Clustering Analysis of Credit Card Transactions

1. Main Objective: The primary goal of this analysis is to apply unsupervised learning techniques to the credit card transactions dataset to identify patterns and group similar observations. By clustering transactions, we can gain insights into different types of behaviors, which could potentially help in identifying fraudulent activities. This analysis utilizes clustering models like K-Means, DBSCAN, and dimensionality reduction techniques (PCA, t-SNE) to visually interpret and understand the data.

2. Description of the Dataset: The dataset consists of 284,807 credit card transactions with 31 features:

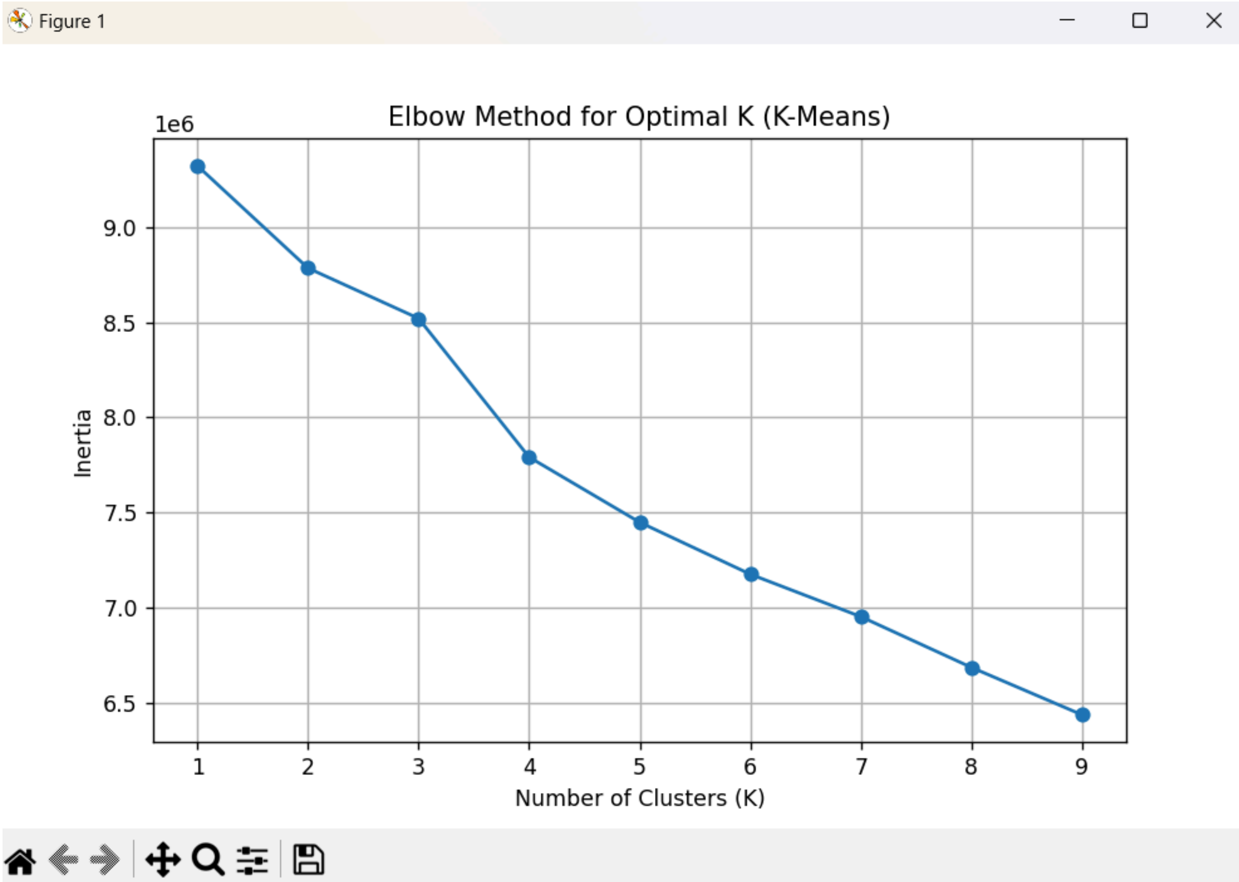
- **Time**: Time elapsed between the transaction and the first transaction.
- **V1 to V28**: Principal Component Analysis (PCA) transformed features.
- **Amount**: The transaction amount.
- **Class**: Target variable where '0' represents legitimate transactions and '1' indicates fraudulent transactions.

The dataset is highly imbalanced, with fraudulent transactions being relatively rare. The analysis focuses on identifying natural groupings without using the `Class` labels.

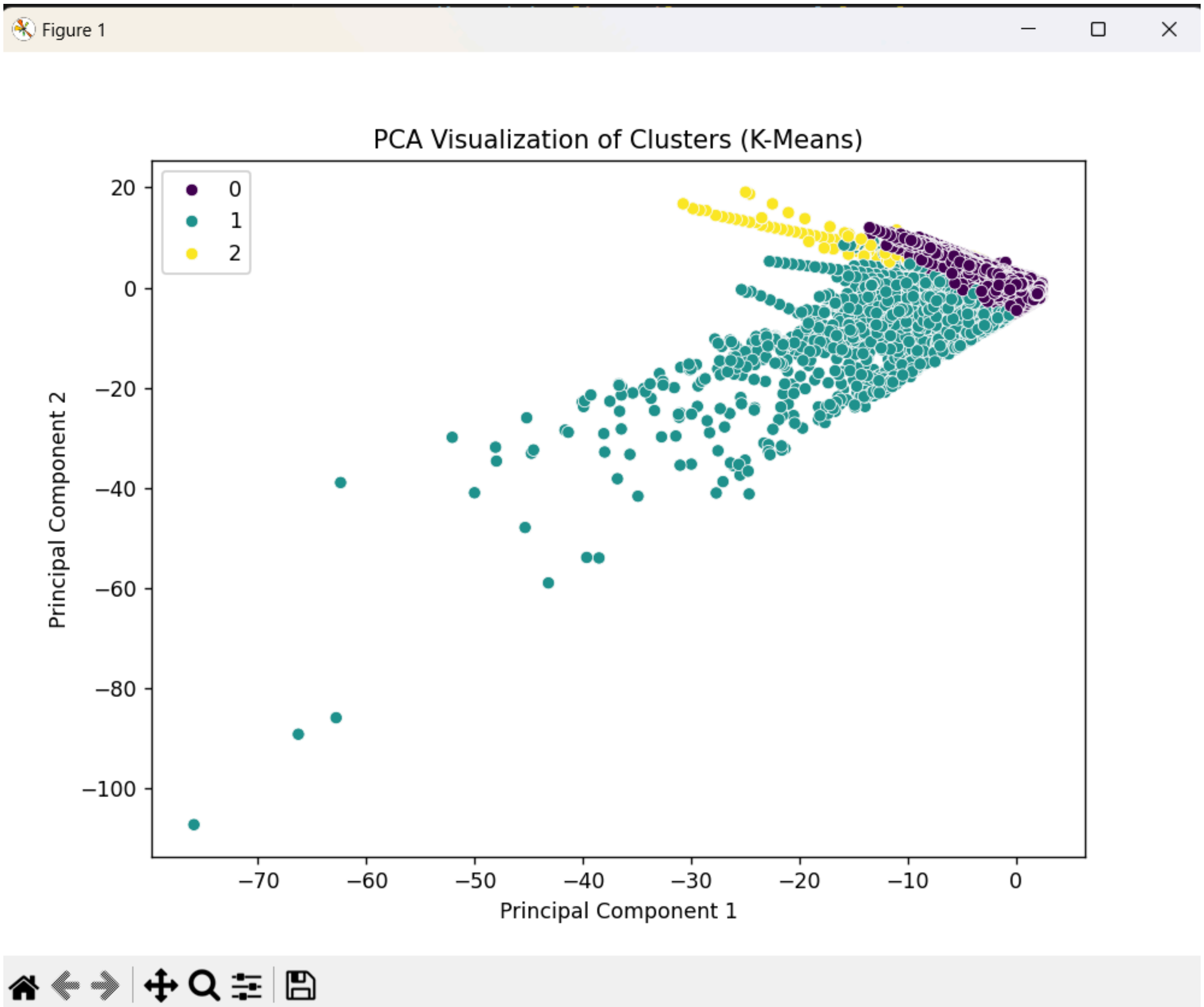
3. Data Exploration and Preprocessing: Initial exploration revealed no missing values. The `Time` and `Amount` features were normalized to ensure they did not bias the clustering process. This step was crucial, as it helped to standardize the data for better performance of clustering algorithms.

4. Models and Analysis: We trained and compared multiple clustering models:

1. **K-Means Clustering:**
    - The elbow method was used to determine the optimal number of clusters. Based on the plot (refer to Figure 1), we selected 3 clusters for the final model.
    - **Results:**
        - Cluster 0: 276,417 transactions
        - Cluster 1: 8,164 transactions
        - Cluster 2: 226 transactions
    - The clustering identified three main groups, showing a distinct separation in behaviors. However, further inspection indicated that the clusters were not as clearly distinguishable as expected, which could be due to the nature of PCA-transformed features.
2. ***Figure 1: Elbow Method for Optimal K (K-Means)*** :

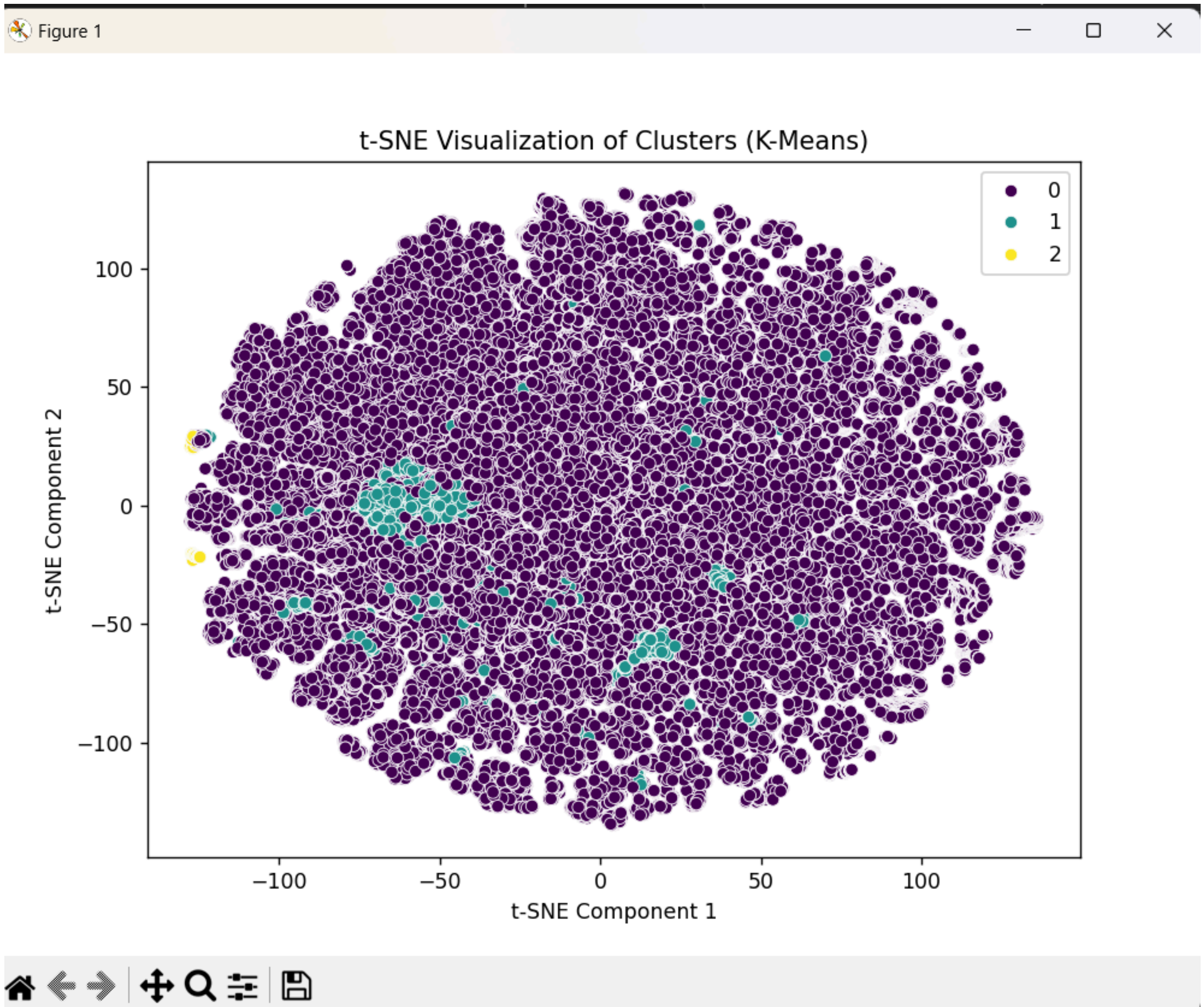


3. **DBSCAN Clustering:**
   - DBSCAN was used to identify clusters without requiring the specification of a set number of clusters.
   - **Results:**
     - The majority of data points (214,666) were labeled as noise, indicated by `-1`.
     - The remaining data was scattered across 2,493 clusters, with varying counts, showing that DBSCAN detected high levels of noise, making it less suitable for this dataset.
   - This model detected noise in the dataset, but clusters were not as well-defined compared to K-Means, making it less suitable for this particular analysis.
4. **Dimensionality Reduction - PCA and t-SNE:**
   - **PCA**: Reduced the dataset to 2 components for visualization. The scatter plot (refer to Figure 2) illustrated that transactions grouped into distinct regions, showing some separation of clusters.
5. ***Figure 2: PCA Visualization of Clusters (K-Means)*** :



- **t-SNE**: Used to further visualize the data. While PCA provided linear separations, t-SNE captured more non-linear patterns, albeit with less clear grouping (refer to Figure 3).

6. ***Figure 3: t-SNE Visualization of Clusters (K-Means)*** *:*



5. Recommended Model: Based on the clustering analysis, the **K-Means model** was selected as the preferred approach. Although the separation between clusters was not entirely distinct, it provided clearer insights compared to DBSCAN. The elbow method also guided us to select an optimal number of clusters, enhancing the interpretability of the results.

6. Key Findings and Insights:

- The clustering showed that there are natural groupings within the transactions, although the distinction between clusters is not very pronounced.
- Visualizations from PCA and t-SNE suggested patterns that could indicate different behaviors in how transactions were processed.
- The analysis highlights the possibility of improving clustering results by enhancing feature engineering or adding more relevant data features.

7. Recommendations and Next Steps:

1. **Additional Feature Engineering**: Consider including more domain-specific features or aggregating transactions over time to provide better inputs for clustering.
2. **Improving Model Parameters**: Fine-tuning hyperparameters of models like DBSCAN or experimenting with other clustering techniques, such as Gaussian Mixture Models, may yield better results.
3. **Combine with Supervised Learning**: Use the clusters obtained to build separate models for fraud detection to understand if different clusters have different risks associated with fraudulent activities.

Conclusion: This project demonstrates the application of unsupervised learning techniques to a real-world dataset, showcasing how clustering and dimensionality reduction can help uncover patterns in data. Although K-Means provided the best results for this analysis, there is potential for further improvement by integrating domain knowledge and refining features.

Terminal Results :

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

```
PS C:\Users\rhyss\OneDrive\Documents\IBM MAchine Learning>  python creditcardproject.py
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
First few rows of the dataset:
   Time        V1        V2        V3        V4        V5        V6  ...       V24       V25       V26       V27       V28  Amount  Class
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  ...  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62      0
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361  ... -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69      0
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  ... -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66      0
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  ... -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50      0
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  ...  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]

Total Missing Values: 0

Summary Statistics:
                Time            V1            V2            V3            V4  ...           V26           V27           V28         Amount            C
lass
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  ...  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000  284807.00
0000
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15  ...  1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619       0.00
1727
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00  ...  4.822270e-01  4.036325e-01  3.300833e-01     250.120109       0.04
1527
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00  ... -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000       0.00
0000
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01  ... -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000       0.00
0000
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02  ... -5.213911e-02  1.342146e-03  1.124383e-02      22.000000       0.00
0000
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01  ...  2.409522e-01  9.104512e-02  7.827995e-02      77.165000       0.00
0000
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01  ...  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000       1.00
0000

[8 rows x 31 columns]

First few rows after normalization:
       Time    Amount
0 -1.996583  0.244964
1 -1.996583 -0.342475
2 -1.996562  1.160686
3 -1.996562  0.140534
4 -1.996541 -0.073403
```

```
K-Means Clustering Results:
KMeans_Cluster
0    276417
1      8164
2       226
Name: count, dtype: int64

DBSCAN Clustering Results:
DBSCAN_Cluster
-1       214666
 1337      1638
 1334      1186
 1341      1160
 1472      1099
          ...
 738          3
 2254         3
 852          3
 1253         2
 2050         1
Name: count, Length: 2493, dtype: int64

Final Insights and Recommendations:
1. The optimal number of clusters was determined using the elbow method.
2. K-Means and DBSCAN clusters were created, and patterns observed in PCA and t-SNE visualizations.
3. Further analysis may include adding additional features, fine-tuning clustering parameters, or testing other clustering algorithms.
PS C:\Users\rhyss\OneDrive\Documents\IBM MAchine Learning>
```

