Krish Sukhani

Batch D , 59

DWM EXP7

```
import numpy as np
import pandas as pd
```

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
```

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive
```

```
df = pd.read_csv('/content/gdrive/My Drive/datasets/weatherAUS.csv',encoding= 'unicode_escape')
```

Data Cleaning

```
categorical = [var for var in df.columns if df[var].dtype=='O']
print('There are {} categorical variables\n'.format(len(categorical)))
print('The categorical variables are :', categorical)
```

```
There are 7 categorical variables

The categorical variables are : ['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday
```

```
cat1 = [var for var in categorical if df[var].isnull().sum()!=0]
print(df[cat1].isnull().sum())
```

```
WindGustDir     9330
WindDir9am     10013
WindDir3pm      3778
RainToday       1406
dtype: int64
```

```
for var in categorical:
    print(var + ' conatins '+str(len(df[var].unique()))+ " labels ")
```

```
Date conatins 3436 labels
Location conatins 49 labels
WindGustDir conatins 17 labels
WindDir9am conatins 17 labels
```

```
WindDir3pm conatins 17 labels
RainToday conatins 3 labels
RainTomorrow conatins 2 labels
```

Splitting the Date column into respective 'Year','Month' & 'Day'.**

```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day

df.drop('Date',axis=1,inplace=True)
```

```
categorical = [var for var in df.columns if df[var].dtype=='O']
print("There are {} categorical variables : ".format(len(categorical)))
print(categorical)
```

```
There are 6 categorical variables :
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

Replacing the missing categorical values by the most frequent value in respective columns.

```
for var in categorical:
    df[var].fillna(df[var].mode()[0],inplace=True)
```

```
numerical = [var for var in df.columns if df[var].dtype!='O']
print(numerical)
```

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3p
```

```
num1 = df[numerical].isnull().sum()
num1 = num1[num1!=0]
num1
```

```
MinTemp            637
MaxTemp            322
Rainfall          1406
Evaporation      60843
Sunshine         67816
WindGustSpeed     9270
WindSpeed9am      1348
WindSpeed3pm      2630
Humidity9am       1774
Humidity3pm       3610
Pressure9am      14014
Pressure3pm      13981
Cloud9am         53657
Cloud3pm         57094
Temp9am            904
Temp3pm           2726
dtype: int64
```

Replacing the missing numercial values by the mean of their respective columns.

```
for col in num1.index:
    col_mean = df[col].mean()
    df[col].fillna(col_mean,inplace=True)
```

```
le = LabelEncoder()
```

```
new_df = df
for col in categorical:
    new_df[col] = le.fit_transform(df[col])
col_names = new_df.columns
```

```
new_df.head()
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	Win
0	2	13.4	22.9	0.6	5.469824	7.624853	13	44.0	13	
1	2	7.4	25.1	0.0	5.469824	7.624853	14	44.0	6	
2	2	12.9	25.7	0.0	5.469824	7.624853	15	46.0	13	
3	2	9.2	28.0	0.0	5.469824	7.624853	4	24.0	9	
4	2	17.5	32.3	1.0	5.469824	7.624853	13	41.0	1	

Feature Scaling using MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler
ss = MinMaxScaler()
new_df = ss.fit_transform(new_df)
new_df = pd.DataFrame(new_df,columns = col_names )
```

```
new_df.describe()
```

