

# Data Analysis using Spectral Clustering Algorithms

Yash Anand Raorane

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfilment of the requirements  
for the degree of

MASTER OF SCIENCE

September 19, 2024

# Student Declaration

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work. I confirm that I have not previously presented the work or part thereof for assessment for another University of Liverpool module. I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that I have not incorporated into this assignment material that has been submitted by me or any other person in support of a successful application for a degree of this or any other university or degree-awarding body.

SIGNATURE Yash Anand Raorane

DATE September 19, 2024

# Acknowledgments

I would want to sincerely thank you to my family for their continuous encouragement and support during this journey. Their belief of me has been a continual inspiration. I also sincerely thank you to my supervisor for their great direction, comments, and patience qualities that were absolutely essential in helping this project to be shaped. Particularly thanks to my colleagues and friends for their moral support and insights amid trying circumstances. Thank finally for making this possible for all those who directly or indirectly helped.

# Data Analysis using Spectral Clustering Algorithms

# Abstract

This work aims to investigate U.S. domestic flight statistics from 1990 to 2009 using spectral clustering methods. The main goal was to find latent trends in behaviour, airport traffic, and geographical clustering that would give aviation sector stakeholders important new perspectives. Comprising more than 3.6 million records, the data was pre-processed and dimensionally reduced using Principal Component Analysis (PCA) to guarantee computational efficiency and better clustering results. Using measures such as Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index, several clustering algorithms like K-mean, DBScan, and various spectral clustering techniques were tested.

By means of the study, Polynomial spectral clustering emerged as the most successful method providing the most well-defined and unique clusters. Designed with Flask and Folium, the interactive web-based application visualised the results allowing users to investigate clusters geographically and by other criteria like location, year, and season. This tool increases the practical relevance of the insights produced by making clustering results available to technical as well as non-technical users. Future research could increase the scalability of the technology, combine real-time data, and include other elements including economic indicators and weather conditions to so improve the analysis.

# Statement of Ethical Compliance

Human volunteers were not part of this project. This publicly available dataset was sourced from the "USA Airport Dataset" on Kaggle [9] and geographic data from Natural Earth's Shapefile [3], guaranteed ethical compliance by eliminating personally identifiable information, thereby guaranteeing compliance with ethical guidelines. Following all ethical standards for data protection and research integrity, data modification and augmentation followed. This project falls under A0. According to BCS criteria, which guarantees no human participants were involved and the research method followed ethical standards to protect privacy and ensure the responsible use of data.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Scope . . . . .	1
1.2	Problem Statement . . . . .	1
1.3	Approach . . . . .	2
1.4	Outcome . . . . .	2
<b>2</b>	<b>Background</b>	<b>4</b>
2.1	Standardised Spectral Clustering . . . . .	4
2.2	Spectral Clustering's Mathematical Background . . . . .	5
2.3	Approach of Data Standardisation . . . . .	5
2.4	Variants in Clustering Algorithm . . . . .	5
<b>3</b>	<b>Design</b>	<b>6</b>
3.1	Proposed design . . . . .	6
3.1.1	Frontend . . . . .	6
3.1.2	Backend . . . . .	7
3.1.3	Customised Method for Clustering . . . . .	8
3.1.4	Cluster Evaluation . . . . .	9
3.1.5	Visualisation component . . . . .	11
3.1.6	Integration of Frontend and Backend . . . . .	12
3.1.7	Proposed GUI . . . . .	13
3.2	Changes to Proposed Design . . . . .	14
3.2.1	GUI modification . . . . .	14
<b>4</b>	<b>Implementation</b>	<b>15</b>
4.1	Workflow of Application . . . . .	15
4.2	Dimensionality Reduction . . . . .	16
4.2.1	Necessity of PCA in the Project . . . . .	16
4.2.2	Conduct PCA . . . . .	17
4.3	Algorithm Variations . . . . .	18
4.3.1	K-Means . . . . .	18
4.3.2	DBScan . . . . .	18
4.3.3	Standardised . . . . .	18
4.3.4	Normalised . . . . .	19
4.3.5	Kernel . . . . .	19
4.3.6	Polynomial . . . . .	19
4.4	Cluster Validation . . . . .	20
4.4.1	Dunn Index (A Dead End) . . . . .	20
4.4.2	Optimise Cluster Validation . . . . .	20
4.5	Analysis of Clustering . . . . .	21
4.6	Folium Visualisation . . . . .	21

<b>5</b>	<b>Testing and Evaluation</b>	<b>24</b>
5.1	Testing . . . . .	24
5.1.1	Test Case One . . . . .	24
5.2	Evaluation of Algorithms . . . . .	26
5.2.1	Explanation for Table 5.1 . . . . .	26
5.2.2	Explanation for Table 5.2 . . . . .	27
5.2.3	General conclusion . . . . .	28
<b>6</b>	<b>Project Ethics</b>	<b>29</b>
<b>7</b>	<b>Conclusion and Future Work</b>	<b>30</b>
7.1	Conclusion . . . . .	30
7.2	Future Work . . . . .	30
<b>8</b>	<b>BCS Criteria and Self Reflection</b>	<b>32</b>
	<b>Bibliography</b>	<b>33</b>
<b>A</b>	<b>Frontend Design Files</b>	<b>36</b>
A.1	HTML Structure . . . . .	36
A.2	CSS styling . . . . .	36
A.3	JavaScript Interaction and AJAX . . . . .	37
<b>B</b>	<b>Supporting Logic behind Clustering Mechanism</b>	<b>38</b>
<b>C</b>	<b>Environment Setup</b>	<b>39</b>
C.1	Install Python and pip . . . . .	39
C.2	Install Required Libraries . . . . .	39
C.3	Operating the Flask Program . . . . .	39
<b>D</b>	<b>Experimental results</b>	<b>40</b>
D.1	Additional Test Case . . . . .	40
D.1.1	Graph Visualisation . . . . .	40
D.1.2	Output of the Console . . . . .	41
D.1.3	Insight on Clusters . . . . .	41
<b>E</b>	<b>Supporting Snapshots</b>	<b>42</b>
E.1	Customised Library for filling colours . . . . .	42
E.2	Customised library for plotting . . . . .	42
E.3	PCA general steps . . . . .	42
E.4	PCA testing . . . . .	43
E.5	Non-Static Graph for Test case 1 . . . . .	43
E.6	Non-Static Graph for Additional test case . . . . .	43
<b>F</b>	<b>Choice of Algorithm</b>	<b>44</b>
F.1	K-Means working . . . . .	44
F.2	DBScan working . . . . .	44
F.3	Standardised working . . . . .	44
F.4	Normalised working . . . . .	44
F.5	Kernel working . . . . .	45
F.6	Polynomial working . . . . .	45



# List of Figures

3.1	Clustering Algorithm Implementation draw from Draw.io . . .	9
3.2	Flowchart for Cluster Evaluation . . . . .	10
3.3	Logical flow of Flask sever drawn on Draw.io . . . . .	12
3.4	Info.HTML . . . . .	13
3.5	Proposed layout of Dashboard . . . . .	13
3.6	New Dashboard . . . . .	14
4.1	Workflow of Implementation drawn on Draw.io . . . . .	15
4.2	PCA Test for explaining Variance Ratio . . . . .	17
4.3	Trial attempt of Dunn Index . . . . .	20
4.4	Usage of Tuple for optimising cluster evaluation code on Vi- sual Studio Code . . . . .	21
4.5	Final Clusters form post final clustering action . . . . .	21
4.6	Logical presentation of Folium logic on Visual Studio Code .	22
4.7	Folium Map marker pop-up displaying cluster information . .	22
5.1	Graphical Visualisation of U.S. Flight Data based on clusters	24
5.2	Console log from Visual Studio Code . . . . .	25
A.1	Actual Frontend Drop-down to choose . . . . .	36
A.2	Read more Button and its styling code snippet . . . . .	36
A.3	Actual code from project which fetch and converts request in JSON form . . . . .	37
D.1	Graphical Visualisation of U.S. Flight Data based on clusters	40
D.2	Console log from Visual Studio Code . . . . .	41
E.1	Customised colour Code Map . . . . .	42
E.2	Customised Plotting Library . . . . .	42
E.3	PCA general logic . . . . .	42
E.4	Console log of the tested PCA variance ratio run on Visual Studio Code . . . . .	43
E.5	Non-Static version of Folium and Flight vs Passenger variation	43
E.6	Non-Static version of Folium and Flight vs Passenger variation	43

# List of Tables

5.1	Evaluating Different Algorithms (Year:1992, Season:Winter, Region:West) . . . . .	26
5.2	Evaluating Different Algorithms (Year:2006, Season:Summer, Region:Northeast) . . . . .	27

# Chapter 1

## Introduction

### 1.1 Scope

This work investigates the application of sophisticated unsupervised machine learning method spectral clustering to examine U.S. domestic flight data 1990 to 2009. Finding latent trends in flight behaviour, airport traffic, spatial clusters is the main objective. The project consists in the creation of a web-based interface so that users may view and interact with the data, thereby making the results accessible. The project ensures the clustering result is interpretable and actionable by including principal component analysis (PCA) for dimensional reduction. Together with other techniques such as K-means and DBScan, Spectral clustering is accessed to find the best one for this dataset.

### 1.2 Problem Statement

Understanding flight traffic patterns can provide insightful information in the aviation sector for operational effectiveness and customer happiness alike. But this study uses a complex, high-dimensional dataset that makes it difficult to find significant trends using conventional clustering techniques. Common in real world flight data, non-convex geometries, high-dimensional data, noisy data all challenge existing clustering methods including K-means and DBScan.

Spectral clustering transforms into a network and partitions it depending on connectivity between nodes (airports), therefore addressing these constraints. This enables spectral clustering to efficiently find groups that conventional techniques find difficult detectable. Though useful, spectral clustering can be challenging to understand, particularly for people not familiar with the mathematics behind it or machine learning techniques.

Two central challenges our initiative seeks to address:

- Using high-dimensional flight data to expose trends and connections between airports, flights and areas by use of clustering methods.
- Providing non-technical users with straightforward clustering findings using an easy online interface that clearly separates spectrum clustering.

Without this initiative, important stakeholders would still be deprived of the sophisticated knowledge of clustering results, therefore restricting the possibilities to maximise airport operations or offer useful information to decision-makers

## 1.3 Approach

This project uses a multifarious strategy to strike a mix between technological proficiency and useful application. First pre-processed to handle missing variables, standardise formats, and get the data ready for clustering analysis, the dataset included thousands of flight records. The data was simplified using dimensionality reduction methods like PCA, therefore preserving just the most important dimensions to enable the most effective and understandable grouping approach.

Several clustering techniques were applied after the data was ready:

- Establishing a baseline using a well-liked clustering technique called K-means
- The capacity of Density-Based Clustering method (DBScan) to manage data noise was evaluated.
- Still, spectral clustering took front stage. Using this approach, the data is turned into a graph model whereby every node represents an airport and edges show flight links. The best fit for the dataset was found by evaluating versions of spectral clustering including normalised and kernel spectral clusters etc.,.

The quality of the clustering was assessed, and the ideal number of clusters was found using validation measure including the Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index following clustering. These measures enable an supervised evaluation of the clustering performance by giving a numerical estimate of the degree of data clustering.

Apart from the technical side of clustering, the project created a browser-based web interface leveraging Flask to dynamically show the outcomes. This interface gives users an easy means of visually exploring clusters and lets them filter data according to zone, year, and season. Incorporating dynamic components like interactive maps and cluster colouring helps to make the data exploration process interesting for technical and non-technical user alike.

## 1.4 Outcome

The result of this project is multi-dimensional. thereby benefiting both the technical domain of machine learning and pragmatic uses in flights:

- Clustering Analysis: By means of spectral clustering, our work effectively exposed significant trends in U.S. domestic flight statistics. These trends show not just the location of airports but also the linkages and traffic flow among them across time.
- Evaluation of Clustering Techniques: The project found spectral clustering to be the most efficient technique for managing the complicated, high-dimensional structure of the dataset by way of a comparison among many clustering algorithms (K-means, DBScan, and spectral clustering). Especially useful was spectral clustering capacity to control irregularly shaped and non-convex clusters.
- Visualisation and Accessibility: One of the main outputs of this project is the creation of an interactive web interface that enables a wide audience to easily access complicated clustering data. Filtering the data by year, area and season lets users explore clustering results depending

on particular criteria in great depth. This instrument improves the interpretability of spectral clustering, therefore enabling the findings to be understandable even to people without knowledge in data science.

