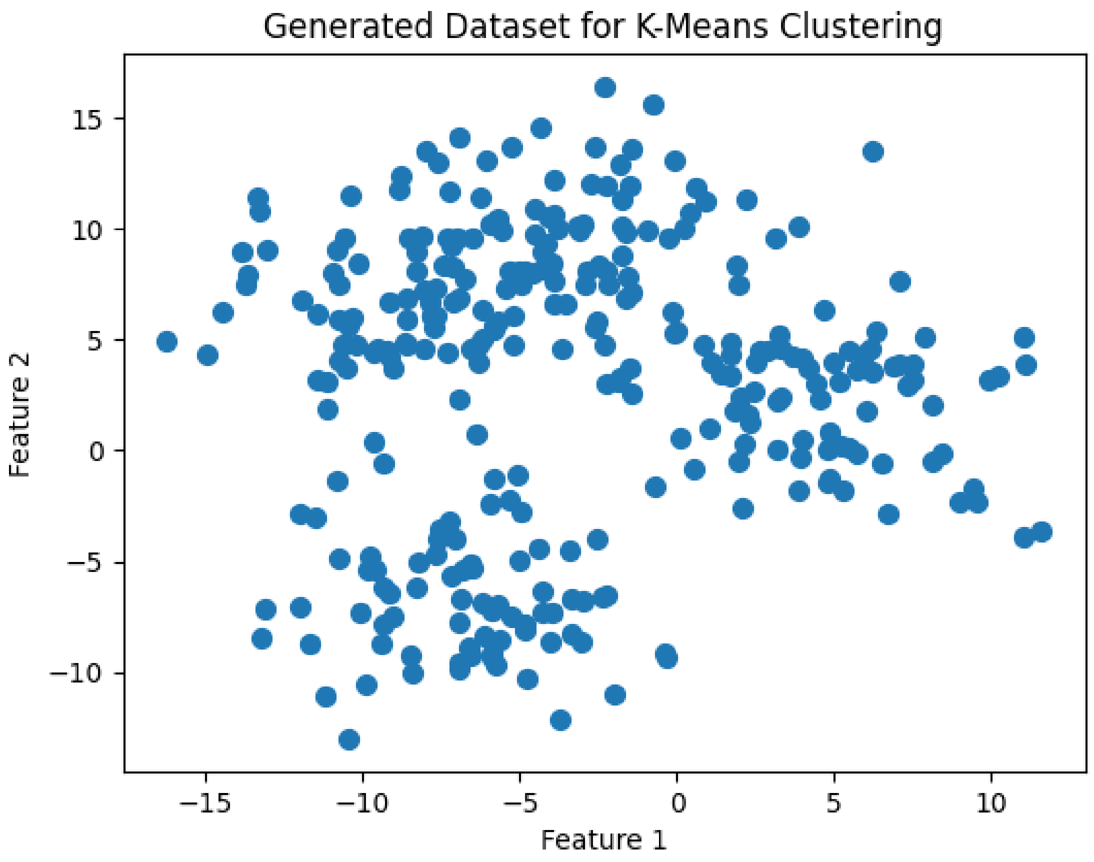
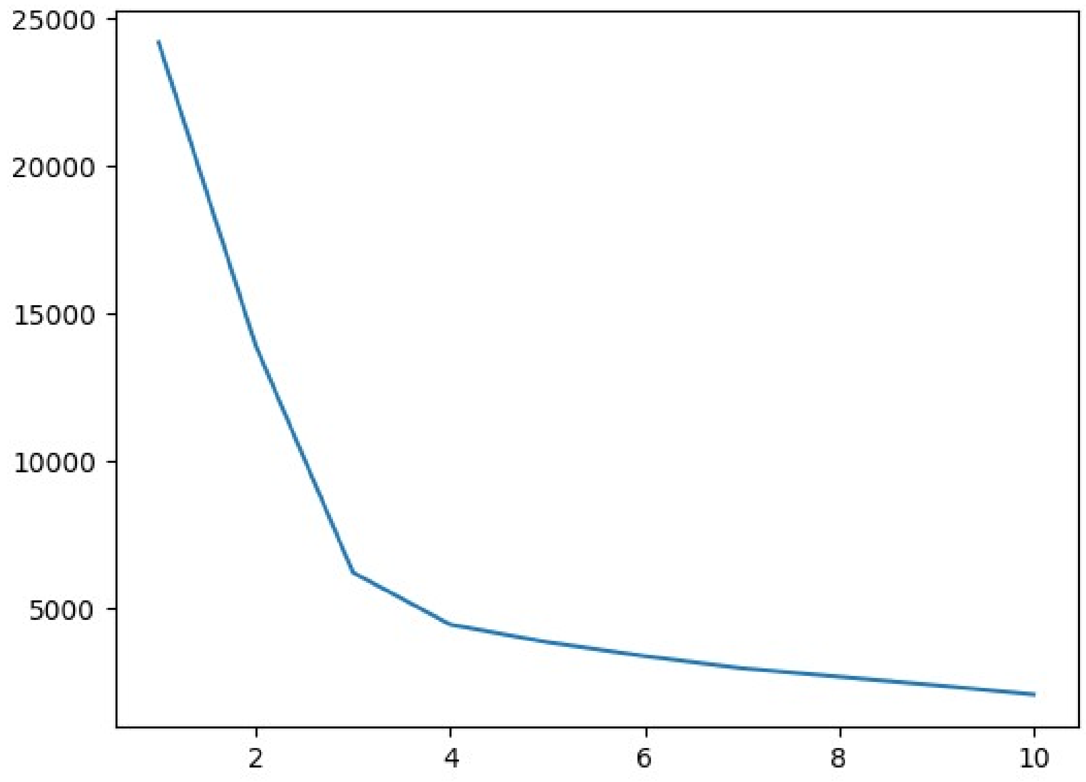
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# Parameters
n_samples = 300
n_features = 2
centers = 4

# Generate data
X, y = make_blobs(n_samples=n_samples, n_features=n_features,
centers=centers, cluster_std=3, random_state=42)

# Visualize the data
plt.scatter(X[:, 0], X[:, 1], s=50, cmap='viridis')
plt.title("Generated Dataset for K-Means Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

```
<ipython-input-11-be17a401e8b3>:14: UserWarning: No data for
colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(X[:, 0], X[:, 1], s=50, cmap='viridis')
```



```python
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_init = 10, n_clusters = i)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
```

```
[<matplotlib.lines.Line2D at 0x7a318b471990>]
```

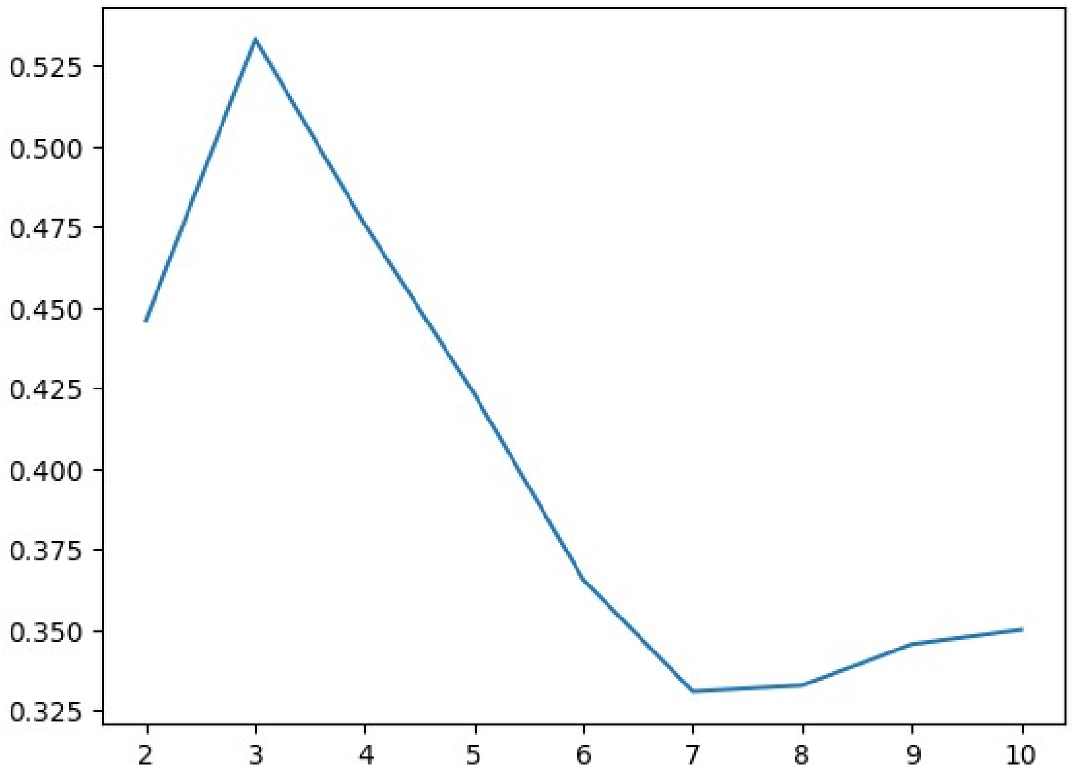


```python
from sklearn.metrics import silhouette_score
sil = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    sil.append(silhouette_avg)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```

```
plt.plot(range(2,11),sil)
```

```
[<matplotlib.lines.Line2D at 0x794a43de8520>]
```



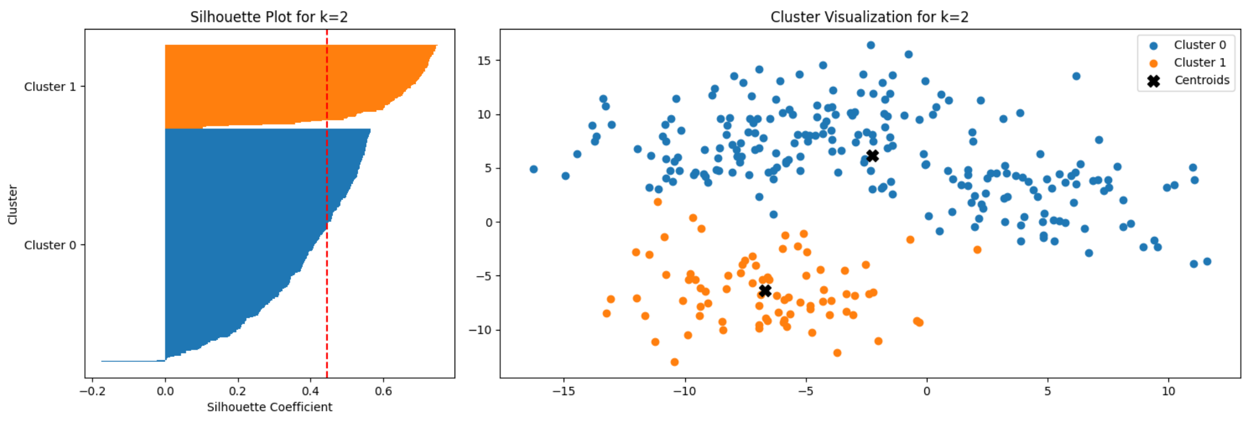
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_init = 10, n_clusters = 4, verbose=100)
kmeans.fit(X)
```

