# ML- KMeans Algorithm

Name: Shradha Agarwal

**Dataset**: Iris - UCI Machine Learning Repository

Number of features: 4 and Number of classes: 3 (Type of Iris Plant)

Libraries used: pandas, numpy, joblib, matplotlib, sklearn

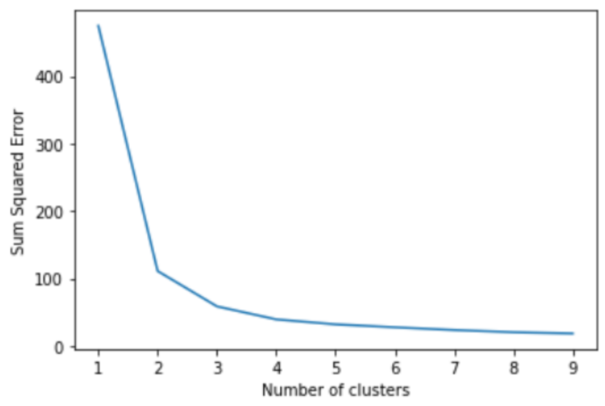
(1) Initially, the labels are converted to codes. Then, the dataset is split into train and validation set with validation size as 30% of the total dataset.

```
def preprocess():
  data = pd.read_csv('/content/drive/My Drive/ML_assignment_4/iris.data', sep=',',header=None)
  data[4] = data[4].astype('category').cat.codes
  label = data.iloc[:,4].copy()
  data.drop(4,axis=1,inplace=True)
  x_train,x_valid,y_train,y_valid = train_test_split(data,label, test_size=0.30, random_state=2)
  return x_train.to_numpy(),x_valid.to_numpy(),y_train.to_numpy(),y_valid.to_numpy()
```

(2). It can be seen from the graph that the elbow of the graph is at number of clusters=3. Thus, the optimal number of clusters is 3.

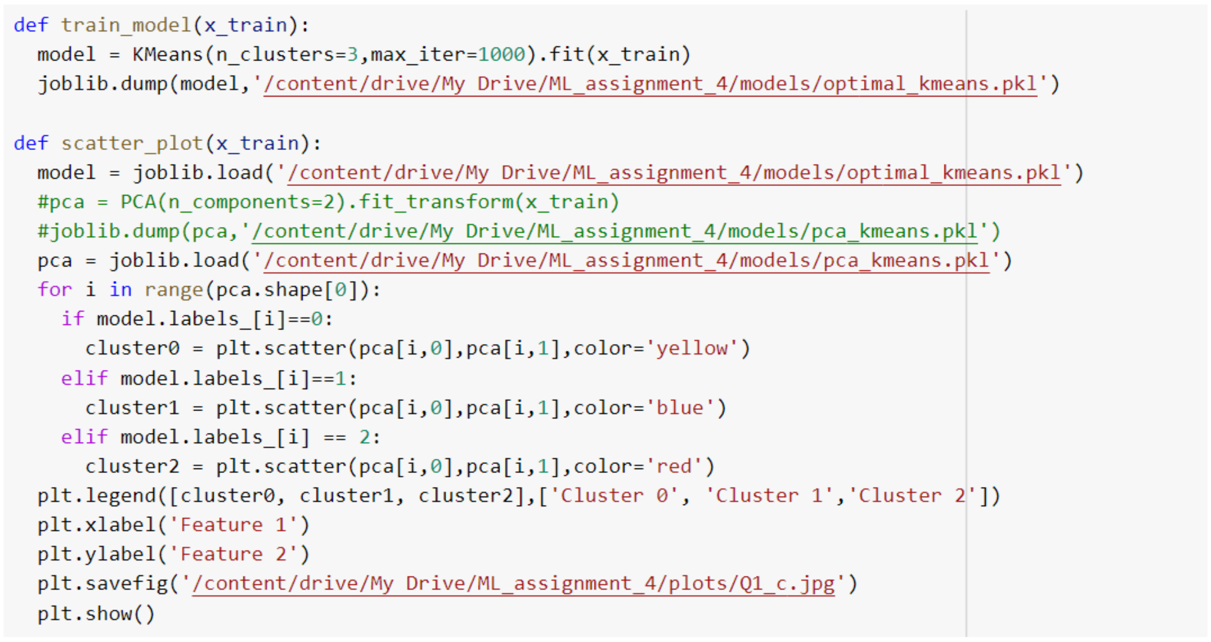
[The elbow method determines the optimal number of clusters for K-Means by plotting the sum of squared errors (SSE) against the number of clusters. As clusters increase, SSE decreases, but after a point, adding more clusters minimally reduces SSE, forming an "elbow". This elbow point balances minimizing SSE and avoiding overfitting due to excessive clusters. Choosing too many clusters risks overfitting, where each data point becomes its own cluster, failing to generalize. The elbow provides a trade-off between clustering quality and model simplicity.]

```
def elbow_method(x_train):
  error=[]
  for i in range(1,10):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(x_train)
    error.append(kmeans.inertia_)
  x_axis=[i for i in range(1,10)]
  plt.plot(x_axis,error)
  plt.xlabel('Number of clusters')
  plt.ylabel('Sum Squared Error')
  #plt.savefig('/content/drive/My Drive/ML_assignment_4/plots/Q1_b.jpg')
  plt.show()
```



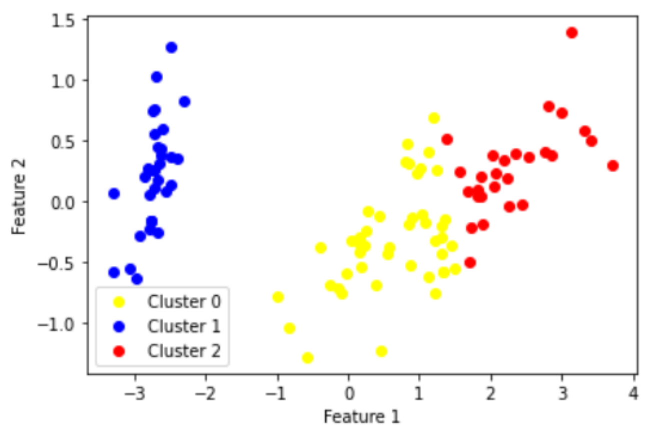
(3). The number of features is reduced to two by PCA and then the clusters are depicted. Three clusters are formed which is shown in the scatter plot below.

[Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving most of the information or variance in the original data. PCA is used to visualize the clusters in a 2D scatter plot since the original Iris dataset has four features, making it difficult to visualize in higher dimensions.]



```python
def train_model(x_train):
  model = KMeans(n_clusters=3,max_iter=1000).fit(x_train)
  joblib.dump(model,'/content/drive/My Drive/ML_assignment_4/models/optimal_kmeans.pkl')

def scatter_plot(x_train):
  model = joblib.load('/content/drive/My Drive/ML_assignment_4/models/optimal_kmeans.pkl')
  #pca = PCA(n_components=2).fit_transform(x_train)
  #joblib.dump(pca,'/content/drive/My Drive/ML_assignment_4/models/pca_kmeans.pkl')
  pca = joblib.load('/content/drive/My Drive/ML_assignment_4/models/pca_kmeans.pkl')
  for i in range(pca.shape[0]):
    if model.labels_[i]==0:
      cluster0 = plt.scatter(pca[i,0],pca[i,1],color='yellow')
    elif model.labels_[i]==1:
      cluster1 = plt.scatter(pca[i,0],pca[i,1],color='blue')
    elif model.labels_[i] == 2:
      cluster2 = plt.scatter(pca[i,0],pca[i,1],color='red')
  plt.legend([cluster0, cluster1, cluster2],['Cluster 0', 'Cluster 1','Cluster 2'])
  plt.xlabel('Feature 1')
  plt.ylabel('Feature 2')
  plt.savefig('/content/drive/My Drive/ML_assignment_4/plots/Q1_c.jpg')
  plt.show()
```



(4). Accuracy obtained on the train set is 92.11% and on the validation set is 91.571%. Both train and validation accuracies are similar. This shows that clusters are well formed by KMeans algorithm.

For calculating iris class label, initially the cluster number is predicted using predict() from sklearn applied on the KMeans model and then the majority iris class label for that cluster is assigned for the given test input.

```python
def accuracy_calculation(x_train,x_valid,y_train,y_valid):
  model = joblib.load('/content/drive/My Drive/ML_assignment_4/models/optimal_kmeans.pkl')
  cluster_label= model.labels_
  unique_clusters = np.unique(cluster_label)
  cluster_iris_label=[]
  for cluster in unique_clusters:
    count =[0,0,0]
    for i in range(len(cluster_label)):
      if cluster_label[i]==cluster:
        count[y_train[i]]+=1
    cluster_iris_label.append(np.array(count).argmax())

  y_pred_train = model.predict(x_train)
  y_pred_train1=[]
  for i in range(len(y_pred_train)):
    y_pred_train1.append(cluster_iris_label[y_pred_train[i]])
  train_accuracy = accuracy_score(y_pred_train1,y_train)
  print('Accuracy Train: ',train_accuracy*100)

  y_pred_valid = model.predict(x_valid)
  y_pred_valid1=[]
  for i in range(len(y_pred_valid)):
    y_pred_valid1.append(cluster_iris_label[y_pred_valid[i]])
  valid_accuracy = accuracy_score(y_pred_valid1,y_valid)
  print('Accuracy Validation: ',valid_accuracy*100)
```

Step 1: Determine the majority class label for each cluster:

For each unique cluster label obtained from the K-Means model, the code counts the number of data points belonging to each class label (0, 1, or 2) within that cluster. The majority class label for a cluster is the class label with the highest count within that cluster. This step is necessary because the K-Means algorithm is an unsupervised learning technique and does not have access to the true class labels during training. By mapping the clusters to the majority class labels, we can associate each cluster with a class label.

Step 2: Predict cluster labels for training and validation data:

The trained K-Means model is used to predict the cluster labels for the training and validation data points. These predicted cluster labels are not the final class labels but rather the cluster assignments based on the K-Means clustering.

Step 3: Map predicted cluster labels to majority class labels:

For each predicted cluster label, the code looks up the corresponding majority class label determined in step 1. This mapping assigns a class label (0, 1, or 2) to each data point based on the cluster it belongs to and the majority class label of that cluster.