Project Report

Clustering Analysis of Air Traffic Passenger Statistics

Ayyub Orujzade

University of Warsaw
Faculty of Economic Sciences

January 2024

Introduction

Finding and analyzing patterns in data is essential for producing actionable insights in the fields of data science and business analytics. Unsupervised learning techniques, particularly clustering, are essential for identifying potential internal structures and relationships in datasets.

Clustering is the process of assembling related observations into groups according to shared traits, which uncovers latent patterns and advances a deeper comprehension of the underlying data. Beyond its technical uses, clustering is important because it has real consequences for business strategy decision-making. Clustering makes it easier to comprehend consumer behavior at a deeper level by defining groups of related entities. This in turn gives businesses the ability to customize strategies according to the particular requirements and preferences of particular customer segments, ranging from focused marketing campaigns to optimal inventory management.

We are looking into the **"Air Traffic Passenger Statistics"** dataset as part of this project, which includes data on the operating airline, geographical areas, passenger counts, and other pertinent aspects of air travel. Some important features to cluster include "Operating Airline," "GEO Region," and "Passenger Count," among others. The dataset was preprocessed before analysis to fix possible problems like missing values and adjust the data so that clustering algorithms could use it.

As we explore the complexities of the "Air Traffic Passenger Statistics," we must choose an appropriate clustering algorithm, look over the generated clusters, and evaluate their quality. This project aims to not only apply unsupervised learning methods but also to interpret the implications of identified clusters.

Dataset

The **Air Traffic Passenger Statistics** dataset is a comprehensive collection of information about air travel activities. It includes several dimensions, giving a comprehensive picture of the airline industry's performance.

The dataset consists of the following key columns:

- **Activity Period**: The time period during which the activity occurred.
- **Operating Airline**: The airline performing the flight operation.
- **Operating Airline IATA Code**: IATA code representing the operating airline.
- **Published Airline**: The airline as published in reports.
- **Published Airline IATA Code**: IATA code for the published airline.
- **GEO Summary:** A summary of the geographic region.
- **GEO Region:** The specific geographic region.

- **Activity Type Code:** Code representing the type of activity.
- **Price Category Code:** Code indicating the price category.
- **Terminal:** The airport terminal associated with the activity.
- **Boarding Area:** The boarding area within the terminal.
- **Passenger Count:** The number of passengers involved in the activity.
- **Adjusted Activity Type Code:** Adjusted code for the activity type.
- **Adjusted Passenger Count:** Adjusted passenger count.
- **Year:** The year in which the activity occurred.
- **Month:** The month in which the activity occurred.

Clustering

There are different types of clustering algorithms, such as those based on connectivity, centroids, density, distribution, and so on, each of which has advantages and disadvantages and is suitable for different purposes and use cases.

In this project, we mainly focus on using two of the clustering techniques: hierarchical clustering (connectivity-based) and k-means (centroid-based). These two unsupervised Machine Learning techniques are both based on proximity (using distance measures). They are simple but very effective and powerful in many clustering tasks.

I started with collecting and analyzing the data.

```
import pandas as pd # Tabular data manipulation library

# Loading data
df = pd.read_csv('C:/Users/orucz/OneDrive/Рабочий стол/UW materials/python/
Air Traffic Passenger Statistics.csv')
print(df.head()) # only first 5 rows will appear
```

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code	Adjusted Passenger Count	Year	Month
0	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271	Deplaned	27271	2005	July
1	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131	Enplaned	29131	2005	July
2	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415	Thru / Transit * 2	10830	2005	July
3	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156	Deplaned	35156	2005	July
4	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090	Enplaned	34090	2005	July

Filtering

The raw data consists of 16 columns in total. To make analysis and demonstration easier, I filtered the clustering analysis.

```
df_filtered = df[(df['Year'] > 2007) &
                 (df['Month'] == 'August') &
                 (df['Passenger Count'] > 20000)]

df_filtered.head()
```

Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code	Adjusted Passenger Count	Year	Month
200808	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 3	E	35117	Deplaned	35117	2008	August
200808	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 3	E	35185	Enplaned	35185	2008	August
200808	AirTran Airways	FL	AirTran Airways	FL	Domestic	US	Deplaned	Low Fare	Terminal 1	B	20860	Deplaned	20860	2008	August
200808	AirTran Airways	FL	AirTran Airways	FL	Domestic	US	Enplaned	Low Fare	Terminal 1	B	21908	Enplaned	21908	2008	August
200808	Alaska Airlines	AS	Alaska Airlines	AS	Domestic	US	Deplaned	Other	Terminal 1	B	48397	Deplaned	48397	2008	August

```
df_filtered.info()# method used to display concise information about a
DataFrame in pandas
```

```
Index: 306 entries, 4320 to 14137
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Activity Period               306 non-null    int64
 1   Operating Airline             306 non-null    object
 2   Operating Airline IATA Code   306 non-null    object
 3   Published Airline             306 non-null    object
 4   Published Airline IATA Code   306 non-null    object
 5   GEO Summary                   306 non-null    object
 6   GEO Region                    306 non-null    object
 7   Activity Type Code            306 non-null    object
 8   Price Category Code           306 non-null    object
 9   Terminal                      306 non-null    object
 10  Boarding Area                 306 non-null    object
 11  Passenger Count               306 non-null    int64
 12  Adjusted Activity Type Code   306 non-null    object
 13  Adjusted Passenger Count      306 non-null    int64
 14  Year                          306 non-null    int64
 15  Month                         306 non-null    object
dtypes: int64(4), object(12)
```

Using the code below, we keep only the fields of interest in our data frame.

```
fields = [
    'Operating Airline',
    'GEO Region',
    'Year',
    'Month',
    'Passenger Count',
    'Activity Type Code',
    'Price Category Code'
]

df = df_filtered[fields].set_index('Operating Airline')
df.head()
```

Operating Airline	GEO Region	Year	Month	Passenger Count	Activity Type Code	Price Category Code
Air Canada	Canada	2008	August	35117	Deplaned	Other
Air Canada	Canada	2008	August	35185	Enplaned	Other
AirTran Airways	US	2008	August	20860	Deplaned	Low Fare
AirTran Airways	US	2008	August	21908	Enplaned	Low Fare
Alaska Airlines	US	2008	August	48397	Deplaned	Other

Now we have 6 columns in total.

Preprocessing

Before delving into clustering analysis, it is crucial to conduct exploratory data analysis (EDA) and preprocess the data as needed. This may involve addressing missing values, scaling numerical features, or encoding categorical variables, ensuring a clean and standardized dataset for robust clustering outcomes.

```
print(df.describe())# method in pandas is used to generate descriptive
statistics of a DataFrame
```

	Year	Passenger Count
count	306.000000	306.000000
mean	2011.614379	90581.630719
std	2.272066	99482.511448
min	2008.000000	20010.000000
25%	2010.000000	24798.750000
50%	2012.000000	52440.000000
75%	2014.000000	131464.000000
max	2015.000000	659837.000000

