Anomaly Detection and Clustering Documentation

This document elaborates on the code provided for anomaly detection and clustering using a dataset of transactions. The code is structured to follow a logical sequence of steps involved in Exploratory Data Analysis (EDA), Preprocessing, Principal Component Analysis (PCA), K-means Clustering, and Anomaly Detection.

Table of Contents

1. Loading Libraries and Dataset

Load necessary libraries and the dataset. Libraries include pandas for data manipulation, matplotlib and seaborn for visualization, and scikit-learn for machine learning tasks.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import silhouette_score

ctr = pd.read_csv(r'C:\Users\Rick-Royal\Documents\PROJECT\ctr_eda1.csv')
```

2. Exploratory Data Analysis (EDA)

Investigate the structure of the data, check for missing values, unique values, and basic statistics of the dataset.

```
ctr.info()
ctr.nunique()
sns.heatmap(ctr.corr(), annot=True)
```

3. Preprocessing

Drop unnecessary columns, encode categorical variables, and scale the numerical variables to standardize the dataset.

```
# Dropping columns
ctr = ctr.drop([...], axis=1)

# Encoding categorical variables
le = LabelEncoder()
ctr['Column'] = le.fit_transform(ctr['Column'])

# Standardizing numerical variables
scaler = StandardScaler()
ctr_scaled = scaler.fit_transform(ctr)
```

4. PCA Decomposition

Perform PCA to reduce the dimensionality of the dataset while retaining most of the variance.

```
pca = PCA(n_components=0.95)
pca.fit(ctr_scaled)
ctr_pca = pca.transform(ctr_scaled)
```

5. K-means Clustering

Apply K-means clustering to group data into clusters and evaluate the optimal number of clusters using the silhouette score.

```
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(ctr_pca)
    silhouette_avg = silhouette_score(ctr_pca, cluster_labels)
    silhouette_scores.append(silhouette_avg)

optimal_k = silhouette_scores.index(max(silhouette_scores)) + 2
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
clusters = kmeans.fit_predict(ctr_pca)
```

6. Anomaly Detection

Detect anomalies in the dataset using Isolation Forest or other anomaly detection algorithms.

```
from sklearn.ensemble import IsolationForest

isolation_forest = IsolationForest(contamination=0.01, random_state=42)
anomalies = isolation_forest.fit_predict(ctr_scaled)
```

7. Visualization

Visualize the clusters and anomalies using various plotting techniques to understand the data distribution and cluster separation.

```
plt.scatter(ctr_pca[:, 0], ctr_pca[:, 1], c=clusters, cmap='viridis', alpha=0.5)
plt.title('K-means Clustering with PCA-Reduced Data')
```

8. Evaluation

Evaluate the performance of the clustering and anomaly detection algorithms using metrics such as silhouette score.

```
sil_score = silhouette_score(ctr_pca, clusters)
print("Silhouette score:", sil_score)
```

This documentation provides an overview of the code structure and the steps involved in the anomaly detection and clustering process. Each section is designed to ensure a thorough understanding and easy navigation through the code.