K-Means Clustering

Clustering is a unsupervised ML technique in which there is no target varible and unlabled data is generally grouped into different groups/ clusters.

we will study following clustering Algorithms

1) K-means clustering
2) DBscan clustering
3) Hierarchical clustering

Lets start with K-means

→ Here K is no. of pre-defined clusters that need to be created in order to initiate algorithm.

→ It is centroid-based Algorithm, where each cluster is associated with a centroid.

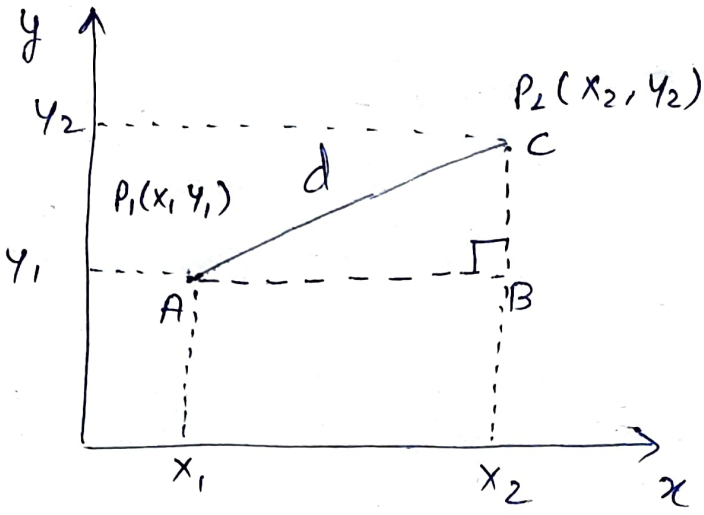
→ The main aim is to minimise the sum of distances blw data point and their corresponding clusters.

⇒ following distance methods are used.

1) Euclidean distance
2) Manhattan distance
3) cosine distance
4) squared Euclidean distance
5) Tanimoto distance

] we will only see these two.

① Euclidean distance



pythagoras theoram in Δ ABC

$$d^2 = AB^2 + BC^2$$

$$d = \sqrt{AB^2 + BC^2}$$

$$\boxed{d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

② Manhattan distance

It is calculated for two vectors.

$\rightarrow a_1 \hat{x} + b_1 \hat{y} + c_1 \hat{z}$

$\rightarrow a_2 \hat{x} + b_2 \hat{y} + c_2 \hat{z}$

$$\boxed{d = |(a_2 - a_1)| + |(b_2 - b_1)| + |(c_2 - c_1)|}$$

⇨ Steps to perform k-means Clustering

1) Decide No. of clusters. (k value)

2) Initialize centroids equal to No. of clusters Randomly

say if you have 3 clusters so there will be 3 centroids and they will be selected randomly.

③ Assign cluster to each data point in dataset. using distance similarity method (say Euclidean distance) ie calculate distance blw centroids and all datapoint. Assign data point to that cluster whose dist is minimum.

=> Now we have distinct clusters.

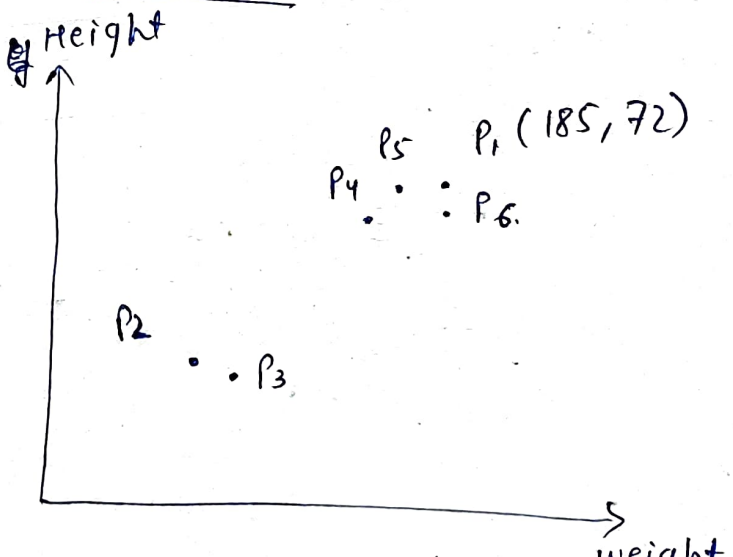
→ calculate New centroids for each cluster by taking mean of all the data point in that cluster.

④ => calculate distances blw all the points wit new centroids and again assign the data points to clusters using minimum distance blw centroid and datapoint.

=> Repeat 3rd and 4th step until there is no further change in centroid.

