**Miner Username: Michael_Scarn**

**Miner Rank Part 1:117,Part 2: 65**

**Accuracy part 1: 0.79, part 2: 0.78**

# Part 1 K means Algorithm on Iris data

**Loading and Preparing Data**: I've used pandas to read data from the train file using *'pd.read_table'* method.

K means Algorithm:

1.To consider the initial centroids I have taken them randomly. This I've written as 'init_cen' function which takes in 'k' number of clusters and 'shape' which is the number of features considered. For example, in iris dataset as there are 4 features 'sepal length in cm','sepal width in cm','petal length in cm','petal width in cm'.And the number of clusters in this case is three.

2.In the next step I used Euclidean distance of each point in the data set with the identified K points — cluster centroids. For this I've written 'distance' function which takes in two data points as it's arguments.

3.The next step in the algorithm is to Assign each data point to the closest centroid using the distance found in the previous step.

4.After the previous step we have to Find the new centroid by taking the average of the points in each cluster group.

5. Foe every iteration steps 2,3,4 is repeated till the centroids don't change. That is till we reach a point of convergence.

V-measure results for K means algorithm on iris data

| Iterations | V_measure |
|---|---|
| 1 | 0.53 |
| 3 | 0.55 |
| 6 | 0.55 |
| 9 | 0.67 |
| 10 | 0.79 |
| 20 | 0.79 |
| 50 | 0.79 |

From the above results we can infer that the k means algorithm has converged at just the $10^{th}$ iteration .That is very few points tend to change the centroids of the clusters.

## Part 2 : K means Algorithm on image dataset

**Loading and Preparing Data**: I've used pandas to read data from the train file using *'pd. read_csv'* method.

Applying Dimensionality reduction:

For Dimensionality reduction I have used principal component analysis and T-distributed Stochastic Neighbor Embedding. PCA is a linear dimensionality reduction technique which converts a set of correlated features in the high dimensional space into a series of uncorrelated features in the low dimensional space. These uncorrelated features are also called principal components.

I have used ' from sklearn.decomposition import PCA' package to implement PCA.

For several variance ranges of PCA. The results are shown below:

| Variance limit | Features After PCA | V_measure |
|---|---|---|
| 0.65 | 20 | 0.67 |
| 0.75 | 32 | 0.78 |
| 0.85 | 57 | 0.76 |
| 0.95 | 149 | 0.71 |

t-SNE is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.[reference from sklearn documentation]

As the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples.

Without tsne and pcs dimensionality reduction if all features are considered I was able to get only V_measure score of 0.50-0.52

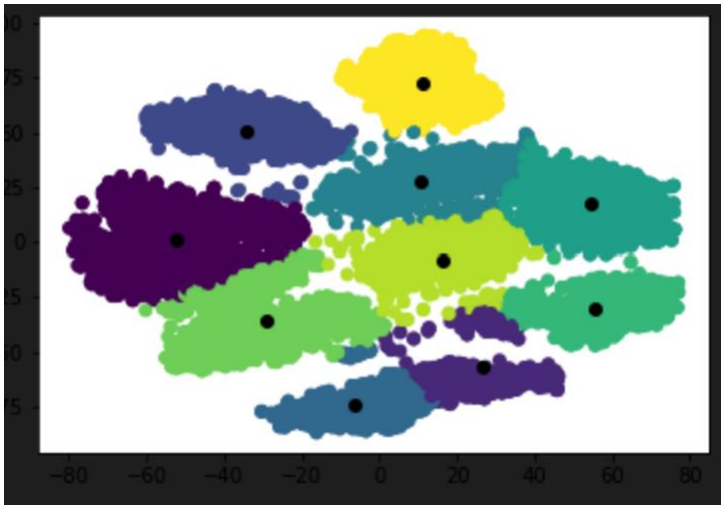
Code :

```
pca = PCA(.95)

pca.fit(datas)

pca_data = pca.transform(datas)
```

*tsne_data = TSNE(n_components=2).fit_transform(pca_data)*




The above visualization shows the clusters after running the kmeans algorithm.

## Internal evaluation metric

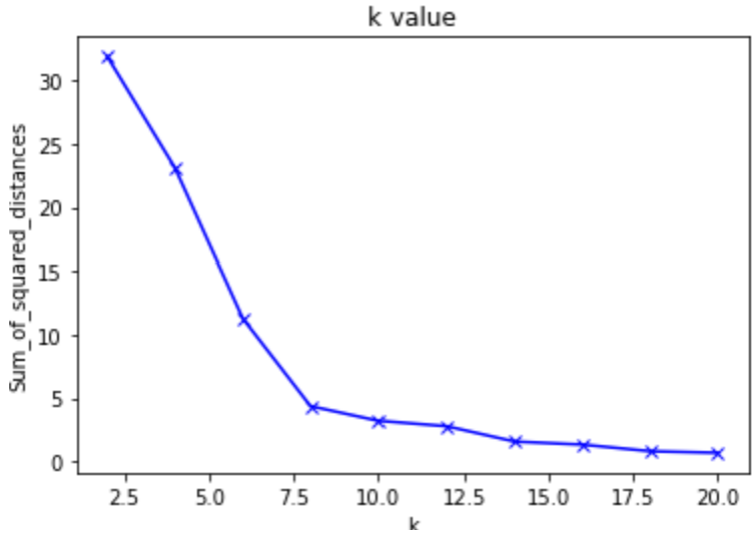
I have used sum of squared error for the internal evaluation metric.This is based on from the lecture slides:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

Where for each point, the error is the distance to the nearest center. To get SSE, we square these errors and sum them. x is a data point in cluster Ci and mi is the representative point for cluster Ci. can show that mi corresponds to the center (mean) of the cluster.

For K increasing from 2 to 20 in steps of 2 for the data. The graph plotted with SSE on y-axis and k values on X axis is shown below:

| K value | SSE |
|---|---|
| 2 | 31.1 |
| 6 | 23.1 |
| 8 | 11.2 |
| 10 | 4.36 |
| 12 | 3.2 |
| 14 | 2.77 |
| 16 | 1.56 |
| 18 | 1.32 |
| 20 | 0.67 |




The above graph should the change in SSE values based on the increasing K on Image dataset. From that we can conclude that as the number of k clusters increased the error rate decreased indicating that the unsupervised  k means performs better with more clusters on large datasets.

Kmeans using Sklearn Library :

When I used kmeans++ from sklearn package I could get around 0.71 v_measure for iris and 0.52 for image dataset. Even if the number of iterations were increased it remained indicating that the kmeans has converged. As it is a unsupervised learning algorithm  to generate better v_measure we should increase the number of clusters k.

References:

Dimensionality Reduction: Principal Component Analysis | by Kiran Parte | Analytics Vidhya | Medium

K-Means Clustering: Python Implementation from Scratch | by Khushijain | Nerd For Tech | Medium