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
```

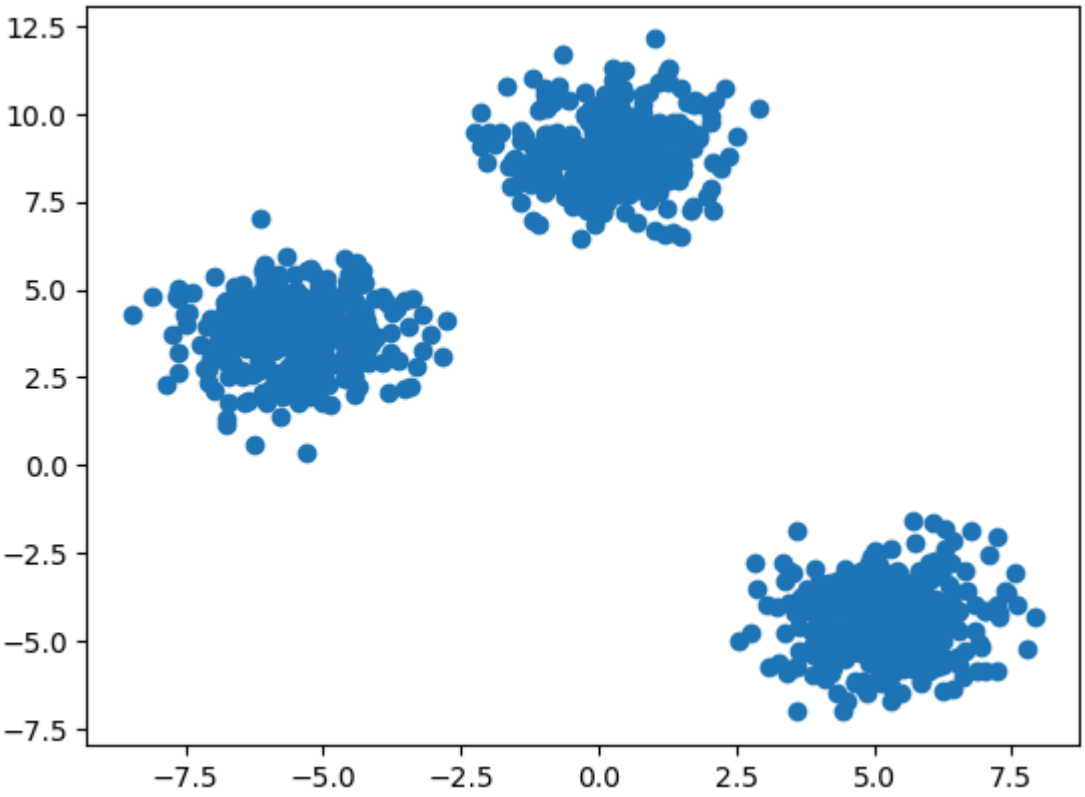
In [2]:
```
X, y = make_blobs(n_samples=1000,centers=3, n_features=2, random_state=23)
```

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
```