- **Technical Contribution:** The project verified the clustering results with strong evaluation criteria like Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index. These measures verified the accuracy and quality of the spectral clustering method in identifying natural groupings in the data, therefore augmenting the corpus of the information on clustering algorithms in practical uses.

# Chapter 2

## Background

In unsupervised machine learning, clustering is a basic method used to find structures and patterns in data by grouping like data points into clusters. For basic, well-behaved datasets, traditional clustering techniques including K-Means and DBScan are generally successful; yet, they have great difficulty applied to complicated, high-dimensional data, such U.S. domestic flight from 1990 to 2009. Conventional approaches find it challenging to identify significant clusters in the dataset with several factors including flight distances, passenger volumes, and airport locations. More complex methods such as spectral clustering have been suggested to solve these difficulties for managing non-convex clusters and revealing latent links inside data [8]

Because Spectral clustering is resilient in handling non-linear, high dimensional data, it turned out to be the best fit method for examine out dataset. Spectral clustering can capture complicated linkages that conventional approaches would overlook by converting data into a graph structure and grouping depending on graph partitioning [14]. To maximise outcomes, this study uses spectral clustering as the main technique, together with other variants including kernel spectral clustering, normalised spectral clustering, and polynomial spectral clustering. Principal Component Analysis (PCA) and data standardising help to facilitate the clustering process by lowering dimensionality and therefore interpretability [13].

### 2.1 Standardised Spectral Clustering

Starting with the similarity graph, where data points are shown as nodes and edges between nodes are weighted depending on their similarity clustering is a graph-based method. Minimising a measure known as the normalised cut will help to divide the graph into clusters of strongly linked nodes [7]. For datasets with non-convex forms or complex associations that conventional approaches such as K-Means cannot readily capture, spectral clustering is very successful [10].

This work uses traditional spectral clustering as main method. The standardising technique guarantees that the clustering findings are not unduly influenced by features of various sizes, such geographical locations and passenger numbers. Standardising the data before using spectral clustering helps us to guarantee that the method considers all features identically, there enhancing the quality of the clusters.

## 2.2 Spectral Clustering's Mathematical Background

Spectral clustering has its mathematical roots in graph theory and the eigenvalue and eigenvector computation from the similarity matrix. Spectral clustering seeks to minimise the normalised cut criteria, which defines as:

$$\text{Normalised Cut} = \sum_{\text{clusters}} \frac{\text{cut}(A, B)}{\text{assoc}(A, V)}$$

where: -  $\text{cut}(A, B)$  quantifies the disconnection between cluster  $A$  and  $B$ , and -  $\text{assoc}(A, V)$  represents the total association of nodes in cluster  $A$  with the whole graph  $V$ . An optimal method for non-convex and non-linear data is to solve for the eigenvectors of the Laplacian matrix of the graph hence partitioning the data into clusters minimising the normalised cut [7].

## 2.3 Approach of Data Standardisation

Standardising is crucial given the varied character of the information, with variables like flight distance and passenger counts having very different scales. This work aimed to normalise the data by means of StandardScalar, therefore converting each feature to have zero mean and unit variance. This phase guarantees that no one feature excessively influence the clustering process. Spectral clustering especially depends on distance-based similarity measure, which can be distorted by non-standardised data [14]. Hence, Standardising is very crucial.

## 2.4 Variants in Clustering Algorithm

Apart from consistent spectral clustering, this work investigates several clustering techniques to assess their performance on this dataset. The following variances were applied:

- K-Means: Designed to reduce within-cluster variance, K-Means is a classic partitioning method. Effective for spherical clusters, it suffer with non-convex forms and high-dimensional data [8].
- DBScan: Density-based clustering method DBScan shines in noise handling and arbitrarily formed cluster discovery. It might, however, fail when the data density changes greatly [10].
- Normalised Spectral: Normalised spectral clustering changes the similarity graph such that, independent of distance from others, every data point makes equal contribution. It enhances non-uniform dataset performance of the algorithm [7] [5].
- Kernel Spectral: Applied a Kernel function to convert the data into a higher-dimensional space, Kernel Spectral clustering helps to separate non-linearly separable data [14] [6].
- Polynomial Spectral: Using a Polynomial kernel to improve the cluster separability in complicated, non-linear datasets, Polynomial Spectral clustering provides adaptability for many kinds of data relations.

To find their efficacy in clustering the flight data, each of these techniques was assessed using clustering validation criteria including the Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index.

# Chapter 3

## Design

Discussing the design of the software built for this topic. Designed to let users interactively investigate clustering patterns, the Spectral clustering visualisation tool is a browser-based program. User input on the frontend, which is handled by the backend, starts the flow; the outcomes are then sent back for view. Every element of the system frontend, backend, data processing, clustering are combined to offer a flawless user interface.

### 3.1 Proposed design

#### 3.1.1 Frontend

Several important frontend components intended to provide flawless user interaction, data filtering, and visualisation define the web application for spectral clustering of U.S. domestic flight data. Through many interactive components, the frontend interface lets users engage with the data, apply clustering methods, and view the clustering outcomes. The parts below show the general organisation of every element and their contribution to the web application's general operation.

##### 3.1.1.1 User Interface (HTML and CSS)

Designed with HTML, CSS, and Bootstrap, the web interface offers consumers a clear canvas on which to choose variables such as region, year, season, and clustering technique. The interface's design ensures responsiveness and simplicity of use, allowing consumers to see clustering results without advanced technical knowledge.

- **HTML structure:** Incorporating drop-down menus, buttons, and placeholders for graph outputs, HTML files `index.html` and `info.html` create the backbone of the interface. As an illustration Refer to Appendix A.1
- **CSS Styling:** Custom CSS file (`style.css`) manages the appearance of the interface elements such as buttons, drop-downs, and graph containers, therefore guaranteeing a consistent and aesthetically pleasing design over many pages and devices. The design uses Bootstrap for adaptability, so the site stays responsive. Refer to Appendix A.2 for the detail styling of all elements.

##### 3.1.1.2 Usage of JavaScript for Data Interaction (JSON and AJAX)

Making the frontend interactive and dynamic falls to JavaScript, mostly within `script.js`. Without running a complete reload, the script manages



user inputs, sends AJAX calls to the backend, and changes the page. The design used for script can be found in Appendix A.3 , which governs data transmission across the frontend and backend, as well as the dynamic updating.

- **Handling User Input:** The data is filtered using the captured drop-down selections zone, season, year and algorithm. Clicking the "Graph Generate" button starts a JavaScript method called `updateGraph()` which gathers the values from the drop-downs and creates a JSON object
- **JSON for Data Exchange:** User-selected choices are sent from frontend to the Flask server using JSON. The response is returned as JSON once the backend handles the request and produces the clustering results.

### **3.1.1.3 Data Flow from User Interface to server**

The flow starts when a user chooses different parameters (location, year, season, and algorithm) interacting with the frontend. AJAX call allows these inputs packed into a JSON object to be transmitted to the backend Flask server. The server answers the request, uses the pertinent clustering technique, and sends back to the frontend the clustering results as JSON. This back-and-forth interaction guarantees that the program is both dynamic and responsive, thereby enabling real-time changes of clustering results depending on user preferences.

### **3.1.2 Backend**

The fundamental system architecture in the backend is found in the `data.py` file's `fine_tune_model()` function. Processing the flight data, getting it ready for clustering, and using several clustering techniques depend critically on this ability. The backend process is fully explained here.

#### **3.1.2.1 Data Cleaning and Preparation**

Loading and cleaning the data comes first. This guarantees the data is ready for more work. Key operation comprise:

- **Splitting Columns:** City and Region components separate the columns `Origin_city` and `Destination_city`. This makes the data more finely tuned and facilitates more geographically focused analysis.
- **Datetime Conversion:** The flights dates are transformed to a date-time style and feature a `Calendar Year` column. This helps to sort information for particular year.
- **Dropping Unnecessary Columns:** The transformation of the flight-date column, it is eliminated together with other superfluous columns to simplify the dataset for analysis.
- **Handling Missing Data:** Any rows including null values are deleted to guarantee data consistency and prevent analysis mistakes.

Important for the clustering procedure are columns in the cleaned data including `Passengers`, `Seats`, `Flights`, `Distance`, `Origin_population`, and Geographic coordinates.

### 3.1.2.2 Feature Selection

Essential columns that provide understanding of flying patterns are narrowed down in feature selection. The important columns consist of:

- Passenger, Seats, and flights mirror the traffic and demand at airports.
- Distances: This clarifies the flying route lengths.
- Origin and Destination Population: These characteristics provides socio-economic background and help to separate urban from rural airports.

These characteristics enable differentiating flights depending on volume, distance, and airport relevance, so facilitating clustering.

### 3.1.2.3 Dimensionality Reduction with PCA

Principal Component Analysis (PCA) is applied for dimensionality reduction in the dataset considering its great dimensionality. PCA reduces the data by keeping just the most important components, therefore optimising the clustering process.

- Standardising the data to have a mean of zero and a variance of one helps to guarantee that every characteristic equally influences the clustering process before using PCA.
- Two Principal components: Two main components make out the dataset. These elements help to simplify the clustering process and catch the most variation in the dataset.

PCA guarantees that the clustering algorithms receive just the most essential data, therefore lowering noise and raising the quality of the clusters. This is further explained in section 4.2

### 3.1.3 Customised Method for Clustering

Standardised skeleton forms the foundation of the design of the clustering process among several algorithms as shown below in Figure 3.1. This design guarantees a consistent approach to clustering independent of the particular method applied. This skeleton's primary objective is to enable scalability and flexibility such that several clustering techniques may be easily combined with little structural change.

The skeleton's basic reasoning consists following steps:

1. The clustering procedure starts with variables meant for tracking the best scores and cluster counts.
2. Iteration is a loop running across several cluster setups using clustering techniques in each one.
3. Clustering outcomes are assessed for every iteration using pre-defined benchmarks like Davies-Bouldin Index, Calinski-Harabasz Index, and Silhouette Score. Explain
4. Score Update: The scores and cluster count are changed should the present iteration produce improved clustering scores.
5. Following iterative way through possible cluster configurations, the optimal model is chosen depending on the highest assessment scores.

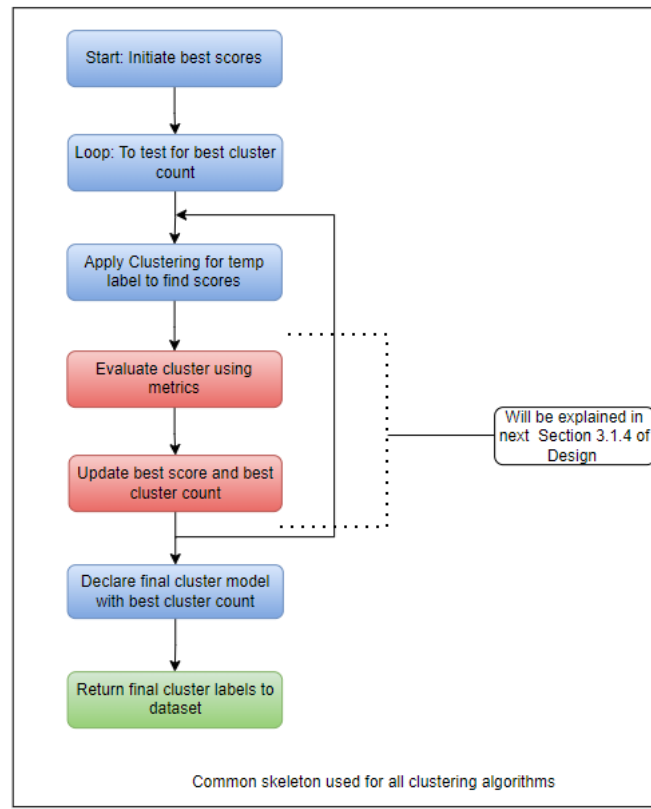


Figure 3.1: Clustering Algorithm Implementation draw from Draw.io

6. For every data point, the last cluster labels are generated ready to classify the airports for visualising purposes.

Due to the consistency of input-output structure, this skeleton facilitates scalability by letting the introduction of new algorithms with minimum code changes. Furthermore, the modular design guarantees that speed may be maintained while simple integration of additional functionality is possible. Additionally, details about parameters used for clustering algorithms shown in above Figure 3.1 can be found in section 4.3. Additionally, pseudo code based on the above architectural diagram, code structure and methodology offered for clustering algorithm can be found in Appendix B.