Complete code :

```python
# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import seaborn as sns

# Load the dataset
file_path = 'creditcard.csv'
creditcard_data = pd.read_csv(file_path)

# Display basic dataset information
print("Dataset Info:")
creditcard_data.info()

print("\nFirst few rows of the dataset:")
print(creditcard_data.head())
```

```python
# Initial data exploration: Check for missing values and basic statistical
summary
missing_values = creditcard_data.isnull().sum().sum()
summary_statistics = creditcard_data.describe()

print("\nTotal Missing Values:", missing_values)
print("\nSummary Statistics:")
print(summary_statistics)

# Normalizing 'Amount' and 'Time' features
scaler = StandardScaler()
creditcard_data[['Amount', 'Time']] =
scaler.fit_transform(creditcard_data[['Amount', 'Time']])

# Displaying the first few rows after normalization
print("\nFirst few rows after normalization:")
print(creditcard_data[['Time', 'Amount']].head())

# Define features for clustering (excluding 'Class' for unsupervised learning)
features = creditcard_data.drop(columns=['Class'])

# Run K-Means with a range of cluster values to determine the optimal number
using the elbow method
inertia = []
k_values = range(1, 10)
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(features)
    inertia.append(kmeans.inertia_)

# Plotting the elbow curve
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o')
plt.title('Elbow Method for Optimal K (K-Means)')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()
```

```python
# Applying K-Means with the optimal number of clusters
optimal_k = 3  # Example value, adjust based on the elbow method output
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
creditcard_data['KMeans_Cluster'] = kmeans.fit_predict(features)

# Summary of K-Means clusters
print("\nK-Means Clustering Results:")
print(creditcard_data['KMeans_Cluster'].value_counts())

# Applying DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
creditcard_data['DBSCAN_Cluster'] = dbscan.fit_predict(features)

# Summary of DBSCAN clusters
print("\nDBSCAN Clustering Results:")
print(creditcard_data['DBSCAN_Cluster'].value_counts())

# Applying PCA for dimensionality reduction (for visualization)
pca = PCA(n_components=2)
pca_result = pca.fit_transform(features)
creditcard_data['PCA1'] = pca_result[:, 0]
creditcard_data['PCA2'] = pca_result[:, 1]

# Scatter plot of PCA results with K-Means clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x='PCA1', y='PCA2', hue='KMeans_Cluster', data=creditcard_data,
palette='viridis')
plt.title('PCA Visualization of Clusters (K-Means)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()

# Applying t-SNE for visualization
tsne = TSNE(n_components=2, random_state=42)
tsne_result = tsne.fit_transform(features)
creditcard_data['TSNE1'] = tsne_result[:, 0]
creditcard_data['TSNE2'] = tsne_result[:, 1]
```

```
# Scatter plot of t-SNE results with K-Means clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x='TSNE1', y='TSNE2', hue='KMeans_Cluster', data=creditcard_data,
palette='viridis')
plt.title('t-SNE Visualization of Clusters (K-Means)')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.legend()
plt.show()

# Final Insights and Next Steps
print("\nFinal Insights and Recommendations:")
print("1. The optimal number of clusters was determined using the elbow method.")
print("2. K-Means and DBSCAN clusters were created, and patterns observed in PCA
and t-SNE visualizations.")
print("3. Further analysis may include adding additional features, fine-tuning
clustering parameters, or testing other clustering algorithms.")

# python creditcardproject.py
```