K Means Clustering Algorithms Implementation

In [1]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import pandas as pd
import numpy as np
%matplotlib inline
```

In [2]:

```
X, y = make_blobs(n_samples=1000,centers=3, n_features=2, random_state=23)
```

In [3]:

```
X.shape
```

Out[3]:

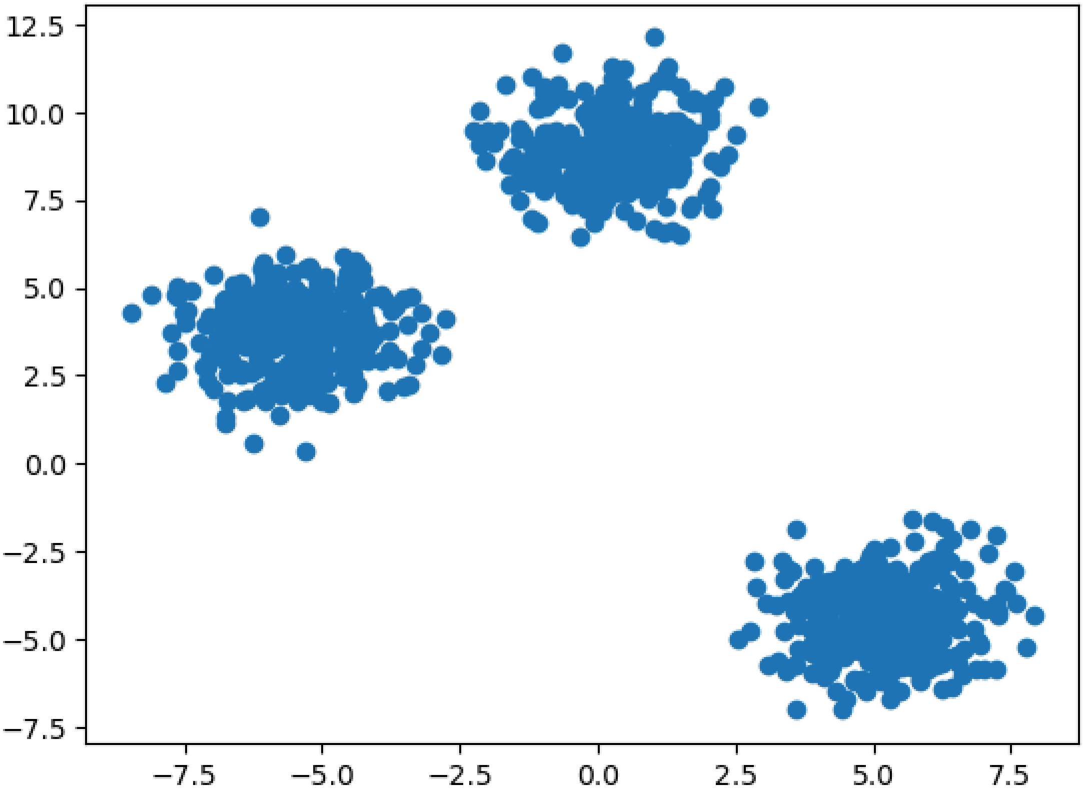
```
(1000, 2)
```

In [6]:

```
plt.scatter(X[:,0],X[:,1])
```

Out[6]:

```
<matplotlib.collections.PathCollection at 0x1ca2588e750>
```



In [5]:

```
from sklearn.model_selection import train_test_split
X_train, y_train, X_test, y_test = train_test_split(X, y , test_size=0.33, randc
```

In [9]:

```
from sklearn.cluster import KMeans
```

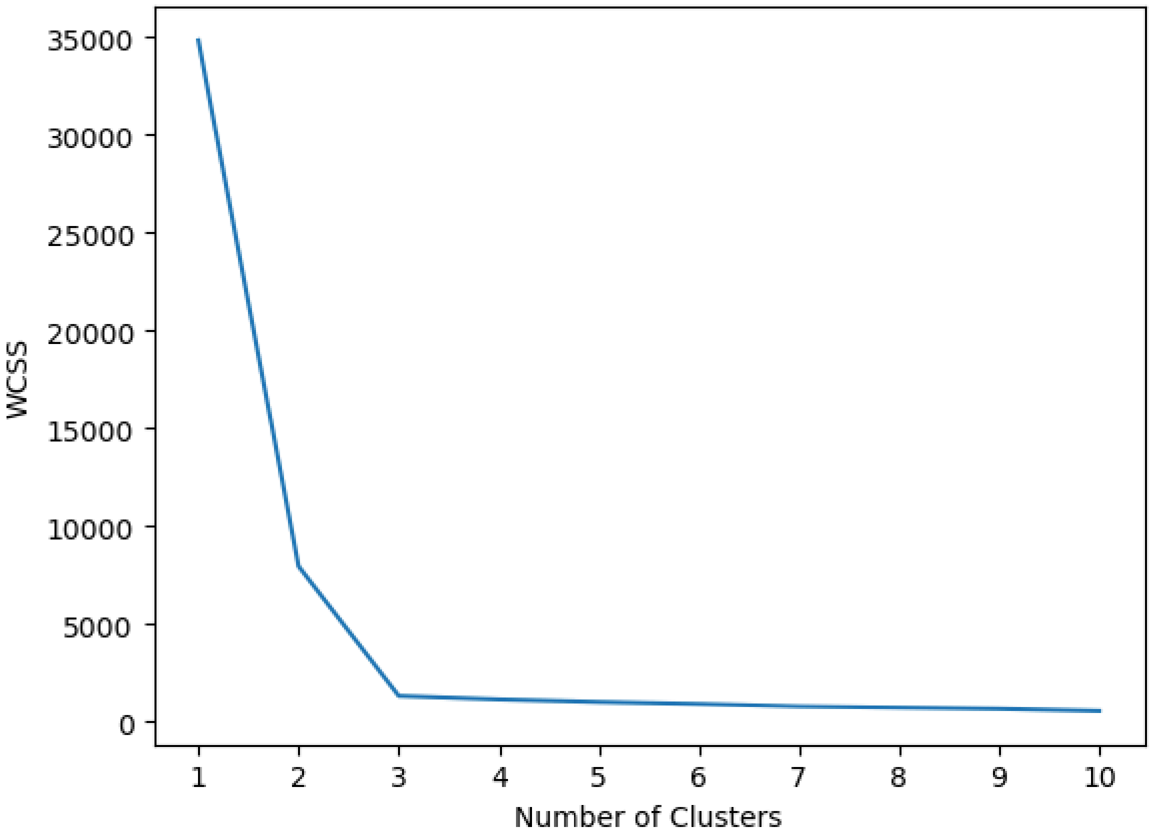
In [41]:

```
# Manual process
## Elbow method to select the k value

wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k,init='k-means++')
    kmeans.fit(X_train)
    wcss.append(kmeans.inertia_)
```

In [42]:
```
wcss
```

Out[42]:
```
[34827.57682552021,
 7935.437286145418,
 1319.2730531585605,
 1140.4677884655123,
 1006.8745739258457,
 902.3383256536542,
 770.7924368518035,
 709.6055857560772,
 644.8743127558693,
 532.3776756218431]
```

In [44]:
```
## plot the elbow curve
plt.plot(range(1,11),wcss)
plt.xticks(range(1,11))
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```



In [45]:
```
kmeans = KMeans(n_clusters=3, init="k-means++")
```

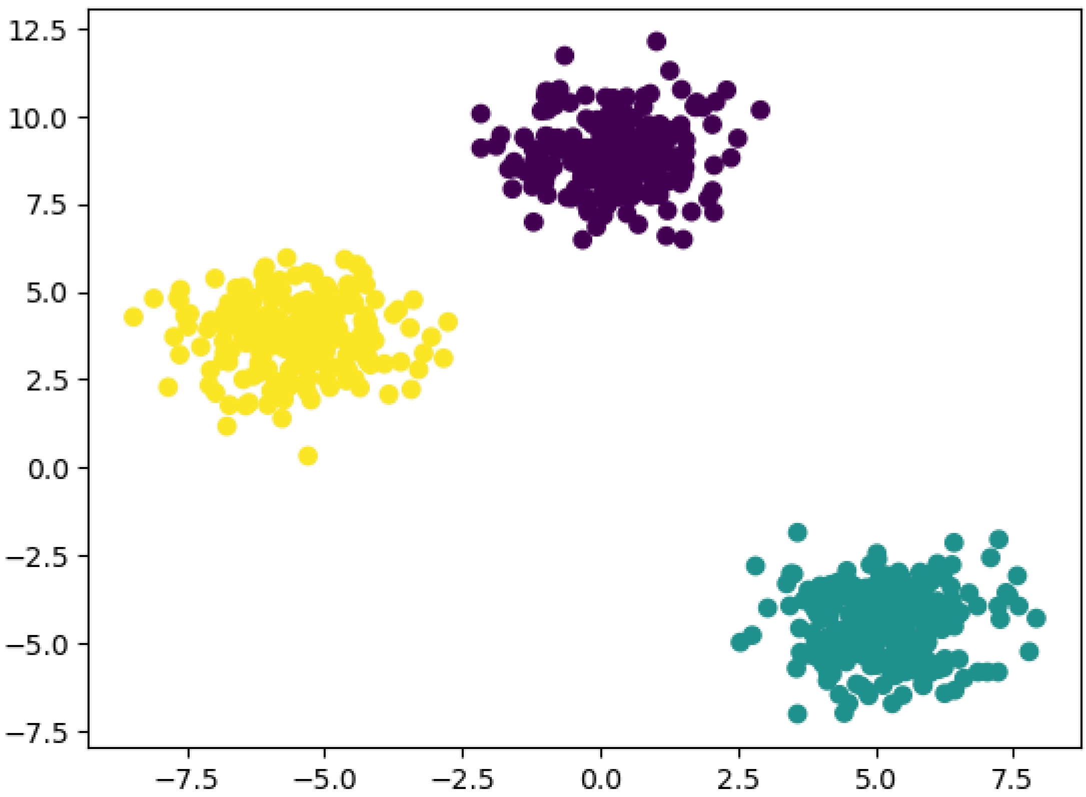
In [46]:
```
y_labels = kmeans.fit_predict(X_train)
```

