**Optimal Number of Clusters**

* **Analysis Results**:
  + The Elbow Method and Silhouette Score were used to determine the optimal number of clusters.
  + The Elbow Method suggested an optimal k of **2 or 3** based on the "elbow point" in the WCSS graph.
  + The Silhouette Score analysis confirmed that **k=2** provided the best clustering structure.

*# Step 1: Import Necessary Libraries for Clustering*

**from** sklearn.cluster **import** KMeans

**import** matplotlib.pyplot **as** plt

*# Reload the scaled training dataset*

file\_path\_train **=** "C:/Users/samee/X\_train\_scaled.csv"

X\_train **=** pd**.**read\_csv(file\_path\_train)

*# Select numerical features for clustering*

features\_for\_clustering **=** ['tenure', 'MonthlyCharges', 'TotalServiceCost']

* **Supporting Visualizations**:
  + **Elbow Method Plot**: Shows the WCSS decreasing with the number of clusters.

*# Step 2: Determine the Optimal Number of Clusters Using the Elbow Method*

*# Initialize variables for the elbow method*

wcss **=** [] *# Within-cluster sum of squares*

*# Try different numbers of clusters (from 1 to 10) and compute WCSS*

**for** n\_clusters **in** range(1, 11):

kmeans **=** KMeans(n\_clusters**=**n\_clusters, n\_init**=**10, random\_state**=**42)

kmeans**.**fit(X\_train[features\_for\_clustering])

wcss**.**append(kmeans**.**inertia\_) *# Inertia measures clustering tightness*

*# Step 3: Plot the Elbow Method Graph*

plt**.**figure(figsize**=**(8, 5))

plt**.**plot(range(1, 11), wcss, marker**=**'o', linestyle**=**'-', color**=**'b')

plt**.**xlabel("Number of Clusters (k)")

plt**.**ylabel("WCSS (Within-Cluster Sum of Squares)")

plt**.**title("Elbow Method to Determine Optimal k")

plt**.**xticks(range(1, 11))

plt**.**grid()

plt**.**show()

* + **Silhouette Score Plot**: Highlights that k=2 has the highest silhouette score, indicating the best clustering separation

**from** sklearn.metrics **import** silhouette\_score

**from** sklearn.cluster **import** KMeans

*# Extract feature columns (excluding the cluster labels)*

X **=** df**.**drop(columns**=**["gender","SeniorCitizen", "Dependents", "PhoneService","MultipleLines","InternetService\_DSL","InternetService\_Fiber optic","Contract\_Month-to-month","Contract\_One year","Contract\_Two year"])

*# Evaluate silhouette scores for different cluster numbers (e.g., from 2 to 10)*

silhouette\_scores **=** {}

**for** k **in** range(2, 11):

kmeans **=** KMeans(n\_clusters**=**k, random\_state**=**42, n\_init**=**10)

labels **=** kmeans**.**fit\_predict(X)

score **=** silhouette\_score(X, labels)

silhouette\_scores[k] **=** score

*# Display the silhouette scores for different cluster values*

**import** matplotlib.pyplot **as** plt

plt**.**figure(figsize**=**(8, 5))

plt**.**plot(list(silhouette\_scores**.**keys()), list(silhouette\_scores**.**values()), marker**=**"o")

plt**.**xlabel("Number of Clusters")

plt**.**ylabel("Silhouette Score")

plt**.**title("Silhouette Score for Different Cluster Numbers")

plt**.**grid(**True**)

plt**.**show()

*# Find the best number of clusters based on silhouette score*

best\_k **=** max(silhouette\_scores, key**=**silhouette\_scores**.**get)

best\_k