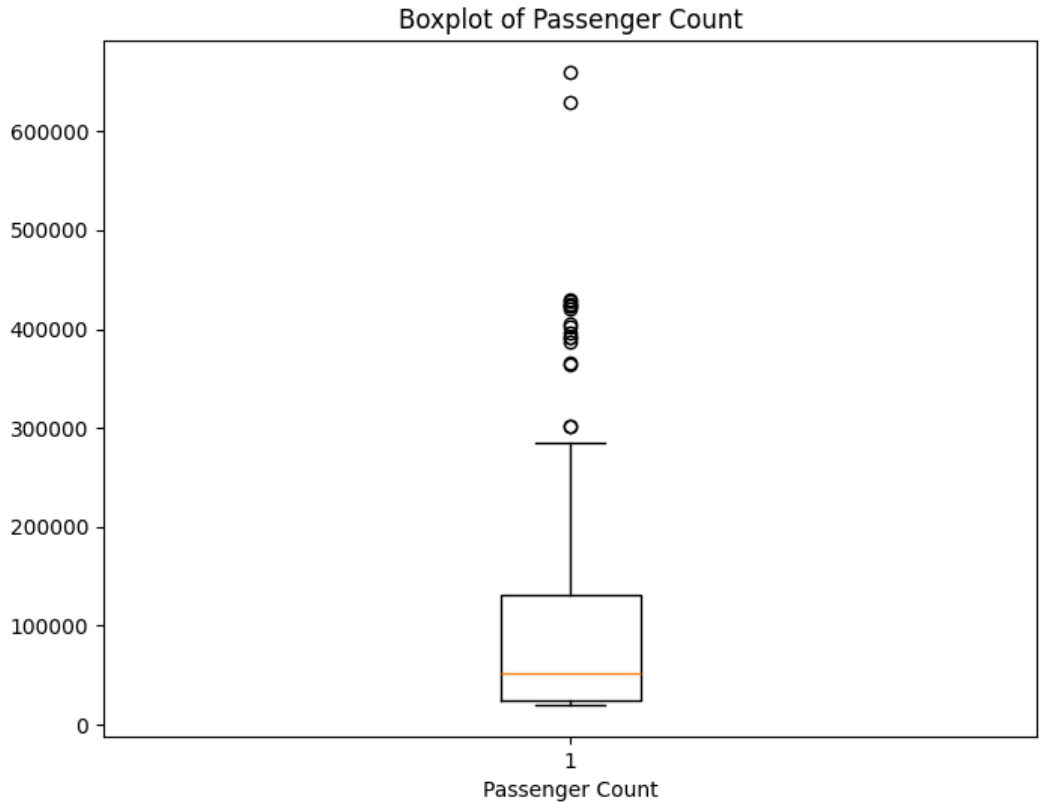
From the statistics reported, it appears that the data do not have major anomalies and no missing values for all columns. Otherwise, it seems that the feature **“Passenger Count”** may have potentially large outliers, while the rest of the features seem to be in a fairly reasonable range.

Treating outliers

I used boxplot to visualize the "Passenger Count" distribution. This will enable us to more fully comprehend the distribution of this numerical feature and help us spot any possible outliers. The ***matplotlib*** library will be used for the visualization:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.boxplot(df['Passenger Count'])
plt.title('Boxplot of Passenger Count')
plt.xlabel('Passenger Count')
plt.show()
```

Before treating



I decided to remove outliers from the **"Passenger Count"** column, and used a criterion such as the **Z-score** to identify and remove the extreme values.

This code calculates Z-scores for the **"Passenger Count"** column and removes the rows where the absolute Z-score is greater than 3.

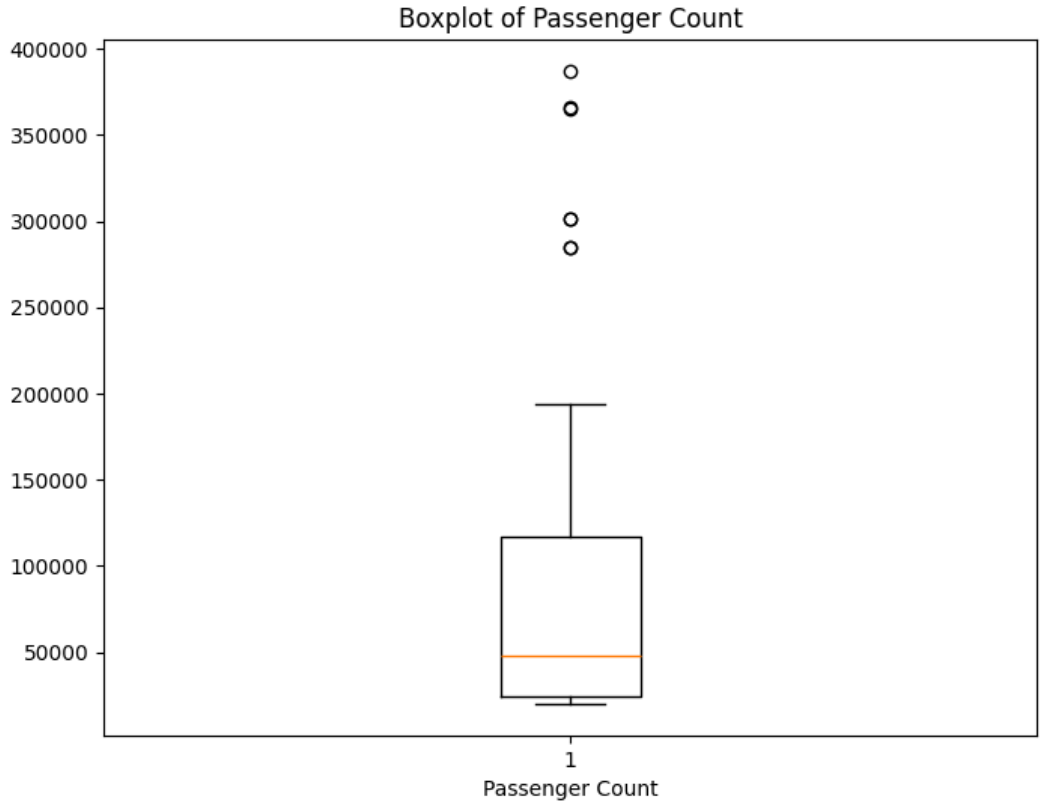
```
from scipy.stats import zscore
import numpy as np