### 3.1.4 Cluster Evaluation

Cluster validation is a crucial phase in unsupervised learning, guaranteeing that clustering outcomes are both optimal and significant. This effort employed an evaluation system based on established clustering validation studies, utilising measures including the Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index to assess the quality of various cluster configurations. This evaluation procedure enables us to ascertain the optimal number of clusters by doing numerous iterations and contrasting the outcomes with these well specified parameters.

This Methodology is corroborated by research by [11] which illustrates that iterative cluster evaluation can ascertain the appropriate number of clusters, hence improving the interpretability and quality of the outcomes. In accordance with their paradigm, I utilised many indicators to thoroughly assess and validate our clustering models. By employing this strategy, the project successfully compared various algorithms, guaranteeing that the final

clusters were both precise and actionable. This approach allowed us to enhance the clustering procedure and identify the most effective algorithm.

#### 3.1.4.1 Process of Evaluation

The fundamental logic of the clustering architecture comprises two essential steps: Evaluation and Score Update as shown in 3.1. The subsequent code excerpt illustrates the function, which manages the scoring procedure during the cluster iterations.

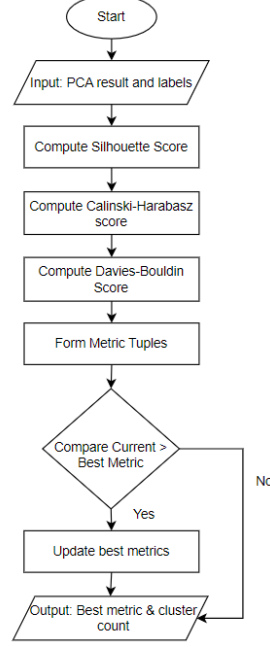


Figure 3.2: Flowchart for Cluster Evaluation

The function operates in the following manner:

- During each clustering iteration, Silhouette Score, Calinski-Harabasz Score, and Davies-Bouldin Score are computed based on the generated clustering labels.
- A tuple is utilised to retain the present collection of metric data and juxtapose them with the optimal set of values identified to date.
- The score update mechanism juxtaposes the current measurements against the optimal metrics.
- If the present metrics are superior, the tuple containing the optimal metrics is revised. This procedure guarantees the preservation of the optimal cluster count following each repetition.

The use of a tuple structure makes it more effective for comparing across several metrics, ensuring the most optimal model is chosen based on several factors rather than relying on one.

#### 3.1.4.2 Metrics for validating cluster

- **Silhouette Score:** This score quantifies the similarity of a data point to its own cluster relative to other clusters. The score varies from -1 to 1, with a higher score signifying that clusters are distinctly separated and points are accurately allocated. This metric is particularly

adept at assessing the cohesiveness and separation of clusters. Research conducted by [4] and [12] has thoroughly examined the use of the Silhouette Score for cluster validation across diverse fields, confirming its dependability.

- **Calinski-Harabasz Index:** Sometimes referred to as the Variance Ratio Criteria, assesses the variance between cluster against the variance inside individual clusters. Greater scores point to better defined clusters. Commonly used for evaluating clustering quality, this statistic has great relevance in field of multivariate analytic study including work of [1]. It helps especially in figuring the cluster separation and compactness.
- **Davies-Bouldin Index:** Gauges, on average, the similarity between any cluster and its most like one. Lower scores indicate better grouping; the measure is based on the ratio of within-cluster scatter to between-cluster separation. As mentioned in the original study by [2], this statistic is particularly useful for spotting distinct from one another clusters.

All three indicators helped to assess cluster setups in our work. Still, Silhouette Score displayed the most consistent results throughout several cluster counts and clustering techniques. These three criteria used together guaranteed a strong well-rounded validation of the clustering outcomes, hence guiding the choice of clustering model for every dataset. Please refer to 5.1 where scores are captured for few test cases.

### **3.1.5 Visualisation component**

Visualising the clustering findings follows clustering validation to help to make them more interpretable. I used Folium for map-based visualisation and NetworkX for graph representation to visualise the clusters for the airport clustering project. Both approaches gives clear, interactive visuals using the reduced PCA data and final cluster labels

#### **3.1.5.1 Folium for Interactive Visualisation**

Primary visualise with the help of Folium to provide an interactive map-based depiction of the clusters. For non-technical users, this approach is more straightforward since it overlays the clusters over a U.S. geographical map. Folium featured the following components:

- Markers for every airport (node), easily identified by cluster label.
- Popups offering comprehensive details on every airport, including origin, destination, flight counts, seats, and passengers.

Folium let us create an interactive map where users could zoom, pan, and click on certain airports to view comprehensive cluster data. This improves user experience and facilitates interpretation of airport geographical distribution and cluster linkages.

#### **3.1.5.2 NetworkX for Map Representation**

Followed by Folium map, thought of adding a static version of cluster formation with the help of NetworkX.

- Using the Origin airport column as a nodes. With weights matching the number of flights between two airports (nodes), edges based on flight data represent links between origin airports.
- With the nodes coloured in line with their designated clusters, this graph offers a clear , structural perspective of the airport connections and clusters.

NetworkX allows us to see the connections between airports in an aesthetically please graph and record their interactions. I also use the Laplacian matrix, which captures the structure of the graph, therefore enabling us to efficiently breakdown it and compute clusters can be found left graph of Appendix E.5.

### 3.1.5.3 Colour Mapping Function

Enhanced visual difference among cluster by using the `get_color_map` tool can refer from Appendix E.1. This utility takes a cluster label as input and generates the corresponding colour name identified by Folium. Giving every cluster a different shade enhances visual difference on the map, therefore enabling users capacity to identify links and patterns.

### 3.1.5.4 Customise Plotting Library

I made a similar to colour code map, a plot library mapping methods names to their appropriate charting functions to effectively manage several clustering techniques. I utilise this customised libraries to call suitable plotting function, and Flask route handling the graph update. The design can be found in Appendix E.2.

### 3.1.6 Integration of Frontend and Backend

This project integrates the frontend and backend using Flask as the web framework, therefore enabling communication between the HTML/JavaScript frontend and the Python backend. Between user encounters and backend data processing, flask act as the link.

#### 3.1.6.1 Flask Backend Architecture

Routes in Flask enable handling of HTTP enquires. The integration goes as shown in the Figure 3.3

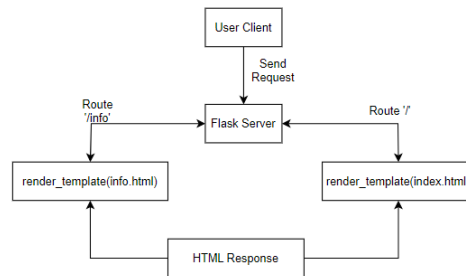


Figure 3.3: Logical flow of Flask sever drawn on Draw.io

Rendering HTML Pages: The Flask `render-template()` tool lets the user interact with the application by rendering HTML files (`index.html` and `info.html`). These path handles frontend files as user visits other pages.

### 3.1.6.2 Data Processing and API Response

When users request a clustering (e.g., choosing a clustering algorithm and filters), the client sends a JSON payload to the backend via a POST request. While in return JSON receive via /updateGraph, Flask handles it performs action, and create visuals send it back to frontend.

### 3.1.7 Proposed GUI

#### 3.1.7.1 Homepage(info.html)

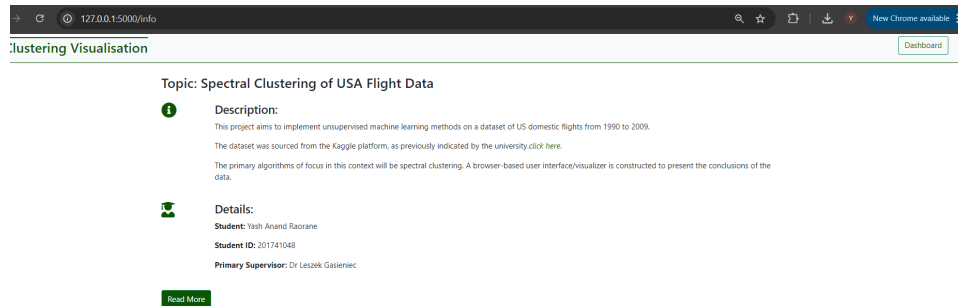


Figure 3.4: Info.HTML

Acting as the online application's homepage, the info.html page shown in Figure 3.4 project authors basic information and project descriptions. It features a "read more" button so users may investigate the goals and aspirations of the project. Users of a dashboard button in the right corner can access the main feature where clustered traffic patterns enable exploration of U.S. flight data.

#### 3.1.7.2 Dashboard



Figure 3.5: Proposed layout of Dashboard

The dashboard was proposed earlier had two buttons in which generate button as shown in the above Figure send request to the users select options and Flask server returns back the response with cluster results. Then EDA button was given right next to it, for getting basic idea of data.

## 3.2 Changes to Proposed Design

There were no major changes in backend procedure of the project just minor changes in User Interface as mentioned below:

### 3.2.1 GUI modification

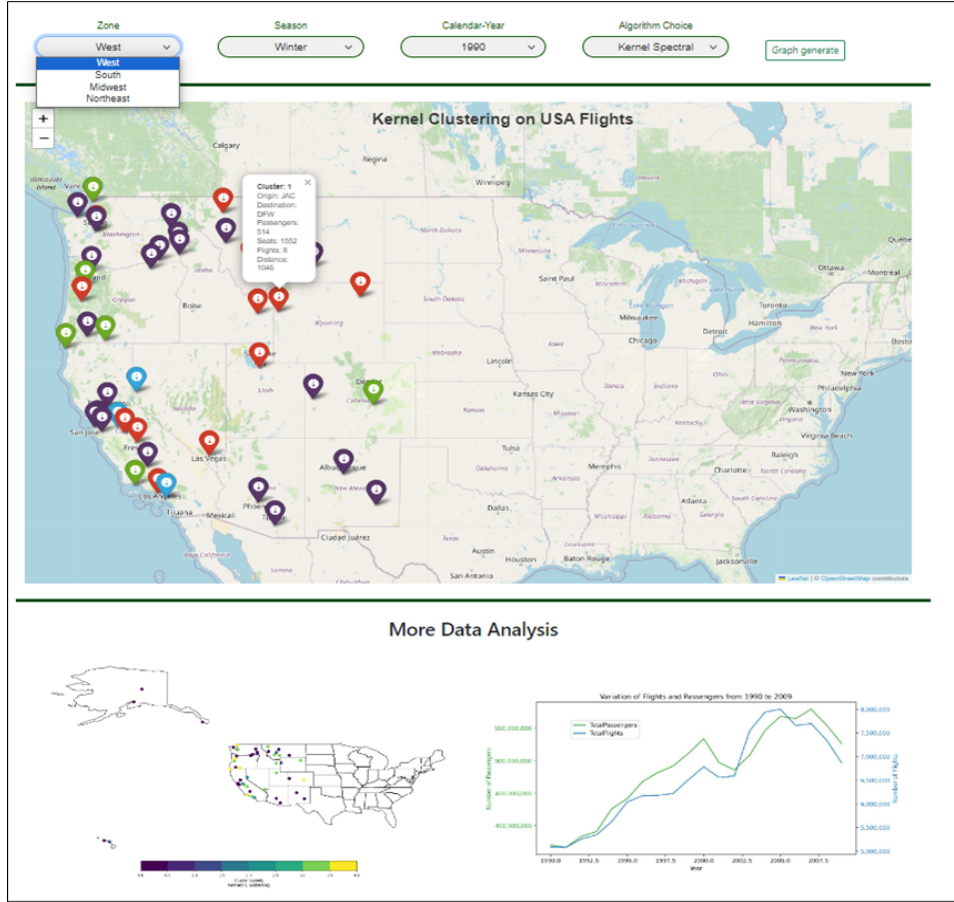


Figure 3.6: New Dashboard

Using JavaScript, users make choice on the dashboard, therefore generating a client request with their preferences (zones, season, year, algorithm). The server handles this request; the answer appears in the user's browser dynamically created and shown in a graph, additional to that below the main folium map, two EDA maps are generated but this feature does not required EDA button , it can be generated via same Graph Generate button at the same time. Hence, I combined the functionality of both in one.



# Chapter 4

## Implementation

### 4.1 Workflow of Application

Starting with the general flow of how the system handles user requests and generates significant clustering insights on U.S. domestic flight data, the implementation part of this project starts built with a frontend user interface for parameter selection including the location, year, season, and clustering technique, the browser-based utility uses a backend driven by Flask to manage data processing and clustering.

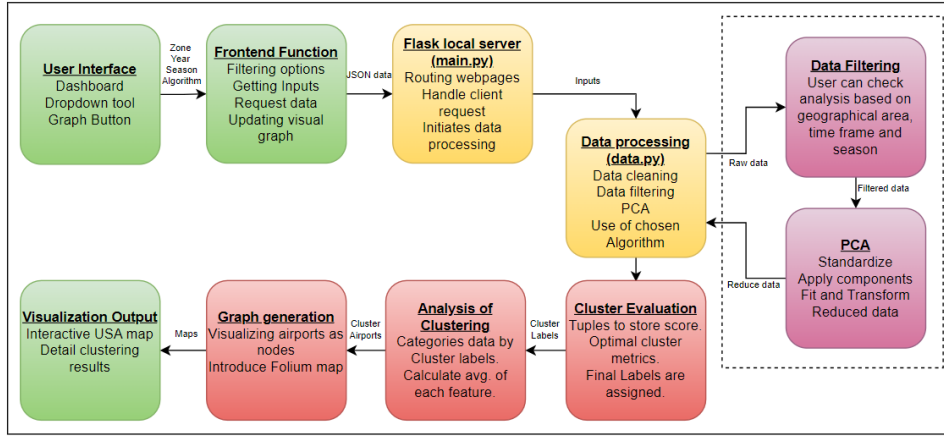


Figure 4.1: Workflow of Implementation drawn on Draw.io

The workflow as shown in above Figure 4.1 begins when a frontend user selects options from the drop-down given to check for that specific tailored data on Index.html, the frontend feature records this set and generates a JSON object. Sent via an AJAX request to the Flask backend as explained earlier through Figure 3.3, which manages server-side processes. Once the backend receive request, the selected data is filtered and cleaned. Missing values are addressed, and extraneous material is eliminated. Dimensionality is reduced using principal component analysis (PCA), therefore guaranteeing the retention of just the most significant dataset features.

Following this, cleaned and reduced data moves using the designated technique. First creating temporary labels for the data points, the clustering procedure runs several times utilising evaluation criteria including the Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index to ascertain the ideal number of clusters. Every iteration's evaluation findings are aggregated and the cluster count with the best metric score chosen. The ultimate cluster labels are assigned once the ideal cluster count has been found, this is previously discussed in section 3.1.3 and pseudo code is

mentioned in Appendix B.

Generated using Folium for visualisation, the clustering's outcomes are returned to the frontend as an interactive map and static graph. This map lets the viewer access information about the airports including traffic data, flight information, cluster details. This real-time reaction guarantees that the user may investigate significant cluster catered to their particular choices.

This implementation guarantees scalability and adaptability, therefore enabling the inclusion of other clustering techniques or future variants without major impact on the process. With precisely specified procedures for data pretreatment, clustering, evaluation, and visualisation, the modular design guarantees that the project can effectively manage challenging datasets and produce interesting results.

For U.S. flight data, this combination of Flask for backend management, PCA for dimensionality reduction, clustering algorithms for pattern detection, Folium for visualisation produces an integrated system that yields strong and interpretable clustering insight.

## 4.2 Dimensionality Reduction

### 4.2.1 Necessity of PCA in the Project

Principal Component Analysis (PCA) is quite important in this work for high-dimensional data simplification. Features including passenger count, flight distances, and geographic coordinates abound in the U.S flight dataset. The high dimensionality and correlations among these features provide a difficult since they can hide patterns in spectral clustering and other clustering methods. Addressing these problems PCA keeps the most significant data variations while cutting the dataset to less space.

PCA preserves the largest amount of variation and helps convert the data into a lower-dimensional space, it is especially appropriate for this work. This guarantees that important data patterns stay unaltered, hence improving the dataset's management and the following clustering performance. Clustering methods may find it difficult to separate less relevant or associated information without PCA, therefore producing less than ideal clusters. Moreover, PCA is a perfect technique for dimensionality reduction in complicated datasets such as flight data since it lets us more successfully detect non-linear features in the data.

PCA's mathematical foundation consist in multiple phases, code can be found in Appendix E.3:

- Centering the data: This ensures zero as the mean of every feature present
- Eigen decomposition: Divides the covariance matrix into eigenvectors and eigenvalues in turn. While eigenvalues gauge the degree of volatility in certain directions, the eigenvectors show the directions of maximum variance.
- Main Component selection: PCA then chooses the top eigenvector that has best capture variance, therefore transforming the data into a lower-dimensional space while preserving important information.

Referencing on the mathematical procedure described by Bishop in [13], Probabilistic PCA:

Let  $X \in \mathbf{R}^{n \times d}$ , where  $n$  is the number of data points and  $d$  is the number of features.

The data is centered, and a covariance matrix is computed:

$$\text{Cov}(X_{\text{centered}}) = \frac{1}{n-1} X_{\text{centered}}^T X_{\text{centered}}$$

Eigen decomposition follows:

$$C = Q\lambda Q^T$$

where  $\lambda$  contains eigenvalues and  $Q$  contains the corresponding eigenvectors.

The data is then project into a lower-dimensional space using the top eigenvectors:

$$X_{\text{textPCA}} = X_{\text{centered}} Q_2$$

This method helps us to efficiently manage the dataset while maintaining the most pretinent data for clustering study. In this next section 4.2.2 tried to explain PCA variance ration by taking small chuck of dataset.

#### 4.2.2 Conduct PCA

In this project, I showed the value of PCA in lowering dataset dimensionality by running a simple test using starting with standardising the data to have a mean of 0 and a variance of 1 for all pertinent variables, including passenger counts, flights, seats, and geographic coordinates. This allowed PCA to be sensitive to the data's scale.

Using eigen decomposition on the covariance matrix to find the eigenvalues and eigenvectors, therefore defining the vital direction of variance. Based on eigenvalues, top two components are chosen to maximise variance in a 2D space under constraint. This simplification let exclude less significant variance and concentrate on the most relevant trends in the data.

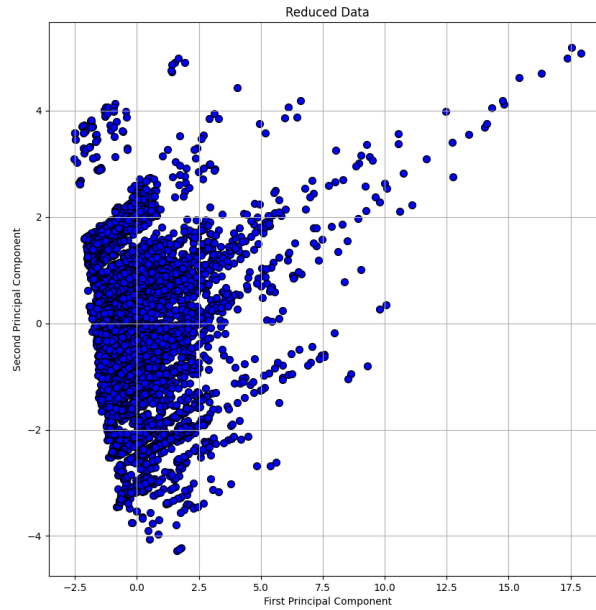


Figure 4.2: PCA Test for explaining Variance Ratio

By computing the explained variance ratio, which revealed that the first two principal components accounted for 55 percent of the overall variation in the data, can be found from the Appendix E.4. The graph produced by this PCA test shown above 4.2 displays the lowered data, there illustrating the dimensionality reduction attained. While still keeping important information from the original dataset, by streamlining this dataset which allowed manageable clustering and analysis.

With the Laplacian matrices and similarity graphs playing a part in the next steps, the processes were vital in preparing the dataset for further investigation, especially for clustering, since they ensure that the complexity of the data was reduced without losing vital information.

## 4.3 Algorithm Variations

As explained earlier in 3.1.3 about how I used same customised structure for each clustering algorithm can be found in Appendix B, only step varies was the parameter used for each clustering methods. In this section I investigated several techniques to find the best way to group airport depending on traffic statistic. I go through every method, the selected parameters, and how the project made use of each below.

### 4.3.1 K-Means

When the data is spherical and well-separated, a common and straightforward clustering method called K-means performs. The following values applied.

- `n_cluster`: Indices the data's partitioning cluster count. Various cluster counts and the most efficient number of cluster for the data were investigated by varying this value.
- `random_state`: By controlling the random number generator, consistent results ensured throughout several runs.

When the clusters were somewhat homogeneous in size and structure, K-means was helpful.

### 4.3.2 DBScan

Strong against noise and outliers, DBScan is a potent clustering method able to identify groups of different forms and sizes. The following were the parameters used:

- `eps (epsilon)`: The highest distance two points could have to be regarded as neighbours. This setting directly affected the cluster formation density threshold.
- `Min_samples`: The least amount of data points needed to create a cluster, or dense region. This prevented overfitting by guaranteeing that smaller, less significant clusters were not generated, there helping to define the scale of the clusters.

### 4.3.3 Standardised

Especially in nonlinear data, standard spectral clustering is a solid technique for identifying well-separated clusters. Applied parameters were:

- `n_init` (25): Ensured better resilience to poor initialisation, therefore lowering the possibility therefore lowering the possibility of converging to less than ideal solution.
- `eigen_tol`(0.0): The convergence tolerance for the eigenvalue. Lower tolerance meant more precise eigenvalue computation, which is absolutely essential for the spectrum clustering procedure.
- `affinity`: Local links in the data were obtained by using `k` nearest neighbour.
- `eigen_solver`: ARPACK was selected for effective eigenvalue computation on big data, hence time and memory utilising was optimised.

#### 4.3.4 Normalised

Data point are normalised to unit norm, therefore enhancing cluster separation for datasets with different cluster densities or sizes. Guarantees in the new space clusters are more different.

- `n_init` (20): Kept a bit smaller than standard variant.
- `eigen_tol`(auto): Designed to set a mix between computing efficiency and precision, automatic tolerance was allowed.
- `affinity`:K nearest neighbour was used.
- `eigen_solver`: ARPACK helped lowering computing costs.

#### 4.3.5 Kernel

Converting higher dimensional space, captures non-linear correlation using Radial Basis Function.

- `rbf_kernel` (RBF): Computes a similarity matrix depending on data point distance using `rbf_kernel`. This change enabled the method to split to split non-linearly separable clusters.
- `gamma`: Managed the RBF spread, higher gamma meant that only really close points were deemed to be similar.
- `affinity` (precomputed): Already a define set of cluster used for this variant, since the data count was too large to visualise.
- `assign_labels` (kmeans):

#### 4.3.6 Polynomial

Similar to kernel, converts data into higher dimensional space, therefore allowing the algorithm to identify increasingly intricate, non-linear structure. Majorly helped to capture complex-linkage for this dataset.

- `degree` (3): Compute third degree polynomial and record more intricate linkages between data points.
- `coef0` (1): Constant offset, balanced the effect of linear and non-linear elements by scaling the inner product.
- `affinity` (precomputed): Already a define set of cluster used for this variant, since the data count was too large to visualise.

- `assign_labels` (discretize): This ensure that points in the same eigenvector subspace were regularly assigned to the same cluster.

More details can be found on above algorithm, how they worked in this project in Appendix F. Every method offered fresh analysis of the trends in the airport traffic data. By means of the flexibility to test several clustering techniques, I was able to choose the optimum methods depending on data features, therefore producing strong, significant clusters that faithfully reflected the underlying patterns in the dataset.

## 4.4 Cluster Validation

### 4.4.1 Dunn Index (A Dead End)

I first tried to validate the clustering with the Dunn Index following PCA application for dimensionality reduction. The indicator guages the smallest inter-cluster distance to the biggest intra-cluster distance ratio which is mentioned by [15] in their work. The main objective was to evaluate the cluster separability using this index. But in our instance, especially because the Dunn Index struggles with more complicated and non-linear data, the outcomes were less than expected. It seemed computationally costly and lacked obvious understanding for the quality of clustering in our dataset. Here is the snapshot taken in the prior stage of the development of the code snippet 4.3 for the trial Dunn Index.

```
def calculate_dunn_index(data, clusters):
    intra_dists = []
    inter_dists = []
    unique_clusters = set(clusters) - {-1} # Exclude noise points labeled as -1

    for cluster in unique_clusters:
        cluster_data = data[clusters == cluster]
        if len(cluster_data) > 1: # Only consider clusters with more than one point
            intra_dists.append(np.mean(pairwise_distances(cluster_data)))

    for i in unique_clusters:
        for j in unique_clusters:
            if i < j: # Ensure that we do not calculate distance twice
                inter_cluster_data_i = data[clusters == i]
                inter_cluster_data_j = data[clusters == j]
                if len(inter_cluster_data_i) > 0 and len(inter_cluster_data_j) > 0:
                    inter_dists.append(np.min(pairwise_distances(inter_cluster_data_i, inter_cluster_data_j)))

    if intra_dists and inter_dists: # Ensure there are valid distances to calculate
        return np.min(inter_dists) / np.max(intra_dists)
```

Figure 4.3: Trial attempt of Dunn Index

But in terms of cluster evaluation, this strategy produced no appreciable outcomes, later removed from main logic and changed to more trustworthy techniques, such as Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index which can be found in section 3.1.4.2 and explained how I used these in next section 4.4.2.

### 4.4.2 Optimise Cluster Validation

I changed to apply three well-known criteria since the Dunn Index produced conflicting findings:

- Silhouette Score: Gauges the cohesiveness of the clusters that is , the degree of similarity among the same cluster's data points.
- Calinski-Harabasz Index: Often referred to as the variance ration criteria, this measures cluster separation.

- Davies-Bouldin Index: Shows better separation between clusters only when cooresponds to a lower score.

```
#Determining the best cluster count, silhouette score, calinski harabasz score,
#davies bouldin score (helper function)
def evaluate_cluster(pca_result, labels, best_silhouette, best_calinski_harabasz,
                    best_davies_bouldin, best_clusters_count, cluster_count):
    #Getting the metrics
    silhouette = silhouette_score(pca_result, labels)
    calinski_harabasz = calinski_harabasz_score(pca_result, labels)
    davies_bouldin = davies_bouldin_score(pca_result, labels)

    #Using tuple to store the current metric and the best metric
    current_metric = (silhouette, calinski_harabasz, -davies_bouldin)
    best_metric = (best_silhouette, best_calinski_harabasz, -best_davies_bouldin)

    #Checking if the current metric is greater than the best metric, if it is then
    # we will update the best metric,
    #for daives bouldin score we are using negative value, as the lower the value
    # the better the clustering
    if current_metric > best_metric:
        best_silhouette = silhouette
        best_calinski_harabasz = calinski_harabasz
        best_davies_bouldin = davies_bouldin
        best_clusters_count = cluster_count

    return best_silhouette, best_calinski_harabasz, best_davies_bouldin, best_clusters_count
```

Figure 4.4: Usage of Tuple for optimising cluster evaluation code on Visual Studio Code

As I mentioned in 3.1.4.1 the structure of evaluation function, which manages the scoring procedure during the cluster iteration. These three metrics gave more consistent and understandable findings. I used tuples as shown in the above Figure 4.4 to save the best and current measurement at every iteration, hence optimising our procedure. I effectively changed the best scores and cluster counts by matching the current findings to the best so far. The tuple format allowed  $O(1)$  complexity for updates, therefore enabling extremely space and time efficient processes.

## 4.5 Analysis of Clustering

Upon determining the optimal cluster count post cluster validating, definitive cluster labels were allocated to each data point. The labels were subsequently reintegrated into the dataset as `cluster_tags`. Post-clustering analysis entailed computing the mean values for diverse characteristic (including passenger count and distance) within each cluster. This facilitated a more profound comprehension of the distinctions between clusters, including the identification of short-haul versus long-haul flight clusters can refer 5.1. As shown in the below figure displays output log of Cluster mean table is created on Visual Studio Code console, also you can refer last step of pseudo code mentioned in Appendix B.

```
127.0.0.1 - - [09/Aug/2024 12:49:21] "GET /static/images/VIM.gif HTTP/1.1" 304 -
Best Silhouette Score: 0.6277843814966138 Best Calinski Harabasz: 1645.815212642432 Best Davies Bouldin: 0.50382168985/4935 Best Clusters Count: 5
Cluster_Tags    Passengers    Seats    Flights    Distance    Origin_population    Org_airport_lat    Org_airport_long    Dest_airport_lat    Dest_airport_long
0    1672.684564    3397.916107    30.651007    545.174497    2.954659e+06    38.997404    -118.727775    37.889073    -116.589288
1    2172.105263    3881.695489    27.657895    1800.048872    9.523022e+06    35.402450    -116.624961    37.988947    -105.240744
2    6229.109524    9690.628571    40.623810    2407.885714    1.484707e+07    30.876099    -129.774759    34.292236    -99.436818
3    11330.476812    19389.847026    141.615942    442.590830    4.457904e+06    36.408688    -122.237485    35.367176    -120.807330
4    382.722581    709.541935    3.361290    1977.987897    1.382797e+07    35.551850    -119.415992    39.197863    -81.860955
```

Figure 4.5: Final Clusters form post final clustering action

## 4.6 Folium Visualisation

To improve user interpretability, the clustered airport data was visualised using Folium, an advanced mapping tool enabling interactive geographic plotting. Every airport was shown as a node; aircraft paths between airports were shown as edges. To help visual distinction among clusters, the cluster labels were colour-coded can be found logic from Appendix E.1.

The Folium map object generates the map, using airports as nodes. Based on the cluster they fall into, each marker is given a colour: these markers offer comprehensive information about the airport including origin, destination, passenger, seat availability, flight distance.

As the code fragment illustrates in below Figure 4.6:

```
#Plotting folium map which can help to visualize the data
map_centre= [39.8283, -98.5795] #center of USA
m=folium.Map(location=map_centre, zoom_start=5, height=750, weight=1280 ) #creating a map
#adding title to the map
title='''<div style="position: fixed; width: 100%; height: 50px;font-size:24px; top:10px;left:50px; font-weight:bold; z-index:9999;
text-align: center;"> Normalized Clustering on USA Flights</div>'''
m.get_root().html.add_child(folium.Element(title))

#Introduction of markers to the map
for node in nodePositions: #iterating through the nodes
    label= G.nodes[node]['label'] #getting the label of the node
    pos = G.nodes[node]['pos'] #getting the position of the node
    node_info= dataset[dataset['Origin_airport']==node].iloc[0] #getting the information of the node
    #creating a text to display on the marker
    text=f'''<b>Cluster: {label}</b><br>
Origin: {node_info['Origin_airport']}<br>Destination: {node_info['Destination_airport']}<br>
Passengers: {node_info['Passengers']}<br>Seats: {node_info['Seats']}<br>
Flights: {node_info['Flights']}<br>Distance: {node_info['Distance']}'''
    #adding marker to the map
    folium.Marker(
        location=[pos[1],pos[0]],
        popup=text,
        icon=folium.Icon(color=get_color_map(label), icon='info-sign')).add_to(m) #color and icon of the marker

#returning the html of the map
map_html=m. repr_html ()
```

Figure 4.6: Logical presentation of Folium logic on Visual Studio Code

- Map Initialisation: The USA latitude and longitude is set. Starting the map with zoom, size, and style parameters, the `Folium.Map()` function.
- Marker Generation: I extract the cluster label, geographic coordinates, and specifics (such as passenger count and flight statistics) for every node that of an airport. These are included as marker with pop-up text that, upon clicking the marker, offers a detailed view.
- Cluster colour Mapping: Using the custom colour function E.1 which dynamically chooses a colour for every cluster. Users may easily distinguish between clusters by means of the colours.
- Map Rendering: Rendering the map into an HTML object(`map_html=m._repr_HTML()`), workflow can be found in 3.3 enables it to be shown straight in the user's browser as part of the web interface.

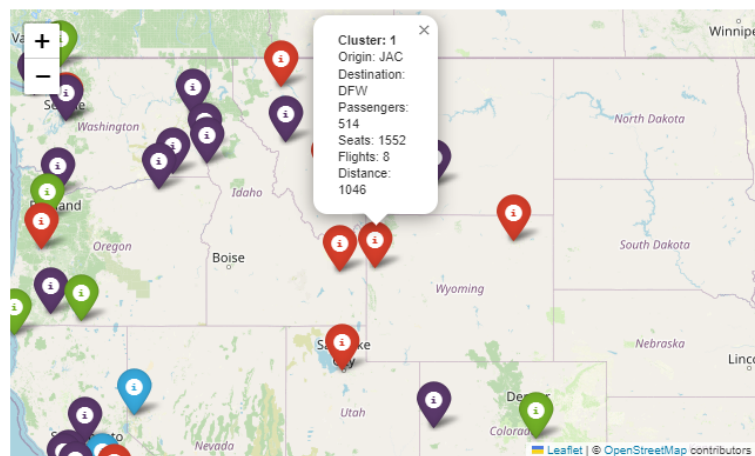


Figure 4.7: Folium Map marker pop-up displaying cluster information



As shown in above Figure 4.7, the graphic outcome shows concentrated nodes all throughout the USA, offering interesting geographic grouping of flight data. Clicking the marker allows user to see a comprehensive cluster information.

## Chapter 5

# Testing and Evaluation

### 5.1 Testing

In this part, I access the system with an eye towards a particular user case so as to ascertain its performance in an actual environment. Customised insights depending on the user's selected inputs for zone, season, year, system manages a user query, visualising the outcomes, offer insights.

#### 5.1.1 Test Case One

The user selected these inputs:

- Zone: South
- Season: Fall
- Calendar Year: 1993
- Algorithm: Kernel Spectral Cluster

##### 5.1.1.1 Graph Visualisation

Clicking 'Graph Generate' the system searches and filters the pertinent data, then uses PCA for dimensionality reduction and kernel spectral clustering to cluster the airport data. As shown in Figure 5.1 the map's developed clusters

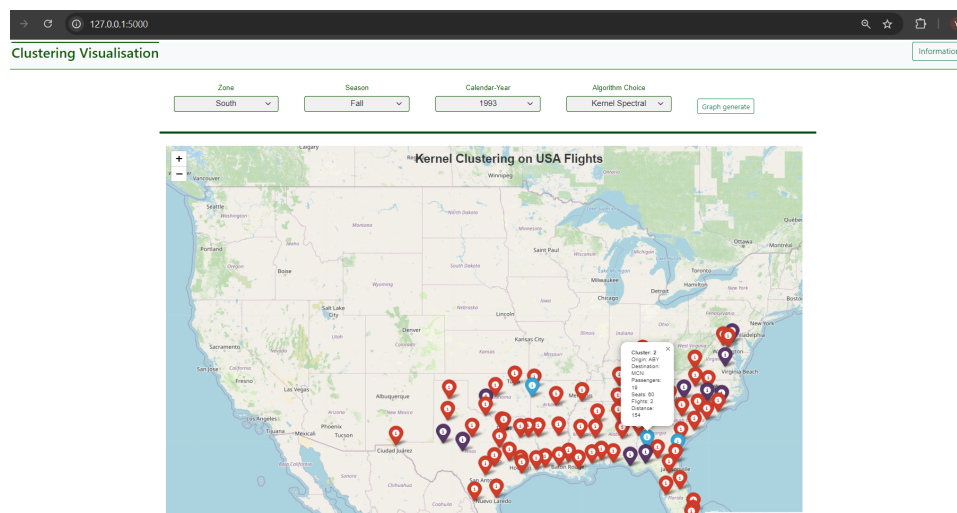


Figure 5.1: Graphical Visualisation of U.S. Flight Data based on clusters together with marks denoting several South Zone airports. These colour-coded marks give a rapid visual grasp of cluster grouping depending on the

cluster the airports fall under. Apart from this, I include two more graphs in Appendix E.5 clarifying the statistics underlying the fluctuation. in flight against passenger from year and non-dynamic version of above Folium map 5.1 map whereby airport data is classified based on clusters label created using a legend.