In [59]:
```
y_test_label = kmeans.predict(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[59], line 1
----> 1 y_test_label = kmeans.predict(X_test)

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1085, in _BaseKMean
s.predict(self, X)
   1067 """Predict the closest cluster each sample in X belongs to.
   1068 
   1069 In the vector quantization literature, `cluster_centers_` is called
   (...)
   1081     Index of the cluster each sample belongs to.
   1082 """
   1083 check_is_fitted(self)
-> 1085 X = self._check_test_data(X)
   1087 # sample weights are not used by predict but cython helpers expect an arr
ay
   1088 sample_weight = np.ones(X.shape[0], dtype=X.dtype)

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:944, in _BaseKMean
s._check_test_data(self, X)
    943 def _check_test_data(self, X):
--> 944     X = validate_data(
    945         self,
    946         X,
    947         accept_sparse="csr",
    948         reset=False,
    949         dtype=[np.float64, np.float32],
    950         order="C",
    951         accept_large_sparse=False,
    952     )
    953     return X

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2954, in validate_
data(_estimator, X, y, reset, validate_separately, skip_check_array, **check_para
ms)
   2952         out = X, y
   2953 elif not no_val_X and no_val_y:
-> 2954     out = check_array(X, input_name="X", **check_params)
   2955 elif no_val_X and not no_val_y:
   2956     out = _check_y(y, **check_params)

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1091, in check_arr
ay(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_writeabl
e, force_all_finite, ensure_all_finite, ensure_non_negative, ensure_2d, allow_nd,
ensure_min_samples, ensure_min_features, estimator, input_name)
   1084         else:
   1085             msg = (
   1086                 f"Expected 2D array, got 1D array instead:\narray={arra
y}.\n"
   1087                 "Reshape your data either using array.reshape(-1, 1) if "
   1088                 "your data has a single feature or array.reshape(1, -1) "
   1089                 "if it contains a single sample."
   1090             )
-> 1091         raise ValueError(msg)
   1093 if dtype_numeric and hasattr(array.dtype, "kind") and array.dtype.kind in
"USV":
   1094     raise ValueError(
   1095         "dtype='numeric' is not compatible with arrays of bytes/strings."
   1096         "Convert your data to numeric values explicitly instead."
   1097     )

ValueError: Expected 2D array, got 1D array instead:
array=[1. 1. 1. 2. 2. 2. 1. 2. 1. 1. 0. 0. 1. 1. 0. 1. 0. 2. 0. 0. 2. 0. 0. 2.
 2. 1. 1. 2. 2. 1. 2. 2. 1. 2. 0. 0. 0. 2. 2. 1. 1. 2. 1. 1. 2. 0. 0. 1.
 2. 1. 1. 0. 1. 1. 1. 1. 1. 0. 2. 2. 1. 1. 2. 1. 0. 0. 0. 0. 2. 1. 0. 2.
 1. 0. 1. 0. 2. 2. 2. 2. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 2. 1. 1.
 2. 0. 0. 0. 0. 2. 0. 1. 2. 2. 0. 1. 0. 1. 2. 0. 1. 2. 2. 2. 2. 0. 1. 0.
 1. 1. 0. 2. 1. 2. 1. 0. 2. 2. 1. 2. 2. 2. 2. 2. 2. 2. 1. 1. 0. 0. 1. 2.
 0. 0. 2. 1. 0. 0. 0. 2. 0. 1. 1. 2. 1. 0. 0. 1. 0. 2. 0. 0. 2. 2. 2. 1.
 2. 2. 1. 0. 2. 2. 0. 2. 0. 1. 0. 2. 0. 2. 1. 2. 2. 2. 0. 1. 0. 2. 1. 0.
 2. 1. 1. 0. 0. 1. 0. 0. 0. 0. 1. 2. 1. 1. 0. 2. 0. 0. 1. 2. 1. 2. 1. 2.
 0. 1. 2. 0. 2. 1. 1. 2. 2. 0. 2. 1. 0. 0. 1. 0. 2. 1. 0. 2. 1. 1. 1. 1.
 0. 1. 1. 2. 1. 0. 0. 2. 1. 2. 1. 1. 2. 2. 2. 1. 0. 0. 2. 2. 2. 0. 0. 1.
 1. 1. 0. 0. 1. 0. 2. 0. 0. 0. 0. 2. 1. 2. 0. 2. 1. 2. 2. 0. 2. 0. 0. 2.
 2. 1. 1. 1. 2. 0. 2. 0. 1. 1. 1. 1. 2. 1. 2. 0. 1. 1. 1. 2. 2. 0. 0. 0.
 2. 0. 2. 2. 2. 1. 1. 0. 2. 1. 0. 1. 2. 0. 1. 1. 0. 1. 0. 2. 0. 2. 2. 2.
 1. 2. 1. 1. 2. 2. 2. 0. 0. 2. 2. 2. 1. 0. 2. 2. 2. 2. 0. 1. 1. 0. 2. 0.
 2. 2. 0. 2. 0. 0. 1. 2. 1. 0. 1. 2. 0. 1. 0. 1. 0. 1. 1. 2. 0. 2. 0. 1.
 1. 0. 1. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 2. 1. 2. 0. 2. 2. 2. 0. 0.
 1. 0. 0. 1. 2. 0. 1. 0. 2. 2. 1. 2. 2. 0. 0. 2. 2. 0. 0. 1. 0. 0. 2. 1.
 1. 0. 1. 2. 0. 2. 0. 1. 0. 2. 0. 1. 1. 0. 1. 0. 1. 1. 0. 1. 1. 0. 0. 0.
 0. 0. 1. 2. 1. 1. 0. 0. 2. 0. 1. 1. 2. 0. 2. 0. 2. 2. 0. 1. 2. 1. 0. 0.
 2. 1. 2. 2. 2. 2. 2. 1. 0. 1. 0. 0. 2. 2. 1. 1. 1. 1. 2. 0. 1. 0. 1. 0.
 1. 0. 2. 1. 0. 2. 1. 2. 1. 2. 1. 1. 2. 2. 1. 2. 1. 1. 1. 2. 2. 2. 0. 1.
 2. 1. 0. 0. 2. 1. 2. 0. 0. 1. 2. 0. 0. 2. 0. 1. 0. 0. 2. 2. 1. 1. 0. 0.
 1. 1. 2. 1. 2. 1. 1. 0. 2. 1. 0. 0. 2. 0. 1. 2. 1. 1. 0. 1. 2. 0. 1. 1.
 1. 0. 0. 1. 2. 0. 2. 1. 1. 1. 2. 0. 0. 0. 0. 1. 0. 0. 2. 0. 0. 1. 0. 0.
 2. 2. 1. 0. 0. 1. 1. 0. 1. 1. 0. 1. 2. 2. 1. 2. 0. 1. 1. 0. 0. 2. 2. 1.
 1. 2. 0. 1. 0. 2. 0. 2. 2. 2. 0. 1. 1. 0. 2. 2. 0. 1. 0. 1. 0. 2. 0. 0.
 2. 2. 2. 1. 0. 2. 2. 2. 2. 1. 0. 1. 0. 1. 0. 1. 0. 2. 0. 1. 2. 1.].
Reshape your data either using array.reshape(-1, 1) if your data has a single fea
ture or array.reshape(1, -1) if it contains a single sample.
```

```
In [52]: plt.scatter(X_train[:,0],X_train[:,1],c=y_labels)
```

```
Out[52]: <matplotlib.collections.PathCollection at 0x1ca323ffa90>
```



In [53]:

```
## kneed locatore

from kneed import KneeLocator
```

In [54]:

```
k1 = KneeLocator(range(1,11),wcss,curve='convex', direction='decreasing')
k1.elbow
```

Out[54]:

```
3
```

In [55]:

```
# performance metrics
# silhoutte score
from sklearn.metrics import silhouette_score
```

In [56]:

```
silhouette_coefficients = []
for k in range(2,11):
    kmeans = KMeans(n_clusters=k,init='k-means++')
    kmeans.fit(X_train)
    score = silhouette_score(X_train, kmeans.labels_)
    silhouette_coefficients.append(score)
```

In [57]:

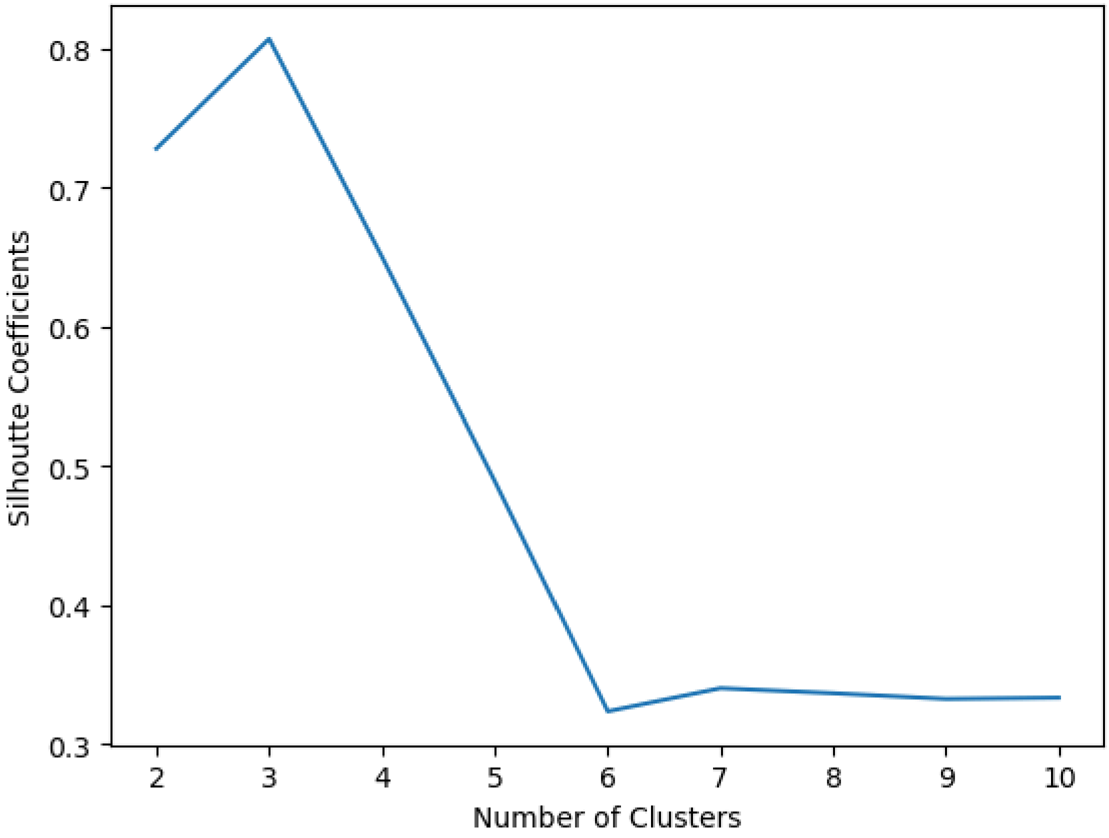
```
silhouette_coefficients
```

Out[57]:

```
[0.7281443868598331,
 0.8071181203797672,
 0.6505454471731087,
 0.4895647834796006,
 0.3237480747005592,
 0.34040524166707997,
 0.33677299942813615,
 0.3327061302389863,
 0.3335585872729096]
```

In [60]:

```python
# plot the silhouette score
plt.plot(range(2,11),silhouette_coefficients)
plt.xticks(range(2,11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhoutte Coefficients")
plt.show()
```



In [ ]: