Step-1

In [157…
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

Step-2

In [158…
```
np.random.seed(0)
#ensures same random numbers are generated every single time
#easy comparison of clustures across iterations
n_samples = 500
X = np.random.randn(n_samples , 2 )
#visualizing data in 2-D where randn represents std normal distribution
```

Step-3

In [159…
```
# mu represents the mean of clusters or where the clusters are centred
# sigma here in 2-D represents the covariance matrix
mu1 = [2, 2]
sigma1 = [[0.9, -0.0255],
          [-0.0255, 0.9]]

mu2 = [5, 5]
sigma2 = [[0.5, 0],
          [0, 0.3]]

mu3 = [-2, -2]
sigma3 = [[1, 0],
          [0, 0.9]]

mu4 = [-4, 8]
sigma4 = [[0.8, 0],
          [0, 0.6]]
```

Step-4

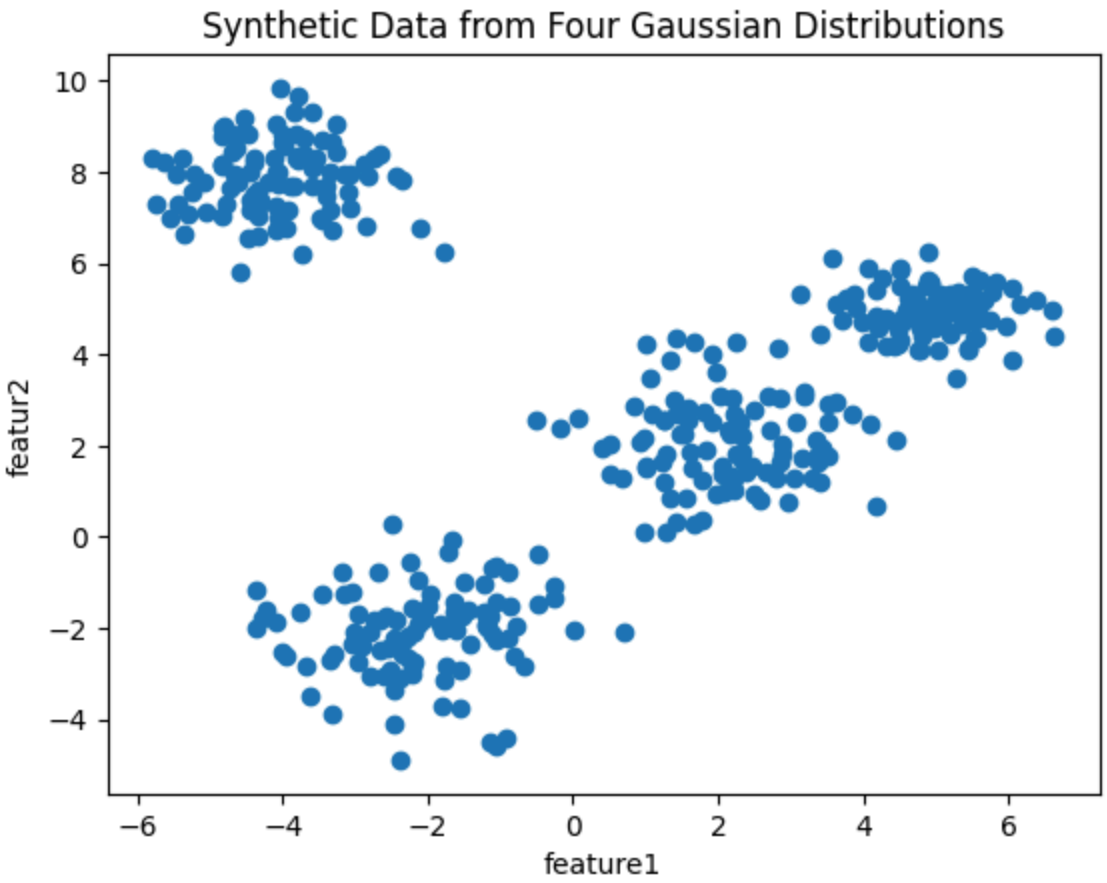
In [160…
```
np.random.seed(0)
n_samples_per_cluster = 100

X1 = np.random.multivariate_normal(mu1, sigma1, n_samples_per_cluster)
X2 = np.random.multivariate_normal(mu2, sigma2, n_samples_per_cluster)
X3 = np.random.multivariate_normal(mu3, sigma3, n_samples_per_cluster)
X4 = np.random.multivariate_normal(mu4, sigma4, n_samples_per_cluster)


X = np.vstack((X1, X2, X3, X4))
#stack all points in a dataset

plt.scatter(X[:, 0], X[:, 1])
plt.title("Synthetic Data from Four Gaussian Distributions")
plt.xlabel("feature1")
```

```
plt.ylabel("featur2")
plt.show()
```