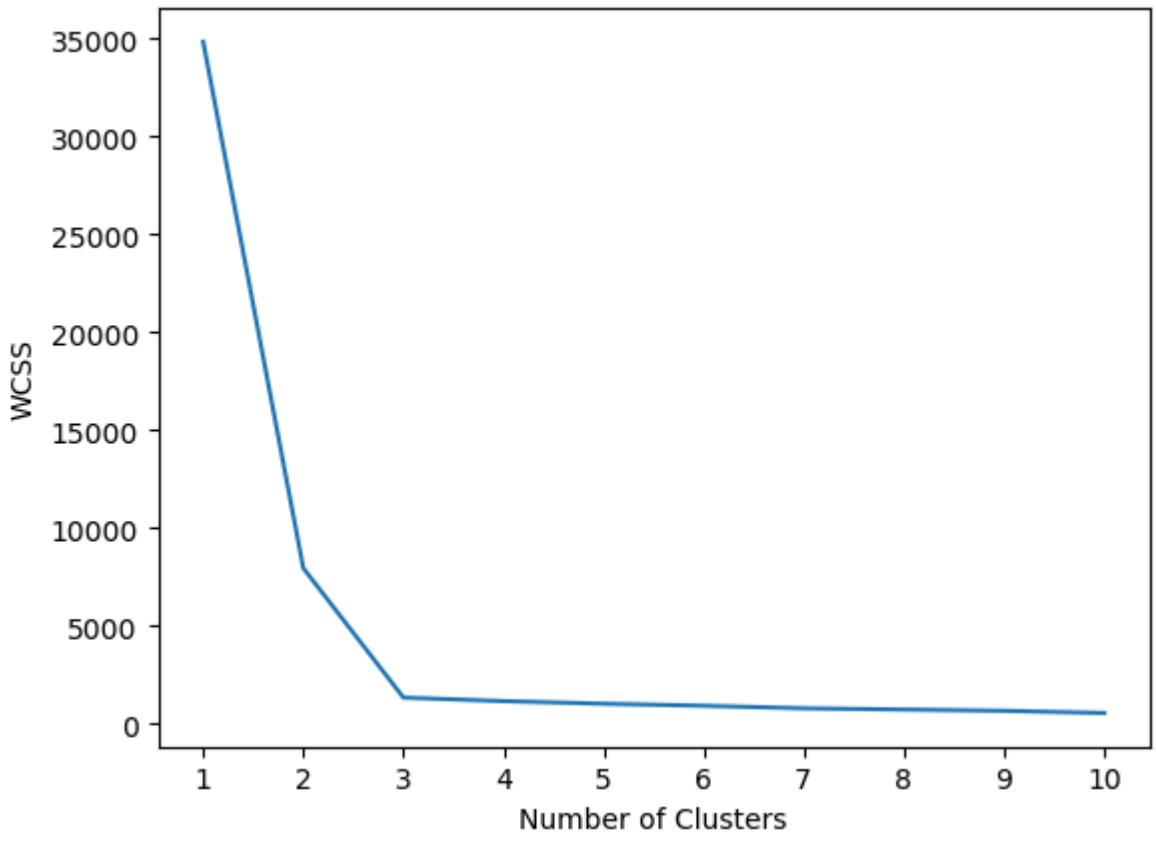
In [41]:
```
# Manual process
## Elbow method to select the k value

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++")
```

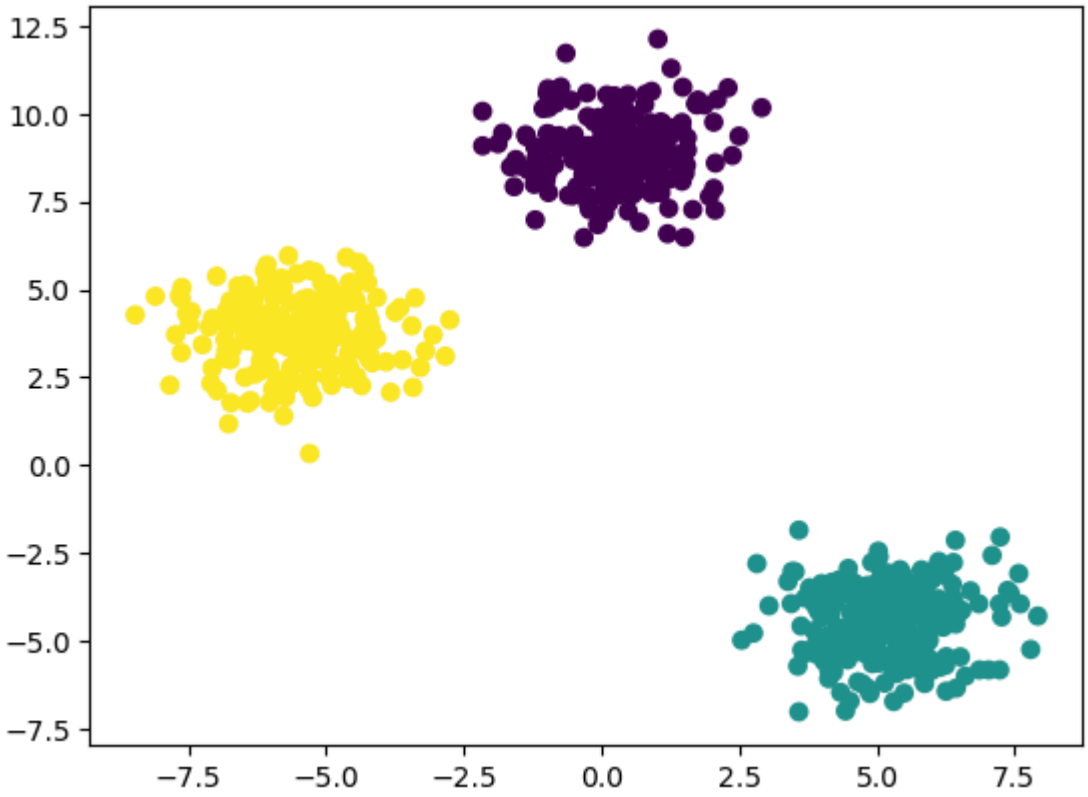
In [46]:
```
y_labels = kmeans.fit_predict(X_train)
```

In [51]:
```
y_test_label = kmeans.predict(X_test.reshape(-1,1))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[51], line 1
----> 1 y_test_label = kmeans.predict(X_test.reshape(-1,1))

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1085, in _BaseKMeans.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 array
   1088 sample_weight = np.ones(X.shape[0], dtype=X.dtype)

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:944, in _BaseKMeans._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:2975, in validate_data(_estimator, X, y, reset, validate_separately, skip_check_array, **check_params)
   2972     out = X, y
   2974 if not no_val_X and check_params.get("ensure_2d", True):
-> 2975     _check_n_features(_estimator, X, reset=reset)
   2977 return out

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2839, in _check_n_features(estimator, X, reset)
   2836     return
   2838 if n_features != estimator.n_features_in_:
-> 2839     raise ValueError(
   2840         f"X has {n_features} features, but {estimator.__class__.__name__}"
   2841         f"is expecting {estimator.n_features_in_} features as input."
   2842     )

ValueError: X has 1 features, but KMeans is expecting 2 features as input.
```

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

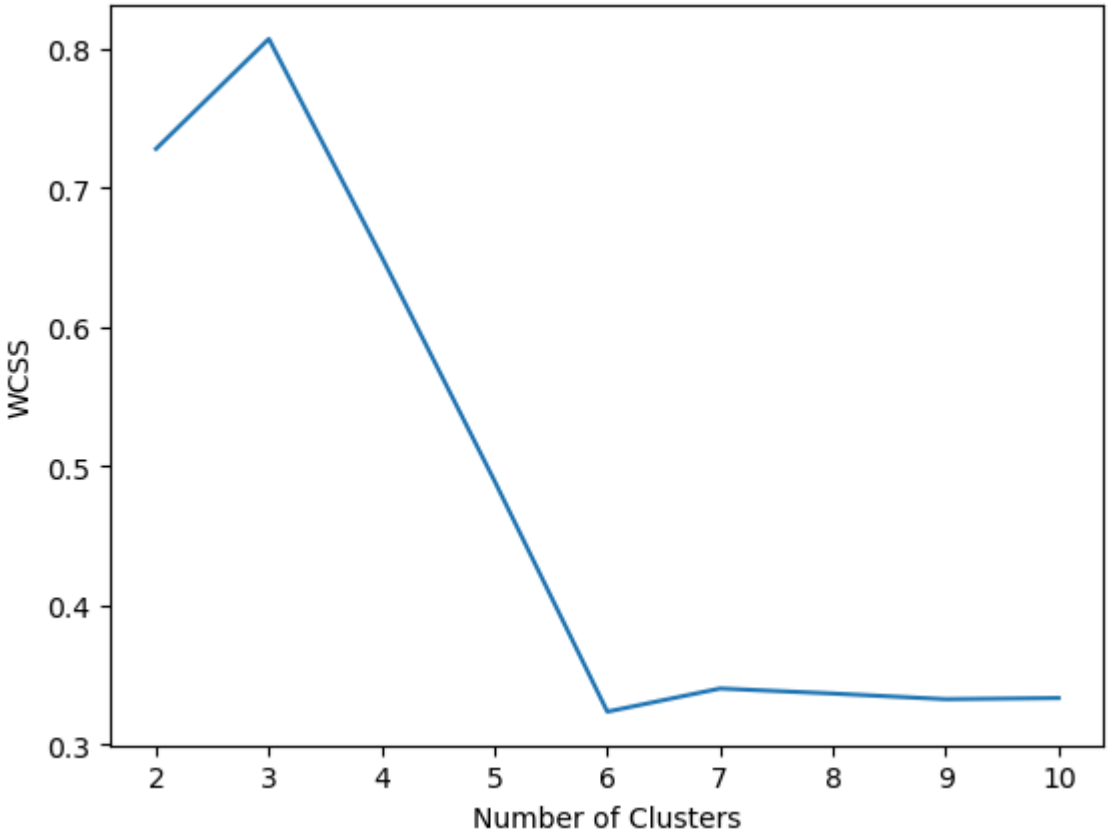
In [55]:
```
# performance metrics
# silhoutte score
from sklearn.metrics import silhouette_score
```

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 [58]:
```
# plot the silhouette score
plt.plot(range(2,11),silhouette_coefficients)
plt.xticks(range(2,11))
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```



In [ ]: