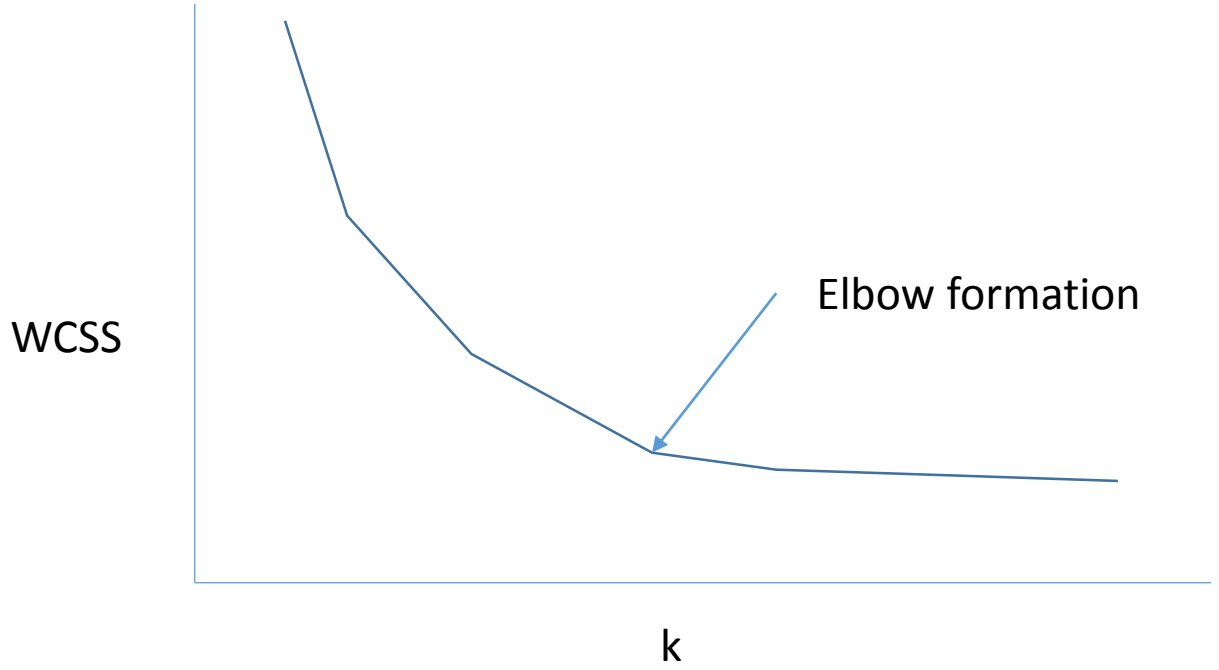
K Mean Clustering

- Picks k number of points for each cluster known as centroids.
- Each data point closest to the centroid become part of that cluster.
- Compute the new centroids based on the members.
- Repeat the above steps, until the process converge (centroids does not change)

Optimal number of clusters

- Plot of Within Cluster sum of Squares (WCSS) and k
- Formation of elbow in the plot – choice of k



Cluster Analysis using K Means Algorithm

- import pandas as pd
- import matplotlib.pyplot as plt
- from sklearn.datasets import load_iris
- iris=load_iris()
- X=iris.data
- y=iris.target

```
#Elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
list1=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,random_state=42)
    kmeans.fit(X)
    list1.append(kmeans.inertia_)


#Scree Plot
plt.plot(range(1,11), list1,marker='o')
plt.title("Elbow method")
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
# Fitting K-means algorithm to the data set
kmeans=KMeans(n_clusters=3, random_state=10)
y_kmeans=kmeans.fit_predict(X)
print("The predicted clusters are: \n", y_kmeans)
print("The centers are: \n", kmeans.cluster_centers_)

#Determining the number of observations in the cluster.
import pandas
data= {'Original': y, 'Predicted' : y_kmeans}
kmeansdf = pandas.DataFrame(data, columns=['Original','Predicted'])
print("Details of predicted clusters are:
\n",kmeansdf["Predicted"].value_counts())
```

```
#creating names of clusters
kmeansdf['Original'].replace(to_replace=int("0"),value="Setosa", inplace=True)
kmeansdf['Original'].replace(to_replace=int("1"),value="Versicolor", inplace=True)
kmeansdf['Original'].replace(to_replace=int("2"),value="Virginaca", inplace=True)
kmeansdf['Predicted'].replace(to_replace=int("0"),value="Setosa", inplace=True)
kmeansdf['Predicted'].replace(to_replace=int("1"),value="Versicolor", inplace=True)
kmeansdf['Predicted'].replace(to_replace=int("2"),value="Virginaca", inplace=True)

from sklearn.metrics import confusion_matrix
results=confusion_matrix(kmeansdf['Original'], kmeansdf['Predicted'])
print("The Confusion Matrix is: \n", results)

#determining the accuracy of the model
from sklearn.metrics import accuracy_score
score=accuracy_score(kmeansdf['Original'], kmeansdf['Predicted'])
print("The accuracy is: ",score.round(2))
```

```
#Visualizing clusters for sepal
plt.figure(2)
plt.figure(2)
plt.subplot(221)
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],c='red')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],c='blue')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],c='green')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=200,c='yellow',label='Centroids')
plt.title('Clusters in iris data set for Sepal')
plt.xlabel('Sepal.Length')
plt.ylabel('Sepal.Width')

#Visualizing clusters for petal
plt.subplot(222)
plt.scatter(X[y_kmeans==0,2],X[y_kmeans==0,3],c='red')
plt.scatter(X[y_kmeans==1,2],X[y_kmeans==1,3],c='blue')
plt.scatter(X[y_kmeans==2,2],X[y_kmeans==2,3],c='green')
plt.title('Clusters in iris data set for Petal')
plt.xlabel('Petal.Length')
plt.ylabel('Petal.Width')
```

```
#Visualizing clusters for sepal.length and petal.length
plt.subplot(223)
plt.scatter(X[y_kmeans==0,1],X[y_kmeans==0,3],c='red',label='Setosa')
plt.scatter(X[y_kmeans==1,1],X[y_kmeans==1,3],c='blue',label='Versicolor')
plt.scatter(X[y_kmeans==2,1],X[y_kmeans==2,3],c='green',label='Virginaca')
plt.title('Clusters in iris data set for Length')
plt.xlabel('Sepal.Length')
plt.ylabel('Petal.Length')

#Visualizing clusters for sepal.width and petal.width
plt.subplot(224)
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,2],c='red',label='Setosa')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,2],c='blue',label='Versicolor')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,2],c='green',label='Virginaca')
plt.title('Clusters in iris data set for Width')
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Width')
plt.show()
```