### 5.1.1.2 Output of the Console

The console logs expose that the data has been handled as follows 5.2:

```
South 1993 Fall kernel
fine_tune_model function called where data is loaded and cleaned
127.0.0.1 - - [10/Sep/2024 14:15:23] "GET /static/images/YlWk.gif HTTP/1.1" 304 -
Number of rows and columns while loading data: (3606803, 15)
Number of rows and columns after cleaning data: (3213418, 20)
filtering_data function called where data is filtered based on zone, calendaryear and season
number of rows and columns after filtering zone: (1218904, 20)
number of rows and columns after filtering calendaryear: (51469, 20)
number of rows and columns after filtering season: (13152, 20)
Number of rows and columns after filtering data: (13152, 20)
perform_pca function to reduce the dimensionality of the dataset
Number of rows and columns use for PCA: (13152, 9)
dataset_scaled (13152, 9)
After performing PCA reduced data (13152, 2)
Best Silhouette Score: 0.5363265185504291 Best Calinski Harabasz: 4745.920740660797 Best Davies Bouldin: 0.7223000239692285 Best Clusters Count: 4
```

Cluster_Tags	Passengers	Seats	Flights	Distance	Origin_population	Org_airport_lat	Org_airport_long	Dest_airport_lat	Dest_airport_long
0	5694.311667	10136.555000	85.043333	578.643333	5.314385e+06	31.438592	-95.110364	33.122723	-94.730138
1	891.084649	1688.170562	14.481996	303.267214	2.519969e+06	37.063220	-78.732339	39.445735	-77.639230
2	14165.511905	22311.357143	142.392857	385.916667	3.052248e+06	34.616133	-83.021751	35.273091	-81.637239
3	2562.707291	4724.267682	38.746464	600.776931	2.949915e+06	31.683517	-85.392123	35.651435	-87.333400

```
Execution time: 52.02958393096924
```

Figure 5.2: Console log from Visual Studio Code

- After cleaning, filtering by the south zone, the year 1993, and the fall season, the initial 3.6 million row dataset was dropped to 1.3 million. This demonstrates how the system uses choice to efficiently reduce the dataset.
- PCA reduced the dimensionality of the data from 9 characteristics to 2 principal components, therefore simplifying the data for clustering.
- Based on three metrics: Silhouette Score (0.5363), Calinski-Harabasz Index (4745.92), and Davies-Bouldin Index (0.7223); The algorithm assessed the clustering and found that 4 was the ideal number of clusters.

### 5.1.1.3 Insight on Clusters

- Cluster 0: Mostly shows airports managing a medium volume of passengers (average of 5694 people) and seats. Given the roughly 578km separating the origin from the destination airports in this cluster, they most certainly represent medium-haul flights. Larger population at the origin defines these airports.
- Cluster 1: With an average flight distance of 303km, airports here manage a lesser amount of passengers(891) and flights (14). These shorter-haul flights imply that local or regional routes could be being served from these airports.
- Cluster 2 (light blue): Comprising heavy traffic airports with an average of 14,165 passengers and 22,311 seats. This cluster is crowded with medium-haul flights since the average flight distance in it is 385km. These airports are also dispersed over bigger populations, which suggests more travel demand.
- Cluster 3: With an average travel distance of 600km, these airports deals a moderate number of passengers (2562) and flights (38). Covering more distance between origin and destination, these airports are also probably serving longer-haul services that clusters 1 and 2.

Based on user-selected parameters, this test case shows how the system can offer comprehensive, cluster-specific insights into air traffic patterns, therefore providing visual and statistical analysis to assist in either observations about travel patterns or informed decision-making. Appendix D.1 contain an additional test case.

## 5.2 Evaluation of Algorithms

Using many clustering techniques (K-Means, DBScan, Standard Spectral, Normalised, Spectral Kernel Spectral, and Polynomial Spectral), the outcomes for two scenarios: [West Zone, Winter, 1992] and [Northeast Zone, Summer, 2006] were examined to compare their clustering efficacy. Allow us to dissect the outcomes for every situation and discuss the algorithm performance:

### 5.2.1 Explanation for Table 5.1

Algorithm	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score	Cluster Count	Execution Time
K-Means	0.38	0.87	4480.0	3	31.1s
DBScan	0.45	0.75	1283.3	2	47.3s
Standard	0.29	0.81	2438.9	5	41.1s
Normalise	0.29	0.80	2483.9	5	40.0s
Kernel	0.59	0.67	2108.3	5	33.0s
Polynomial	0.76	0.42	5097.8	7	27.9s

Table 5.1: Evaluating Different Algorithms (Year:1992, Season:Winter, Region:West)

#### 5.2.1.1 Polynomial Spectral Clustering

- Score of Silhouette:0.76
- Davies-Bouldin Index:0.42 (Lower is preferable)
- Calinski-Harabasz Score: 5097.8 (better higher)

In this scenario, Polynomial Spectral clustering did the best. The low Davies-Bouldin index denotes minimal cluster overlap; the high silhouette score indicates that the clusters are nearly formed. The high Calinski-Harabasz score indicates that the variance between clusters is large relative to the variance inside clusters, so the clusters are compact and well separated.

#### 5.2.1.2 Kernel Spectral Clustering

- Score of Silhouette:0.59
- Davies-Bouldin Index:0.67
- Calinski-Harabasz Score: 2108.3

Though not as powerfully as Polynomial, Kernel Spectral clustering did really well. The Calinski-Harabasz Score is lower than Polynomial, suggesting that the clusters are not well separated; the Silhouette score is reasonable. Slightly more overlap between clusters is shown by the Davies-Bouldin index being somewhat higher than polynomial's.

### 5.2.1.3 DBScan

- Score of Silhouette:0.45
- Davies-Bouldin Index:0.75
- Calinski-Harabasz Score: 1283.3

With a low Silhouette score and high Davies-Bouldin Index, DBScan has modest performance suggesting some cluster overlap. The quite low Calinski-Harabasz score indicates that DBScan finds poorly defined clusters in this dataset difficultly.

### 5.2.1.4 Standardised and Normalised Spectral Clustering

- Silhouette Scores: 0.29
- Davies-Bouldin Index: 0.8 and 0.81
- Calinski-Harabasz score: 2438.9

With both indicate somewhat similar performance. These algorithm's modest Davies-Bouldin index and lower silhouette score suggest difficulties in constructing well-separated clusters. The clusters were not as well defined as those from Polynomial and Kernel Spectral clustering even if they could classify the data.

## 5.2.2 Explanation for Table 5.2

Algorithm	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score	Cluster Count	Execution Time
K-Means	0.48	0.84	5169.4	3	35.3s
DBScan	0.54	11.6	230.1	5	57.2s
Standard	0.25	0.83	2775.4	7	44.7s
Normalise	0.25	0.83	2775.1	7	44.8s
Kernel	0.51	0.67	1310.1	5	62.0s
Polynomial	0.53	0.53	2262.0	7	26.2s

Table 5.2: Evaluating Different Algorithms (Year:2006, Season:Summer, Region:Northeast)

### 5.2.2.1 Polynomial Spectral Clustering

- Score of Silhouette:0.53
- Davies-Bouldin Index:0.42
- Calinski-Harabasz Score:2262.0

Once more displaying excellent performance by Polynomial Spectral Clustering. The Davies-Bouldin Index similar to that in the west region, implying somewhat well-defined clusters; this silhouette score shows well formed clusters. Still, the technique kept a decent mix of compactness and cluster separation.

### 5.2.2.2 Kernel Spectral Clustering

- Score of Silhouette:0.51
- Davies-Bouldin Index:0.67
- Calinski-Harabasz Score: 1310.1

With a decent Silhouette score and Davies-Bouldin Index on par with polynomial, kernel did really nicely. On the other hand, the calinski-Harabasz score is on the lower end, suggesting that kernel clustering failed to identify in this dataset well separated clusters.

#### 5.2.2.3 DBScan

- Score of Silhouette:0.54
- Davies-Bouldin Index:11.6
- Calinski-Harabasz Score: 230.1

With a low Calinski-Harabasz score and a high Davies-Bouldin index, DBScan performed poorly in this instance and indicated that the clusters created by it lacked clear boundaries and had significant overlap.

#### 5.2.3 General conclusion

- Polynomial Spectral Clustering is the best approach for both circumstances since it regularly beats other methods by generating clearly defined with high Silhouette scores and low Davies-Bouldin Indexes.
- Strong potential exists in kernel spectral clustering, especially in non-linear datasets; but its performance varies with the region and season.
- Based on the high Davies-Bouldin scores and low Calinski-Harabasz scores, DBScan struggles with both test cases typically generating clusters with substantial overlap and poor separation.
- Though neither could match the efficacy of Polynomial or Kernel in distinguishing clusters and preserving cohesiveness, Standard and Normalised Spectral Clustering exhibit modest performance.

Applying these tests to U.S. airport traffic data across several areas and seasons helps one understand the relative strengths and shortcomings of certain clustering techniques.

## Chapter 6

# Project Ethics

This initiative guaranteed complete compliance with data privacy and protection rules by following the ethical guidelines set by the university. There were no human volunteers; all the data used comes from publicly available, anonymised, accessible sources. Particularly, the Shapefile from Natural Earth [3] and the "USA Airport Dataset" from Kaggle were used [9]; both of which are open access and copyright compliant. The initiative upholds all relevant ethical standards, including observance of the GDPR; the data comes under Category A (no Human data).

## Chapter 7

# Conclusion and Future Work

### 7.1 Conclusion

Ultimately, this work effectively showed how well spectral clustering analysed U.S domestic flight data from 1990 to 2009. I found significant trends in flight traffic, airport behaviour, and geographical clusters by using Principal Component Analysis (PCA) for dimensionality reduction and different clustering algorithms including K-means, DBScan, and spectral clustering variations.

After utilising spectral clustering turned out to be successful, outperforming conventional clustering methods in managing the high-dimensional, complicated dataset. For both technical and non-technical users, the web interface created throughout the project greatly improved the accessibility and interpretability of these clustering results, therefore enabling their comprehension.

By means of extensive testing and analysis, I found that Polynomial Spectral Clustering regularly offered the greatest results, thereby generating clearly defined, unique clusters. Although Kernel Spectral Clustering produced encouraging results as well, it struggled on some datasets. Though good for managing noise, the DBScan algorithm suffered with the density and complexity of the dataset.

The project's main objective were ultimately to find latent trends in flight data and provide interactive, user-friendly platforms for access to these discoveries.

### 7.2 Future Work

Although this project has effectively met its goals, there are various chances for expansion and enhancement even now:

- **Enhanced Scalability:** Expanding the system to manage global flight data or bigger datasets would help to validate the scalability and reliability of the utilised clustering techniques. This could call for including cloud based infrastructure or distributed computing methods.
- **Integration of Additional Data Sources:** Future research could involve combining other data sources such as weather data, economic issues, or airline delays to find more precisely the elements impacting flight patterns and airport traffic.
- **Real Time Analysis:** Using real-time data processing and clustering could provide stakeholders with a dynamic tool for tracking and re-



acting to changing flight trends, hence increasing the system’s applicability in practical, operational settings.

- **User Experience Enhancements:** Offering various clustering visualisation techniques or extensive reporting capabilities will help to improve the user experience for even more engagement and visual clarity, thereby making the platform more interesting for a larger audience.
- **Algorithm Optimisation:** Particularly for bigger datasets, further improving spectral clustering techniques to lower computational complexity and enhance efficiency would be a useful next step. Investigating hybrid approaches combining strengths of several clustering techniques could also provide more complex understanding.

## Chapter 8

# BCS Criteria and Self Reflection

The project reflects the British society (BCS) project criteria, therefore highlighting the following:

- Emphasising data pre-treatment, dimensionality reduction, and clustering analysis, this project shows great technical ability that is, practical and analytical skills. These techniques provide efficient handling of challenging datasets and preparation for clustering, hence facilitating insights on flight data trends.
- This project stresses the creation of an interactive interface for users incorporating several clustering techniques to visualise aviation traffic data, therefore promoting creativity and innovation. This new method lets users investigate several techniques and scenarios, therefore enabling easy and interesting access to difficult data analysis.
- Key design decisions including metric choice and algorithm design were informed by background study. For dimensionality reduction, for instance, PCA was selected because it has shown proven capacity to retain the most significant characteristic from higher dimensional data while minimising computing cost.
- Meeting a Real Need: Accessible flight traffic analysis is in more sought for. This project provides an easy to use interface for traffic congestion and travel pattern analysis thereby meeting that demand. Airlines, airports, and academics trying to maximise flight operations and control air traffic could all find use for this instrument.
- Self-Management was absolutely important for the project to be completed successfully. Managing this project called for ongoing iteration, careful planning, and time allocation. Starting with a thorough project schedule and dissecting the whole procedure into stages: data pre-processing, clustering method choice, implementation, testing, and lastly, interactive web interface building. Using an iterative approach guaranteed that every part of the project was completely operational before moving on to the next. As I faced unanticipated difficulties, this approach let me make small changes, fast mistakes corrections, and gradual improvements. Furthermore impacted by pragmatic factors including the scalability and adaptability of the project was the technology stack chosen: Python for data processing and clustering, Flask for the web framework, and Folium for visualisation. The development cycle included regular testing as well, so making sure that

additions or new features did not bring flaws or conflicts. Managing the complexity of the data also demonstrated good self-management, particularly in relation to big datasets including over 3.6 million records. Reducing data dimensionality with PCA and preprocessing guaranteed that my process stayed effective and orderly. Always updating targets and deadlines as per development. This enabled me to manage tasks including cluster evaluations and performance optimisation.