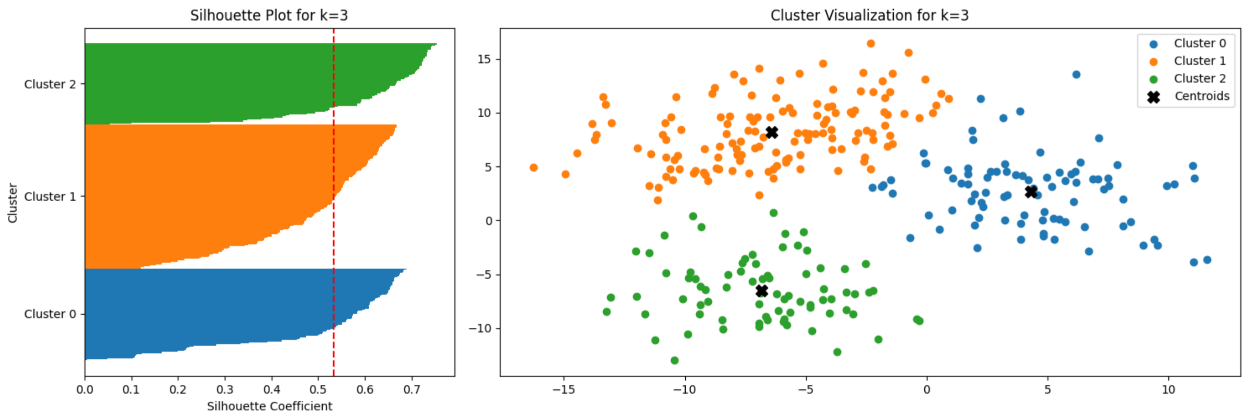
```
Initialization complete
Iteration 0, inertia 6473.033808617001.
Iteration 1, inertia 4529.970450579422.
Iteration 2, inertia 4485.261793377747.
Iteration 3, inertia 4476.280950769462.
Iteration 4, inertia 4475.918395021896.
Converged at iteration 4: strict convergence.
Initialization complete
Iteration 0, inertia 7269.787782262799.
Iteration 1, inertia 4937.615912478641.
Iteration 2, inertia 4675.371687906602.
Iteration 3, inertia 4565.51698217038.
Iteration 4, inertia 4521.120738498696.
Iteration 5, inertia 4492.562586381868.
Iteration 6, inertia 4477.333098866022.
Iteration 7, inertia 4475.918395021896.
Converged at iteration 7: strict convergence.
Initialization complete
Iteration 0, inertia 5833.552054345968.
Iteration 1, inertia 4720.636414164014.
Iteration 2, inertia 4580.389037563589.
Iteration 3, inertia 4521.120738498696.
Iteration 4, inertia 4492.562586381868.
Iteration 5, inertia 4477.333098866022.
Iteration 6, inertia 4475.918395021896.
Converged at iteration 6: strict convergence.
Initialization complete
Iteration 0, inertia 6083.281519295091.
Iteration 1, inertia 4553.354165146864.
Iteration 2, inertia 4483.2714625758035.
Iteration 3, inertia 4476.36716823383.
Iteration 4, inertia 4475.918395021896.
Converged at iteration 4: strict convergence.
Initialization complete
Iteration 0, inertia 7003.664230558079.
Iteration 1, inertia 4626.839075431153.
Iteration 2, inertia 4536.927791749251.
Iteration 3, inertia 4491.34516060812.
Iteration 4, inertia 4476.367348636015.
Iteration 5, inertia 4475.918395021896.
Converged at iteration 5: strict convergence.
Initialization complete
Iteration 0, inertia 7382.997930380328.
Iteration 1, inertia 4751.065888535701.
Iteration 2, inertia 4550.497824148024.
Iteration 3, inertia 4486.810184425599.
Iteration 4, inertia 4476.36716823383.
Iteration 5, inertia 4475.918395021896.
Converged at iteration 5: strict convergence.
Initialization complete
Iteration 0, inertia 6301.024915065829.
Iteration 1, inertia 4834.595082474243.
Iteration 2, inertia 4605.669752000156.
Iteration 3, inertia 4513.691604111213.
Iteration 4, inertia 4493.169438933145.
Iteration 5, inertia 4490.395440330573.
Iteration 6, inertia 4481.523897930505.
Iteration 7, inertia 4475.918395021896.
Converged at iteration 7: strict convergence.
Initialization complete
Iteration 0, inertia 10324.877715586237.
Iteration 1, inertia 5545.738923007804.
Iteration 2, inertia 4564.775489217898.
Iteration 3, inertia 4494.139925923097.
Iteration 4, inertia 4479.174755841133.
Iteration 5, inertia 4475.918395021896.
Converged at iteration 5: strict convergence.
Initialization complete
Iteration 0, inertia 10311.25450048352.
Iteration 1, inertia 5595.313434712934.
Iteration 2, inertia 4834.361994274447.
Iteration 3, inertia 4595.2821882189955.
Iteration 4, inertia 4532.773186808699.
Iteration 5, inertia 4490.3655725953595.
Iteration 6, inertia 4485.7464721156575.
Iteration 7, inertia 4483.070290877196.
Iteration 8, inertia 4477.333098866022.
Iteration 9, inertia 4475.918395021896.
Converged at iteration 9: strict convergence.
Initialization complete
Iteration 0, inertia 7102.418689410471.
Iteration 1, inertia 5789.900278754151.
Iteration 2, inertia 5770.342241391474.
Iteration 3, inertia 5741.309771920889.
Iteration 4, inertia 5733.446314730845.
Iteration 5, inertia 5718.509444050268.
Iteration 6, inertia 5688.099194404587.
Iteration 7, inertia 5647.0220216610605.
Iteration 8, inertia 5593.313991207889.
Iteration 9, inertia 5534.560392359043.
Iteration 10, inertia 5446.532041767841.
Iteration 11, inertia 5367.91811388054.
Iteration 12, inertia 5302.177882198167.
Iteration 13, inertia 5207.836877496525.
Iteration 14, inertia 4975.015301046065.
Iteration 15, inertia 4787.974915797297.
Iteration 16, inertia 4585.36055780838.
Iteration 17, inertia 4532.773186808699.
Iteration 18, inertia 4490.3655725953595.
Iteration 19, inertia 4485.7464721156575.
Iteration 20, inertia 4483.070290877196.
Iteration 21, inertia 4477.333098866022.
Iteration 22, inertia 4475.918395021896.
Converged at iteration 22: strict convergence.
```