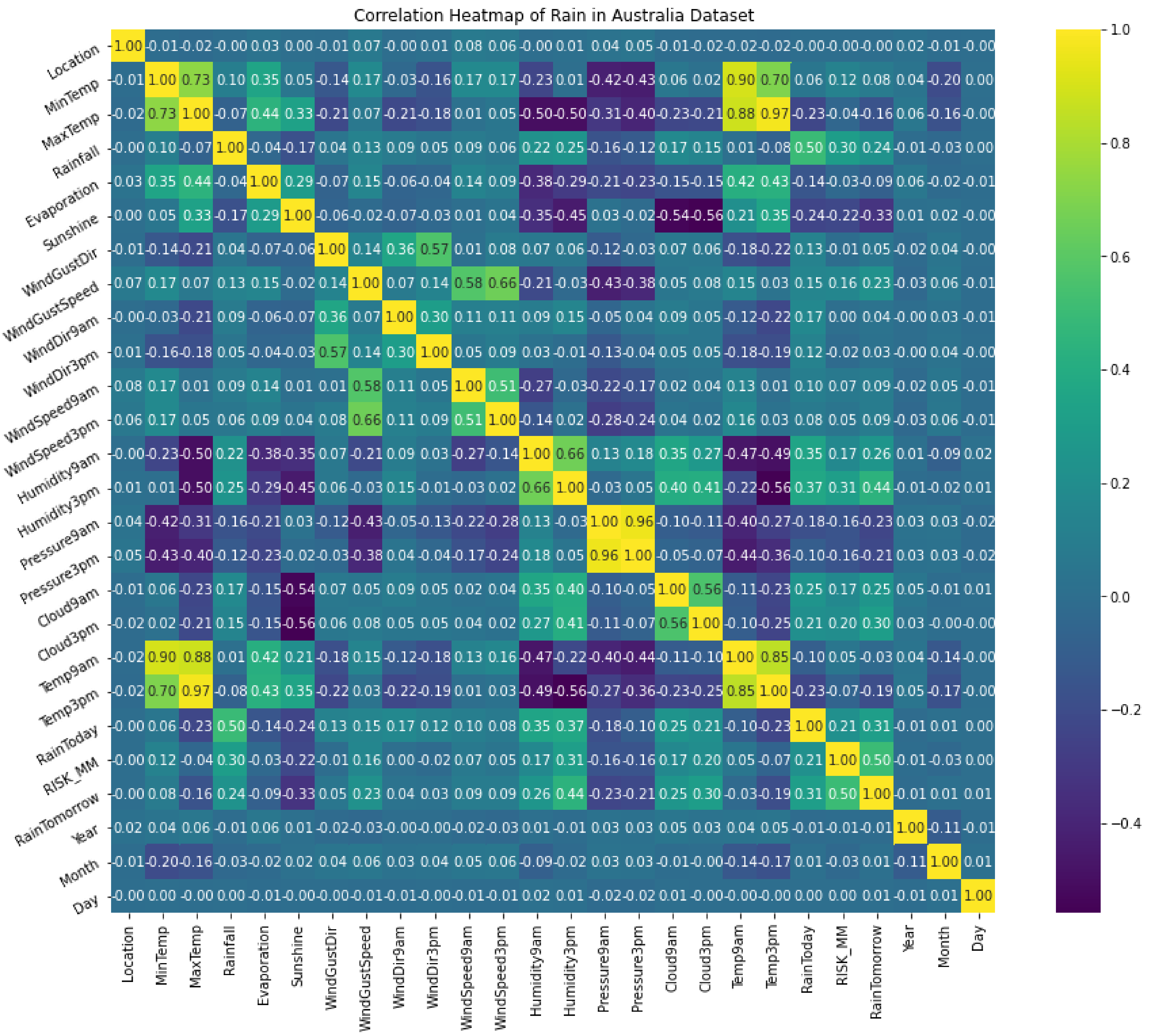
	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDi
count	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.00000
mean	0.494597	0.487887	0.529807	0.006334	0.037723	0.525852	0.5372
std	0.296615	0.150682	0.134396	0.022704	0.021849	0.188616	0.3129
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.229167	0.379717	0.429112	0.000000	0.027586	0.525852	0.2666
50%	0.500000	0.483491	0.519849	0.000000	0.037723	0.525852	0.6000
75%	0.750000	0.596698	0.623819	0.002156	0.037723	0.600000	0.8666
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

```
# new_df.to_csv("weatherCleaned.csv")
```

