components analysis (PCA) to your preprocessed data, testing with different numbers of components (dimensions). You may use the PCA implementation from a package such as scikit-learn. Provide an interpretation of the top three principal components, identifying which original features contribute most strongly to each component and therefore what development concepts they might represent. Compare the results of K-means clustering on the original feature space to that of clustering the PCA-transformed data.

Visualize the clusters with t-SNE, and additionally also visualize the clusters in 2D and 3D PCA space, using the first two or three principal components. Discuss whether the dimensionality reduction reveals clearer cluster structure or loses important information.

Optional: Analyze differences in clustering quality using the Silhouette score and compare computational efficiency (time and iterations to convergence).

6: Cluster Interpretation (3 marks)

The ultimate goal of clustering is to discover meaningful patterns in the data. Provide comprehensive interpretation of your final clusters. One way to do this is to identify representative countries that exemplify each cluster's characteristics. Develop cluster "personas" that capture the essential development patterns of each group, giving them descriptive names like "Emerging Industrial Economies" or "Resource-Rich Developing Nations."

Compare your clusters against the country classifications given by the Human Development Index (also included in the provided data), discussing agreements and discrepancies.

Optional: Analyze how countries transition between clusters over time. Conduct temporal analysis to understand development trajectories, identifying countries that have moved between clusters and what changes drove these transitions. Identify surprising or counterintuitive groupings and provide potential explanations for these unexpected associations.

7. Creative Extensions (8 marks)

Implement at least two extensions to demonstrate advanced understanding and creativity. Below are some options, but you may also propose your own extensions. The optional functionality listed in the descriptions above will also be scored under this section.

Options:

- **Feature Engineering**: Define composite features that capture development concepts - for example by combining indicators related to health, education, and income. Perform clustering based on those features and compare the results with clustering the raw features. Note that you can use the structure of the indicator codes in the data to help to identify groups of related features.
- **Outlier Analysis**: Identify outliers in the dataset and analyse the effect of excluding the outliers when performing clustering.
- **Hierarchical Clustering**: Implement any hierarchical clustering method to create a multi-level analysis. Analyse whether the method discovers a meaningful hierarchy of clusters, and compare to the results from K-means clustering with a varying number of clusters.
- **Alternative Clustering Algorithms**: Implement and thoroughly compare K-means with another clustering algorithm such as DBSCAN or Gaussian Mixture Models.
- **Efficient implementation**: Identify computational bottlenecks in your implementation and optimize the performance of the code. One option is to use vectorized operations with NumPy as much as possible to eliminate Python loops. You can also consider other strategies to speed up processing of large datasets such as parallel processing. Compare

memory usage and runtime between your optimized and baseline implementations across different dataset sizes.

Submission Requirements

Submit a single compressed archive named as your student number, e.g. GWRBRA001./tar.xz. The code should be submitted as a Jupyter notebook with clear documentation. Version control with git is recommended but not required. The code should be able to run based with only the provided data and standard libraries such as numpy, scikit-learn and matplotlib (check with a tutor if you are unsure about using other libraries). A README file should include setup instructions and a description of each submitted file. Do not submit any data.

Also include a report of at most 6 pages (submitted as a PDF document) that describes your design decisions, reports and discusses your results, including visualizations. Marks will be given primaily based on showing understanding of the concepts being applied and demonstrating ability to develop appropriate models and to analyse the results.

Please ensure that your tarball works and is not corrupt (you can check this by trying to downloading your submission and extracting the contents of your tarball - make this a habit!). Corrupt or non-working tarballs will not be marked - no exceptions.

A 10% penalty will be incurred for a submission that is one day late (handed in within 24 hours after the deadline). Any submissions later than 1 day will be given a 0% mark.

Academic Integrity

All code and analysis must be your own work. You may use standard libraries (in line with the specification of what is permitted for the different components of the assignment) and reference documentation, but must cite all sources. You may discuss your work with other students, but code sharing is prohibited. Use of AI tools must be declared and limited to debugging assistance only. AI assistance is not permitted for writing the report.

Marking Rubric

Category	Marks
Data Preprocessing	6
Initialization Strategies	6
Convergence Criteria	3
Optimal K Selection	3
PCA	5
Cluster Interpretation	3
Code Quality	3
Overall Report Quality	3

Optional and Creative Extensions	8
Total	**40**