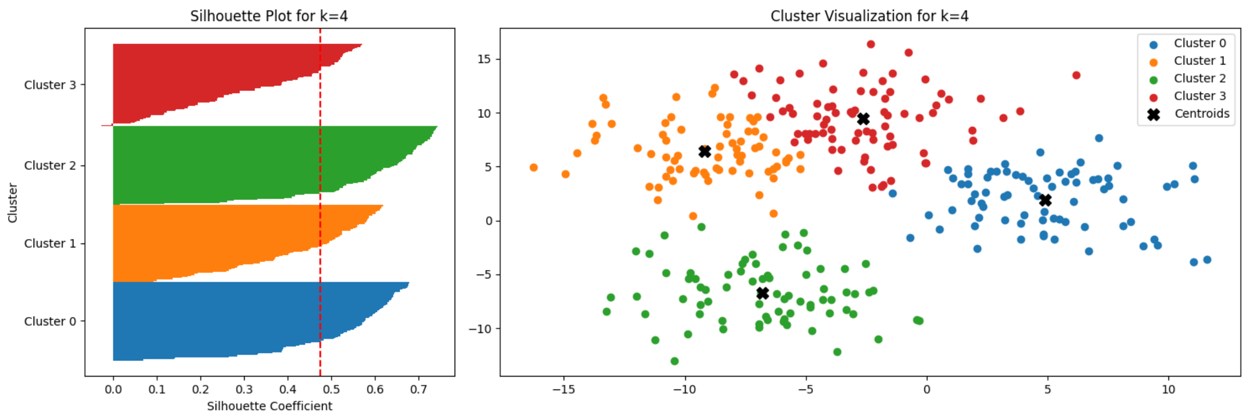
```
KMeans(n_clusters=4, n_init=10, verbose=100)
```

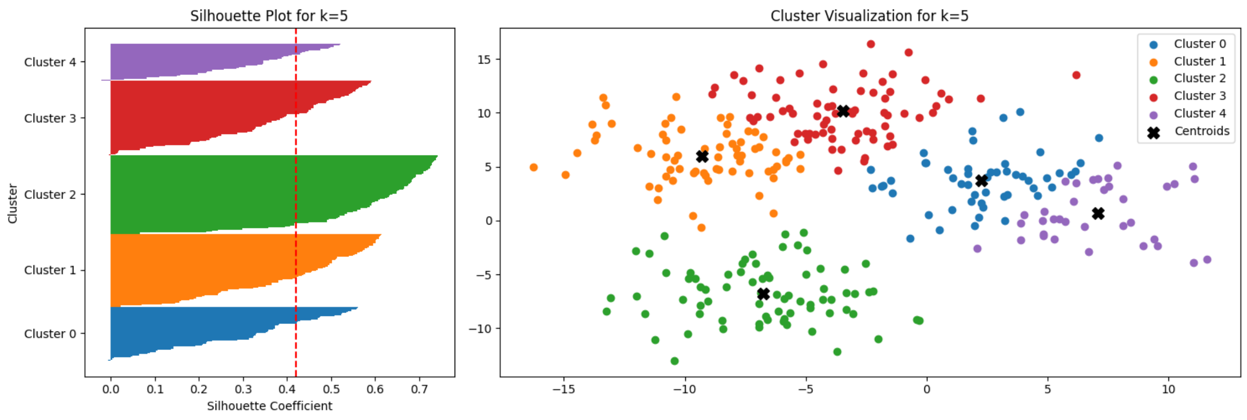
```python
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from sklearn.datasets import make_blobs

from sklearn.metrics import silhouette_samples, silhouette_score
import numpy as np


# Setting up the subplot grid
fig = plt.figure(figsize=(15, 20))
gs = gridspec.GridSpec(4, 2, width_ratios=[1, 2])
```

```python
# Creating silhouette and scatter plots for different values of k (2,
3, 4, 5)
for i, k in enumerate([2, 3, 4, 5]):
    # Clustering with k-means
    kmeans = KMeans(n_init=10, n_clusters=k, random_state=10)
    cluster_labels = kmeans.fit_predict(X)

    # Silhouette values
    silhouette_vals = silhouette_samples(X, cluster_labels)
    avg_score = silhouette_score(X, cluster_labels)

    # Silhouette plot
    ax1 = plt.subplot(gs[i, 0])
    y_lower, y_upper = 0, 0
    yticks = []
    for j, cluster in enumerate(np.unique(cluster_labels)):
        cluster_silhouette_vals = silhouette_vals[cluster_labels ==
cluster]
        cluster_silhouette_vals.sort()
        y_upper += len(cluster_silhouette_vals)
        ax1.barh(range(y_lower, y_upper), cluster_silhouette_vals,
edgecolor='none', height=1)
        yticks.append((y_lower + y_upper) / 2)
        y_lower += len(cluster_silhouette_vals)
    ax1.axvline(avg_score, color="red", linestyle="--")
    ax1.set_yticks(yticks)
    ax1.set_yticklabels([f'Cluster {x}' for x in
np.unique(cluster_labels)])
    ax1.set_ylabel('Cluster')
    ax1.set_xlabel('Silhouette Coefficient')
    ax1.set_title(f'Silhouette Plot for k={k}')

    # Scatter plot
    ax2 = plt.subplot(gs[i, 1])
    for j in np.unique(cluster_labels):
        ax2.scatter(X[cluster_labels == j, 0], X[cluster_labels == j,
1], label=f'Cluster {j}')
    ax2.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], s=100, c='black', marker='X',
label='Centroids')
    ax2.set_title(f'Cluster Visualization for k={k}')
    ax2.legend()

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

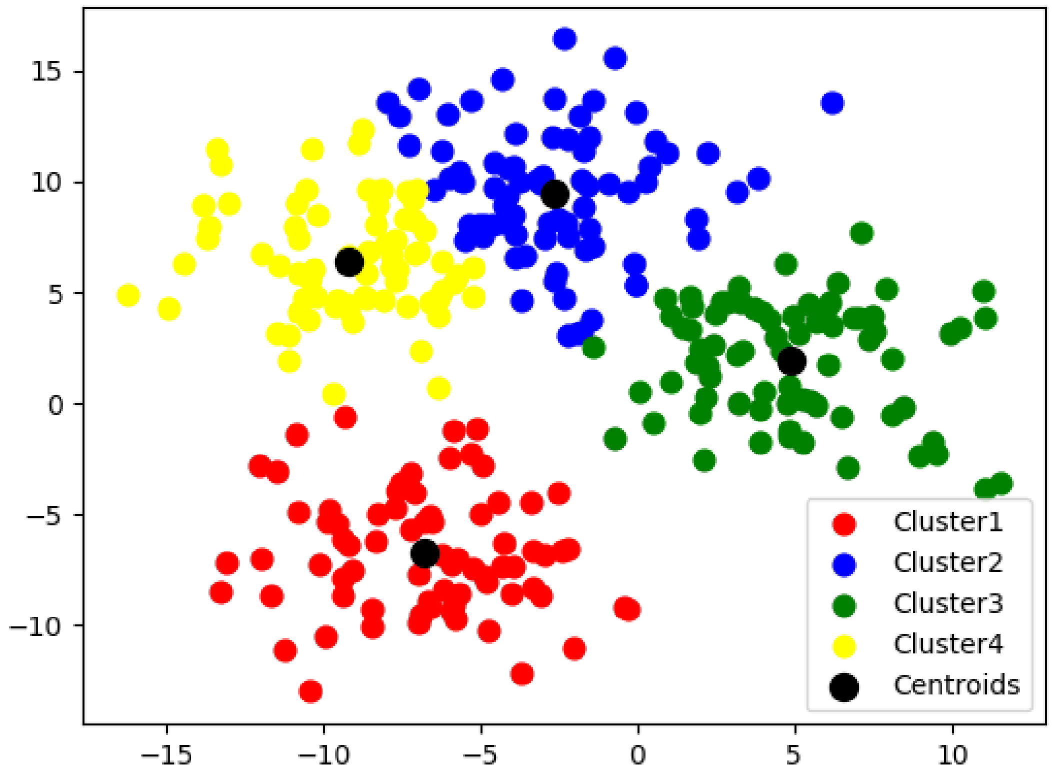

Silhouette Plot for k=2
Cluster 1
Cluster 0
Cluster
−0.2
0.0
0.2
0.4
0.6
Silhouette Coefficient
Cluster Visualization for k=2
Cluster 0
Cluster 1
Centroids
15
10
5
0
−5
−10
−15
−10
−5
0
5
10


Silhouette Plot for k=3
Cluster 2
Cluster 1
Cluster 0
Cluster
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Silhouette Coefficient
Cluster Visualization for k=3
Cluster 0
Cluster 1
Cluster 2
Centroids
15
10
5
0
−5
−10
−15
−10
−5
0
5
10

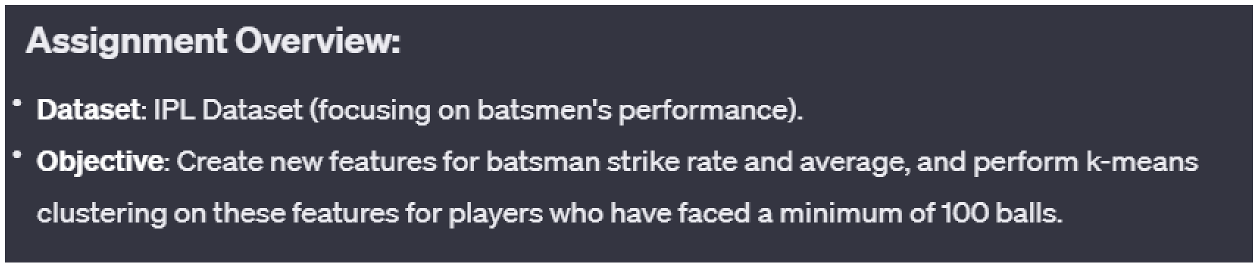

Silhouette Plot for k=4
Cluster 3
Cluster 2
Cluster 1
Cluster 0
Cluster
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Silhouette Coefficient
Cluster Visualization for k=4
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Centroids
15
10
5
0
−5
−10
−15
−10
−5
0
5
10


Silhouette Plot for k=5
Cluster 4
Cluster 3
Cluster 2
Cluster 1
Cluster 0
Cluster
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Silhouette Coefficient
Cluster Visualization for k=5
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroids
15
10
5
0
−5
−10
−15
−10
−5
0
5
10

