Lab 10: Clustering (K Means)

R Abhijit Srivathsan
2448044

In [1]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
```

In [2]:
```
df = pd.read_csv("credit_card_customer_data.csv")
df.head()
```

Out[2]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1	87073	100000	2	1	1	0
1	2	38414	50000	3	0	10	9
2	3	17341	50000	7	1	3	4
3	4	40496	30000	5	1	1	4
4	5	47437	100000	6	0	12	3

In [3]:
```
df.shape
```

Out[3]:
```
(660, 7)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sl_No                660 non-null    int64
 1   Customer Key         660 non-null    int64
 2   Avg_Credit_Limit     660 non-null    int64
 3   Total_Credit_Cards   660 non-null    int64
 4   Total_visits_bank    660 non-null    int64
 5   Total_visits_online  660 non-null    int64
 6   Total_calls_made     660 non-null    int64
dtypes: int64(7)
memory usage: 36.2 KB
```

No null values present

Dropping unnecessary features

In [5]:
```
features = df.drop(columns = ["Sl_No", "Customer Key"])
```

Standardizing the features

In [6]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
```

Finding optimal number of clusters

In [7]:
```
inertia = []
silhouette_scores = []
k_range = range(2, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
```

```
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_scaled, kmeans.labels_))
```

```
c:\Users\abhijit\AppData\Local\Programs\Python\Python313\Lib\site-packages\joblib\externals\loky\backend\context.py:136: UserWa
rning: Could not find the number of physical cores for the following reason:
[WinError 2] The system cannot find the file specified
Returning the number of logical cores instead. You can silence this warning by setting LOKY_MAX_CPU_COUNT to the number of core
s you want to use.
  warnings.warn(
  File "c:\Users\abhijit\AppData\Local\Programs\Python\Python313\Lib\site-packages\joblib\externals\loky\backend\context.py", l
ine 257, in _count_physical_cores
    cpu_info = subprocess.run(
        "wmic CPU Get NumberOfCores /Format:csv".split(),
        capture_output=True,
        text=True,
    )
  File "c:\Users\abhijit\AppData\Local\Programs\Python\Python313\Lib\subprocess.py", line 556, in run
    with Popen(*popenargs, **kwargs) as process:
         ~~~~~^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\abhijit\AppData\Local\Programs\Python\Python313\Lib\subprocess.py", line 1038, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
    ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                        pass_fds, cwd, env,
                        ^^^^^^^^^^^^^^^^^^^
    ...<5 lines>...
                        gid, gids, uid, umask,
                        ^^^^^^^^^^^^^^^^^^^^^^
                        start_new_session, process_group)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\abhijit\AppData\Local\Programs\Python\Python313\Lib\subprocess.py", line 1550, in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^
                            # no special security
                            ^^^^^^^^^^^^^^^^^^^^^
    ...<4 lines>...
                            cwd,
                            ^^^^
                            startupinfo)
                            ^^^^^^^^^^^^
```

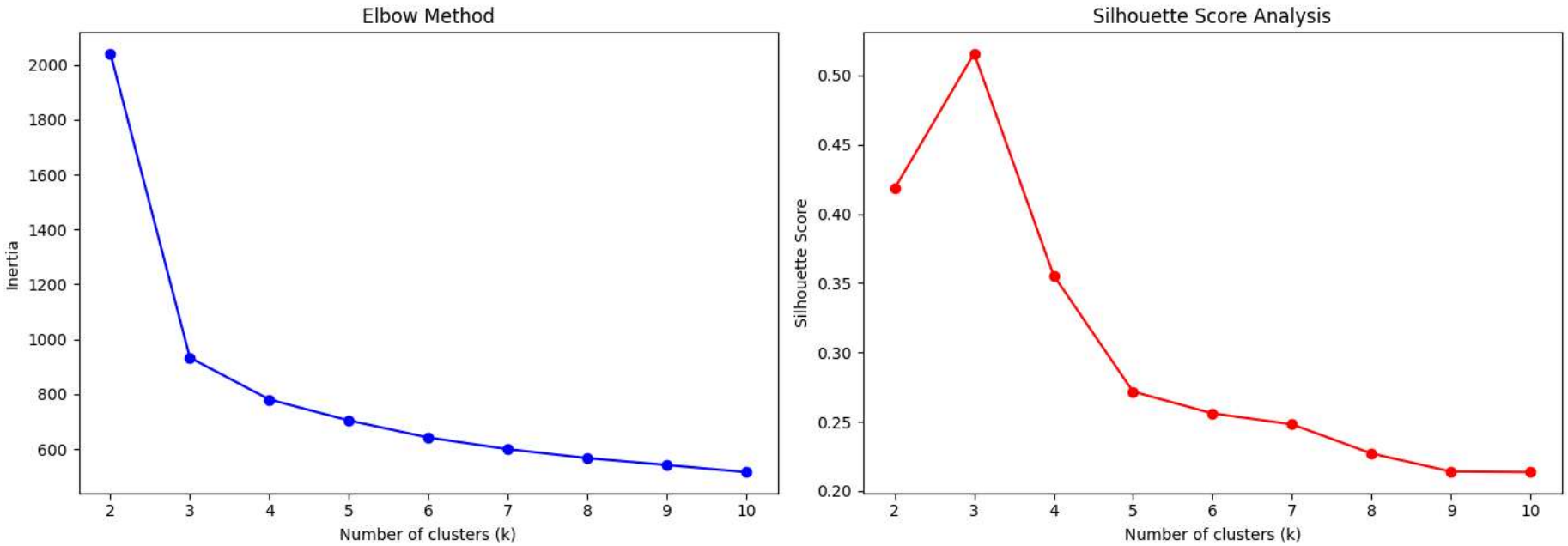
Plot Elbow Method and Silhouette Score

In [8]:
```
plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
plt.plot(k_range, inertia, 'bo-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method')

plt.subplot(1, 2, 2)
plt.plot(k_range, silhouette_scores, 'ro-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score Analysis')

plt.tight_layout()
plt.show()
```