Step-5

In [161…

```
wcss = []

for k in range(1, 21):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
    print("k",k,"wcss",wcss)
```

```
k 1 wcss [10955.801110550676]
k 2 wcss [10955.801110550676, 6467.148554319869]
k 3 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146]
k 4 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603]
k 5 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628]
k 6 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428]
k 7 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946]
k 8 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665]
k 9 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427]
k 10 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195]
k 11 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895]
k 12 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157]
k 13 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184]
k 14 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184, 219.6444081395697]
k 15 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184, 219.6444081395697, 204.97528433040736]
k 16 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184, 219.6444081395697, 204.97528433040736, 194.53588305152115]
k 17 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184, 219.6444081395697, 204.97528433040736, 194.53588305152115, 181.53932195909607]
k 18 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184, 219.6444081395697, 204.97528433040736, 194.53588305152115, 181.53932195909607, 171.11713125677565]
k 19 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 333.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40457188883184, 219.6444081395697, 204.97528433040736, 194.53588305152115, 181.53932195909607, 171.11713125677565, 157.57245709616785]
```

```
k 20 wcss [10955.801110550676, 6467.148554319869, 1384.0618801265146, 591.5149010190
603, 546.0031720452628, 482.103624090428, 456.17862942877946, 376.3637134506665, 33
3.02633614063427, 292.99340568241195, 264.19833198599895, 248.63898084811157, 233.40
457188883184, 219.6444081395697, 204.97528433040736, 194.53588305152115, 181.5393219
5909607, 171.11713125677565, 157.57245709616785, 148.60317186154663]
```

Step-6

In [162…

```
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(X)
labels= kmeans.labels_
centroids = kmeans.cluster_centers_
print(centroids)
```

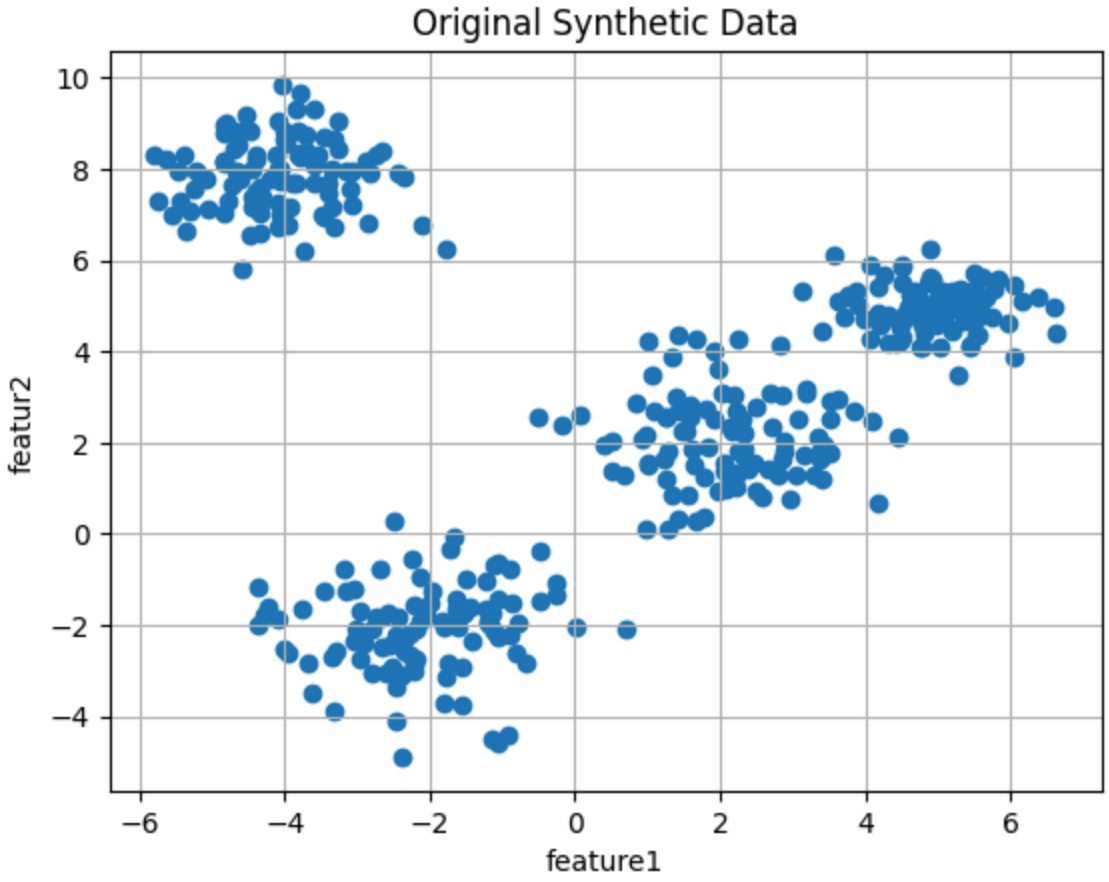
```
[[-2.11235752 -2.09784499]
 [-4.09173419  7.87917301]
 [ 4.89882834  4.93813439]
 [ 2.0950637   2.09376076]]
```

Step-7

Original Synthetic Data

In [ ]:

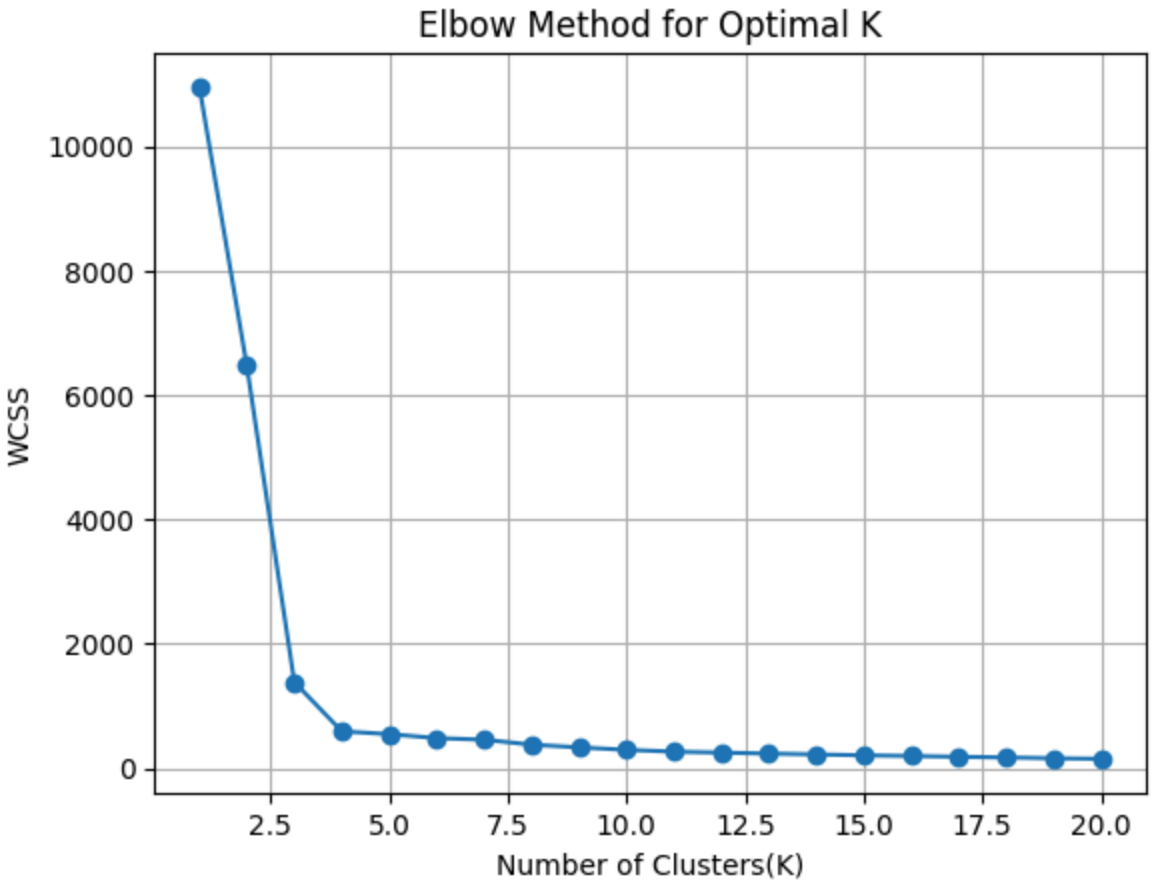
```
plt.scatter(X[:, 0], X[:, 1])
plt.title("Original Synthetic Data")
plt.xlabel("feature1")
plt.ylabel("feature2")
plt.grid()
plt.show()
```



Elbow Plot

In [164…

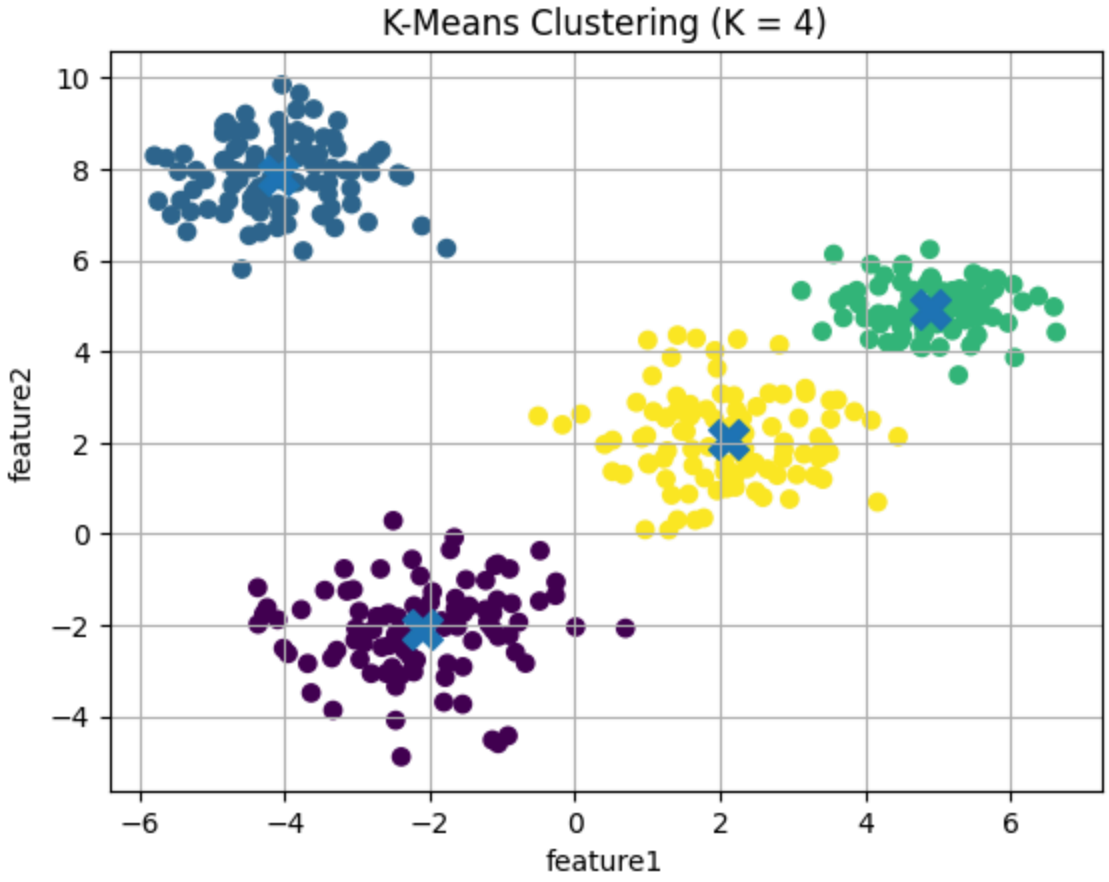
```
plt.plot(range(1, 21), wcss, marker='o')
plt.title("Elbow Method for Optimal K")
plt.xlabel("Number of Clusters(K)")
plt.ylabel("WCSS")
plt.grid()
plt.show()
```



Clustered Data PLOT

In [165…

```
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='X', s=200)
plt.title("K-Means Clustering (K = 4)")
plt.xlabel("feature1")
plt.ylabel("feature2")
plt.grid()
plt.show()
```


K-Means Clustering (K = 4)
feature2
feature1
10
8
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6

MLPR LAB-2 REPORT

1) **What happens if the number of clusters is set too high or too low?**
   If K is too low (K<4), distinct Gaussian clusters get merged and important structure is lost.
   If K is too high (K>4) , clusters break into multiple small clusters (splitting happens beyond natural splitting) increasing WCSS reduction .

2) **Effect of initializing centroids too far apart or too close together?**
   If centroids are too close, multiple centroids may converge to the same cluster.
   If they are poorly placed far apart, K-Means may converge slowly to a sub-optimal solution which requires more iterations

3) How might you optimize the centroid initialisation to lead to better clustering ?
Using K-Means++ initialization places centroids far apart in high-density regions of the data. In this lab, it helps centroids align closely with the means of the four Gaussian distributions.

4) Why might the Elbow Method not always provide a clear solution for choosing the optimal number of clusters?
If Gaussian clusters overlap or have similar spreads, the WCSS curve may decrease smoothly without a sharp elbow. In such cases, as seen in real datasets (unlike this clean lab data), identifying the optimal K becomes difficult and should possibly be treated with other techniques . (Like Silhouette Method)

5) How can WCSS be influenced by the presence of outliers or noise in the data?
Outliers increase WCSS by pulling centroids away from the true Gaussian centers. If there are outliers in the dataset , the elbow plot will be distorted and it may show us higher number of clusters than required in the data .