z_scores = zscore(df['Passenger Count']) # Calculation of Z-scores

outliers = (np.abs(z_scores) > 3) # Defining a threshold for identifying
outliers

df_cleaned = df[~outliers].copy()# Removing outliers
```

After treating



Performing PCA

Before entering the features into a clustering algorithm, I ran PCA on the data.

Using PCA, a dimensionality reduction technique, we can reduce the number of principal components in our data and concentrate on a small number that account for the majority of the information and variance. PCA's primary benefit is actually visualization.

I used **Standard Scaling** (Z-score normalization) method.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

numerical_features = df_cleaned[['Passenger Count', 'Year']]

# Standardization of the numerical features
scaler = StandardScaler()
numerical_features_standardized = scaler.fit_transform(numerical_features)

# Applying PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(numerical_features_standardized)

# Creating a DataFrame with PCA results
pca_df = pd.DataFrame(data=pca_result, columns=['Principal Component 1',
'Principal Component 2'])

# Resetting the index of the original DataFrame
df_cleaned_reset = df_cleaned.reset_index(drop=True)

# Concatenating the PCA results with the reset DataFrame
final_df = pd.concat([df_cleaned_reset, pca_df], axis=1)

# Displaying the first few rows of the final DataFrame
print(final_df.head())
```

The data frame is ready for clustering algorithm.

Hierarchical clustering

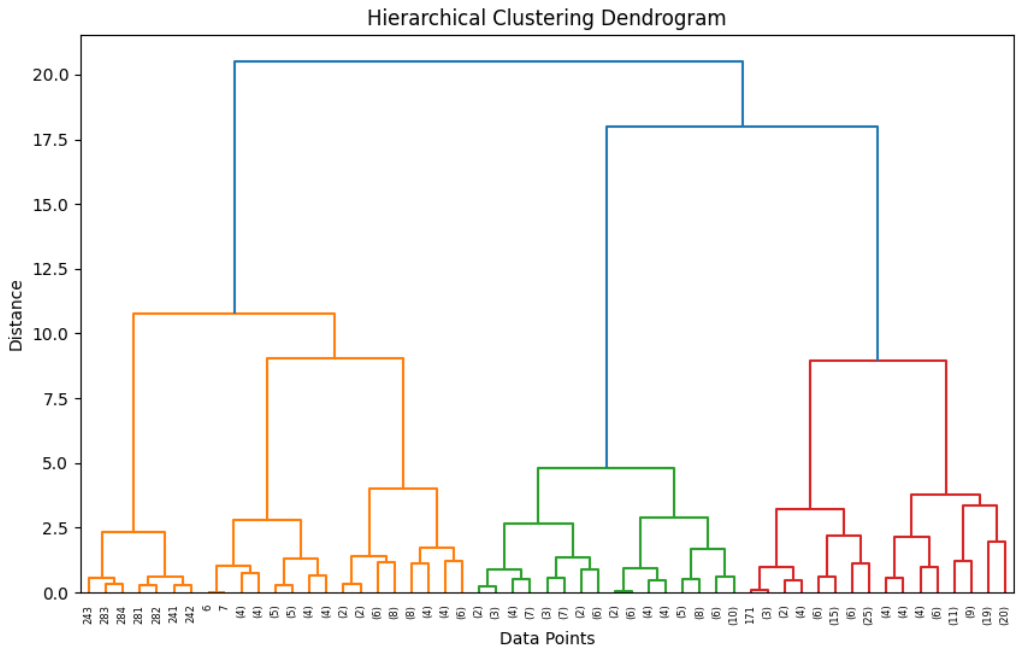
I tried hierarchical clustering first. Hierarchical clustering allows to build a cluster tree **("dendrogram")** that displays the clustering steps.

The tree can then be sliced horizontally to determine the number of clusters that make the most sense given its structure.

```
# 'final_df' contains the DataFrame with PCA results
pca_components = final_df[['Principal Component 1', 'Principal Component
2']]

# Performing hierarchical clustering
linkage_matrix = linkage(pca_components, method='ward')

# Plotting the dendrogram
plt.figure(figsize=(10, 6))
dendrogram(linkage_matrix, p=5, truncate_mode='level', orientation='top',
labels=final_df.index)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()
```



• The **linkage** function calculates the hierarchical clustering of principal components. The 'ward' linkage method is an effective option for reducing variance between clusters.

• The **dendrogram** function helps visualize the clustering hierarchy.

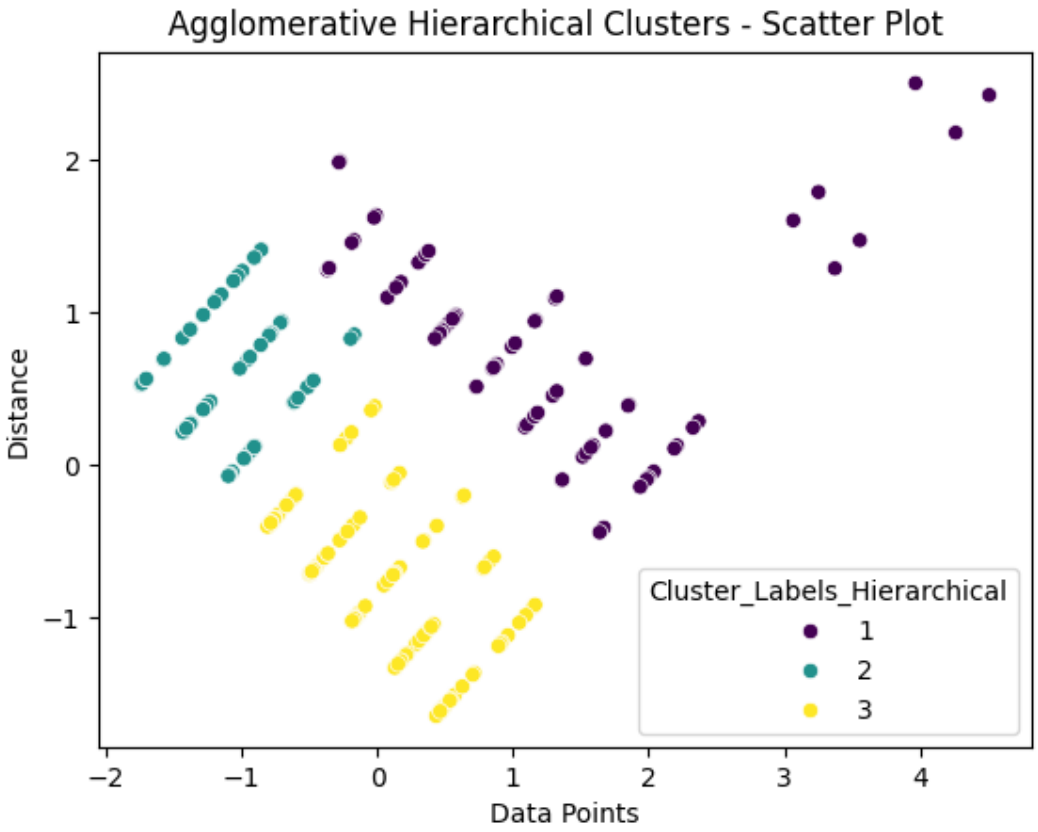
Scatter plot

The **scatter plot** visualizes the distribution of data points in the reduced two-dimensional space defined by the principal components obtained from PCA. Each point represents an observation, and the axes correspond to the first and second principal components. The color differentiation showcases the clustering patterns, offering insights into how well the hierarchical clustering algorithm grouped similar data points in this lower-dimensional feature space.

```
from scipy.cluster.hierarchy import fcluster
import seaborn as sns

# Specifying the number of clusters or the height at which to cut the
dendrogram
num_clusters = 3
# Cutting the dendrogram and getting cluster labels
cluster_labels = fcluster(linkage_matrix, t=num_clusters,
criterion='maxclust')
# Adding cluster labels to the DataFrame
final_df['Cluster_Labels_Hierarchical'] = cluster_labels

sns.scatterplot(x='Principal Component 1', y='Principal Component 2',
hue='Cluster_Labels_Hierarchical', data=final_df, palette='viridis')
plt.title('Agglomerative Hierarchical Clusters - Scatter Plot    ')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()
```



This provides us with a basic understanding of how many clusters could be present in this data and how they would show up in a scatter plot.

While hierarchical clustering is a great tool for showing the clustering process as a tree structure, it is not without its drawbacks. Hierarchical clustering only goes through the data once. This implies that records that are assigned or incorrectly assigned at the start of the process cannot be redistributed afterwards.

We rarely end our analysis with the hierarchical clustering solution; instead, we use it to obtain a "visual" representation of potential cluster shapes and then employ an iterative technique (such as k-means) to enhance and optimize the clustering solutions.

Using k-means

A common clustering technique called **K-means** involves choosing a target number of clusters, **k**, and allocating each record to one of the **k** clusters in order to reduce a measure of dispersion within the clusters.

The process is iterative and involves assigning and reallocating data points in order to reduce the distance between each record and the cluster centroid. When some progress is made at the conclusion of each iteration, the process is repeated until very little progress is made.

We must first specify k (number of clusters) in order to perform k-means clustering. We have two options for doing this:

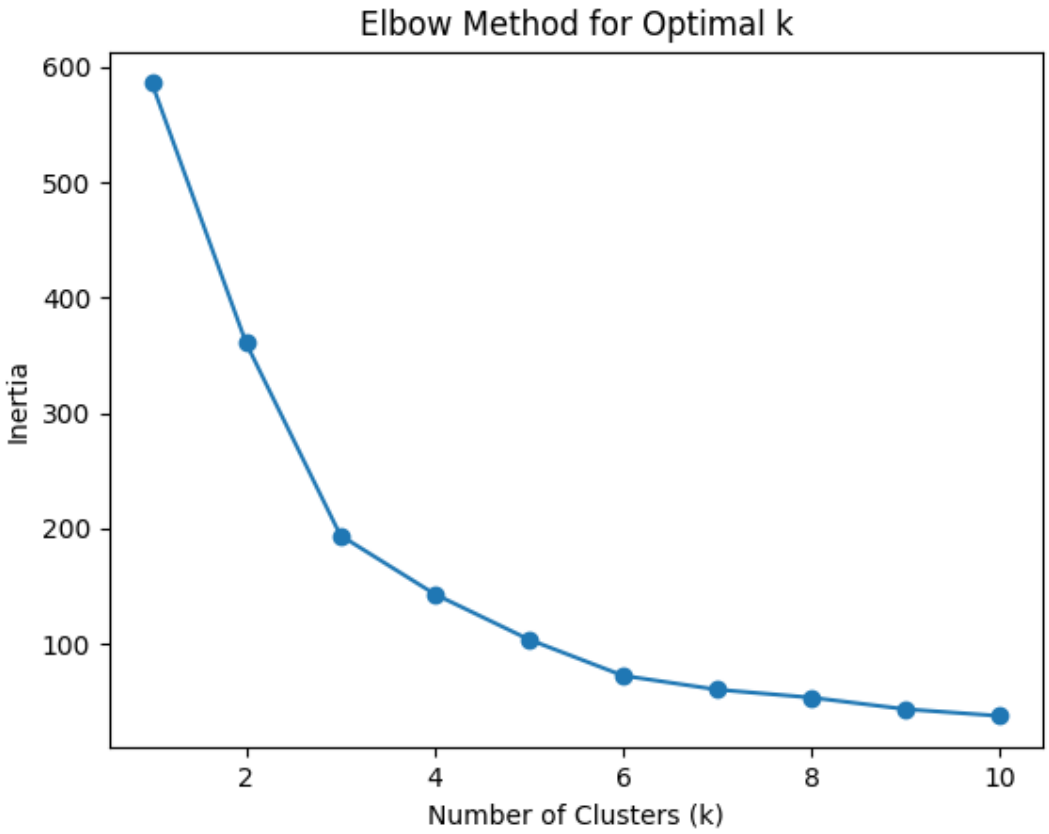
- To determine how many clusters to attempt, first perform hierarchical clustering.
- To find k, use the "Elbow" method.

The "Elbow" graph, which illustrates the turning point for the ideal number of clusters, is plotted by the code below.

```
from sklearn.cluster import KMeans

# 'pca_components' contains the principal components obtained from PCA
X = pca_components
k_values = range(1, 11)
inertia_values = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia_values.append(kmeans.inertia_)
plt.plot(k_values, inertia_values, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()
```



In this case, five clusters are the ideal number because the value does not drop noticeably beyond five.

Silhouette score

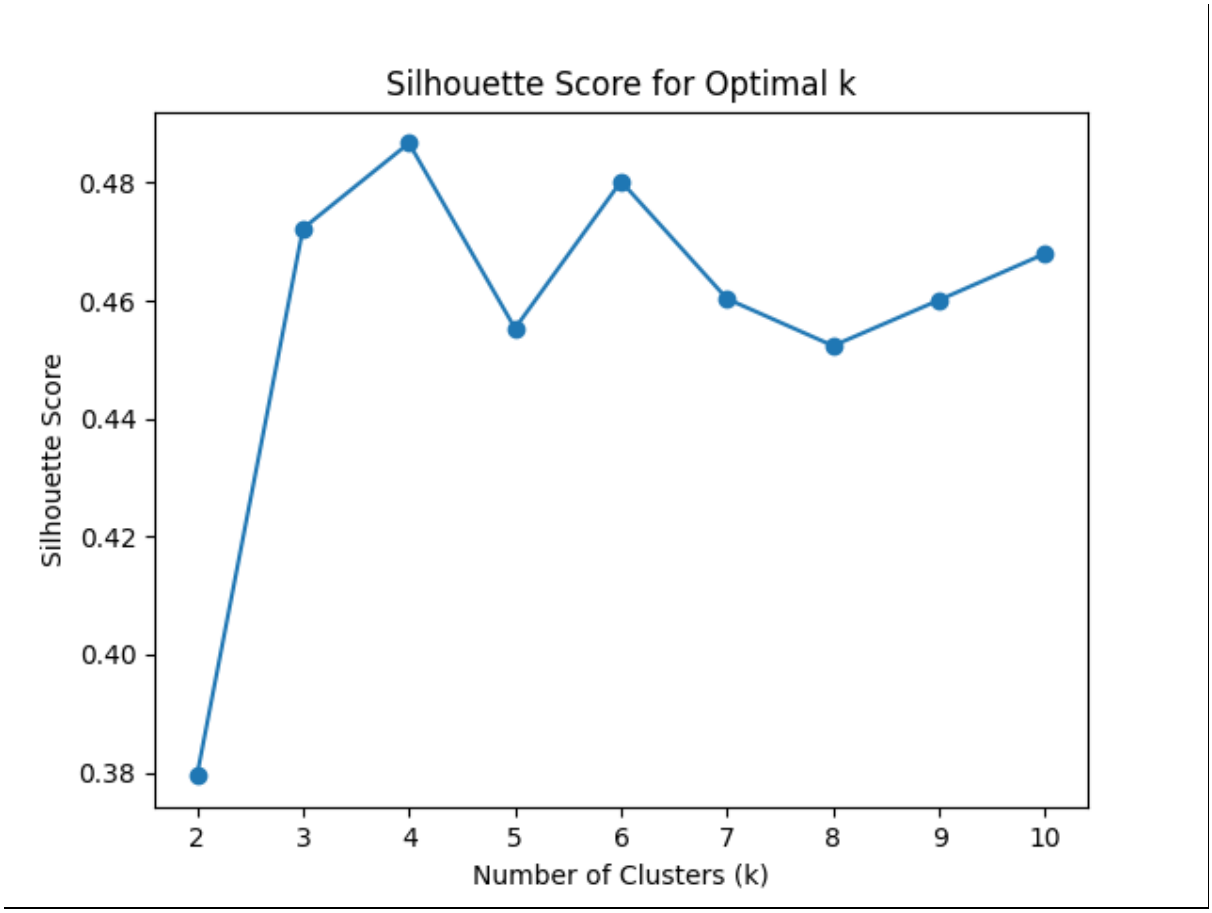
The **silhouette score** is an additional technique for figuring out the ideal number of clusters. The silhouette score compares an object's separation from other clusters to how similar it is to its own cluster (cohesion). A high score means the object is well matched to its own cluster and poorly matched to clusters nearby. The score ranges from -1 to 1.

```
from sklearn.metrics import silhouette_score

X = final_df[['Principal Component 1', 'Principal Component 2']]
k_values = range(2, 11)
silhouette_scores = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    silhouette_scores.append(silhouette_avg)

plt.plot(k_values, silhouette_scores, marker='o')
plt.title('Silhouette Score for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.show()
```



Conclusion

Data Overview:

- Air traffic passenger statistics were analyzed, with a focus on columns like "Passenger Count" and "Year."

Data Preparation:

- Identified and removed outliers from the "Passenger Count" column using appropriate methods. The dataset was then scaled, and principal component analysis (PCA) was used to reduce its dimensionality.

Cluster Analysis:

- Principal components were used in hierarchical clustering to identify patterns of similarity among data points in a reduced feature space.
- The elbow method and silhouette score were used to determine the best number of clusters for K-means clustering. The chosen number of clusters creates meaningful groupings within the data.

Visualization:

- Scatter plots showed the distribution of data points in the reduced PCA space, indicating potential clusters.

The research suggests that the air traffic passenger dataset exhibits meaningful patterns that can be uncovered through clustering techniques. The optimal number of clusters, derived from both the elbow method and silhouette score, provides insights into the natural grouping of air traffic data, assisting to identify specific trends or segments. Further analysis and interpretation of these clusters can provide valuable information for stakeholders in the air travel industry, guiding decision-making and strategic planning.

Overall, the clustering analysis, visualization, and data preprocessing work together to provide a thorough grasp of the underlying structures in the air traffic passenger dataset. This information can be used to make more informed decisions and gain a deeper understanding of the variables affecting patterns of air travel.

References

1. ***Air Traffic Passenger Statistics Dataset. (2016).***
   https://data.world/data-society/air-traffic-passenger-data

2. ***Machine Learning in Python. Pedregosa, F., et al. (2011). Journal of Machine Learning Research, 12, 2825–2830.***
   https://jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

3. ***Harris, C. R., et al. (2020). Nature, 585, 357–362*.**
   https://www.nature.com/articles/s41586-020-2649-2

4. ***Powerful data analysis tools for Python. McKinney, W. (2010). Proceedings of the 9th Python in Science Conference, 51–56.***
   https://conference.scipy.org/proceedings/scipy2010/mckinney.html

5. ***Visualization with Python. Hunter, J. D. (2007). Computing in Science & Engineering, 9(3), 90–95.***
   https://ieeexplore.ieee.org/document/4160265