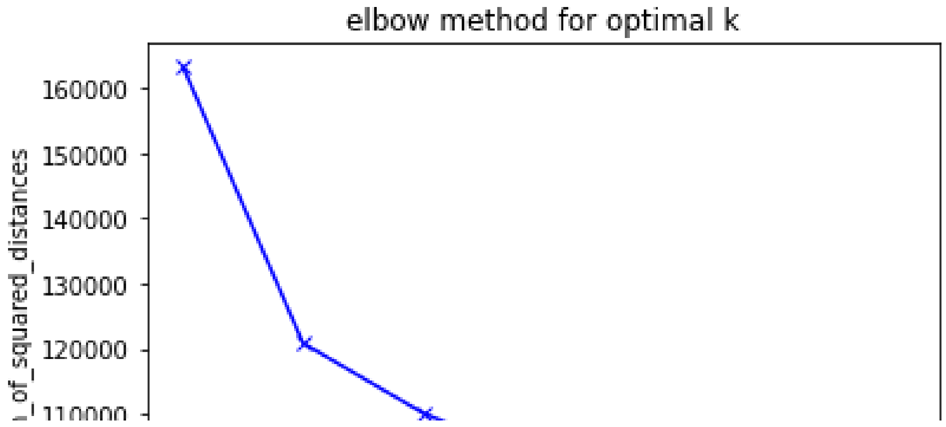
Data Visualization

Heatmap of correlation among the columns of data.

```
correlation = new_df.corr()
plt.figure(figsize=(16,12))
plt.title('Correlation Heatmap of Rain in Australia Dataset')
ax = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white',cmap='viridis')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
ax.set_yticklabels(ax.get_yticklabels(), rotation=30)
plt.show()
```




```
y = new_df.RainTomorrow
X = new_df.drop('RainTomorrow',axis=1)
x = df[['Humidity3pm','RISK_MM']]


from sklearn.cluster import KMeans
from sklearn import metrics


sum_of_squared_distances = []
K = range(1,15,2)
for k in K:
    k_means = KMeans(n_clusters=k)
    model = k_means.fit(X)
    sum_of_squared_distances.append(k_means.inertia_)


plt.plot(K, sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('sum_of_squared_distances')
plt.title('elbow method for optimal k')
plt.show()
```



Observations : KMeans

The Elbow Method is one of the most popular methods to determine this optimal value of k. From the above plot it is clear that for n_clusters = 2 we get the elbow point which means that k = 2 is ideal for clustering

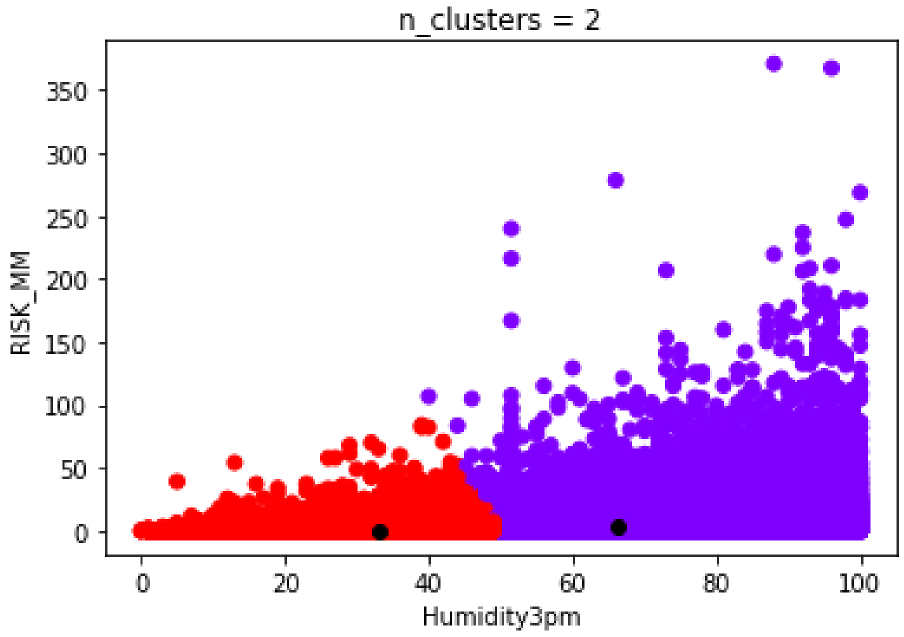
```
kmeans = KMeans(n_clusters=2, random_state=100)
kmeans.fit(x)
labels_1 = kmeans.labels_
np.unique(kmeans.labels_)
print(kmeans.cluster_centers_)
```

```
[[66.16304843  3.83785825]
 [32.96770244  0.49767349]]
```

Double-click (or enter) to edit

```
plt.scatter(x.iloc[:,0],x.iloc[:,1], c=kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], color='black')
plt.xlabel('Humidity3pm')
plt.ylabel("RISK_MM")
plt.title("n_clusters = 2")
```

```
Text(0.5, 1.0, 'n_clusters = 2')
```



Hence we find the kMeans distribution

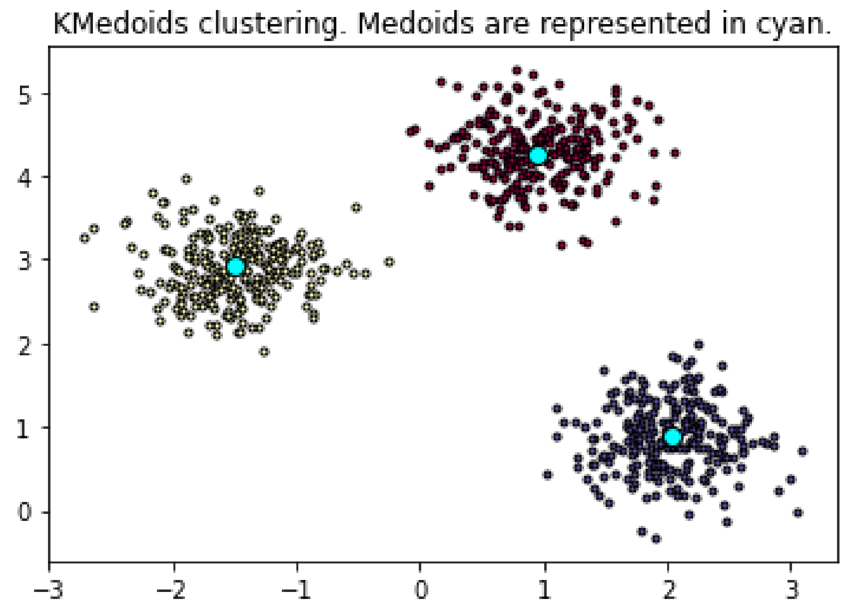
```
!pip install https://github.com/scikit-learn-contrib/scikit-learn-extra/archive/master.zip
```

```
Collecting https://github.com/scikit-learn-contrib/scikit-learn-extra/archive/master.zip
  Using cached https://github.com/scikit-learn-contrib/scikit-learn-extra/archive/master.zip
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Requirement already satisfied (use --upgrade to upgrade): scikit-learn-extra==0.1.0b2 from https://github.
Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.7/dist-packages (from scikit-
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.7/dist-packages (from scikit-learn-
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-learn-
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=
```

```
Building wheels for collected packages: scikit-learn-extra
  Building wheel for scikit-learn-extra (PEP 517) ... done
  Created wheel for scikit-learn-extra: filename=scikit_learn_extra-0.1.0b2-cp37-cp37m-linux_x86_64.whl si
  Stored in directory: /tmp/pip-ephem-wheel-cache-jhg1o50h/wheels/d3/a5/a8/411bc2d0939f2cc9d17f34f0d345704
Successfully built scikit-learn-extra
```

```
from sklearn_extra.cluster import KMedoids
from sklearn.datasets import make_blobs
X, labels_true = make_blobs(
    n_samples=750,cluster_std=0.4, random_state=0
)
cobj = KMedoids(n_clusters=3).fit(X)
labels = cobj.labels_


unique_labels = set(labels)
colors = [
    plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))
]
for k, col in zip(unique_labels, colors):

    class_member_mask = labels == k

    xy = X[class_member_mask]
    plt.plot(
        xy[:, 0],
        xy[:, 1],
        "o",
        markerfacecolor=tuple(col),
        markeredgecolor="k",
        markersize=3,
    )

plt.plot(
    cobj.cluster_centers_[:, 0],
    cobj.cluster_centers_[:, 1],
    "o",
    markerfacecolor="cyan",
    markeredgecolor="k",
    markersize=8,
)

plt.title("KMedoids clustering. Medoids are represented in cyan.")
```

```
Text(0.5, 1.0, 'KMedoids clustering. Medoids are represented in cyan.')
```



Hence we plot and find the K Medoids