- **Self-Evaluation:** This project has given me a chance to grow and hone numerous technical and analytical abilities, both rewarding and demanding at the same time. My capacity to combine several machine learning approaches including spectral clustering into a practical use was among my strongest suit. By use of the sophisticated clustering techniques applied practically to U.S. domestic flight data, I was able to derive important insights applicable to aviation industry stakeholders. Moreover, the development of an easy to use interactive online interface shows the effective implementation of creative ideas to challenging issues. Still, the initiative also revealed a number of areas needing work. Optimising the clustering techniques to function well on such a vast dataset presented the first difficulty I encountered. Managing big amounts of data first caused performance problems, especially with regards to the visualisation tools and clustering assessments. While using PCA for dimensionality reduction helped solve these problems, I came to see that more sophisticated optimisation methods such as distributed computing or parallel processing could have greatly raised the scalability of the system. Managing technological depth against usability presented still another difficulty. Although the backend handled highly technical tasks including data preparation and clustering, the outputs were accessible to non-technical consumers depending on consideration effort. Looking back, the tool might have been more successful if the visualisations had been clearer and more thorough explanations for user without a background in data science had been included. At last, this initiative has underlined the need of introspection and flexibility in handling challenging assignments. I learnt to spot when a strategy was not producing the expected outcomes, as in the case of the Dunn Index for validation, and to turn fast to more appropriate substitutes. In next projects, this flexibility will be quite helpful.

# Bibliography

- [1] Tadeusz Caliński and Harabasz JA. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3:1–27, 01 1974.
- [2] David Davies and Don Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227, 05 1979.
- [3] Natural Earth. Natural earth shapefile, 2023. <https://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-1-states-provinces/>.
- [4] Leonard Kaufman and Peter Rousseeuw. *Finding Groups in Data: An Introduction To Cluster Analysis*. 01 1990.
- [5] Ulrike Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36, 05 2008.
- [6] Dmitrii Marin, Meng Tang, Ismail Ben Ayed, and Yuri Boykov. Kernel clustering: Density biases and solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 06 2017.
- [7] Andri Mirzal. Statistical analysis of clustering performances of nmf, spectral clustering, and k-means. In *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, pages 1–4, 2020.
- [8] K. M. Archana Patel and Prateek Thakral. The best clustering algorithms in data mining. *2016 International Conference on Communication and Signal Processing (ICCSP)*, pages 2042–2046, 2016.
- [9] Jacob Perkins. Usa airport dataset, 2021. <https://www.kaggle.com/datasets/flashgordon/usa-airport-dataset/data>.
- [10] Kristina P.S and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, PP:1–1, 04 2020.
- [11] J.C. Rojas-Thomas, M. Santos, and M. Mora. New internal index for clustering validation based on graphs. *Expert Systems with Applications*, 86:334–349, 2017.
- [12] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [13] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622, 1999.

- [14] Lijuan Wang, Shifei Ding, and Hongjie Jia. An improvement of spectral clustering via message passing and density sensitive similarity. *IEEE Access*, PP:1–1, 07 2019.
- [15] Chunmei Yang, Xuehong Zhao, Ning Li, and Yan Wang. Arguing the validation of dunn’s index in gene clustering. In *2009 2nd International Conference on Biomedical Engineering and Informatics*, pages 1–4, 2009.

# Appendix A

## Frontend Design Files

### A.1 HTML Structure

Drop-downs for choosing the zone, year, season and algorithm let users engage with the clustering tool and personalise their searches. In this appendix a quick review of the key elements such as drop-down menus are shown, this enables one to visualise the frontend's arrangement.

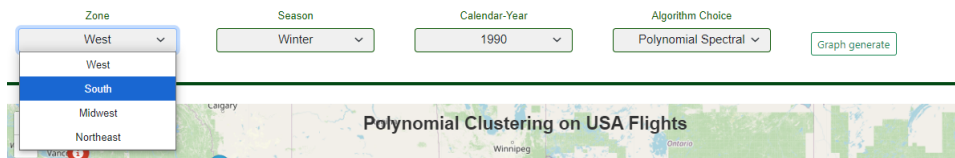


Figure A.1: Actual Frontend Drop-down to choose

### A.2 CSS styling

Buttons are styled for consistency across different browsers using Bootstrap's classes. Enhanced for responsiveness using media queries for user friendly site.

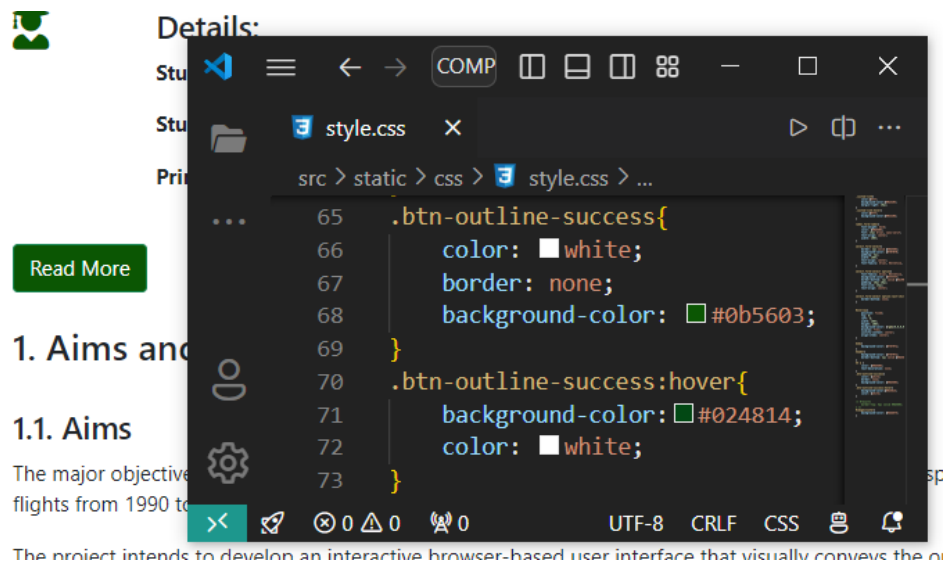
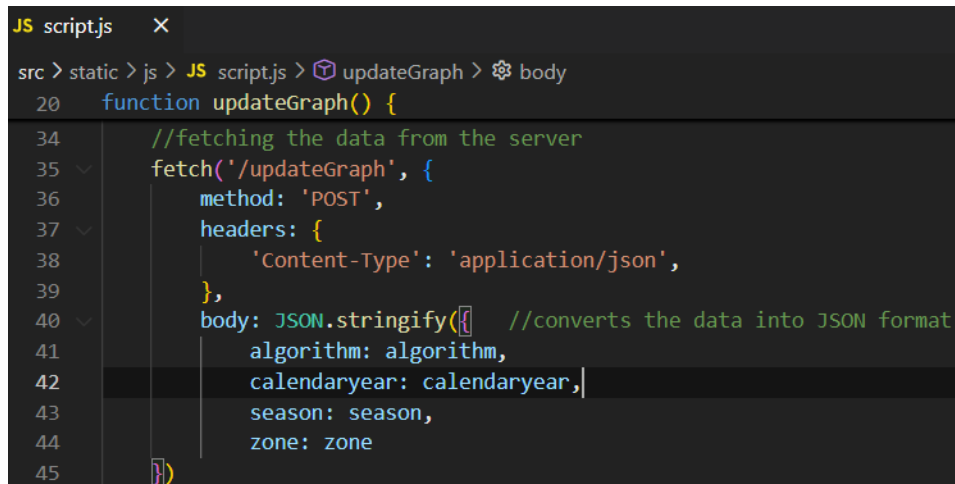


Figure A.2: Read more Button and its styling code snippet

## A.3 JavaScript Interaction and AJAX

Showcasing data flow between the client and backend, example of the main purpose of JSON handling shown in below Figure

A screenshot of a code editor window titled "JS script.js". The breadcrumb path at the top is "src > static > js > JS script.js > updateGraph > body". The code defines a function "updateGraph()" starting at line 20. Inside the function, there is a comment "//fetching the data from the server" at line 34. Line 35 shows a "fetch('/updateGraph', {" call. Line 36 shows "method: 'POST'," as a property. Line 37 shows "headers: {" as a property. Line 38 shows "'Content-Type': 'application/json'," as a property. Line 39 shows a closing brace "},". Line 40 shows "body: JSON.stringify({" as a property, followed by a comment "//converts the data into JSON format" on the same line. Line 41 shows "algorithm: algorithm," as a property. Line 42 shows "calendaryear: calendaryear," as a property. Line 43 shows "season: season," as a property. Line 44 shows "zone: zone" as a property. Line 45 shows the closing brace for the object and the closing parenthesis for the fetch call: "});".

```
JS script.js  X
src > static > js > JS script.js > updateGraph > body
20  function updateGraph() {
34      //fetching the data from the server
35      fetch('/updateGraph', {
36          method: 'POST',
37          headers: {
38              'Content-Type': 'application/json',
39          },
40          body: JSON.stringify({ //converts the data into JSON format
41              algorithm: algorithm,
42              calendaryear: calendaryear,
43              season: season,
44              zone: zone
45          })
}
```

Figure A.3: Actual code from project which fetch and converts request in JSON form

## Appendix B

# Supporting Logic behind Clustering Mechanism

---

**Algorithm 1:** Common Skeleton for Clustering Methods 3.1.3

---

**Input:** Data to be clustered

**Output:** Final cluster labels and summary statistics

**Initialize Best Score;**

$best\_silhouette \leftarrow -\infty;$

$best\_calinski\_harasz \leftarrow -\infty;$

$best\_davies\_boudlin \leftarrow \infty;$

$best\_cluster\_count \leftarrow 1;$

**for**  $cluster\_count \leftarrow 3$  **to** 7 **do**

    Apply chosen clustering algorithm (K means, Standardised, etc.);

    Generate cluster labels using the algorithm;

**Evaluate clusters based on metrics;**

$silhouette\_score \leftarrow calculate\_silhouette(labels);$

$calinski\_harasz \leftarrow calculate\_calinski(labels);$

$davies\_boudlin \leftarrow calculate\_davies(labels);$

**if** *new scores are better* **then**

        Update  $best\_silhouette$ ,  $best\_calinski\_harasz$ ,  
         $best\_davies\_boudlin$ ;

        Update  $best\_cluster\_count$ ;

**Final Model Selection;**

    Apply the chosen algorithm (K means, Standardised, etc.) with the  
    best chosen  $best\_cluster\_count$ ;

    Assign final cluster labels to the dataset;

    Return final cluster labels and summary statistics;

---

The pseudo code which build covers the iterative method for finding the appropriate number of clusters depending on the specified metrics. This structure offers scalability across several clustering methods as K means, DBScan and spectral clustering variants.