* lets see above with an example

	Height	weight
P_1	185	72
P_2	170	56
P_3	168	60
P_4	179	68
P_5	180	71
P_6	182	72



1) lets consider two clusters ie $k=2$

2) Randomly selecting 2 points

$P_4 \Rightarrow (179, 68)$ and $P_6 \Rightarrow (182, 72)$

as two centroids

	Height	weight	dist P_4 (A)	dist P_6 (B)	cluster(A/B)
P_1	185	72	7.2	3	B
P_2	170	56	15	20	A
P_3	168	60	13.6	18.43	A
P_5	180	71	3.16	2.23	B

$P_4 = 179, 68$

$P_6 = 182, 72$

consider P_1 point

1) dist of P_1 from centroid A $[P_4 (179, 68)]$

$$d_{A_1} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$= \sqrt{(72-68)^2 + (185-179)^2}$$

$$dA_1 = 7.2 \text{ units}$$

2) dist of P_1 from centroid B, $P_6 (182, 72)$.

$$dB_1 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$= \sqrt{(72-72)^2 + (185-182)^2}$$

$$dB_1 = 3 \text{ units.}$$

$\therefore$ P_1 belongs to B cluster as distance b/w P_1 and B centroid is less as compared to distance b/w P_1 and A centroid.

Now similarly lets calculate All distances

Cluster A

$P_4 \Rightarrow 179, 68$

$P_2 \Rightarrow 170, 56$

$P_3 \Rightarrow 168, 60$

Cluster B

$P_6 \Rightarrow 182, 72$

$P_1 \Rightarrow 185, 72$

$P_5 \Rightarrow 180, 71$

New centroids

Cluster A

$$\left(\frac{179+170+168}{3}, \frac{68+56+60}{3}\right)$$

$\Rightarrow (172.33, 61.33)$

Cluster B

$$\left(\frac{182+185+180}{3}, \frac{72+72+71}{3}\right)$$

$\Rightarrow (182.33, 71.67)$

	Height	weight	dist A	dist B	Cluster A/B
P_1	185	72	16.56	2.69	B
P_2	170	56	5.82	19.94	A
P_3	168	60	4.53	18.48	A
P_4	179	68	9.43	4.95	B
P_5	180	71	12.34	2.42	B
P_6	182	72	14.4	0.47	B

Cluster A

$$\left(\frac{170+168}{2}, \frac{56+60}{2}\right)$$

$\Rightarrow (169, 58)$

Cluster B

$$\left(\frac{185+179+180+182}{4}, \frac{72+68+71+72}{4}\right)$$

$(181.5, 70.75)$

Height	weight	dist A (169, 58)	dist B (181.5, 70.75)	cluster A/B.
185	72	21.26	3.72	B
170	56	2.24	18.7	A
168	60	2.24	17.26	A
179	68	14.14	3.72	B
180	71	17.03	1.52	B
182	72	19.1	1.35	B

New centroids

cluster A

$\left(\frac{170+168}{2}, \frac{56+60}{2}\right)$

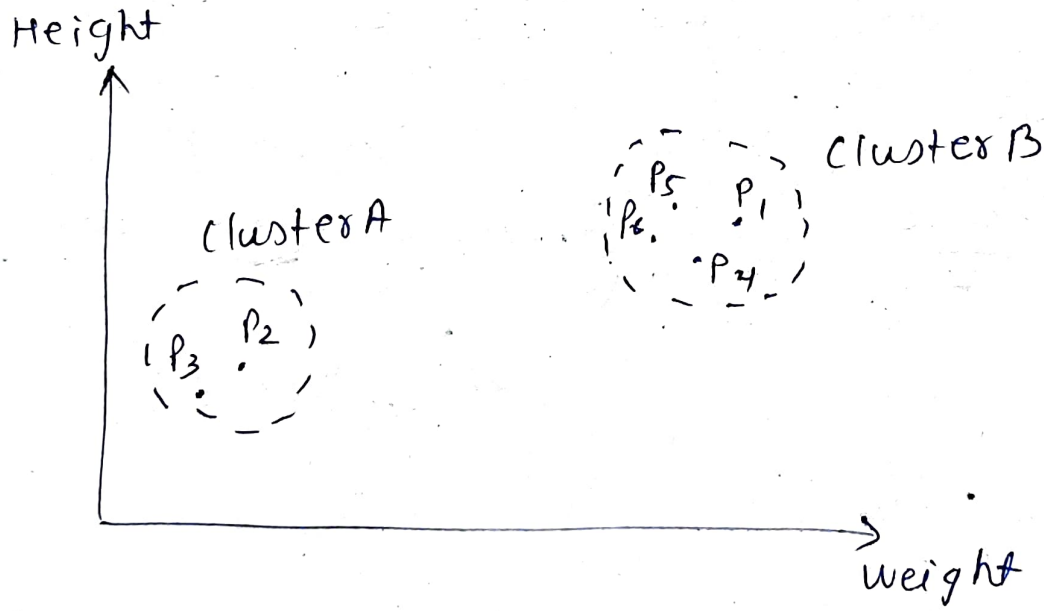
(169, 58)

