***Abstract--*** *For the given Iris data set, aim is to full fill the K-means clustering algorithm. Depending on the given attributes of the flower the clustering method should be able to cluster the observation into K clusters such that instances of same species lies in same cluster. This project illustrates the advantages, accuracy and limitations of using K-means clustering on Iris data. It also focuses on choosing the value of K to get the best clustering results.*

**Introduction**

The Iris data is available on the following URL “<http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>”. This data is classified into 3 related species of Iris flower. The classification is done using variation in different attributes of distinct IRIS species. Each specie is sampled for 50 observations. Using K-means clustering we try to recreate this classification and cluster the species having similar attributes together. We thereby compare the clustered model with the original classification. This comparison is based on how accurately the instances in a cluster relate to their original labels.

**What is Clustering Analysis?**

Clustering Analysis is a task of grouping a set of objects in such a way that objects in a group are similar to each other than those in other groups. There are many algorithms that perform clustering analysis like hierarchical clustering, K-means clustering, E-M clustering. [1]

**K-means Clustering**

It is an unsupervised learning algorithm where instances of a data set are clustered into K clusters given some parameters for clustering. These parameters are the attributes of the observations upon which model is supposed to find the similarity between the observations and cluster the similar observations together. For the given dataset the parameters are Petal length, Sepal Length, Sepal width, Petal Width.

K-means partitions the dataset into different clusters depending on the distance of the observations in dataset from the cluster mean called Centroids. The distance of each observation is measured from the K centroids. The observation is then allocated to the cluster whose centroid has Minimum Euclidean Distance from it.

**The K-means Algorithm**

**Initialization Step:**

* Randomly select K observation from the dataset as initial centroids.

**Assignment Step:**

* Measure Euclidean from each observation to the k centroids.
* Assign the observation to the cluster with minimum distance of its centroid from the observation.
* Re-compute the centroid for the cluster.

Run the model till the centroids for each cluster changes with each iteration.

**Convergence:**

* Model converges when the centroid for each cluster does not change and no new instance changes its cluster.

**Complexity of K-means Clustering Algorithm**

The complexity of K-means is O(n\*k\*d\*i). Where

n= number of instances

k= number of clusters

d= parameters

i= number of iterations required to converge.

**Measure of variation in K-means**

The variation in K-means is measure by calculating the sum of square distances of observation in a cluster from its centroid. It is also called as **Sum of Square Error** **(SSE)**

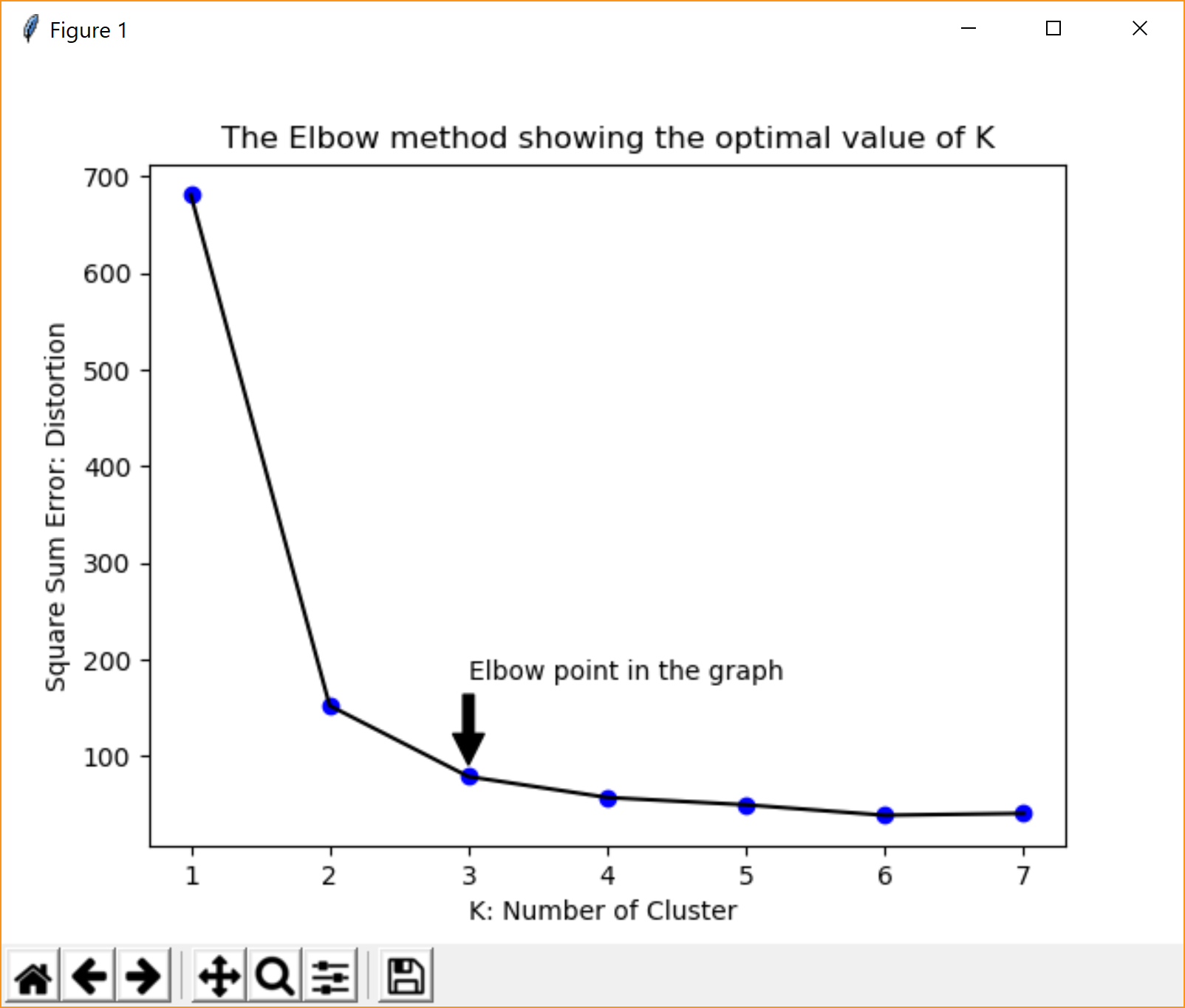
**Large SSE** = Large variations in the clusters (inefficient clustering)

