# Assignment Project Report

## Spherical K-Means: Pattern Discovery in Textures

**Name :** Meghna Maan

**Course :** AI & ML (Batch 4)

### Given Question

Generate a dummy dataset using Scikit-Learn having high dimensionality (number of features >10) and total 4 classes. For this dataset, first implement K-Means clustering and then use the clusters for classification purpose. Now using the same dataset, implement spherical clustering and then check accuracy for classification. Notice the change in accuracy. You may also plot the obtained clusters from both the methods using t-SNE plots or by projecting data into two dimensions using PCA.

### Prerequisites

1. Software:

    Python 3

2. Tools:

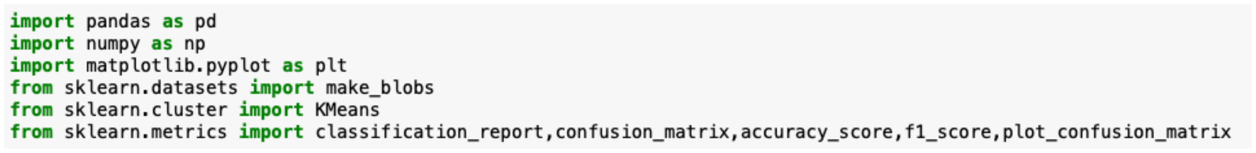
    - Numpy
    - Pandas
    - OpenCv

### Methods Used

The classical k-means method of clustering minimizes the sum of squared distances between cluster centres and cluster members. The intuition is that the radial distance from the Cluster- Centre should be similar for all elements of that cluster. The spherical k-means algorithm, however, is equivalent to the k-means algorithm with cosine similarity, a popular method for clustering high-dimensional data. The idea is to set the centre of each cluster such that it makes the angle between components both uniform and minimal.

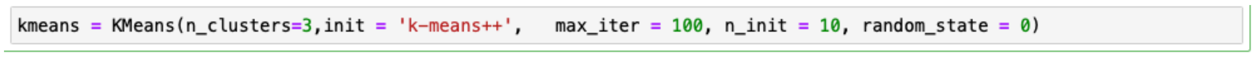
## Implementation

1. Loading Libraries and Dataset

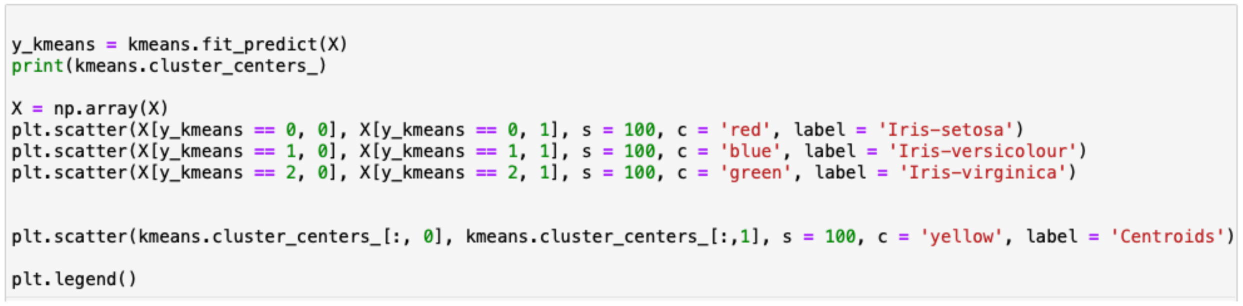


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,f1_score,plot_confusion_matrix
```

2. Implementing K-Means classifier and plotting centroid of cluster

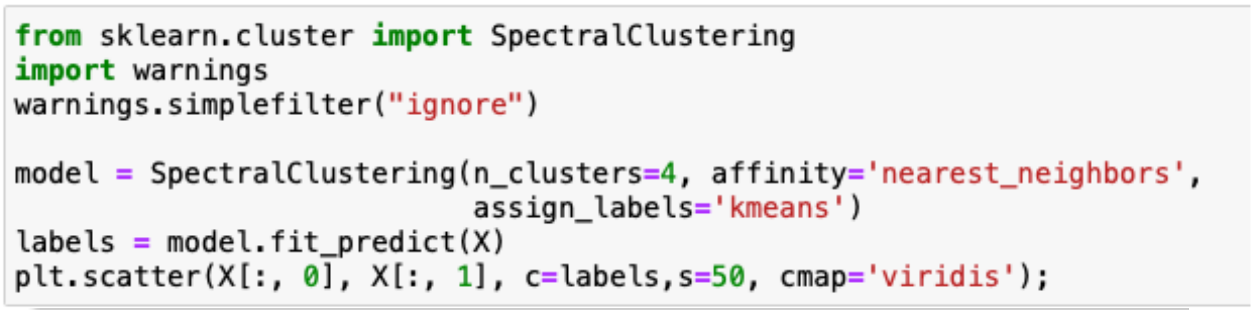


```
kmeans = KMeans(n_clusters=3,init = 'k-means++',   max_iter = 100, n_init = 10, random_state = 0)
```

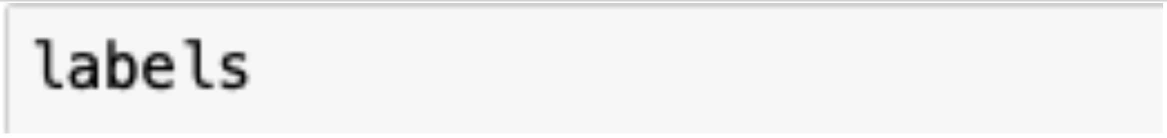


```
y_kmeans = kmeans.fit_predict(X)
print(kmeans.cluster_centers_)

X = np.array(X)
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')


plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

3. Importing Sklearn and printing labels

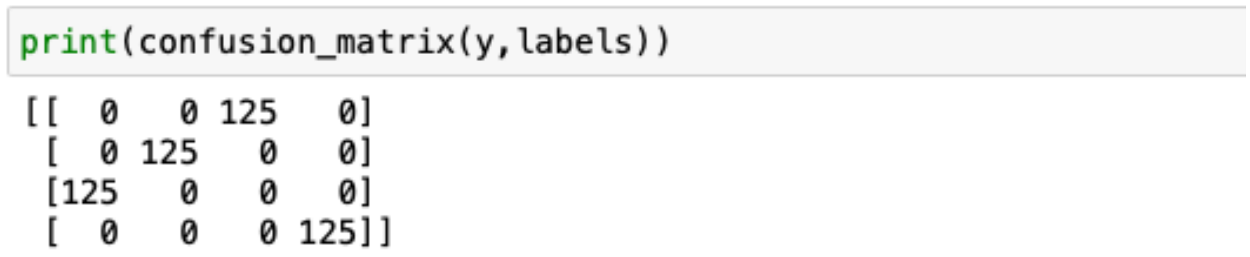


```
from sklearn.cluster import SpectralClustering
import warnings
warnings.simplefilter("ignore")

model = SpectralClustering(n_clusters=4, affinity='nearest_neighbors',
                           assign_labels='kmeans')
labels = model.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels,s=50, cmap='viridis');
```



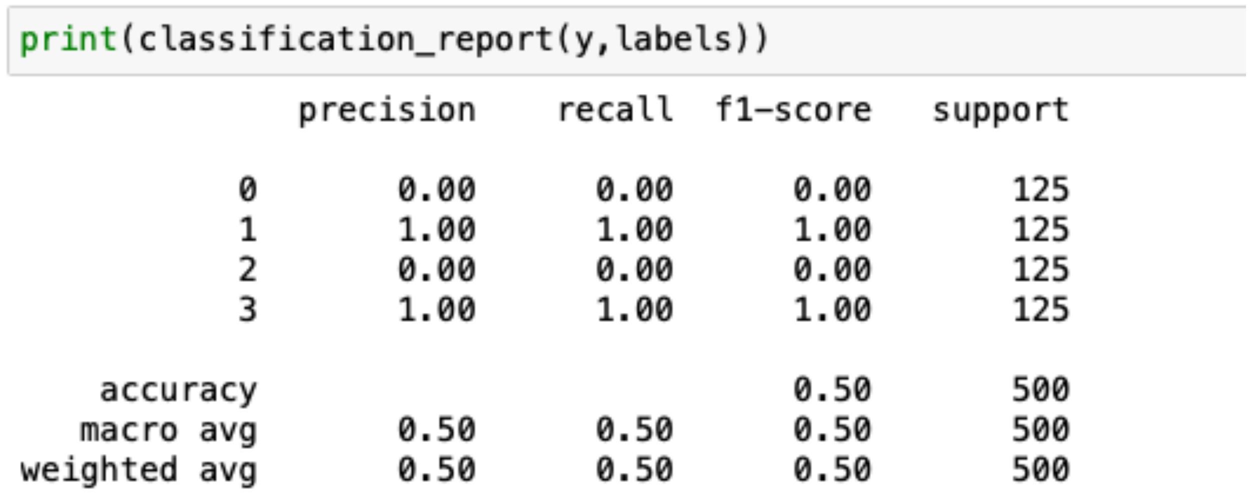
```
labels
```

4. Printing Confusion Matrix and Classification Report



```
print(confusion_matrix(y,labels))
```

```
[[  0   0 125   0]
 [  0 125   0   0]
 [125   0   0   0]
 [  0   0   0 125]]
```



```
print(classification_report(y,labels))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       125
           1       1.00      1.00      1.00       125
           2       0.00      0.00      0.00       125
           3       1.00      1.00      1.00       125

    accuracy                           0.50       500
   macro avg       0.50      0.50      0.50       500
weighted avg       0.50      0.50      0.50       500
```

Fin.