Machine Learning Lab program 8

Name – Pranshu Aggarwal
Batch– CSE 3
Enrollment No. – 40576802717

AIM : Select two datasets. Each dataset should contain examples from multiple classes. For training purposes assume that the class label of example is unknown. Apply the k-means algorithm and apply it to the selected data. Evaluate the process by measuring the sum of euclidean distance of each example from its class center. Test the performance of the algorithm as a function of the parameter k.

Github Link :- https://github.com/pranshuag9/machine-learning-lab/blob/main/lab9/Program8.ipynb

Algorithm

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way kmeans algorithm works is as follows:

- Specify number of clusters K.
- Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
- Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all-data points that belong to each cluster.

Program Snippets :-

1. Importing Modules

```
K-Means algorithm on two different datasets

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.style.use('dark_background')
```

2. Loading dataset :-

```
Iris Dataset

In [15]: data = pd.read_csv('Iris.csv')

In [29]: data.head()
```

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82

```
Wine Dataset

Loading dataset

In [20]: data = pd.read_csv('wine.csv')
```

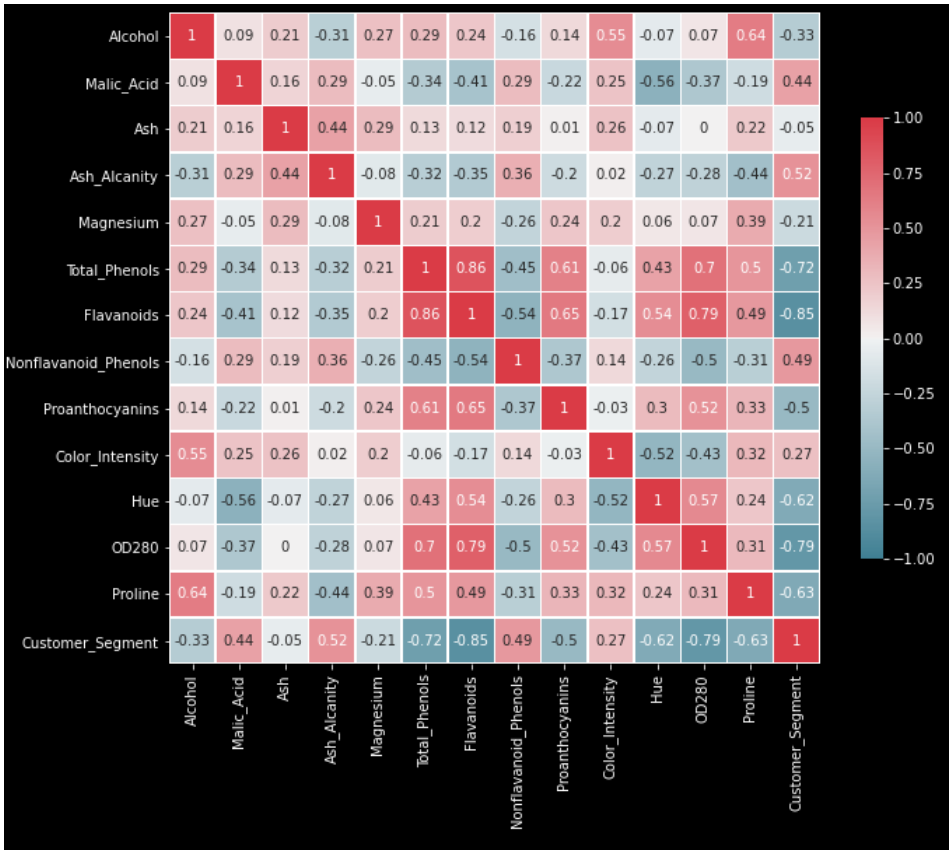
3. Data Visualization

Heatmap

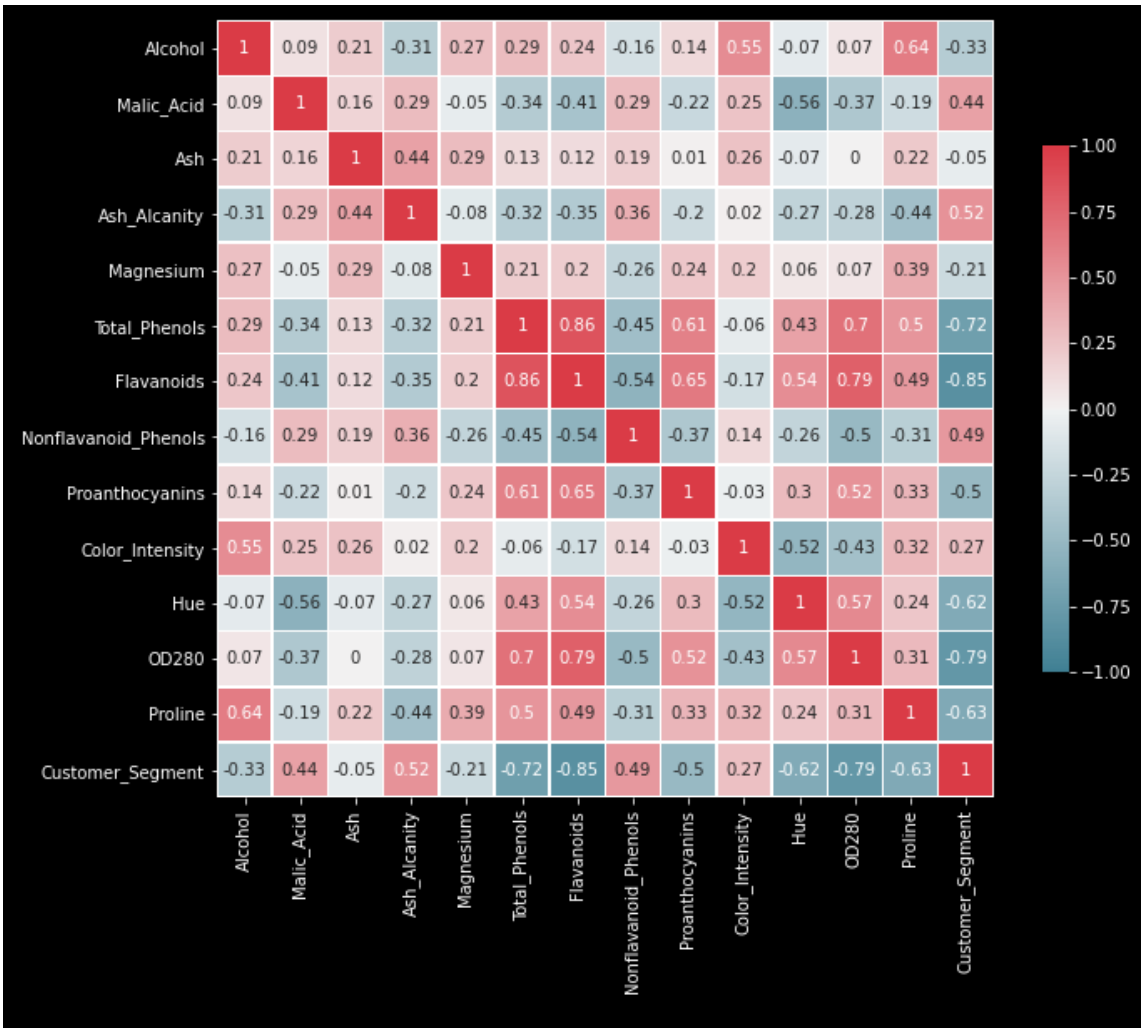
In [31]:

```
#generate the corelation matrix
corr=data.corr().round(2)
#mask for the upper triangle
mask=np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)]
# Set figure size
f, ax = plt.subplots(figsize=(10, 10))

#define custom colormap
cmap=sns.diverging_palette(220,10, as_cmap=True)

#draw the heatmap
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)

plt.tight_layout()
plt.savefig("heatmap.png")
```