cluster B

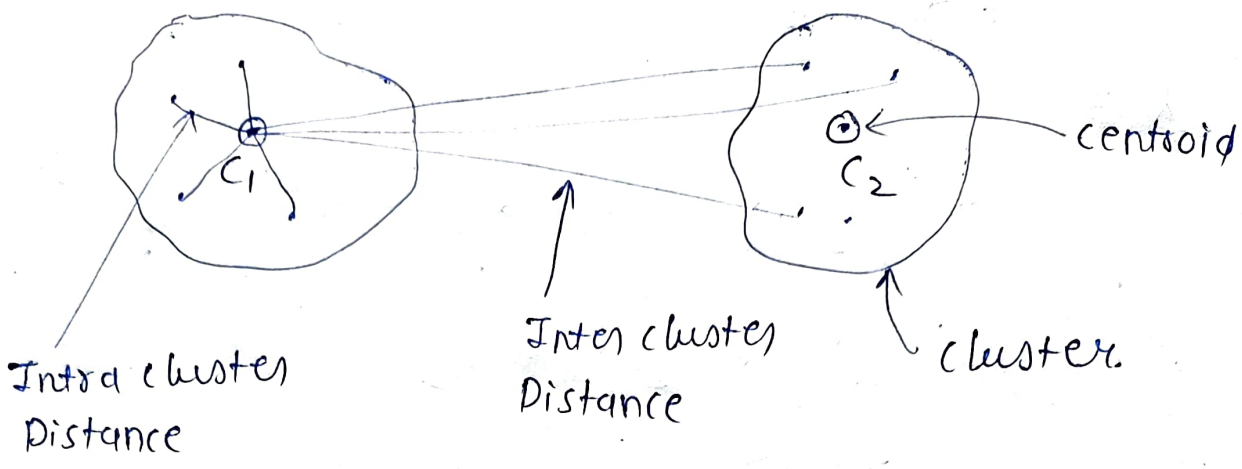
$\left(\frac{185+179+180+182}{4}, \frac{72+68+71+72}{4}\right)$

(181.5, 70.75)

∵ centroids in previous step and this step are same we have successfully clustered our data points



Terminology Related to clusters



Here $K = 2$, and 2 centroids

Qu: How to decide No. of clusters in Step 1

⇒ For that we have Elbow method

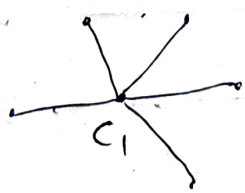
→ In Elbow method we plot graph blw wcss and k value

→ Here we take $K = 1, 2, 3, 4, \ldots$ max 20, and for each k value we calculate WCSS. ie within cluster sum of squares. and plot.

→ The K point at which there is no abrupt change in wcss value is our optimum cluster value.

For example:

$K = 1$



wcss,

$K = 2$

$wcss_2 = (wcss)_{c_1} + (wcss)_{c_2}$

c_1 c_2

$K = 3$

c_1 c_2 c_3

$wcss_3 = (wcss)_{c_1} + (wcss)_{c_2} + (wcss)_{c_3}$

Also $(wcss)_1 > (wcss)_2 > (wcss)_3$.

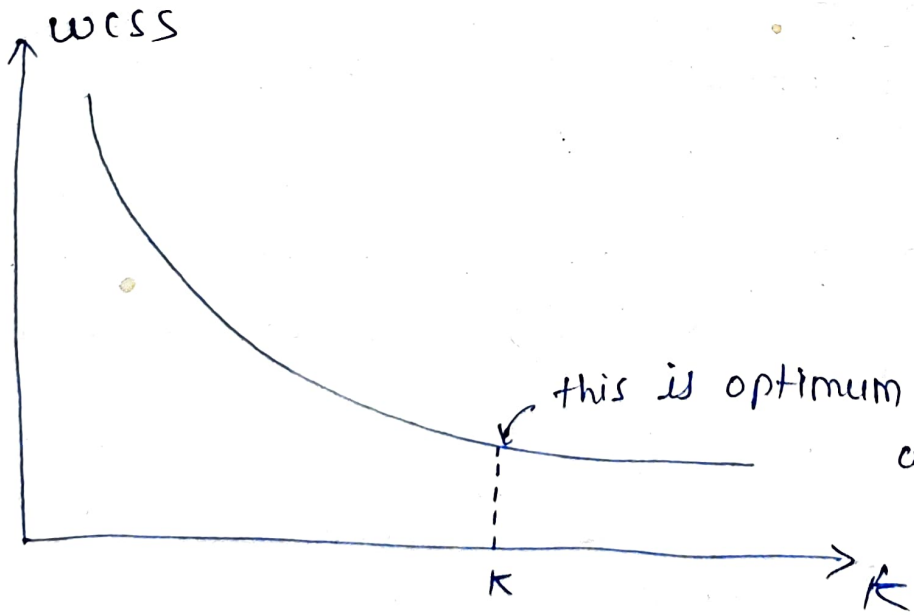
The above statement can be understood using below Argument:

consider 10 data points

→ for clusters $K = 1$ (wcss), will be maximum.

→ for K = No. of data points ie $K = 10$.
each point will act as centroid
ie wcss will be zero.

so we can say that as K value increases wcss value decreases.



$$WCSS = \sum_{i=1}^{n} d(c, x_i)^2$$

where $n \equiv$ no. of points in c cluster.

$c \equiv$ centroid

$x_i \equiv$ i^{th} point in c cluster.

How to evaluate quality of clusters?

=> This is done using

1) dunn Index

2) silhoutte score

1) Dunn index

$$\text{Dunn index} = \frac{\min(\text{inter cluster distance})}{\max(\text{Intra cluster distance})}$$

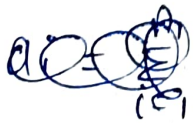
Dunn index ↑ Quality of cluster ↑

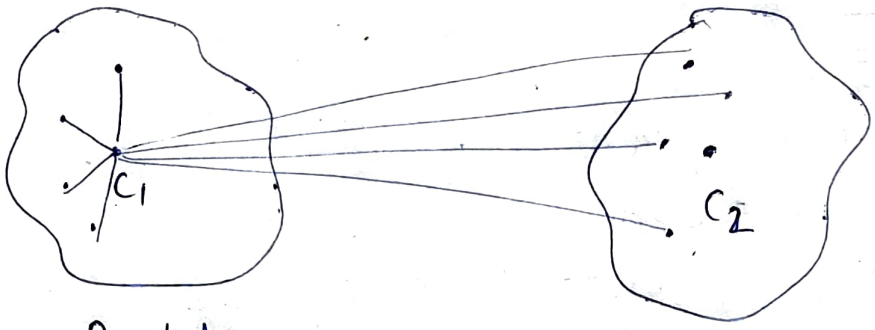
2) silhoutte score

$$\text{silhoute score} = \frac{bi - ai}{\max(bi, ai)}$$

where $bi \equiv$ (inter cluster distance) => mean

$ai \equiv$ (intra cluster distance) => mean





n_1 data points

n_2 data points

$$a_i = \sum_{i=1}^{n_1} \frac{d(C_1, x_{1i})}{n_1}$$

⇒ mean of intra cluster distance

$$b_i = \sum_{i=1}^{n_2} \frac{d(C_1, x_{2i})}{n_2}$$

⇑ mean of Inter cluster distance.

where

x_{1i} ⇒ i^{th} point in 1^{st} cluster.

x_{2i} ⇒ i^{th} point in 2^{nd} cluster.

silhouette score ranges b/w -1 to 1 with -1 being worst and 1 being best.

K-Means Clustering

1.0 Importing required libraries

In [26]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import plotly.express as px

