K-Means_make_blobs

June 2, 2026

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

[2]:
```
X, y_true = make_blobs(
    n_samples=300,
    centers=4,
    cluster_std=1.10,
    random_state=42
)

df = pd.DataFrame(X, columns=["feature_1", "feature_2"])
print(df.head())
print("Shape:", df.shape)
```

```
   feature_1  feature_2
0  -9.343625   6.388694
1  -9.784782   6.900512
2  -1.604398   7.671358
3  -7.119077  -5.671455
4 -11.080265   6.214628
Shape: (300, 2)
```

[3]:
```
# K-Means repeats three main actions:
# 1) choose k cluster centers
# 2) assign each point to the nearest center
# 3) move each center to the mean of its assigned points
#
# It repeats until the centers stop changing enough
# or the maximum number of iterations is reached.
```

[4]:
```
kmeans = KMeans(
    n_clusters=4,
    init="k-means++",
```

```
    n_init=10,
    random_state=42
)

kmeans.fit(df)
```

```
C:\Users\Flynas54223\AppData\Local\Programs\Python\Python311\Lib\site-
packages\joblib\externals\loky\backend\context.py:136: UserWarning: 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 cores you want to use.
  warnings.warn(
  File "C:\Users\Flynas54223\AppData\Local\Programs\Python\Python311\Lib\site-
packages\joblib\externals\loky\backend\context.py", line 257, in
_count_physical_cores
    cpu_info = subprocess.run(
               ^^^^^^^^^^^^^^^
  File "C:\Users\Flynas54223\AppData\Local\Programs\Python\Python311\Lib\subproc
ess.py", line 548, in run
    with Popen(*popenargs, **kwargs) as process:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Flynas54223\AppData\Local\Programs\Python\Python311\Lib\subproc
ess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "C:\Users\Flynas54223\AppData\Local\Programs\Python\Python311\Lib\subproc
ess.py", line 1538, in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

[4]:
```
KMeans(n_clusters=4, n_init=10, random_state=42)
```

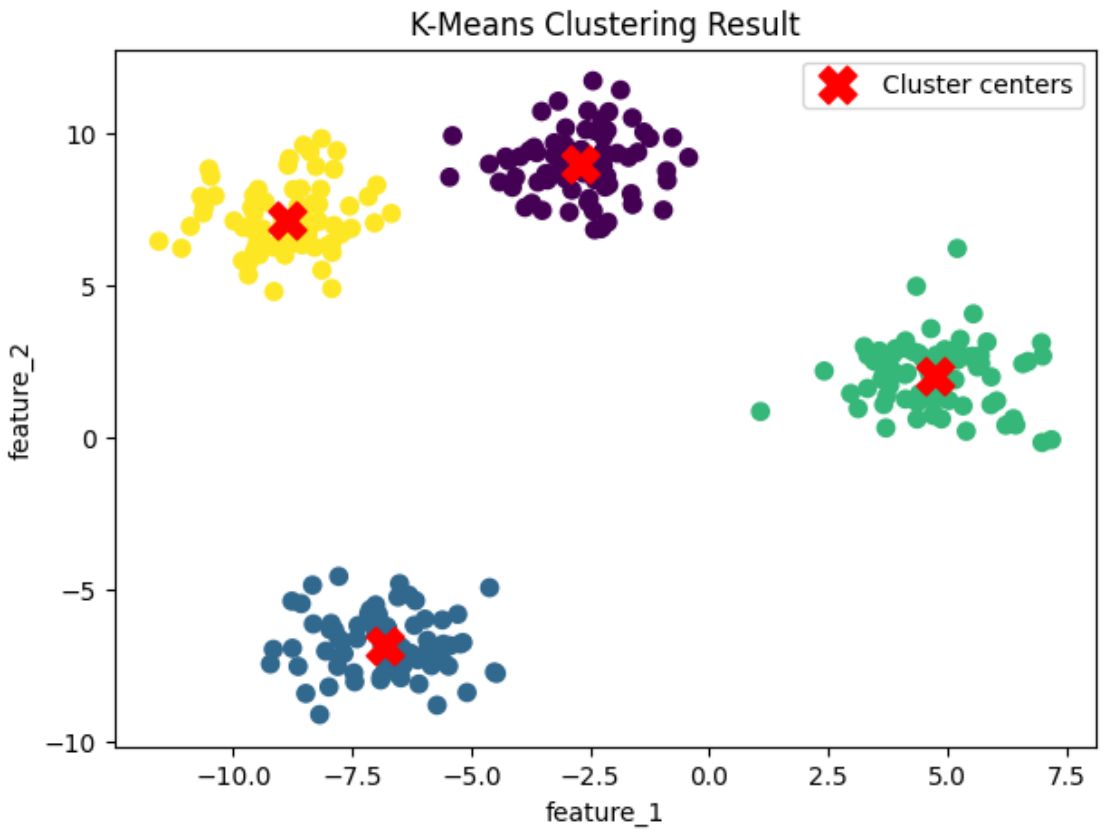
[5]:
```
cluster_labels = kmeans.labels_
cluster_centers = kmeans.cluster_centers_

print("First 10 cluster labels:", cluster_labels[:10])
print("Cluster centers:\n", cluster_centers)
print("Inertia:", round(kmeans.inertia_, 2))
```

```
First 10 cluster labels: [3 3 0 1 3 1 2 1 0 2]
Cluster centers:
 [[-2.72154493  8.96937277]
 [-6.82762454 -6.82549227]
 [ 4.7260375   2.04865946]
 [-8.87709663  7.15968947]]
Inertia: 679.53
```

[6]:
```
df["cluster"] = cluster_labels
print(df.groupby("cluster").mean())
```

```
         feature_1  feature_2
cluster
0        -2.721545   8.969373
1        -6.827625  -6.825492
2         4.726038   2.048659
3        -8.877097   7.159689
```

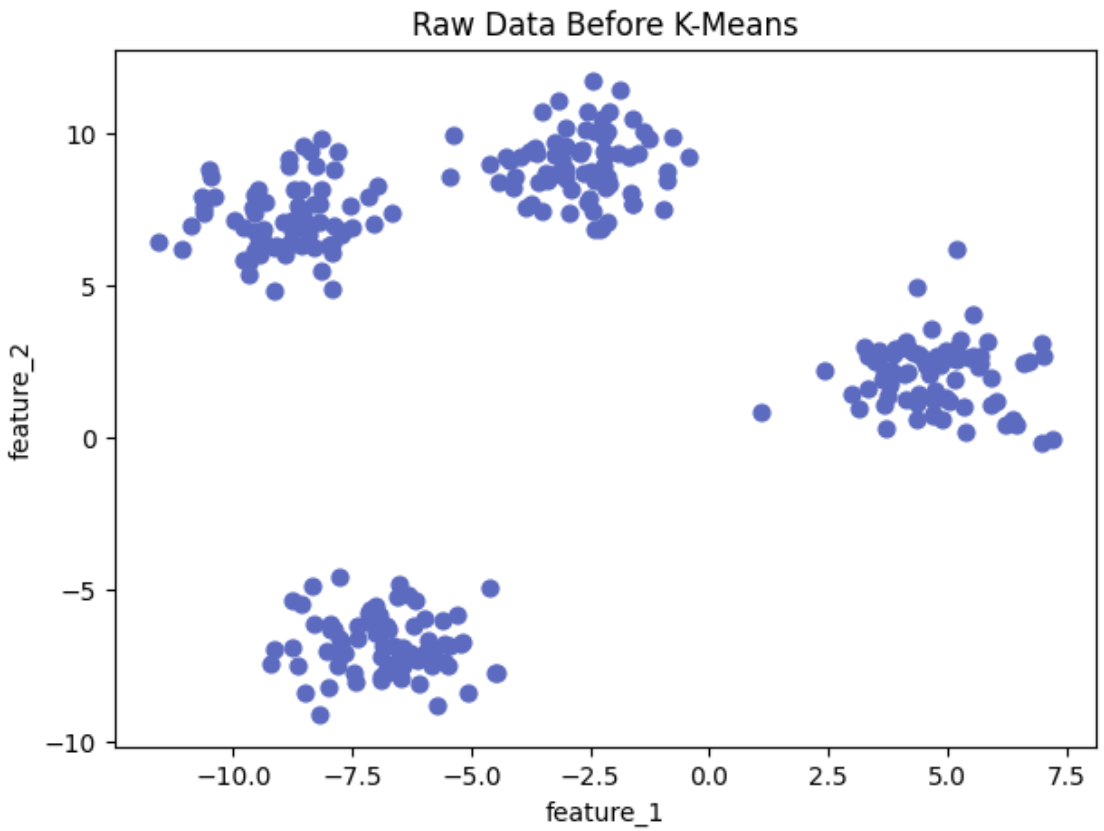
[7]:
```
plt.figure(figsize=(7, 5))
plt.scatter(df["feature_1"], df["feature_2"], c=df["cluster"], cmap="viridis",␣
↪s=45)
plt.scatter(
    cluster_centers[:, 0],
    cluster_centers[:, 1],
    c="red",
    s=220,
    marker="X",
    label="Cluster centers"
)
plt.xlabel("feature_1")
plt.ylabel("feature_2")
plt.title("K-Means Clustering Result")
plt.legend()
plt.show()
```



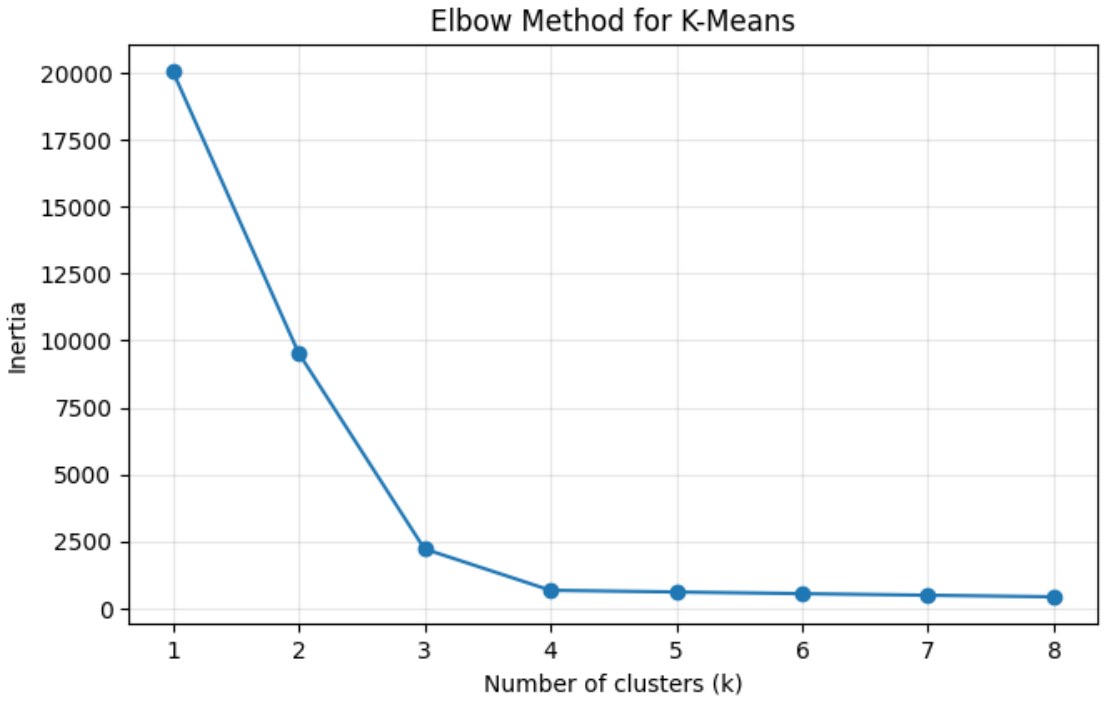
[8]:
```
score = silhouette_score(df[["feature_1", "feature_2"]], cluster_labels)
print("Silhouette score:", round(score, 4))
```

```
Silhouette score: 0.7718
```

[10]:
```
plt.figure(figsize=(7, 5))
plt.scatter(df["feature_1"], df["feature_2"], color="#5c6bc0", s=40)
plt.xlabel("feature_1")
plt.ylabel("feature_2")
plt.title("Raw Data Before K-Means")
plt.show()
```

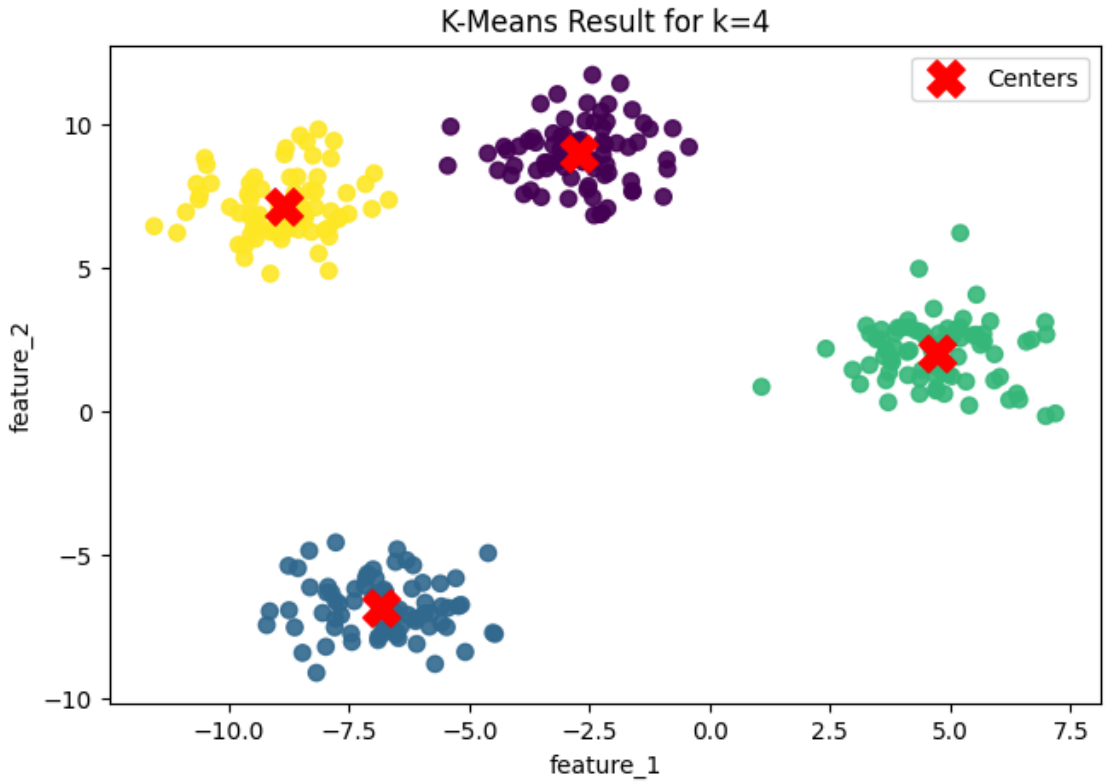


[11]:
```
inertia_values = []
k_values = range(1, 9)

for k in k_values:
    model = KMeans(n_clusters=k, init="k-means++", n_init=10, random_state=42)
    model.fit(df[["feature_1", "feature_2"]])
    inertia_values.append(model.inertia_)

plt.figure(figsize=(7.5, 4.5))
plt.plot(k_values, inertia_values, marker="o")
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia")
plt.title("Elbow Method for K-Means")
plt.grid(alpha=0.3)
plt.show()
```

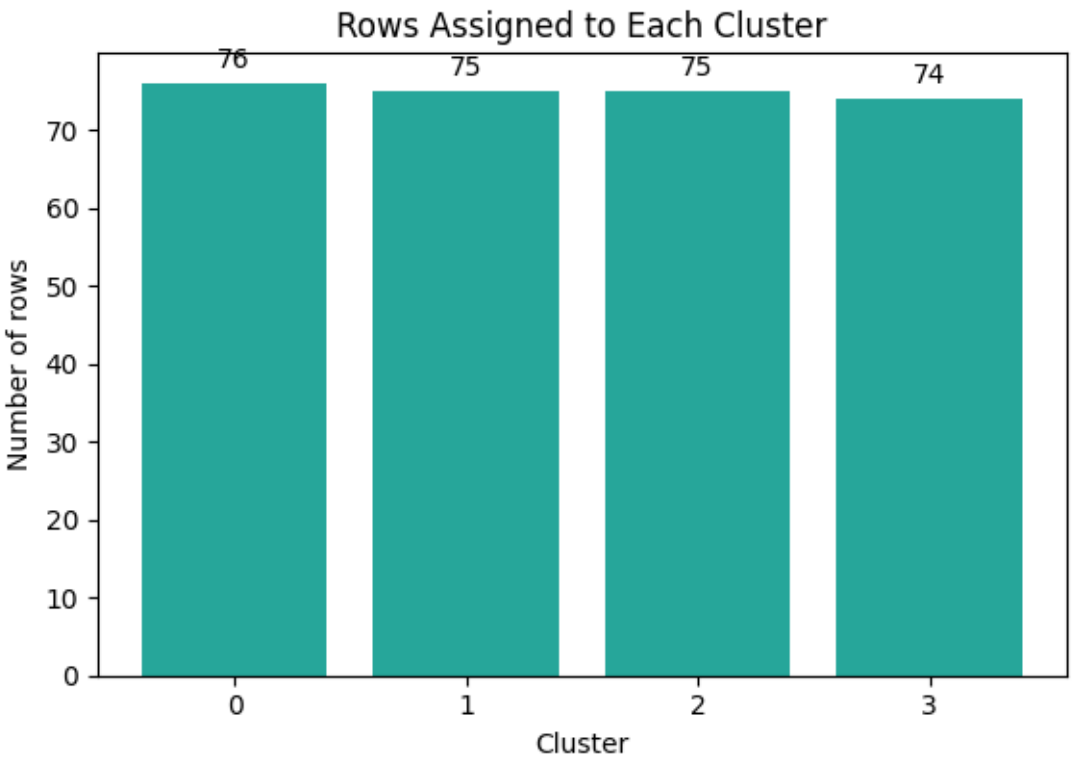


[12]:
```
silhouette_scores = []
k_values = range(2, 9)

for k in k_values:
    model = KMeans(n_clusters=k, init="k-means++", n_init=10, random_state=42)
    labels = model.fit_predict(df[["feature_1", "feature_2"]])
    score = silhouette_score(df[["feature_1", "feature_2"]], labels)
    silhouette_scores.append(score)

plt.figure(figsize=(7.5, 4.5))
plt.plot(k_values, silhouette_scores, marker="o", color="#43a047")
plt.xlabel("Number of clusters (k)")
plt.ylabel("Silhouette score")
plt.title("Silhouette Score for Different k Values")
plt.grid(alpha=0.3)
plt.show()
```

[13]:
```
best_k = 4
final_model = KMeans(n_clusters=best_k, init="k-means++", n_init=10,␣
 ↪random_state=42)
final_labels = final_model.fit_predict(df[["feature_1", "feature_2"]])
final_centers = final_model.cluster_centers_

plt.figure(figsize=(7.5, 5))
plt.scatter(df["feature_1"], df["feature_2"], c=final_labels, cmap="viridis",␣
 ↪s=45, alpha=0.9)
plt.scatter(final_centers[:, 0], final_centers[:, 1], c="red", s=240,␣
 ↪marker="X", label="Centers")
plt.xlabel("feature_1")
plt.ylabel("feature_2")
plt.title(f"K-Means Result for k={best_k}")
plt.legend()
plt.show()
```



[14]:
```
cluster_counts = pd.Series(final_labels).value_counts().sort_index()

plt.figure(figsize=(6.5, 4.2))
plt.bar(cluster_counts.index.astype(str), cluster_counts.values,␣
 ↪color="#26a69a")
plt.xlabel("Cluster")
plt.ylabel("Number of rows")
plt.title("Rows Assigned to Each Cluster")
for i, value in enumerate(cluster_counts.values):
    plt.text(i, value + 2, str(value), ha="center")
plt.show()
```


Rows Assigned to Each Cluster
76
75
75
74
70
60
50
40
30
20
10
0
Number of rows
0
1
2
3
Cluster