CCA-9 Create visualisations for interpretation

Overview:

This clustering documentation guide you through generating variance visualisation to interpret the clustering results of customer segmentation visualisation are included scatter plots, box plots, distribution plots and heat maps. Which help us to understanding the characteristics and distribution of the clusters formed using K-Means clustering.

Steps:

Step 1: Importing libraries and loading configuration

From this step, we are import the necessary libraries and load the configuration settings. The configuration file carry the paths to the database with the cluster assignments, which will be used for visualisation.

```
import os
import json
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Define paths
notebook_dir = os.path.dirname(os.path.abspath(''))
project_root = os.path.abspath(os.path.join(notebook_dir, '..'))
config_path = os.path.join(project_root, 'config.json')

# Load configuration file
with open(config_path, 'r') as f:
    config = json.load(f)

# Convert relative paths to absolute paths
min_max_scaled_4_clusters_path = os.path.join(project_root, config['min-max_scaled_4_clusters_path'])
standard_scaled_4_clusters_path = os.path.join(project_root, config['standard_scaled_4_clusters_path'])

# Load the datasets
df_min_max_scaled_4_clusters = pd.read_csv(min_max_scaled_4_clusters_path)
df_standard_scaled_4_clusters = pd.read_csv(standard_scaled_4_clusters_path)

# Verify the datasets
print("Min-Max Scaled 4 Clusters Dataset:")
display(df_min_max_scaled_4_clusters.head())
```

```
print("Standard Scaled 4 Clusters Dataset:")
display(df_standard_scaled_4_clusters.head())
```

Step 2: Defining the visualisation functions

We will define many functions to make different visualisations that will help us in analysis the clusters:

- **Scatter plot with centroids** : visualizes the clusters and their centroids.
- **Box plots:** result the distribution of tenure and monthly charges with in each cluster.
- **Heat map of cluster characteristics:** defined the average tenure and monthly charges for each cluster.

```
# Define the path for saving visualizations
visualizations_path = os.path.join(project_root, 'Clustering_Analysis', 'visualizations')
os.makedirs(visualizations_path, exist_ok=True)

# Function to compute centroids
def compute_centroids(df, n_clusters):
    kmeans = KMeans(n_clusters=n_clusters, init='k-means++', n_init=10, random_state=42)
    kmeans.fit(df[['tenure', 'MonthlyCharges']])
    centroids = kmeans.cluster_centers_
    return centroids

# Function to plot scatter plot of clusters with centroids
def plot_cluster_scatter(df, scaling_label, save_path, n_clusters):
    centroids = compute_centroids(df, n_clusters)

    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x='tenure', y='MonthlyCharges', hue='Cluster', palette='viridis')
    plt.scatter(centroids[:, 0], centroids[:, 1], s=300, c='red', marker='X', label='Centroids')
    plt.title(f'Customer Segments based on Tenure and Monthly Charges ({scaling_label})')
    plt.xlabel('Tenure')
    plt.ylabel('Monthly Charges')
    plt.legend(title='Cluster')
    plt.tight_layout()
    file_path = os.path.join(save_path, f'cluster_scatter_{scaling_label.lower().replace(" ", "_")}.png')
    plt.savefig(file_path)
    plt.show()

# Function to plot boxplots of clusters
def plot_cluster_boxplots(df, scaling_label, save_path):
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Cluster', y='tenure', data=df, hue='Cluster', palette='viridis', legend=False)
    plt.title(f'Tenure by Cluster ({scaling_label})')
    plt.xlabel('Cluster')
    plt.ylabel('Tenure')
    plt.tight_layout()
    file_path_tenure = os.path.join(save_path, f'cluster_boxplot_tenure_{scaling_label.lower().replace(" ",
"_")}.png')
    plt.savefig(file_path_tenure)
    plt.show()

    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Cluster', y='MonthlyCharges', data=df, hue='Cluster', palette='viridis', legend=False)
    plt.title(f'Monthly Charges by Cluster ({scaling_label})')
    plt.xlabel('Cluster')
    plt.ylabel('Monthly Charges')
```

```
    plt.tight_layout()
    file_path_charges = os.path.join(save_path, f'cluster_boxplot_charges_{scaling_label.lower().replace(" ", "_")}.png')
    plt.savefig(file_path_charges)
    plt.show()

# Function to plot distribution of clusters
def plot_cluster_distribution(df, scaling_label, save_path):
    plt.figure(figsize=(10, 6))
    sns.countplot(x='Cluster', data=df, hue='Cluster', palette='viridis', legend=False)
    plt.title(f'Cluster Distribution ({scaling_label})')
    plt.xlabel('Cluster')
    plt.ylabel('Count')
    plt.tight_layout()
    file_path = os.path.join(save_path, f'cluster_distribution_{scaling_label.lower().replace(" ", "_")}.png')
    plt.savefig(file_path)
    plt.show()

# Function to plot heatmap of cluster characteristics
def plot_cluster_heatmap(df, scaling_label, save_path):
    cluster_summary = df.groupby('Cluster').agg({
        'tenure': 'mean',
        'MonthlyCharges': 'mean'
    }).reset_index()

    plt.figure(figsize=(10, 6))
    sns.heatmap(cluster_summary.set_index('Cluster').T, annot=True, cmap='viridis')
    plt.title(f'Cluster Heatmap ({scaling_label})')
    plt.tight_layout()
    file_path = os.path.join(save_path, f'cluster_heatmap_{scaling_label.lower().replace(" ", "_")}.png')
    plt.savefig(file_path)
    plt.show()

```

Step 3 : Generating Visualizations

In this step we will generate the visualizations for both min-max scaled and standard scaled datasets:

```
# Generate visualizations for Min-Max scaled clusters and Standard scaled clusters
plot_cluster_scatter(df_min_max_scaled_4_clusters, 'Min-Max Scaled', visualizations_path, n_clusters=4)
plot_cluster_scatter(df_standard_scaled_4_clusters, 'Standard Scaled', visualizations_path, n_clusters=4)

plot_cluster_boxplots(df_min_max_scaled_4_clusters, 'Min-Max Scaled', visualizations_path)
plot_cluster_boxplots(df_standard_scaled_4_clusters, 'Standard Scaled', visualizations_path)

plot_cluster_distribution(df_min_max_scaled_4_clusters, 'Min-Max Scaled', visualizations_path)
plot_cluster_distribution(df_standard_scaled_4_clusters, 'Standard Scaled', visualizations_path)

plot_cluster_heatmap(df_min_max_scaled_4_clusters, 'Min-Max Scaled', visualizations_path)
plot_cluster_heatmap(df_standard_scaled_4_clusters, 'Standard Scaled', visualizations_path)

```

Results:

- **Cluster assessments:** the dataset were segmented into 4 clusters using the K-means algorithm.
- **Scatter plots:** provided a clear visualisation of customer segments based on tenure and monthly charges, with distinct centroids.

- **Box plots:** illustrated the variation in tenure and monthly charges with in each cluster.
- **Clusters distribution:** showed the number of customers in each cluster, revealing the distribution of segments.
- **Heat maps:** summarized the average tenure and monthly charges for each cluster, highlighting the key characteristics.

Conclusion:

Consistent clustering: the visualisations confirm that 4 clusters offer a meaningful segmentation of customers with clear differences in tenure and monthly charges across clusters.

Actionable insights: these clusters provide insights into customer behaviour and that can inform targeted marketing planning and customer retention efforts.

Next Steps

After Visualizations to aid in the interpretation of clustering results. we will utilise the clusters to predict customer churn and develop targeted retention strategies and also refine the clustering model based on further analysis and testing.