```
y_kmeans = kmeans.fit_predict(X)

y_kmeans

array([3, 3, 1, 0, 3, 0, 2, 0, 1, 2, 1, 2, 1, 3, 3, 1, 3, 2, 1, 1, 2,
1,
       0, 3, 1, 1, 3, 0, 0, 2, 1, 2, 3, 2, 3, 1, 3, 0, 3, 0, 2, 1, 3,
0,
       1, 1, 3, 2, 3, 2, 0, 3, 0, 1, 0, 1, 3, 2, 2, 1, 3, 2, 2, 1, 0,
0,
       0, 0, 0, 1, 0, 0, 3, 2, 1, 3, 0, 0, 1, 0, 1, 1, 3, 1, 0, 3, 1,
2,
       2, 2, 3, 1, 3, 1, 1, 3, 0, 1, 3, 1, 2, 2, 2, 1, 1, 1, 1, 1, 0,
3,
       2, 1, 1, 1, 1, 2, 3, 0, 3, 0, 0, 0, 2, 3, 0, 3, 3, 1, 3, 0, 2,
3,
       1, 3, 3, 2, 2, 3, 1, 0, 1, 2, 2, 1, 2, 2, 2, 2, 0, 1, 3, 1, 2,
0,
       1, 2, 0, 3, 3, 2, 3, 3, 0, 3, 2, 3, 0, 1, 1, 1, 1, 1, 0, 2, 2,
0,
       0, 1, 2, 0, 3, 1, 3, 2, 2, 3, 0, 1, 0, 2, 0, 0, 0, 3, 2, 0, 0,
2,
       2, 3, 1, 1, 0, 2, 1, 0, 0, 3, 0, 1, 1, 0, 0, 2, 3, 0, 1, 3, 3,
3,
       3, 0, 3, 0, 2, 2, 3, 1, 2, 2, 2, 1, 1, 0, 2, 0, 3, 2, 1, 3, 1,
0,
       0, 2, 1, 0, 0, 0, 3, 0, 3, 0, 3, 2, 0, 3, 2, 1, 3, 1, 2, 1, 3,
3,
       0, 2, 0, 2, 2, 1, 1, 0, 2, 2, 3, 3, 0, 3, 1, 2, 2, 2, 2, 0, 3,
2,
       0, 2, 2, 0, 1, 0, 2, 1, 3, 1, 2, 3, 3, 1], dtype=int32)

plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 60, c =
'red', label = 'Cluster1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 60, c =
'blue', label = 'Cluster2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 60, c =
'green', label = 'Cluster3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 60, c =
'yellow', label = 'Cluster4')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 100, c = 'black', label = 'Centroids')

plt.legend()

plt.show()
```

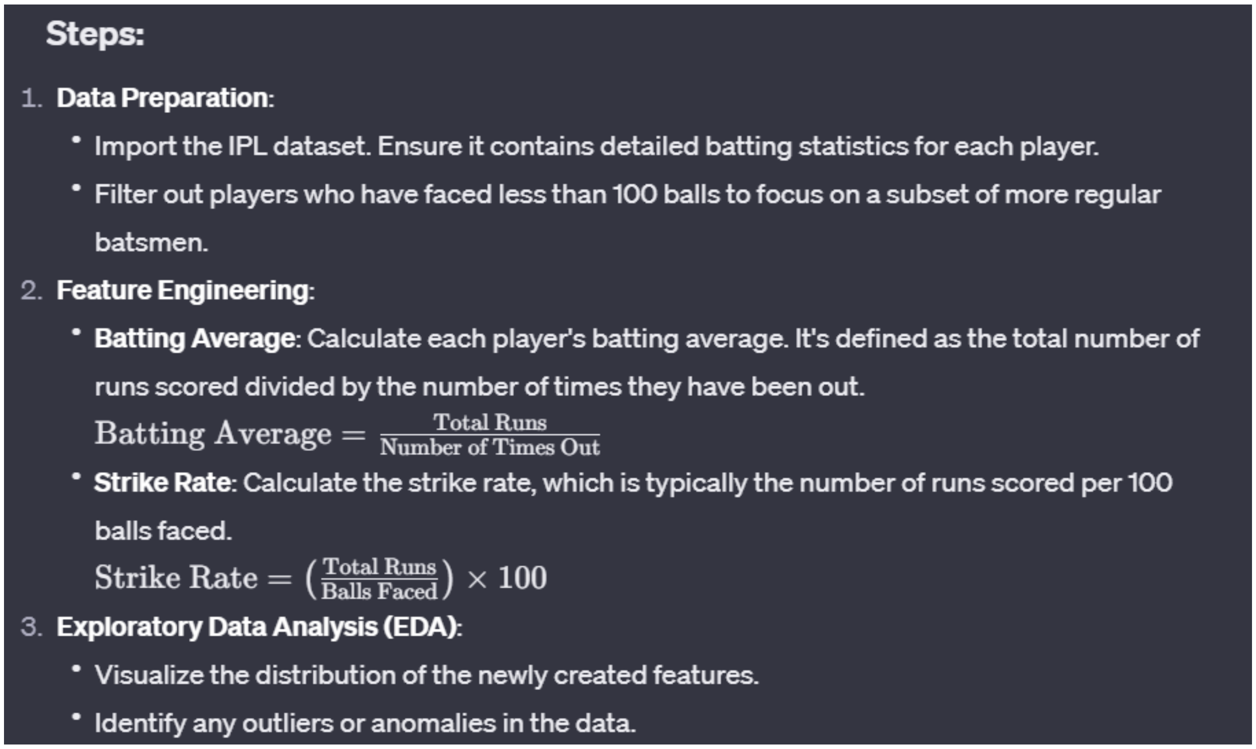


Assignment



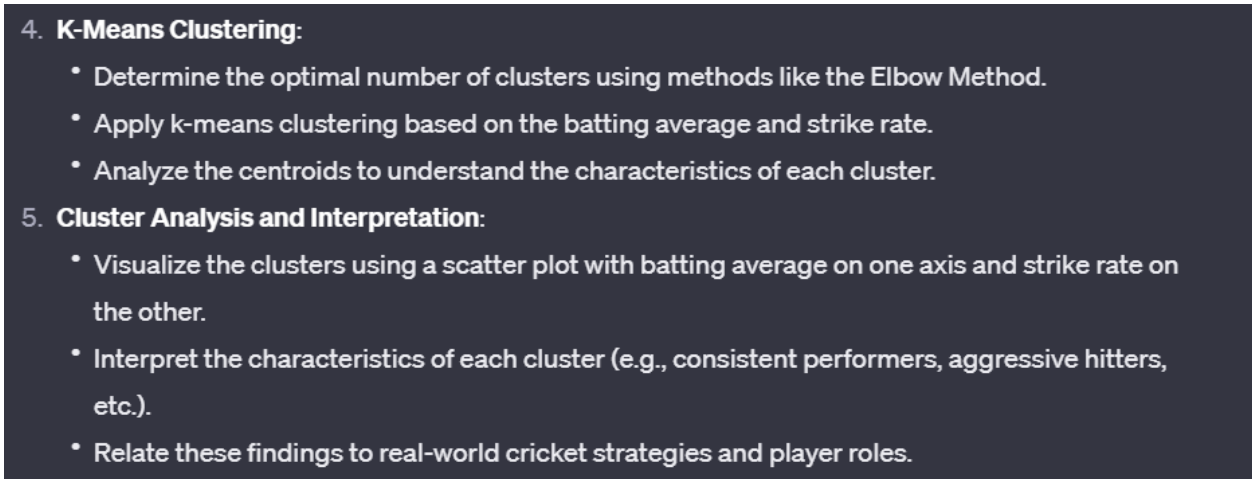
Assignment Overview:

- **Dataset**: IPL Dataset (focusing on batsmen's performance).
- **Objective**: Create new features for batsman strike rate and average, and perform k-means clustering on these features for players who have faced a minimum of 100 balls.



Steps:

1. **Data Preparation**:
   - Import the IPL dataset. Ensure it contains detailed batting statistics for each player.
   - Filter out players who have faced less than 100 balls to focus on a subset of more regular batsmen.
2. **Feature Engineering**:
   - **Batting Average**: Calculate each player's batting average. It's defined as the total number of runs scored divided by the number of times they have been out.
     $\text{Batting Average} = \frac{\text{Total Runs}}{\text{Number of Times Out}}$
   - **Strike Rate**: Calculate the strike rate, which is typically the number of runs scored per 100 balls faced.
     $\text{Strike Rate} = \left(\frac{\text{Total Runs}}{\text{Balls Faced}}\right) \times 100$
3. **Exploratory Data Analysis (EDA)**:
   - Visualize the distribution of the newly created features.
   - Identify any outliers or anomalies in the data.



4. **K-Means Clustering**:
   - Determine the optimal number of clusters using methods like the Elbow Method.
   - Apply k-means clustering based on the batting average and strike rate.
   - Analyze the centroids to understand the characteristics of each cluster.
5. **Cluster Analysis and Interpretation**:
   - Visualize the clusters using a scatter plot with batting average on one axis and strike rate on the other.
   - Interpret the characteristics of each cluster (e.g., consistent performers, aggressive hitters, etc.).
   - Relate these findings to real-world cricket strategies and player roles.

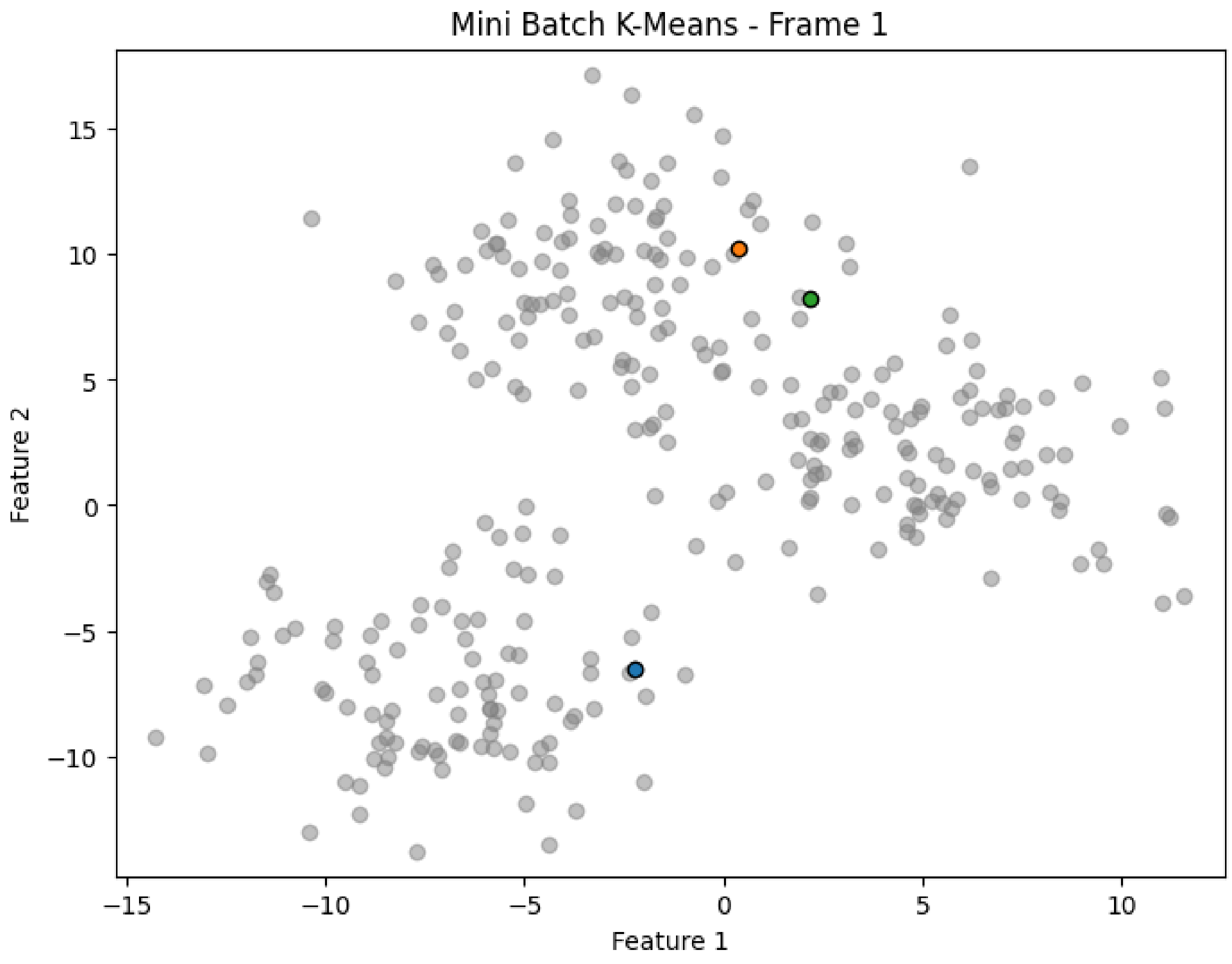
```python
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import time

# Generating synthetic data for clustering
n_samples = 100000
n_features = 10
n_clusters = 5
X, _ = make_blobs(n_samples=n_samples, n_features=n_features,
centers=n_clusters)

# Function to measure time taken by KMeans using different algorithms
```

```python
def measure_kmeans_time(algorithm):
    kmeans = KMeans(n_clusters=n_clusters, algorithm=algorithm,
random_state=0)
    start_time = time.time()
    kmeans.fit(X)
    return time.time() - start_time

# Measuring time for Lloyd's algorithm
lloyd_time = measure_kmeans_time('full')

# Measuring time for Elkan's algorithm
elkan_time = measure_kmeans_time('elkan')

lloyd_time, elkan_time
```

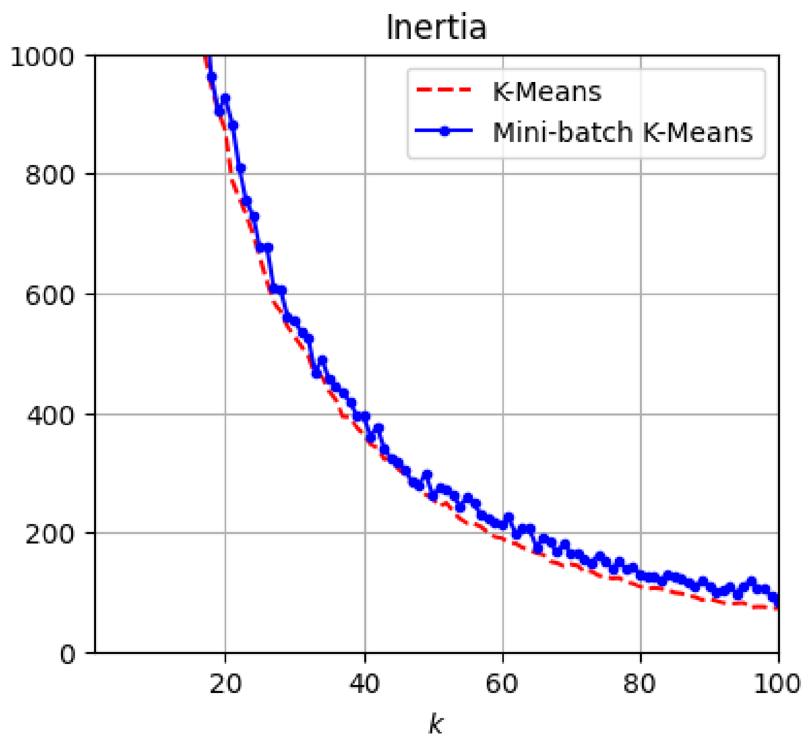
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:136
6: FutureWarning: algorithm='full' is deprecated, it will be removed
in 1.3. Using 'lloyd' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```

```
(1.3539178371429443, 1.468332052230835)
```

```
<IPython.core.display.HTML object>
```



```
100/100
```


Inertia
1000
800
600
400
200
0
20
40
60
80
100
k
K-Means
Mini-batch K-Means