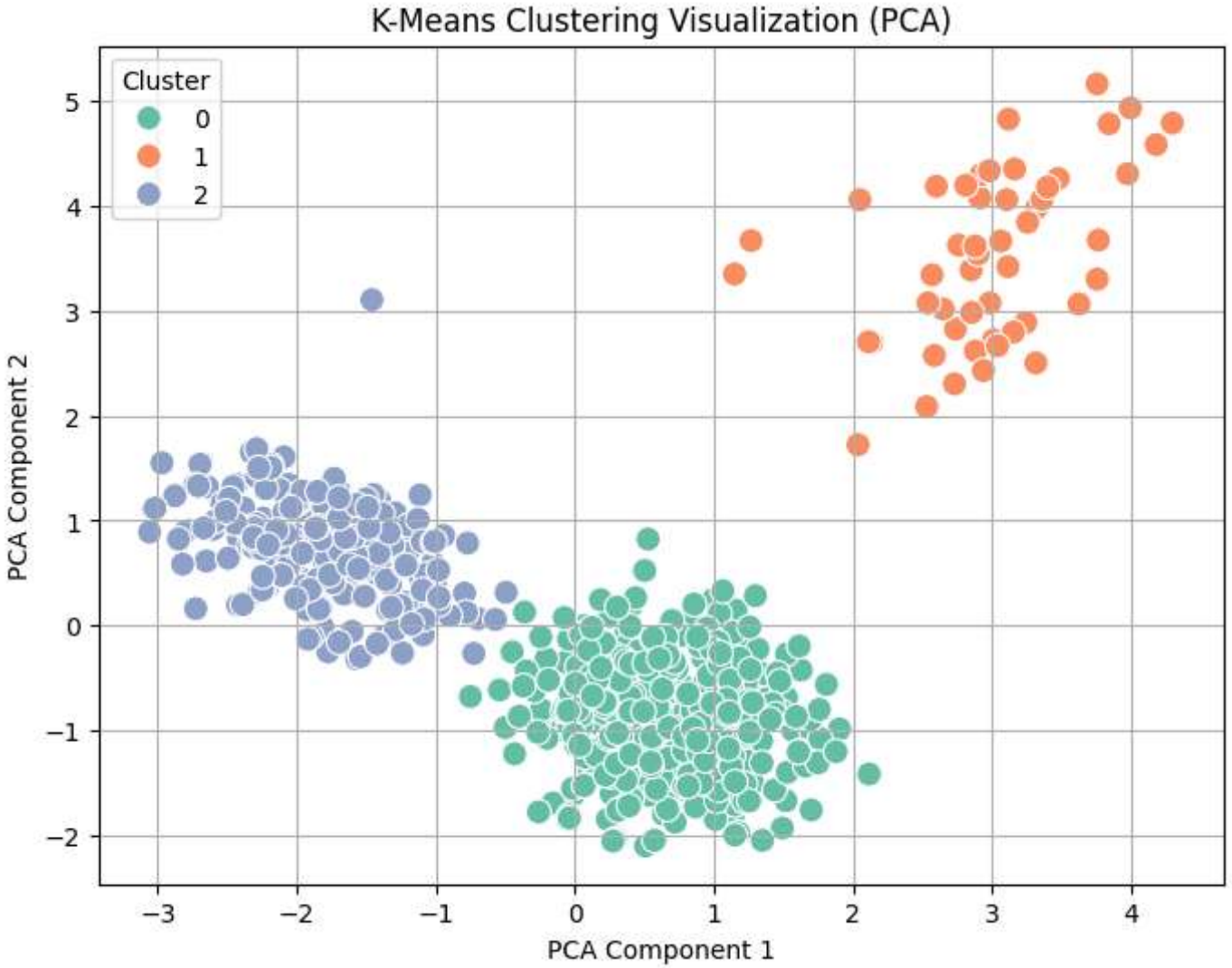
In [9]:
```
optimal_k = 3
kmeans_final = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
df['Cluster'] = kmeans_final.fit_predict(X_scaled)
```

In [10]:
```
final_silhouette = silhouette_score(X_scaled, df['Cluster'])
print(f"Final Silhouette Score: {final_silhouette:.2f}")
```

```
Final Silhouette Score: 0.52
```

In [11]:
```
pca = PCA(n_components=2)
pca_components = pca.fit_transform(X_scaled)
df['PCA1'] = pca_components[:, 0]
df['PCA2'] = pca_components[:, 1]
```

In [12]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PCA1', y='PCA2', hue='Cluster', palette='Set2', s=100)
plt.title('K-Means Clustering Visualization (PCA)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



Final Inference

After applying K-Means clustering on the credit card customer dataset, the following conclusions were drawn:

- The **optimal number of clusters** was determined to be **3**, based on:

    - The **highest Silhouette Score** at $k = 3$ (Score = **0.52**), which indicates well-separated and dense clusters.
    - The **Elbow Method**, which showed a noticeable "elbow" or drop in inertia at $k = 3$, supporting the same choice.
- The **Silhouette Score of 0.52** suggests that the clusters formed are **reasonably well-defined** and the separation between them is meaningful.

- A **PCA-based 2D visualization** confirmed the clustering quality, showing three distinct groupings with minimal overlap, reinforcing the effectiveness of clustering on the scaled features.