For every clustering approach (K means, Normalized, Polynomial etc.), the underlying logic remains the same guaranteeing the modularity of the code structure



# Appendix C

## Environment Setup

Use these guidelines to guarantee appropriate project configuration and implementation:

### C.1 Install Python and pip

Ensure Python 3.x is installed. You may confirm the installation by:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\Aanand\Desktop\COMP702> python --version
Python 3.12.2
PS C:\Users\Aanand\Desktop\COMP702> pip --version
pip 24.0 from C:\Program Files\Python312\Lib\site-packages\pip (python 3.12)
PS C:\Users\Aanand\Desktop\COMP702>
```

### C.2 Install Required Libraries

Install Flask and other required libraries running the following command:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\Aanand\Desktop\COMP702> pip install flask pandas scikit-learn folium matplotlib
```

### C.3 Operating the Flask Program

The project operates on a local server. Go to the project directory and execute to initiate the server:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\Aanand\Desktop\COMP702> cd src
PS C:\Users\Aanand\Desktop\COMP702> python .\main.py
* Serving Flask app 'main'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 129-880-898
127.0.0.1 - - [14/Sep/2024 05:39:42] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Sep/2024 05:39:42] "GET /static/images/img_1.jpg HTTP/1.1" 304 -
127.0.0.1 - - [14/Sep/2024 05:39:42] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [14/Sep/2024 05:39:42] "GET /static/images/img_2.jpg HTTP/1.1" 304 -
127.0.0.1 - - [14/Sep/2024 05:39:42] "GET /static/images/default.jpg HTTP/1.1" 304 -
127.0.0.1 - - [14/Sep/2024 05:39:42] "GET /static/js/app.js HTTP/1.1" 304 -
```

This configuration lets the clustering visualisation tool be developed locally with Flask's lightweight server. Homepage 3.1.7.1 will load on respective users browser.

# Appendix D

## Experimental results

Additional Test case performed on the given dataset to explain more on clustering analysis.

### D.1 Additional Test Case

The user selected these inputs:

- Zone: South
- Season: Winter
- Calendar Year: 1990
- Algorithm: Polynomial Spectral Cluster

#### D.1.1 Graph Visualisation

After choosing "Graph Generate", the system searches and filters the pertinent information, does PCA for dimensional reduction, then groups the airport data using the normalised spectral clustering technique. The clus-

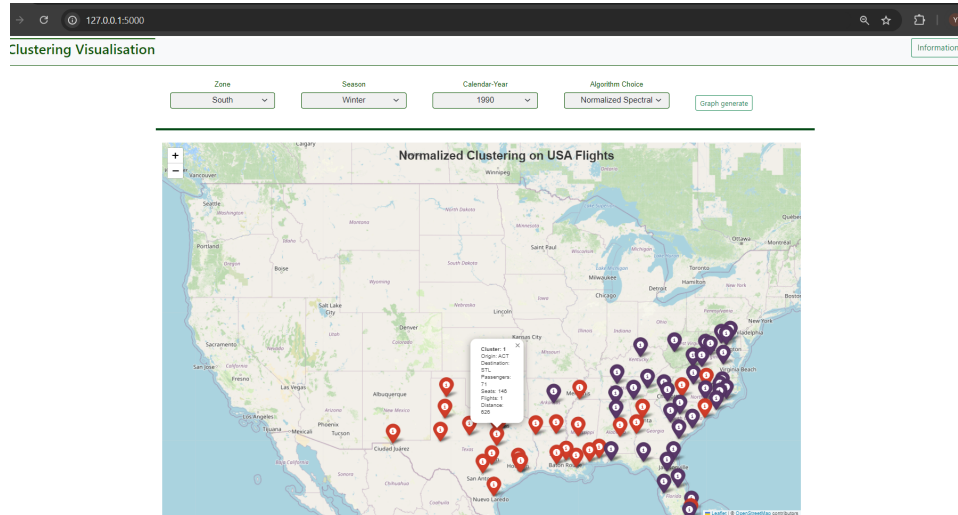


Figure D.1: Graphical Visualisation of U.S. Flight Data based on clusters

ters on the Map are shown in the Figure D.1 above together with marks denoting many Southern airports. These colour-coded markers show airport associations clearly depending on their respective clusters. Apart from this, I include two more graphs in Appendix E.6 clarifying the statistics

underlying the fluctuation. in flight against passenger from year and non-dynamic version of above Folium map D.1 map whereby airport data is classified based on clusters label created using a legend.

### D.1.2 Output of the Console

The console logs exhibit the stages of data processing are as follows D.2:

```

south 1990 winter normalized
fine_tune_model function called where data is loaded and cleaned
Number of rows and columns while loading data: (3606803, 15)
Number of rows and columns after cleaning data: (3213418, 20)
filtering_data function called where data is filtered based on zone, calendaryear and season
number of rows and columns after filtering zone: (1218904, 20)
number of rows and columns after filtering calendaryear: (54090, 20)
number of rows and columns after filtering season: (13825, 20)
Number of rows and columns after filtering data: (13825, 20)
perform_pca function to reduce the dimensionality of the dataset
Number of rows and columns use for PCA: (13825, 9)
dataSet_Scaled (13825, 9)
After performing PCA reduced data (13825, 2)
Best Silhouette Score: 0.28298993588696896 Best Calinski Harabasz: 2696.6955463338013 Best Davies Bouldin: 0.9619219454447001 Best Clusters Count: 3
Cluster_Tags    Passengers    Seats    Flights    Distance    Origin_population    Org_airport_lat    Org_airport_long    Dest_airport_lat    Dest_airport_long
0    1070.696624    2388.405510    20.540064    414.111204    2.536414e+06    34.822322    -81.234947    37.110755    -80.666709
1    3421.656612    6202.163421    44.367316    691.359679    3.690706e+06    31.483833    -89.649387    34.924084    -91.088211

```

Figure D.2: Console log from Visual Studio Code

- Following cleaning, the initial 3.6 million row dataset dropped to 1.3 million rows.
- With the help of PCA clustering process get simplify by reducing the feature just by using only 2 essential components.
- Based on three metrics: Silhouette Score (0.283), Calinski-Harabasz Index (2696.70), and Davies-Bouldin Index (0.9619); The algorithm assessed the clustering and found that 2 was the ideal number of clusters not well separated than earlier Test case 5.1.1 using Kernel Spectral.

### D.1.3 Insight on Clusters

- Cluster 0: With an average of 1070 passengers and 20 flights a node, this cluster's airport manage modest passenger volume. Given the average flight distance of 414 km, these are mostly short to medium haul paths. These flights go to reasonably populous regional location.
- Cluster 1 (red): With average passenger count of 3421 and 44 flights, this cluster has airports with more traffic. These mostly medium to long haul flights since the flight distance in this cluster average 691 km. These airports serve greater, more densely populated areas with increasing travel demand.

All things considered, this test case demonstrates how the system uses user inputs to customise data to offer significant analysis of air traffic trends over several areas, seasons, and years. Based on traffic data, the normalised spectral clustering technique arranged airports in not distinguished way comparing to Test case 5.1.1.

# Appendix E

## Supporting Snapshots

### E.1 Customised Library for filling colours

This graphic shows the custom colour filling tool used to colour several cluster category in folium map.

```
#Function to get color map
def get_color_map(label):
    color_map = {}
    0: 'darkpurple',
    1: 'indigo',
    2: 'blue',
    3: 'cyan',
    4: 'green',
    5: 'yellowgreen',
    6: 'yellow',
    7: 'lightyellow',
    return color_map.get(label, 'gray')
```

Figure E.1: Customised colour Code Map

### E.2 Customised library for plotting

This graphic shows the custom plotting tool used to call several project cluster methods.

```
main.py > -
#A dictionary to call specific plotting function of algorithms
PLOT_DICTIONARY = {
    'kmeans': plot_kmeans, 'dbscan': plot_dbscan, 'standard': plot_standard,
    'normalized': plot_normalized, 'kernel': plot_kernel, 'polynomial': plot_polynomial,
}
```

Figure E.2: Customised Plotting Library

### E.3 PCA general steps

Here this figure is just showing the basic mandatory steps needed for pca to perform.

```
#Determining reduce data with pca (helper function)
def perform_pca(dataset, columns, n_comps=2, rand=42):
    print('perform pca function to reduce the dimensionality of the dataset')
    #getting the clean dataset by using the columns
    clean_dataset = dataset[columns]
    print(f'Number of rows and columns use for PCA:', clean_dataset.shape)
    #Standardizing the dataset
    scaler = StandardScaler()
    #Fitting the scaler to the clean dataset
    dataset_Scaled = scaler.fit_transform(clean_dataset)
    print(f'dataset_Scaled', dataset_Scaled.shape)
    #Reducing the dimensionality of the dataset to 2 and
    #using the random state of 42 for reproducibility
    pca = PCA(n_components=n_comps, random_state=rand)
    #Fitting the PCA to the scaled dataset
    reduce = pca.fit_transform(dataset_Scaled)
```

Figure E.3: PCA general logic

## E.4 PCA testing

This snapshot shows the console output of visual studio code PCA variance ratio run. it follows the main stages in dimensionality reduction together with the related PCA outcomes

```
Number of rows and columns after filtering data: (6866, 20)
Number of rows and columns use for PCA: (6866, 9)
PCA explained variance ratio: [0.32187237 0.2046858 ]
Reduced data info: (6866, 2)
After performing PCA reduced data (6866, 2)
```

Figure E.4: Console log of the tested PCA variance ratio run on Visual Studio Code

## E.5 Non-Static Graph for Test case 1

Both below graphs are additional data analysed for respective test cases.

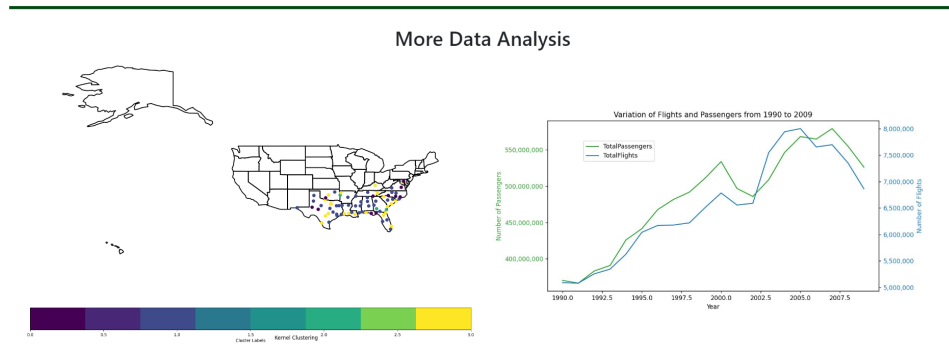


Figure E.5: Non-Static version of Folium and Flight vs Passenger variation

## E.6 Non-Static Graph for Additional test case

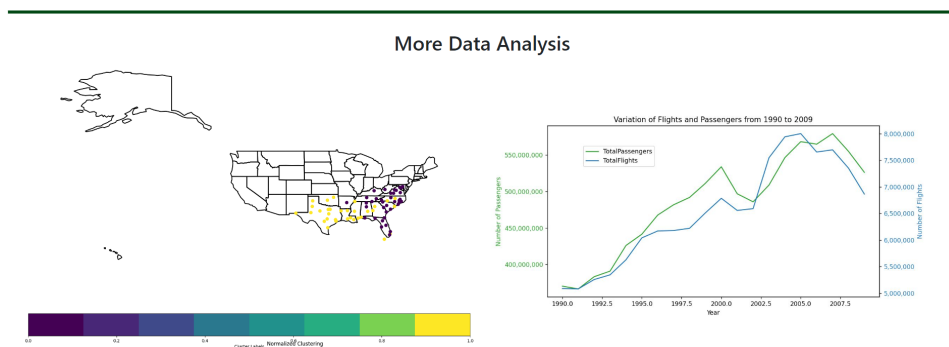


Figure E.6: Non-Static version of Folium and Flight vs Passenger variation

# Appendix F

## Choice of Algorithm

Here you can find extra detail of working steps of each clustering algorithm of the project, mostly all are having in build libraries.

### F.1 K-Means working

K-Means starts with choosing  $k$  random centroids and iteratively assigns data points to the closest centroid, therefore reevaluating the centroids depending on present assignments. This iterative process keeps on until the centroids steady or the improvement is negligible.

This project reduces the data using PCA then feeds it to K-means method. This guarantees simplicity of high-dimensional data, therefore enhancing the clustering outcomes.

### F.2 DBScan working

Defining a neighbourhood based on two criteria epsilon, the greatest distance between two points to be considered neighbour and min-samples, the minimum number of points needed to build a dense region allows DBScan to operate. DBScan grows cluster from these dense areas; points within these dense areas comprise the core of the cluster. DBScan groups point according on density using PCA reduced data, therefore guaranteeing the successful identification of clusters with different densities and forms.

### F.3 Standardised working

Designed to be successful for well-separated data, standardised spectral clustering it uses graph theory to build a similarity matrix capturing the connections between data points, then uses an Eigen decomposition to expose the main data dimension. Because Eigen decomposition is required especially for big datasets this spectral clustering is computationally expensive. It is successful, nevertheless, in capturing non-linear correlations that standardised techniques would overlook.

### F.4 Normalised working

This variation brings the data points into line to increase cluster separation. For datasets with varied sized and densities in particular, it is quite successful. Through better handling of data with varying densities, normalised spectral clustering enhances on the standard approach.

## F.5 Kernel working

For non-linear datasets, kernel spectral clustering is quite successful; nonetheless the performance of the method depends much on the kernel and its parameters (such as gamma).

## F.6 Polynomial working

A similarity matrix is generated using polynomial kernel, therefore converting the data into a higher-dimensional space. I build a Laplacian matrix and Eigen decomposition is used to lower the data dimensionality. Clustering is done in this modified environment where data point correlations are more readily found.