**Small SSE**= Small variation in the clusters (efficient clustering)

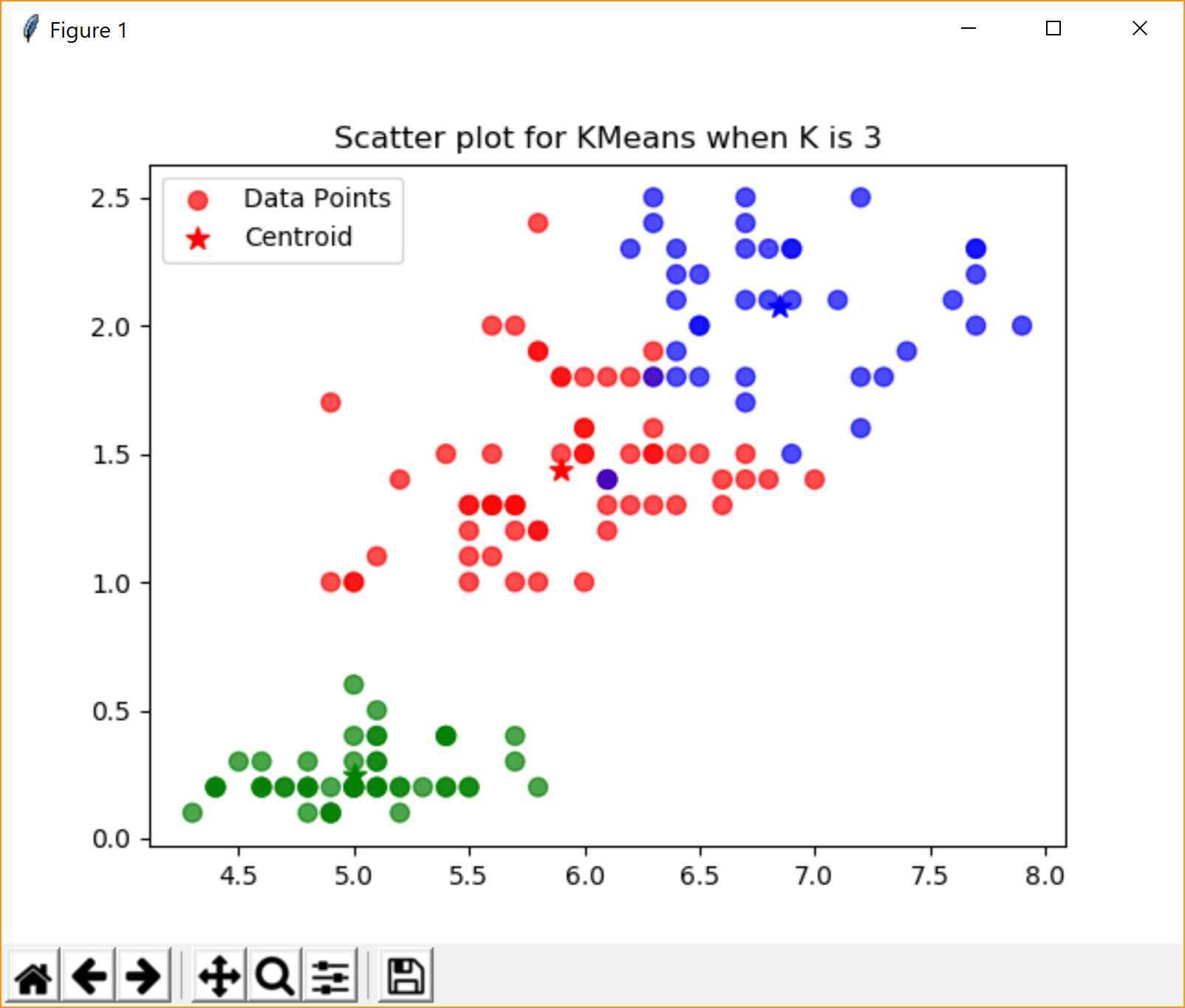
**What should be the value of K**

The value of k in K means can be determined by the **Elbow Method**. If we plot the graph of K (number of clusters) versus Variation(SSE). We can see that increasing the value of K leads to significant decrease in the variation(SSE) [2]. The idea is to choose K at which the SSE decreases abruptly. This produces an elbow effect in the graph.

