K-Means Clustering on Geographical Data

Dr Ravi Teja

September 16, 2024

1 Introduction

In this document, we will explain the process of performing K-Means clustering on geographical data (longitude and latitude) using Python. We will be using the K-Means algorithm from `scikit-learn` to group countries based on their geographic coordinates.

2 Step 1: Importing Required Libraries

We start by importing the necessary libraries for data manipulation, visualization, and clustering.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

- `pandas` is used for data manipulation and loading the dataset.
- `matplotlib` and `seaborn` are used for data visualization.
- `KMeans` from `scikit-learn` is the algorithm used for clustering.

We also use `sns.set()` to set the default style for better visuals in `seaborn` plots.

```
sns.set()
```

3 Step 2: Loading the Data

Next, we load the dataset which contains information about different countries, including their longitude and latitude.

```
raw_data = pd.read_csv('Countries_exercise.csv')
```

- The dataset is loaded into a pandas DataFrame named `raw_data`. This allows us to easily manipulate and analyze the
  We then inspect the data by displaying the `raw_data` DataFrame.

```
raw_data
```

4 Step 3: Copying the Data

To avoid making changes to the original data, we create a copy of the dataset.

```
data = raw_data.copy()
```

This creates a new DataFrame called `data` that we will work with, preserving the original data.

5 Step 4: Visualizing the Geographical Data

We create a scatter plot to visualize the geographical locations (longitude and latitude) of the countries in the dataset.

```
plt.scatter(data['Longitude'], data['Latitude'])
plt.xlim(-180,180)
plt.ylim(-90, 90)
plt.show()
```

- `plt.scatter()` plots the countries on a map using longitude and latitude.
- `plt.xlim()` and `plt.ylim()` set the limits for the x-axis (longitude) and y-axis (latitude), respectively.
- `plt.show()` displays the plot.

This helps us understand the distribution of countries geographically before applying clustering.

6 Step 5: Selecting Features for Clustering

We select the longitude and latitude columns as the features for clustering.

```
x = data.iloc[:,1:3]
```

- The `iloc[:,1:3]` function selects all rows and columns 1 and 2, which correspond to the longitude and latitude columns.
- The selected data is stored in the variable `x`, which will be used as input for the K-Means algorithm.

7 Step 6: Initializing the K-Means Algorithm

We initialize the K-Means algorithm with the number of clusters set to 7.

```
kmeans = KMeans(7)
```

- `KMeans(7)` initializes the K-Means algorithm to partition the data into 7 clusters. The number of clusters is a hyperparameter chosen based on the problem.

8 Step 7: Training the K-Means Algorithm

Next, we train the K-Means model using the geographical data.

```
kmeans.fit(x)
```

- The `fit()` function trains the K-Means algorithm by grouping the data points (countries) based on their longitude and latitude into 7 clusters.

9 Step 8: Making Predictions with K-Means

We use the trained K-Means model to predict the clusters for each data point.

```
identified_clusters = kmeans.fit_predict(x)
```

- `fit_predict()` fits the model and simultaneously assigns each data point (country) to a cluster. The cluster labels are stored in the `identified_clusters` variable.

10 Step 9: Adding Cluster Information to the DataFrame

We create a new DataFrame that includes the predicted cluster information.

```
data_with_clusters = data.copy()
data_with_clusters['Cluster'] = identified_clusters
```

- We make a copy of the `data` DataFrame and add a new column called `Cluster`, which contains the cluster labels for each country.

11 Step 10: Visualizing the Clusters

Finally, we visualize the clusters by coloring the data points based on their cluster assignments.

```
plt.scatter(data['Longitude'], data['Latitude'], c=
    data_with_clusters['Cluster'], cmap='rainbow')
plt.xlim(-180,180)
plt.ylim(-90, 90)
plt.show()
```

- `plt.scatter()` creates a scatter plot of the countries, with the color of each point determined by its cluster.
- The `c` argument specifies the color based on the cluster assignment, and `cmap='rainbow'` ensures that different clusters are represented with different colors.
- The plot is displayed using `plt.show()`.

12 Conclusion

In this tutorial, we demonstrated how to use K-Means clustering to group countries based on their geographical coordinates (longitude and latitude). We visualized the original data distribution and the clusters produced by the algorithm. K-Means helps us find patterns in the data by grouping countries that are geographically close together into the same cluster.