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

import warnings
warnings.filterwarnings('ignore')
```

2.0 Importing 2 feature dataset for clustering

In [3]:
```
data=pd.read_csv("data.csv")
data.head()
```

Out[3]:

	x	y	color
0	516.012706	393.014514	0
1	436.211762	408.656585	0
2	512.052601	372.022014	0
3	489.140464	401.807159	0
4	446.207986	338.516682	0

In [4]:
```
### dropping label color feature
dataset=data.drop('color', axis=1)
dataset.head()
```

Out[4]:

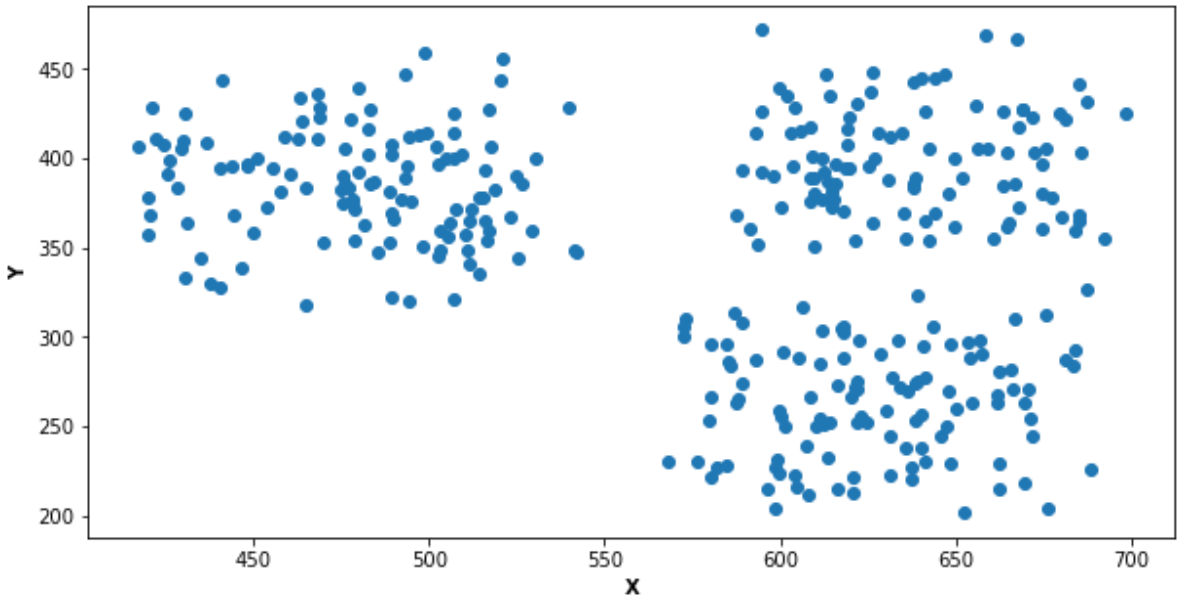
	x	y
0	516.012706	393.014514
1	436.211762	408.656585
2	512.052601	372.022014
3	489.140464	401.807159
4	446.207986	338.516682

In [7]:
```
dataset.shape
```

Out[7]:
```
(336, 2)
```

In [49]:
```
### Visualising dataset
plt.figure(figsize=(10,5))
plt.scatter(data= dataset, x='x', y='y')
plt.xlabel("X", fontsize=10, fontweight='bold')
plt.ylabel("Y", fontsize=10, fontweight='bold')
```

Out[49]:
```
Text(0, 0.5, 'Y')
```

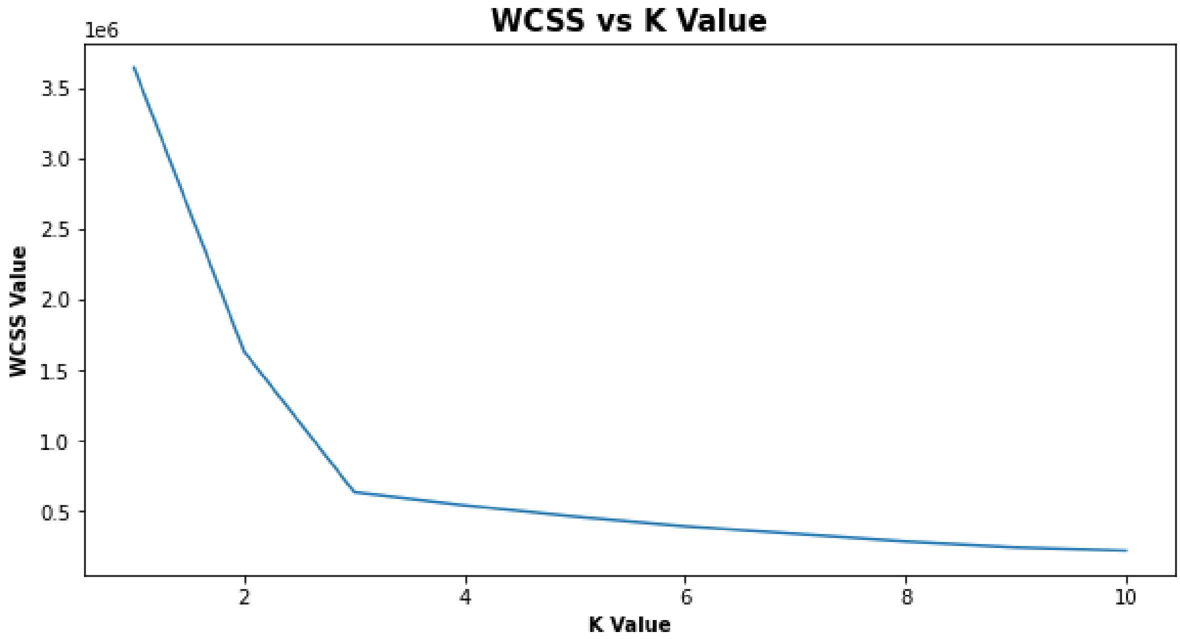


In [41]:
```
### Calculating WCSS for K=1 to K=11 clusters
wcss=[]


for i in range(2,11):
    km = KMeans(n_clusters=i)
    km.fit_predict(dataset)
    wcss.append(round(km.inertia_,2))

print(f"WCSS list: {wcss}")
```

```
WCSS list: [1634662.14, 637691.33, 544675.04, 467501.54, 401945.06, 338585.28, 287
823.43, 246019.25, 224975.35]
```

In [10]:
```
### Plotting WCSS with K
plt.figure(figsize=(10, 5))
plt.plot(range(1,11), wcss)
plt.xlabel("K Value", fontsize=10, fontweight='bold')
plt.ylabel("WCSS Value", fontsize=10, fontweight='bold')
plt.title("WCSS vs K Value", fontsize=15, fontweight='bold')
```

Out[10]:
```
Text(0.5, 1.0, 'WCSS vs K Value')
```



In [12]:
```
### Getting labels of clustered records
X= dataset.iloc[:,:].values
km = KMeans(n_clusters=3)
y_means=km.fit_predict(X)
y_means
```

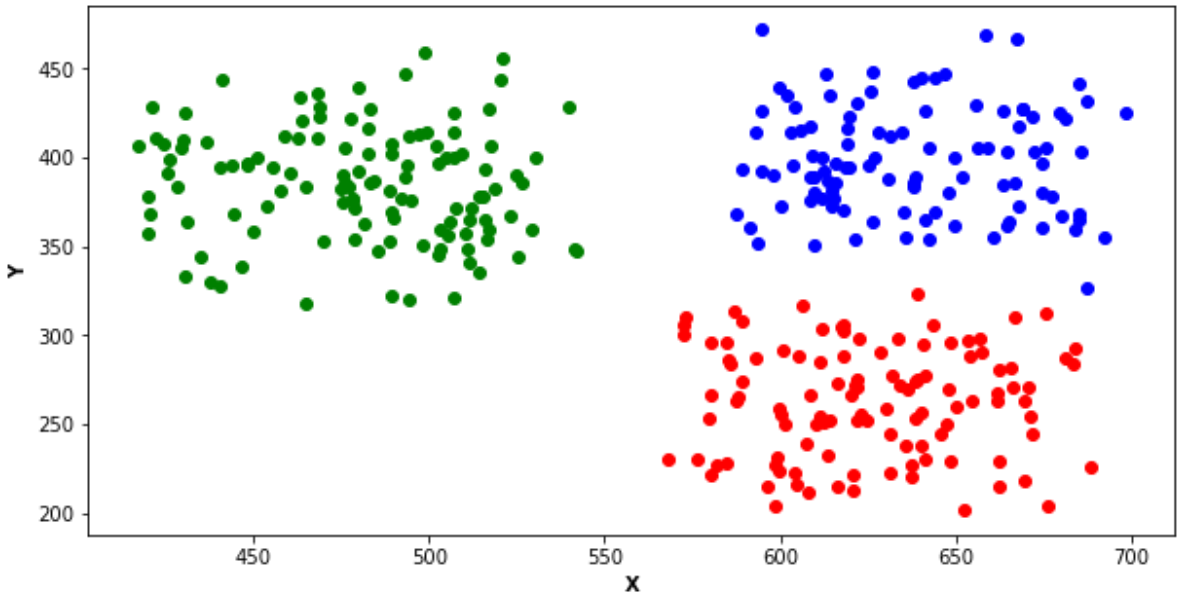
Out[12]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 0,
       0, 1, 2, 0, 2, 0])
```

In [30]:
```
### Silhoutte score
score= round(silhouette_score(X, km.labels_, metric='euclidean'),3)
score
```

Out[30]:
```
0.735
```

In [13]:
```
### Plotting different clusters after K-Means Clustering
plt.figure(figsize=(10,5))
plt.scatter(X[y_means==0,0], X[y_means==0,1], color='blue')
plt.scatter(X[y_means==1,0], X[y_means==1,1], color='red')
plt.scatter(X[y_means==2,0], X[y_means==2,1], color='green')
plt.xlabel("X", fontsize=10, fontweight='bold')
plt.ylabel("Y", fontsize=10, fontweight='bold')
```

Out[13]:
```
Text(0, 0.5, 'Y')
```



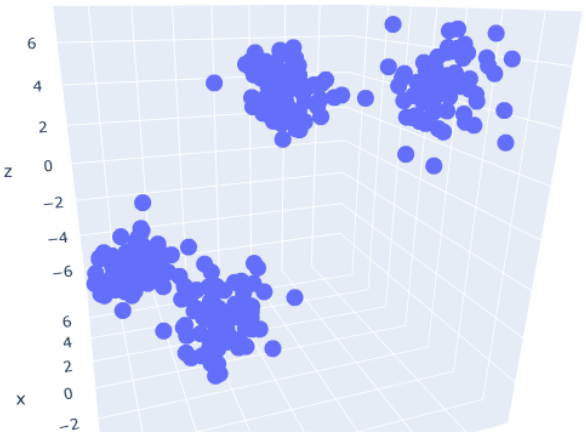
3.0 Creating 3-D dataset for clustering

In [43]:
```
### defining four centroid
centroids=[(-5,-5,5), (5,5,-5), (3.5,-2.5, 4), (-2.5, 2.5,-4)]

### standard deviation for cluster
cluster_std=[1,1,1,1]

### Creating 300 samples
X,y=make_blobs(n_samples=300, cluster_std=cluster_std, centers=centroids, n_feature
```

In [44]:
```
### Visualizing dataset
fig= px.scatter_3d(x=X[:,0], y=X[:,1], z=X[:,2])
fig.show();
```

In [51]:
```
from IPython import display
display.Image("wo_cluster.png")
```

Out[51]:



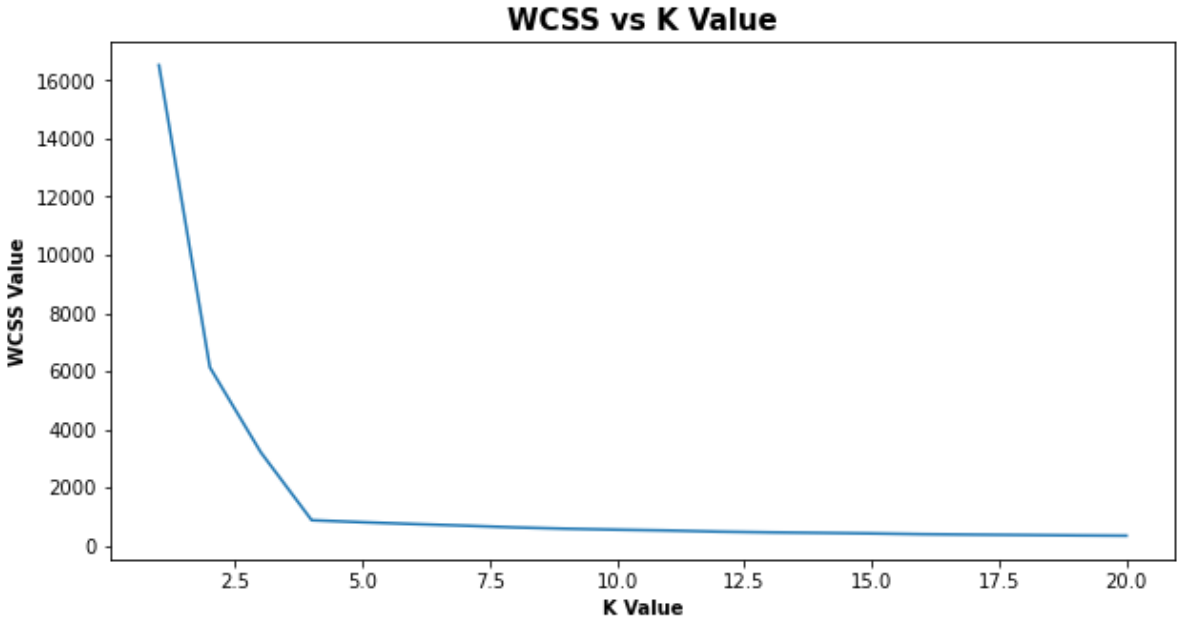
In [45]:
```
### Calculating WCSS for K=1 to K=21 clusters
wcss=[]

for i in range(1,21):
    km = KMeans(n_clusters=i)
    km.fit_predict(X)
```

```
    wcss.append(round(km.inertia_,2))
print(wcss)
```

```
[16520.33, 6135.27, 3215.25, 879.2, 810.02, 748.86, 693.28, 632.22, 587.3, 554.04,
525.92, 486.34, 461.4, 444.53, 424.88, 398.86, 384.19, 372.72, 357.65, 347.18]
```

In [46]:
```
### Plotting WCSS with K
plt.figure(figsize=(10, 5))
plt.plot(range(1,21), wcss)
plt.xlabel("K Value", fontsize=10, fontweight='bold')
plt.ylabel("WCSS Value", fontsize=10, fontweight='bold')
plt.title("WCSS vs K Value", fontsize=15, fontweight='bold')
```

Out[46]:
```
Text(0.5, 1.0, 'WCSS vs K Value')
```



In [47]:
```
### Getting labels of clustered records
km = KMeans(n_clusters=4)
y_pred=km.fit_predict(X)
y_pred
```

Out[47]:
```
array([0, 1, 1, 1, 0, 0, 1, 2, 0, 2, 3, 3, 3, 2, 3, 1, 2, 2, 1, 1, 0, 1,
       2, 1, 0, 2, 2, 1, 2, 0, 1, 0, 3, 3, 1, 1, 0, 2, 3, 0, 3, 1, 0, 2,
       2, 1, 2, 1, 3, 1, 0, 0, 1, 2, 0, 3, 0, 1, 0, 0, 3, 1, 1, 3, 0, 3,
       3, 0, 3, 0, 3, 3, 0, 3, 0, 3, 2, 3, 3, 2, 2, 1, 3, 3, 2, 2, 3, 3,
       3, 1, 0, 3, 2, 0, 0, 2, 1, 1, 1, 1, 0, 3, 2, 1, 3, 3, 2, 1, 0, 1,
       1, 1, 0, 0, 2, 1, 3, 2, 1, 1, 2, 2, 3, 0, 0, 3, 1, 1, 2, 2, 2, 3,
       2, 2, 0, 2, 3, 0, 3, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 3, 3, 2, 2, 2,
       0, 1, 2, 1, 3, 1, 2, 0, 2, 2, 3, 0, 0, 1, 2, 3, 0, 3, 2, 3, 1, 1,
       3, 2, 1, 0, 0, 1, 2, 2, 3, 3, 1, 3, 2, 1, 1, 0, 2, 3, 1, 1, 3, 2,
       2, 1, 0, 2, 3, 1, 2, 2, 3, 2, 0, 0, 2, 0, 3, 2, 0, 3, 0, 3, 2, 0,
       0, 1, 2, 3, 2, 0, 0, 0, 3, 2, 1, 3, 3, 3, 1, 3, 2, 1, 2, 3, 3, 2,
       0, 1, 0, 3, 1, 1, 3, 1, 0, 0, 3, 3, 1, 0, 3, 0, 2, 2, 1, 1, 1, 2,
       3, 0, 0, 3, 0, 1, 2, 3, 2, 2, 1, 2, 2, 0, 2, 3, 3, 0, 2, 2, 1, 1,
       0, 1, 3, 1, 3, 0, 0, 2, 0, 1, 3, 0, 1, 1])
```

In [48]:
```
### Silhoutte score
score= round(silhouette_score(X, km.labels_, metric='euclidean'),3)
score
```

Out[48]:
```
0.735
```

In [23]:
```
### Creating dataset with labels and clustered data
df = pd.DataFrame()
```

```
df['col1']= X[:,0]
df['col2']= X[:,1]
df['col3']= X[:,2]
df['label']= y_pred

df.head()
```

Out[23]:

	col1	col2	col3	label
0	-4.470535	-4.862299	5.077821	1
1	-4.576802	3.447501	-4.503974	0
2	-2.349984	2.963534	-3.602120	0
3	-1.481448	3.730558	-5.181103	0
4	-4.801700	-4.880991	4.329338	1

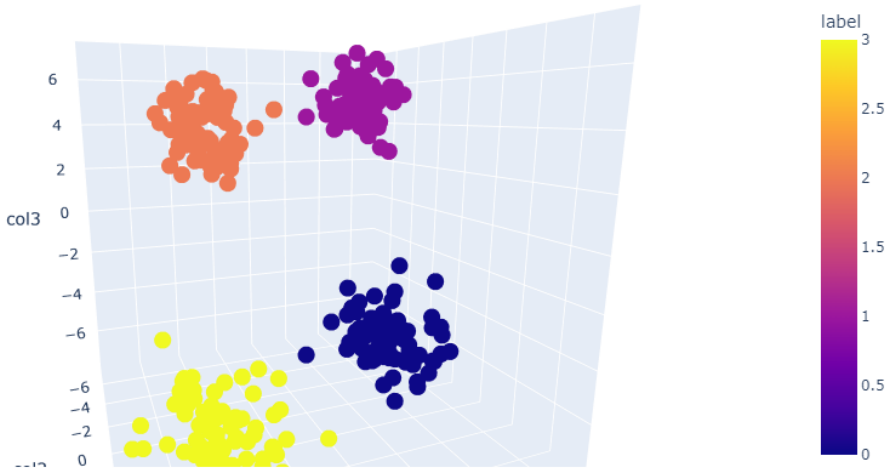
In [24]:

```
### Visualizing the clustered data
fig= px.scatter_3d(df, x='col1', y='col2', z='col3', color='label')
fig.show()
```

In [52]:

```
from IPython import display
display.Image("w_cluster.png")
```

Out[52]:



In [ ]: