CCA-8 Train the clustering model and interpret results

Overview:

From the elbow and silhouette analysis, we identified the number of clusters as 4. Now we will fit K-Means clustering algorithms with the identified number of clusters. We will now segments the customers based on the features tenure and monthly charges. That will help us to determine the characteristics of the clusters that will be used for the future decision and analysis.

Steps:

Step 1: Loading datasets

This steps include the loading both min-max scaled and standard scaled datasets.

```
import os
import pandas as pd
from sklearn.cluster import KMeans
import json

# Define the paths to the scaled datasets
min_max_scaled_data_path = os.path.join(project_root,
'data_preparation/scaling_techniques/min_max_scaled_dataset.csv')
standard_scaled_data_path = os.path.join(project_root,
'data_preparation/scaling_techniques/standard_scaled_dataset.csv')

# Load the datasets
df_min_max_scaled = pd.read_csv(min_max_scaled_data_path)
df_standard_scaled = pd.read_csv(standard_scaled_data_path)

print("Min-Max scaled and Standard scaled datasets loaded successfully.")

```

Step 2: Fitting K-Means Model

Now we will apply the K-Means Model for the both scaled datasets. For this, we will be using 4 as a optimal number of clusters determined from elbow and silhouette analysis. As the result it will helps us to identified the characteristics of each clusters based on the customers segmentation features tenure and monthly charges.

```
# Utility function to convert absolute path to relative path
def to_relative_path(absolute_path, start_path):
    return os.path.relpath(absolute_path, start=start_path).replace('\\', '/')

# Function to fit K-Means and analyze clusters
```

```
def fit_kmeans_and_analyze(df, scaling_label):
    # Fit the K-Means model with k=4
    kmeans = KMeans(n_clusters=4, init='k-means++', n_init=10, random_state=42)
    df['Cluster'] = kmeans.fit_predict(df[['tenure', 'MonthlyCharges']])

    # Display the first few rows with cluster assignments
    print(f"{scaling_label} Dataset with Cluster Assignments:")
    print(df.head())

    # Ensure the correct paths for saving results
    kmeans_model_path = os.path.join(project_root, 'Clustering_Analysis', 'kmeans_model')
    os.makedirs(kmeans_model_path, exist_ok=True)

    # Save the DataFrame with cluster assignments
    result_filename = f'{scaling_label.lower().replace(" ", "_")}_4_clusters.csv'
    result_filepath = os.path.join(kmeans_model_path, result_filename)
    df.to_csv(result_filepath, index=False)
    print(f'Saved cluster assignments to: {result_filepath}')

    # Update the config file with the new path for cluster assignments
    relative_result_filepath = to_relative_path(result_filepath, project_root)
    config[f'{scaling_label.lower().replace(" ", "_")}_4_clusters_path'] = relative_result_filepath

    with open(config_path, 'w') as f:
        json.dump(config, f, indent=4)
    print(f"Updated config.json with new path for {scaling_label} cluster assignments.")

# Perform clustering analysis on both datasets
fit_kmeans_and_analyze(df_min_max_scaled, 'Min-Max Scaled')
fit_kmeans_and_analyze(df_standard_scaled, 'Standard Scaled')

```

Step 3 : Analysing the cluster characteristics

This steps includes analysing the cluster characteristics to the future insights and decision making.

```
# Function to analyze and save cluster characteristics
def analyze_and_save_cluster_characteristics(df, scaling_label):
    # Ensure the correct paths for saving results
    kmeans_model_path = os.path.join(project_root, 'Clustering_Analysis', 'kmeans_model')
    os.makedirs(kmeans_model_path, exist_ok=True)

    # Analyze the cluster characteristics
    cluster_characteristics = df.groupby('Cluster').agg({
        'tenure': ['mean', 'median', 'std'],
        'MonthlyCharges': ['mean', 'median', 'std']
    })
    print(f"\nCluster Characteristics ({scaling_label}):")
    print(cluster_characteristics)

    # Save the cluster characteristics
    characteristics_filename = f'{scaling_label.lower().replace(" ", "_")}_cluster_characteristics.csv'
    characteristics_filepath = os.path.join(kmeans_model_path, characteristics_filename)
    cluster_characteristics.to_csv(characteristics_filepath)
    print(f'Saved cluster characteristics to: {characteristics_filepath}')

    # Update the config file with the new path for cluster characteristics
    relative_characteristics_filepath = to_relative_path(characteristics_filepath, project_root)
```

```
    config[f'{scaling_label.lower().replace(" ", "_")}_cluster_characteristics_path'] =
relative_characteristics_filepath

    with open(config_path, 'w') as f:
        json.dump(config, f, indent=4)
    print(f"Updated config.json with new path for {scaling_label} cluster characteristics.")

# Analyze and save characteristics for both datasets
analyze_and_save_cluster_characteristics(df_min_max_scaled, 'Min-Max Scaled')
analyze_and_save_cluster_characteristics(df_standard_scaled, 'Standard Scaled')

```

Summarize

High Tenure, High Charges (Premium Customers): These are the long-term, high-value customers, likely subscribed to premium plans

Low Tenure, Low Charges(New or Basic Customers): These are new or lower customers, ideal targets for engagement strategies

High tenure, Low Charges(Loyal but Economical Customers): These are loyal customers on economical plans, potential for upselling.

Moderate Tenure and Charges(Mid-Tier Customers): These are Stable mid-tier customers, potential for growth.

Next Steps:

After we train the cluster based on the characteristics, we will develop the visualization to interpret clustering results using the tools like matplotlib or Seaborn.