**Class 05**

Uber trip data:

* Load the data [uber-data.csv]
* Discover and comment clusters of Uber data based on locations (longitude & latitude)
* Analyze the cluster centers by time
* Analyze the cluster centers by date
* Remember to choose the right algorithm, compute the optimal number of clusters and quality measures
* Develop adequate plots
* Apply the dataset for forecasting

**Solution:**

# Load necessary libraries

library(tidyverse) # For data manipulation and visualization

# Load the dataset

uber\_data

# Preview the data

head(uber\_data)

> head(uber\_data)

# A tibble: 6 × 4

`Date/Time` Lat Lon Base

<chr> <dbl> <dbl> <chr>

1 9/1/2014 0:01:00 40.2 -74.0 B02512

2 9/1/2014 0:01:00 40.8 -74.0 B02512

3 9/1/2014 0:03:00 40.8 -74.0 B02512

4 9/1/2014 0:06:00 40.7 -74.0 B02512

5 9/1/2014 0:11:00 40.8 -73.9 B02512

6 9/1/2014 0:12:00 40.7 -74.0 B02512

>

# Select relevant columns: Lat and Lon to create location\_data

location\_data <- uber\_data[,2:3]

# Find columns with missing values

names(uber\_data)[colSums(is.na(uber\_data)) > 0]

character(0)

#output above shows that no column has any missing value

# Randomly sample 10,000 rows (tried running code for whole data as it was throwing memory issue)

set.seed(42)

sampled\_data <- location\_data[sample(1:nrow(location\_data), size = 10000), ]

# Scale the sampled data for K-Means

scaled\_data <- scale(sampled\_data)

# Scale the sampled data for K-Means

scaled\_data <- scale(sampled\_data)

# Install and load factoextra package

install.packages("factoextra")

library(factoextra)

# Compute and plot the Elbow Method

fviz\_nbclust(sampled\_data, kmeans, method = "wss", k.max = 20) +

labs(title = "Elbow Method for Determining Optimal Clusters")

A graph with a line

Description automatically generated

The **elbow method** is a technique used to determine the optimal number of clusters (**k**) in clustering algorithms like K-Means. It helps identify the point where adding more clusters doesn't significantly improve the clustering solution.

Based on this defintion we can see that approx at K= can be our Elbow point as adding more clusters beyond this doesn't significantly reduce WCSS.

# Set the number of clusters = optimal value of k visible in elbow method

k <- 12

# Run K-Means clustering

set.seed(42)

kmeans\_result <- kmeans(sampled\_data, centers = k, nstart = 25)

# Assign cluster labels to the sampled data

sampled\_data1 <- sampled\_data

sampled\_data1$Cluster <- as.factor(kmeans\_result$cluster)

# Plot the clusters using ggplot2

library(factoextra)

# Plot the clusters

ggplot(sampled\_data1, aes(x = Lon, y = Lat, color = Cluster)) +

geom\_point(alpha = 0.6) +

labs(title = "K-Means Clustering of Uber Data (k = 12)",

x = "Longitude",

y = "Latitude") +

scale\_color\_manual(values = grDevices::rainbow(12)) + # Use a palette with 12 colors

theme\_minimal()

A graph with colorful dots

Description automatically generated

#k mean clustering results

kmeans\_result

> kmeans\_result

K-means clustering with 12 clusters of sizes 280, 179, 2300, 1805, 1123, 380, 682, 28, 756, 102, 22, 2343

Cluster means:

Lat Lon

1 40.65239 -73.78213

2 40.85113 -73.92674

3 40.76029 -73.97930

4 40.71861 -74.00205

5 40.77958 -73.95632

6 40.75972 -73.86474

7 40.66728 -73.97862

8 40.77145 -73.52467

9 40.70935 -73.94656

10 40.69295 -74.20267

11 40.98660 -73.78910

12 40.74086 -73.99432

# Perform DBSCAN clustering

library(dbscan)

# Set eps and minPts

#eps = 0.01: This is the maximum distance between two points to be considered as #neighbors

#minPts = 5: This is the minimum number of points required to form a dense region.

location\_data2 <- location\_data[sample(1:nrow(location\_data), size = 10000), ]

dbscan\_result <- dbscan(location\_data2, eps = 0.01, minPts = 5)

# Assign cluster labels to the sampled\_data

location\_data2$Cluster <- as.factor(dbscan\_result$cluster)

# Plot the clusters using ggplot2

# Generate a color palette with as many colors as clusters

# You can choose a color palette that suits your needs (e.g., RColorBrewer, viridis)

#library(RColorBrewer)

# Get the number of unique clusters (including noise points as 0)

num\_clusters <- length(unique(location\_data2$Cluster))

color\_palette <- brewer.pal(n = num\_clusters, name = "Set3")

ggplot(location\_data2, aes(x = Lon, y = Lat, color = Cluster)) +

geom\_point(alpha = 0.6) +

labs(title = "DBSCAN Clustering of Uber Data (Lat/Lon) eps = 0.01, minPts = 5",

x = "Longitude", y = "Latitude") +

scale\_color\_manual(values = color\_palette) + # Apply dynamic color palette

theme\_minimal()

A screen shot of a graph

Description automatically generated

# Perform DBSCAN clustering for different value of eps and minPts

library(dbscan)

# Set eps and minPts

#eps = 0.05: This is the maximum distance between two points to be considered as #neighbors

#minPts = 20: This is the minimum number of points required to form a dense region.

location\_data3 <- location\_data[sample(1:nrow(location\_data), size = 10000), ]

dbscan\_result <- dbscan(location\_data3, eps = 0.05, minPts = 20)

# Assign cluster labels to the sampled\_data

location\_data2$Cluster <- as.factor(dbscan\_result$cluster)

# Plot the clusters using ggplot2

# Generate a color palette with as many colors as clusters

# You can choose a color palette that suits your needs (e.g., RColorBrewer, viridis)

#library(RColorBrewer)

# Get the number of unique clusters (including noise points as 0)

num\_clusters <- length(unique(location\_data2$Cluster))

color\_palette <- brewer.pal(n = num\_clusters, name = "Set3")

ggplot(location\_data2, aes(x = Lon, y = Lat, color = Cluster)) +

geom\_point(alpha = 0.6) +

labs(title = "DBSCAN Clustering of Uber Data (Lat/Lon) eps = 0.05, minPts = 20",

x = "Longitude", y = "Latitude") +

scale\_color\_manual(values = color\_palette) + # Apply dynamic color palette

theme\_minimal()

A screen shot of a graph

Description automatically generated

#Silhouette Coefficient

# Calculate Silhouette Coefficient

silhouette\_values <- silhouette(kmeans\_result$cluster, dist(sampled\_data))

# Display summary of Silhouette Coefficient

summary(silhouette\_values)

> summary(silhouette\_values)

Silhouette of 10000 units in 12 clusters from silhouette.default(x = kmeans\_result$cluster, dist = dist(sampled\_data)) :

Cluster sizes and average silhouette widths:

280 179 2300 1805 1123 380 682 28 756

0.8109980 0.3184873 0.4069691 0.3692169 0.3042565 0.5479205 0.2759134 0.3517255 0.4324861

102 22 2343

0.7129240 0.4370105 0.3205333

Individual silhouette widths:

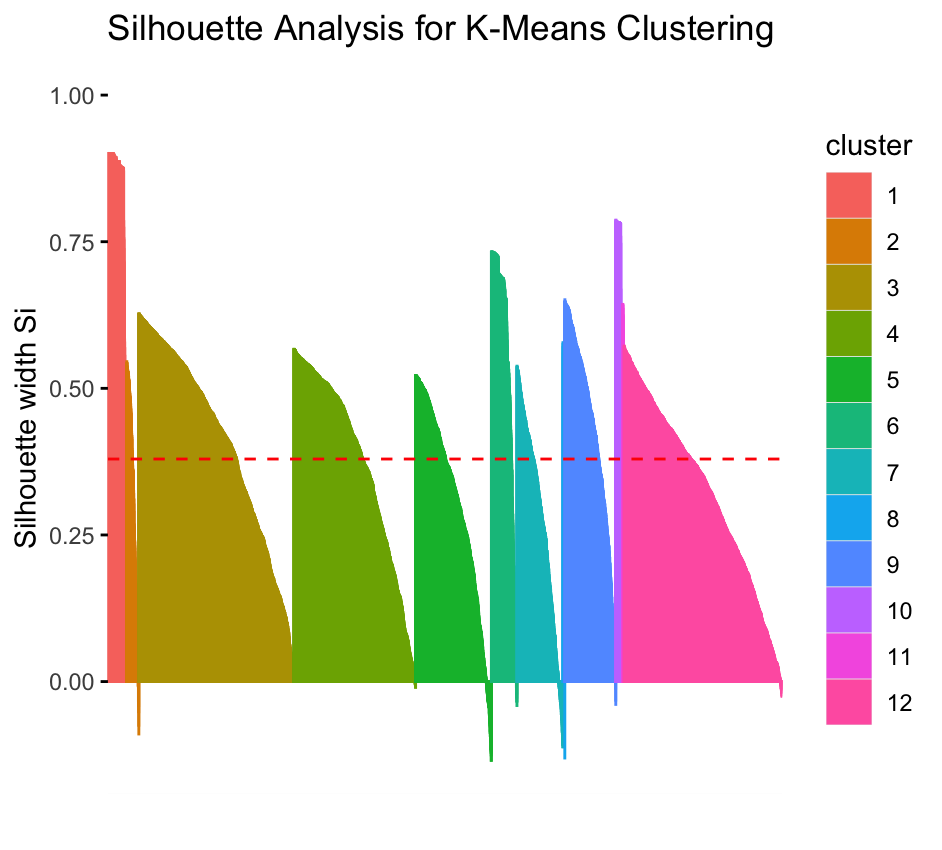
Min. 1st Qu. Median Mean 3rd Qu. Max.

-0.1343 0.2416 0.4013 0.3795 0.5137 0.9006

# Visualize Silhouette Plot

fviz\_silhouette(silhouette\_values) +

labs(title = "Silhouette Analysis for K-Means Clustering")



cluster size ave.sil.width

1 1 280 0.81

2 2 179 0.32

3 3 2300 0.41

4 4 1805 0.37

5 5 1123 0.30

6 6 380 0.55

7 7 682 0.28

8 8 28 0.35

9 9 756 0.43

10 10 102 0.71

11 11 22 0.44

12 12 2343 0.32

**#Interpretation of Result: sWe can see that Silhouette Width for the 12 clusters maded using K means are close to 1 supporting the optimal k value 12 obtained from elbow method**

* **Silhouette Width (Score)**:
  + **1**: Points are well-matched to their own cluster.
  + **0**: Points are on the boundary between clusters.
  + **Negative**: Points may have been assigned to the wrong cluster.
* Check the **average silhouette width** to evaluate overall clustering quality:
  + **0.5–1.0**: Good clustering.
  + **0.25–0.5**: Clustering is reasonable but could be improved.
  + **< 0.25**: Poor clustering.

# Calculate Calinski-Harabasz Index

install.packages("fpc")

library(fpc)

ch\_index <- cluster.stats(dist(sampled\_data), kmeans\_result$cluster)$ch

ch\_index

# Function to calculate CH index for various k

calculate\_ch\_index <- function(data, max\_k) {

ch\_values <- numeric(max\_k - 1) # To store CH values

for (k in 2:max\_k) { # Start from k = 2

# Perform K-means clustering

kmeans\_result <- kmeans(data, centers = k, nstart = 25)

# Calculate the CH index using calinhara function

ch\_index <- calinhara(data, kmeans\_result$cluster, k)

ch\_values[k - 1] <- ch\_index

}

return(ch\_values)

}

# Calculate CH index for k = 2 to k = 20

max\_k <- 20

ch\_values <- calculate\_ch\_index(sampled\_data, max\_k)

# Visualize the CH index values

plot(2:max\_k, ch\_values, type = "b", pch = 19, col = "blue",

xlab = "Number of Clusters (k)", ylab = "Calinski-Harabasz Index",

main = "CH Index for Different Values of k")

A graph with blue dots

Description automatically generated

> ch\_index

[1] 7677.043

Interpretation of the results: The **Calinski-Harabasz (CH) Index** is a clustering evaluation metric that helps determine the optimal number of clusters (k). It compares the ratio of the sum of between-cluster dispersion to within-cluster dispersion. Higher values of the CH index indicate better-defined clusters.

The **optimal k** is the number of clusters where the CH index reaches its maximum. Here in our case we can see that max value of ch\_index occurs approx. around k=12. This indicates that our conclusion from elbow method for optimal k is in line with CH Index.

#Analyze the cluster centers by time for sample 10000 data from uber data

# Load necessary libraries

library(ggplot2)

library(dplyr)

library(lubridate)

# Sample 10,000 random rows from Uber data

set.seed(42)

sampled\_data3 <- uber\_data[sample(1:nrow(uber\_data), size = 10000), ]

sampled\_data3

# Convert the "Date/Time" column to a proper datetime format

sampled\_data3$DateTime <- mdy\_hms(sampled\_data3$`Date/Time`)

# Extract hour of the day

sampled\_data3$Hour <- hour(sampled\_data3$DateTime)

# Extract latitude and longitude for clustering

location\_data1 <- sampled\_data3[, c("Lat", "Lon")]

# Add cluster labels to the sampled data

sampled\_data3$Cluster <- as.factor(kmeans\_result$cluster)

sampled\_data3

kmeans\_result

# Analyze clusters by time

cluster\_analysis <- sampled\_data3 %>%

group\_by(Cluster, Hour) %>%

summarise(Count = n(), .groups = "drop")

# Print cluster analysis

print(cluster\_analysis)

> cluster\_analysis

# A tibble: 266 × 3

Cluster Hour Count

<fct> <int> <int>

1 1 0 5

2 1 1 4

3 1 2 3

4 1 3 3

5 1 4 8

6 1 5 6

7 1 6 10

8 1 7 11

9 1 8 4

10 1 9 11

# ℹ 256 more rows

# ℹ Use `print(n = ...)` to see more rows

>

install.packages("RColorBrewer")

library(ggplot2)

library(RColorBrewer)

# Assume sampled\_data3 is your data with Lat, Lon, and Cluster columns

# Get 12 colors from the "Set3" palette in RColorBrewer

cluster\_colors <- brewer.pal(12, "Set3")

# Visualize cluster activity over time

ggplot(cluster\_analysis, aes(x = Hour, y = Count, color = Cluster, group = Cluster)) +

geom\_line(size = 1) +

geom\_point(size = 2) +

labs(title = "Cluster Activity Over Time",

x = "Hour of the Day",

y = "Number of Trips") +

theme\_minimal() +

scale\_color\_manual(values = RColorBrewer::brewer.pal(12, "Set3")) # Use 12 colors for clusters

A graph of different colored lines

Description automatically generated

**Interpretation:** Here we see a line plot where each line represents a cluster's activity over time (i.e., number of trips for each hour of the day). The x-axis represents the hour of the day, and the y-axis shows the number of trips. Each cluster has its own line with a distinct color. Based on the graphical representation we can see that approx around 17’o clock the number of trips across all the clusters have been maximum indicating evening rush hour.

#Analyze the cluster centers by date for sample 10000 data from uber data and k=12

# Sample 10,000 random rows from Uber data

set.seed(42)

sampled\_data4 <- uber\_data[sample(1:nrow(uber\_data), size = 10000), ]

sampled\_data4

# Convert the "Date/Time" column to a proper datetime format

sampled\_data4$DateTime <- mdy\_hms(sampled\_data4$`Date/Time`)

# Extract the date from DateTime

sampled\_data4$Date <- as.Date(sampled\_data4$DateTime)

# Add cluster labels to the sampled data

sampled\_data4$Cluster <- as.factor(kmeans\_result$cluster)

# Analyze clusters by date

cluster\_analysis\_by\_date <- sampled\_data4 %>%

group\_by(Cluster, Date) %>%

summarise(Count = n(), .groups = "drop")

# Print cluster analysis by date

print(cluster\_analysis\_by\_date)

> print(cluster\_analysis\_by\_date)

# A tibble: 334 × 3

Cluster Date Count

<fct> <date> <int>

1 1 2014-09-01 14

2 1 2014-09-02 13

3 1 2014-09-03 9

4 1 2014-09-04 6

5 1 2014-09-05 6

6 1 2014-09-06 2

7 1 2014-09-07 15

8 1 2014-09-08 10

9 1 2014-09-09 6

10 1 2014-09-10 6

# ℹ 324 more rows

# ℹ Use `print(n = ...)` to see more rows

# Visualize cluster activity by date

ggplot(cluster\_analysis\_by\_date, aes(x = Date, y = Count, color = Cluster, group = Cluster)) +

geom\_line(size = 1) +

geom\_point(size = 2) +

labs(title = "Cluster Activity Over Time (By Date)",

x = "Date",

y = "Number of Trips") +

theme\_minimal() +

scale\_color\_manual(values = RColorBrewer::brewer.pal(12, "Set3")) # Use 12 colors for clusters

**Interpretation:**

The line plot here displays the cluster activity for each date, with each line corresponding to a different cluster. Based on the pattern we can see that cabs in cluster 3 have comparatively high number of rides in comparison to rest other clusters . Around 13th-14th September cab rides were high across all clusters

A graph of different colored lines

Description automatically generated

#Analyze the cluster centers by geographic location for sample 10000 data from uber data and k=12

# Sample 10,000 random rows from Uber data

set.seed(42)

sampled\_data5 <- uber\_data[sample(1:nrow(uber\_data), size = 10000), ]

sampled\_data5

# Add cluster labels to the sampled data

sampled\_data5$Cluster <- as.factor(kmeans\_result$cluster)

# Plot the geographic locations with K-means clusters

install.packages("leaflet")

library(leaflet)

# Create an interactive leaflet map with the K-means clusters

leaflet(data = sampled\_data5) %>%

addTiles() %>% # Add OpenStreetMap tiles

addCircleMarkers(~Lon, ~Lat, color = ~factor(Cluster),

radius = 3, opacity = 0.6, fillOpacity = 0.5,

popup = ~paste("Cluster:", Cluster)) %>%

addLegend("bottomright", pal = colorFactor(palette = "Set3", levels = 1:12),

values = ~Cluster, title = "Cluster")

**Interpretation:**

**Based on cluster plot on a geographic map we can see that most of the cabs are operating in Manhattan area and that too in regions around central park.**

A map of a city

Description automatically generated

#perform forecasting on clustering using 10,000 random samples from the top 50,000 Uber data and check how the model works on the last 10 data points

# Load necessary libraries

library(dplyr)

library(lubridate)

library(ggplot2)

library(forecast)

# Sample 10,000 random rows from the top 50,000 Uber data

set.seed(42)

sampled\_data6 <- uber\_data[1:50000, ]

sampled\_data6 <- sampled\_data6[sample(1:nrow(sampled\_data), size = 10000), ]

# Extract latitude and longitude for clustering

location\_data2<- sampled\_data6[, c("Lat", "Lon")]

# Perform K-means clustering with k = 12

set.seed(42)

kmeans\_result\_new <- kmeans(location\_data2, centers = 12, nstart = 25)

# Add cluster labels to the sampled data

sampled\_data6$Cluster <- as.factor(kmeans\_result\_new$cluster)

sampled\_data6

# Forecasting - let's assume we want to forecast based on the cluster activity (e.g., trips per hour)

# Create a time series of the number of trips per hour per cluster

sampled\_data6$Hour <- hour(mdy\_hms(sampled\_data6$`Date/Time`)) # Extract Hour from Date/Time

# Aggregate the data by Hour and Cluster, calculating the number of trips per hour

hourly\_activity <- sampled\_data6 %>%

group\_by(Cluster, Hour) %>%

summarise(TripCount = n(), .groups = "drop")

# Forecast the trip count for the last 10 hours for a specific cluster (e.g., Cluster 1)

cluster\_1\_data <- hourly\_activity %>% filter(Cluster == 1)

# Prepare the data for time series forecasting (ARIMA model)

cluster\_1\_ts <- ts(cluster\_1\_data$TripCount, frequency = 24) # Assuming 24 hours per day

# Split the data into training and testing sets

train\_data <- head(cluster\_1\_ts, -10) # Training data (excluding last 10 observations)

test\_data <- tail(cluster\_1\_ts, 10) # Test data (last 10 observations)

# Fit an ARIMA model on the training data

arima\_model <- auto.arima(train\_data)

# Forecast the next 10 values (corresponding to the last 10 hours)

forecasted\_values <- forecast(arima\_model, h = 10)

# Print forecast results

print(forecasted\_values)

# Plot the actual vs forecasted values

autoplot(forecasted\_values) +

autolayer(test\_data, series = "Actual", PI = FALSE) +

labs(title = "ARIMA Forecast vs Actual for Cluster 1",

x = "Time", y = "Number of Trips") +

theme\_minimal()

# Evaluate forecast accuracy (for example, Mean Absolute Error)

mae <- mean(abs(forecasted\_values$mean - test\_data))

print(paste("Mean Absolute Error: ", mae))

Above code performs forecasting on clustering using 10,000 random samples from the top 50,000 Uber data and after that i have tried to check how the model works on the last 10 data points.

The plot shows the forecasted values (from the ARIMA model) along with the actual values from the last 10 data points

The Mean Absolute Error (MAE) gives an indication of how well the model is forecasting. A lower MAE means better accuracy here in our case its value is 0.73 means that, on average, the forecasted values are off by **0.73 units** from the actual values.

A graph showing a graph of a graph

Description automatically generated with medium confidence

"Mean Absolute Error: 0.731511335198826"