- **For Iris dataset**



- **For wine dataset**



4. Converting data into X

```
In [16]: x1 = data.iloc[:, [1, 2, 3, 4]].values
```

5. Model Kmeans

Model

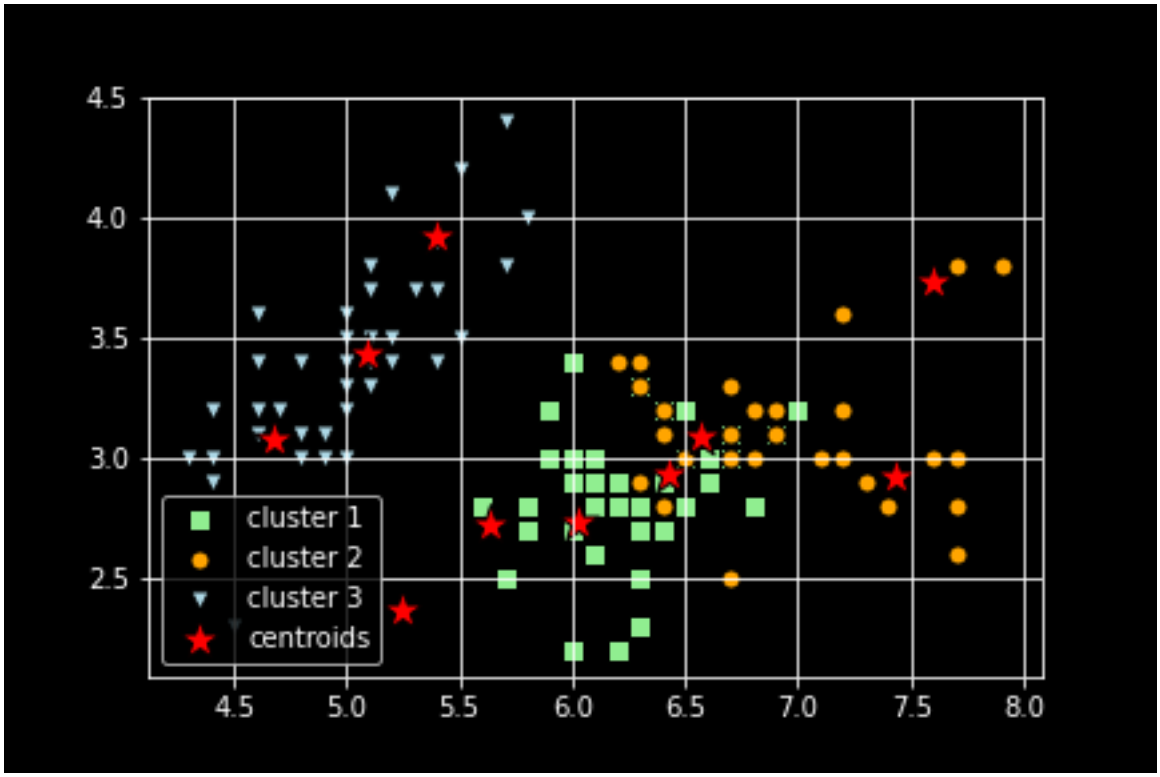
```
In [17]: from sklearn.cluster import KMeans

         km = KMeans(n_clusters=4, init='random',n_init=30, max_iter=300, tol=1e-04, random_state=42)
         y1 = km.fit_predict(x1)
```

6. New points introduced to check k-means Model

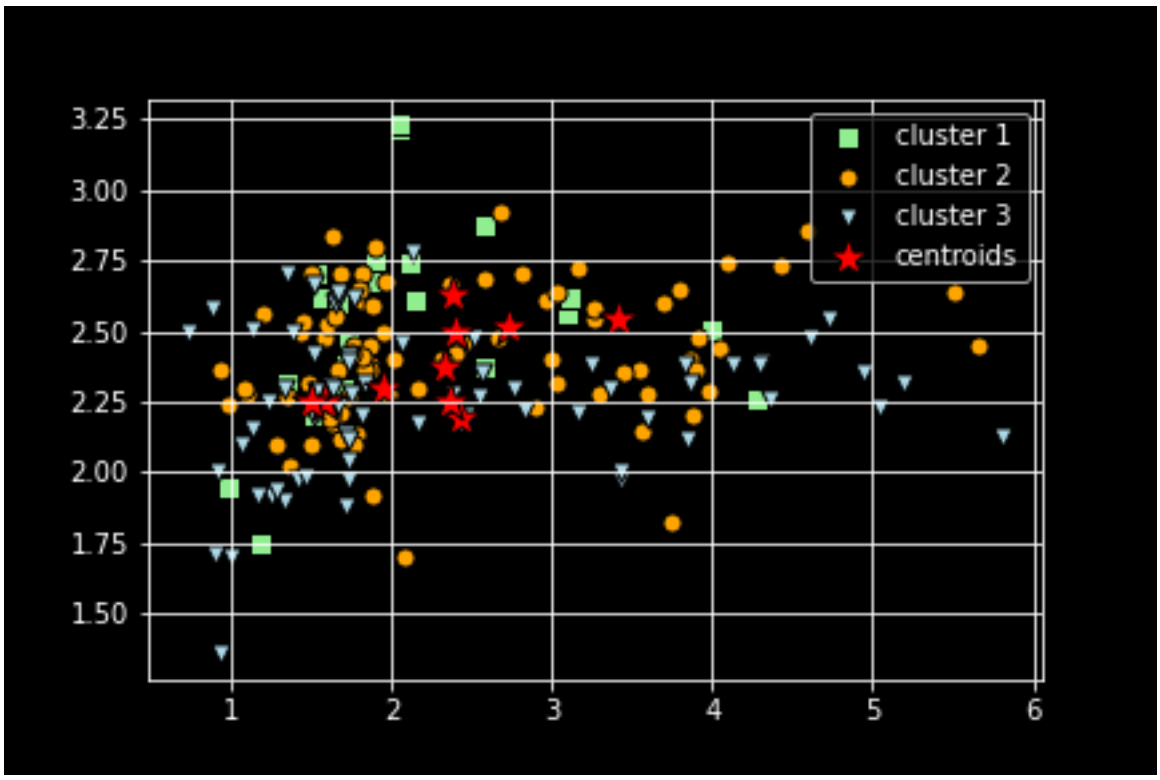
In [28]:

```
plt.scatter(
    x1[y1 == 0, 0], x1[y1 == 0, 1],
    s=50, c='lightgreen',
    marker='s', edgecolor='black',
    label='cluster 1'
)
plt.scatter(
    x1[y1 == 1, 0], x1[y1 == 1, 1],
    s=50, c='orange',
    marker='o', edgecolor='black',
    label='cluster 2'
)
plt.scatter(
    x1[y1 == 2, 0], x1[y1 == 2, 1],
    s=50, c='lightblue',
    marker='v', edgecolor='black',
    label='cluster 3'
)


# plot the centroids
plt.scatter(
    km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],
    s=250, marker='*',
    c='red', edgecolor='black',
    label='centroids'
)
plt.legend(scatterpoints=1)
plt.grid()
plt.savefig("clusters_for-iris.png")
plt.show()
```