For the given data set, SSE was calculated when K (1,2,3,4,5,6,7) and graph was plotted. SSE fell abruptly from 152 at K=2 to 75 K=3 and for K=4 and ons the change in SSE was negligible. Hence the elbow of the graph was K=3.



The clustering was thus performed for K =3 and the clustered formed are shown in the figure below



**Accuracy of the Model**

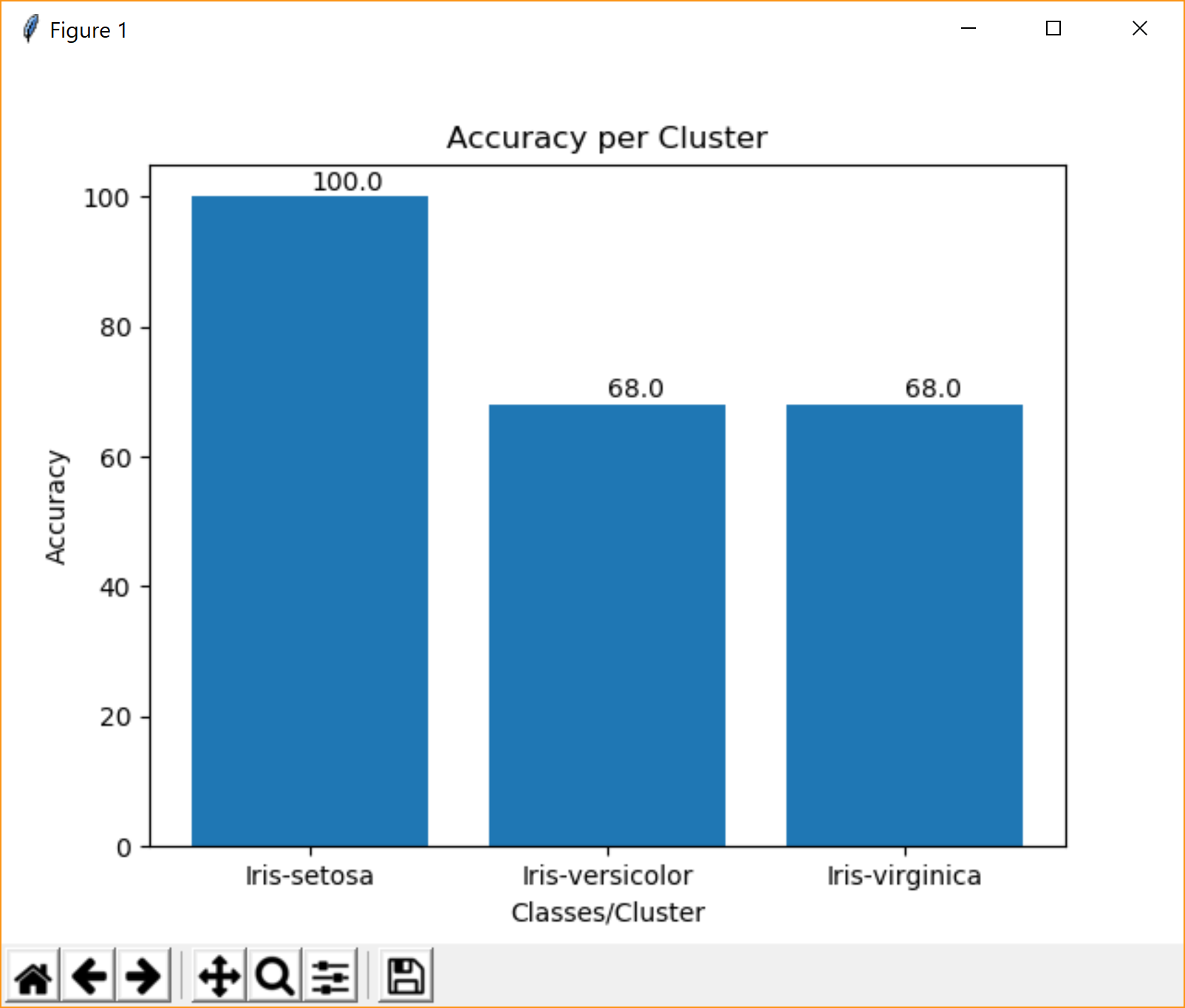
The accuracy of the model for this dataset is based how accurate were the observations clustered to the original classification. The original model had 3 labels and 50 observations under each label.

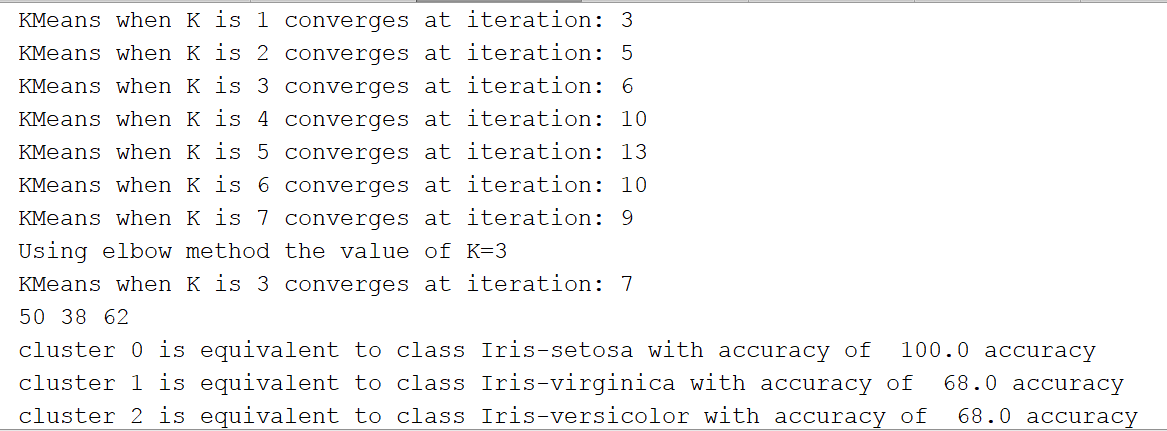
**For accuracy:**

1. Observation of each cluster of the K-means was compared with observation under 3 labels
2. The class was considered equivalent to the cluster in which the matching observation were maximum. For example: For Iris data set the common observation between cluster 1 and class ‘Iris-Setosa’ were maximum. And hence cluster 1 was considered equivalent to class ‘Iris-Setosa’ and similarly cluster 2 ~ ‘Iris Verginica’ and cluster 3 ~ ‘Iris Versicolor’
3. For each equivalent class, number of observations that were dissimilar were calculated.
4. Accuracy for a cluster was then justified as:

(1-(Number of wrong observation in cluster + Number of correct observation missing from the cluster)/50) \*2

For Iris Dataset the accuracy obtained was:





**Limitation of K-means**

The convergence of K-means depends on the choice of the initial centroids. Hence, there is no guarantee that the algorithm will converge to a global optimum. The algorithm converges to local optimum which is subject to change with different initial centroid. Thus, desirable choice of initial centroid is very important.

The results of Algorithm are based on Local optimum, even after running the algorithm multiple times once cannot prove that the results produced by the K-means are global. Thus K-means gives only an idea of how observations might be similar to each other on the specified parameters

The Algorithm is iterated till there is no change in the values of the centroid for each cluster. Although the algorithm is generally fast and converges in less iteration but sometimes it can be very slow in converging. There are proofs that algorithm may take exponential time to converge.

**Conclusion and Results**

Given that K-means clustering is an unsupervised learning algorithm, it does well in clustering the different observations of species in one cluster and distinct species in different clusters.

Algorithm can form 3 clusters each with accuracy:

Cluster 1: 100%

Cluster 2: 68%

Cluster 3: 68%

(*\*****Please note****: These accuracies are subject to change according to the value of initial centroids*)

**References:**

1. <https://en.wikipedia.org/wiki/K-means_clustering>
2. <https://bl.ocks.org/rpgove/0060ff3b656618e9136b>

**Project Code:**

**import** random,csv,math  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** ply  
  
iris\_data=**"C:\\Users\\prach\\Desktop\\New folder\\UTA Ms\\Fall'17\\Machine Learning\\Project\_1\\1001234789\_Prachi\_Goel\\iris.csv"  
  
def** actual\_del(reader):  
 class1=[]  
 class2=[]  
 class3=[]  
 actual\_cluster=[]  
 **with** open(reader,**'r'**) **as** csvread:  
 data=csv.reader(csvread)  
 data=list(data)  
 **for** i **in** range (len(data)):  
 **for** j **in** range(len(data[i])-1):  
 data[i][j]=float(data[i][j])  
 **for** i **in** range(len(data)):  
 **if** data[i][4]==**'Iris-setosa'**:  
 class1.append(data[i][:4])  
 **elif** data[i][4]==**'Iris-versicolor'**:  
 class2.append(data[i][:4])  
 **elif** data[i][4]==**'Iris-virginica'**:  
 class3.append(data[i][:4])  
 actual\_cluster.append(class1)  
 actual\_cluster.append(class2)  
 actual\_cluster.append(class3)  
 **return** actual\_cluster  
  
**def** importing\_data(file):  
 **with** open (file,**'r'**) **as** csvfile:  
 dataset=csv.reader(csvfile)  
 dataset=list(dataset)  
 **for** x **in** range(len(dataset)):  
 dataset[x] = dataset[x][:4]  
 **for** y **in** range(len(dataset[0])):  
 dataset[x][y]=float(dataset[x][y])  
 **return** dataset  
  
**def** initial\_centeroids(dataset,K):  
 number\_of\_clusters=K  
 initial\_centeroids\_of\_clusters=[]  
 rand = random.sample(range(len(dataset)),number\_of\_clusters\*2)  
 **for** i **in** rand:  
 initial\_centeroids\_of\_clusters.append(dataset[i])  
 **return** initial\_centeroids\_of\_clusters  
  
**def** evaluating\_new\_centroid(clusters):  
 temp=np.array(clusters)  
 new\_centroid=[]  
 **for** i **in** range(len(temp)):  
 x=(np.average(temp[i],axis=0))  
 y=x.tolist()  
 new\_centroid.append(y)  
 **return** (new\_centroid)  
  
**def** euclidian\_distance(instance,centroid):  
 summation=0  
 **for** attribute **in** range(len(instance)):  
 summation+=(instance[attribute]-centroid[attribute])\*\*2  
 euclidian=math.sqrt(summation)  
 **return** euclidian  
  
**def** value\_K(dataset):  
 summation = []  
 cluster\_range = [i **for** i **in** range(1,8)]  
 **for** i **in** range(1,8):  
 initial\_centeroid\_Kmeans = initial\_centeroids(dataset, K=i)  
 cluster = kMeans(dataset, initial\_centeroid\_Kmeans, K=i)  
 summation.append(cluster\_validity(cluster[0], cluster[1]))  
 ply.plot(cluster\_range, summation ,**'bo'**, cluster\_range, summation, **'k'**)  
 ply.xlabel(**"K: Number of Cluster "**)  
 ply.ylabel(**"Square Sum Error: Distortion"**)  
 ply.title(**"The Elbow method showing the optimal value of K"**)  
 ply.annotate(**'Elbow point in the graph'**, xy=(cluster\_range[2], summation[2] + 8), xytext=(cluster\_range[2], summation[2] + 100),  
 arrowprops=dict(facecolor=**'black'**, shrink=0.05),  
 )  
 ply.show()  
 print (**"Using elbow method the value of K=3"**)  
  
  
**def** cluster\_validity(cluster,centroid):  
 summation=0  
 **for** i **in** range(len(cluster)):  
 **for** j **in** range(len(cluster[i])):  
 summation+=(euclidian\_distance(cluster[i][j],centroid[i])\*\*2)  
 **return** (summation)  
  
**def** kMeans(dataset,initial\_centroid,K):  
 n=len(initial\_centroid)  
 new\_centroid=initial\_centroid[0:int(n/2)]  
 prev\_centroid=initial\_centroid[int(n/2):n]  
 iteration=0  
 **while** len([p **for** p **in** (new\_centroid) **if** p **not in** prev\_centroid])!=0 **and** iteration<100:  
 clusters=[[] **for** i **in** range (0,K)]  
 **for** i **in** range (len(dataset)):  
 euclidian\_metric=[]  
 **for** j **in** range (len(new\_centroid)):  
 euclidian\_metric.append(euclidian\_distance(dataset[i],new\_centroid[j]))  
 cluster\_value=np.argmin(np.array(euclidian\_metric))  
 clusters[cluster\_value].append(dataset[i])  
 prev\_centroid=new\_centroid  
 new\_centroid=evaluating\_new\_centroid(clusters)  
 iteration+=1  
 print(**"KMeans when K is"**,K,**"converges at iteration:"**, iteration + 1)  
 **return** clusters,new\_centroid  
  
**def** scatter\_plot(clusters,K,centroid):  
 x=[]  
 y=[]  
 center\_x=[]  
 center\_y=[]  
 center\_label=[]  
 labels=[]  
 **for** i **in** range (len(centroid)):  
 center\_x.append(centroid[i][0])  
 center\_y.append(centroid[i][3])  
 center\_label.append(i)  
 **for** i **in** range (len(clusters)):  
 **for** j **in** range(len(clusters[i])):  
 x.append(clusters[i][j][0])  
 y.append(clusters[i][j][3])  
 labels.append(i)  
 label\_color\_map = {0: **'r'**,  
 1: **'g'**,  
 2: **'b'**,  
 3: **'n'**,  
 4: **'k'**,  
 5: **'c'**,  
 6: **'w'**}  
 label\_color = [label\_color\_map[l] **for** l **in** labels]  
 center\_label\_color = [label\_color\_map[l] **for** l **in** center\_label]  
 x=ply.scatter(x,y,s=50,alpha=0.7,c=label\_color,label=**'Data Points'**)  
 y=ply.scatter(center\_x,center\_y,s=80,alpha=1, c=center\_label\_color, marker=**'\*'**,label=**'Centroid'**)  
 ply.title(**'Scatter plot for KMeans when K is '**+str(K) )  
 ply.legend(loc=2)  
 ply.show()  
 **return** x,y  
  
  
  
**def** accuracy(cluster,actual\_data):  
 clusters=[**'Iris-setosa'**,**'Iris-versicolor'**,**'Iris-virginica'**]  
 accu=[]  
 print(len(cluster[0]),len(cluster[1]),len(cluster[2]))  
 **for** i **in** range (len(cluster)):  
 add = []  
 **for** j **in** range(len(actual\_data)):  
 add.append(len([p **for** p **in** cluster[i] **if** p **not in** actual\_data[j]])+len([p **for** p **in** actual\_data[j] **if** p **not in** cluster[i]]))  
 **if** np.argmin(np.array(add))==0:  
 classification=**'Iris-setosa'  
 elif** np.argmin(np.array(add))==1:  
 classification=**'Iris-versicolor'  
 elif** np.argmin(np.array(add))==2:  
 classification=**'Iris-virginica'** print(**"cluster"**,i,**"is equivalent to class"**,classification,**"with accuracy of "**,100-((float(add[np.argmin(np.array(add))])/50)\*100),**"accuracy"**)  
 accu.append(100 - ((float(add[np.argmin(np.array(add))]) / 50) \* 100))  
 ply.bar(clusters,accu)  
 ply.xlabel(**'Classes/Cluster'**)  
 ply.ylabel(**'Accuracy'**)  
 ply.title(**'Accuracy per Cluster'**)  
 **for** i **in** range(len(clusters)):  
 ply.text(clusters[i],accu[i]+1,accu[i],)  
 ply.show()  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 dataset=importing\_data(iris\_data)  
 initial\_centeroid\_Kmeans = initial\_centeroids(dataset, K=3)  
 value\_K(dataset)  
 cluster=kMeans(dataset,initial\_centeroid\_Kmeans,K=3)  
 scatter\_plot(cluster[0],3,cluster[1])  
 actual\_data=actual\_del(iris\_data)  
 accuracy(cluster[0],actual\_data)