- **For Iris Dataset**



- **For Wine Dataset**



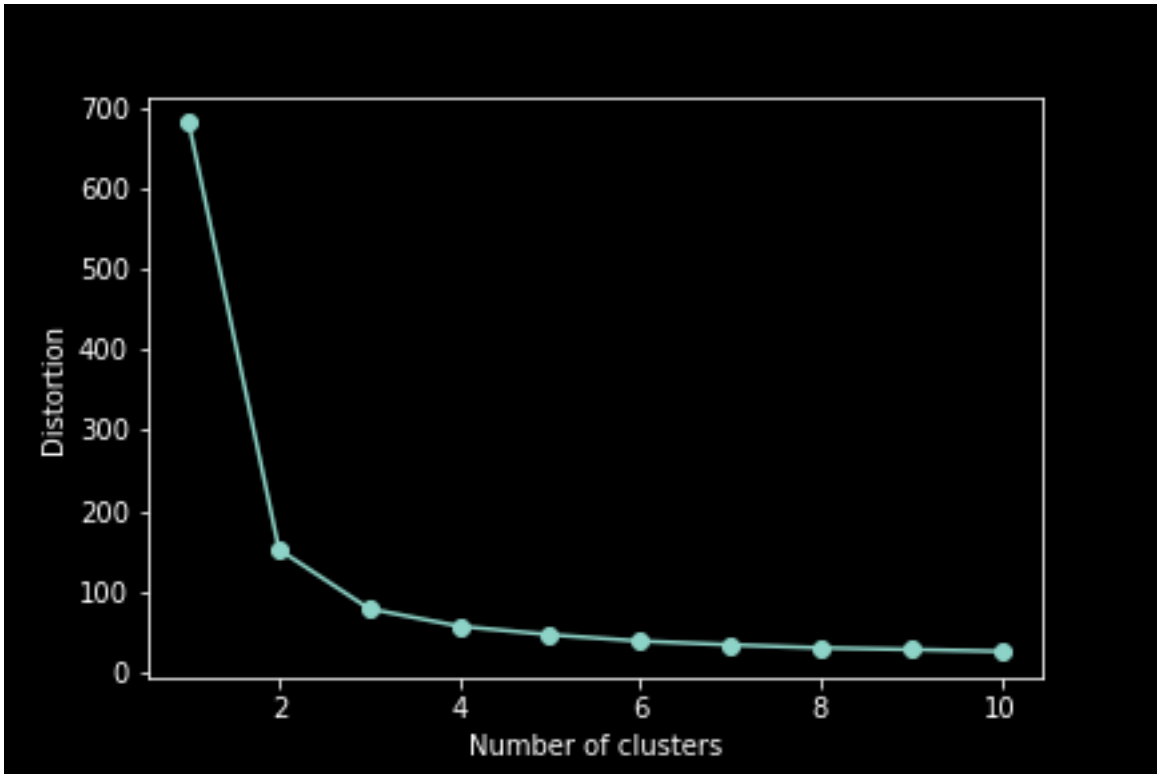
7. Calculate distortion for a range of number of cluster

calculate distortion for a range of number of cluster

In [27]:

```
distortions = []
for i in range(1, 11):
    km = KMeans(
        n_clusters=i, init='random',
        n_init=10, max_iter=300,
        tol=1e-04, random_state=0
    )
    km.fit(x1)
    distortions.append(km.inertia_)

# plot
plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.savefig("distortion_iris.png")
plt.show()
```

- **For Iris Dataset**



- **For wine Dataset**