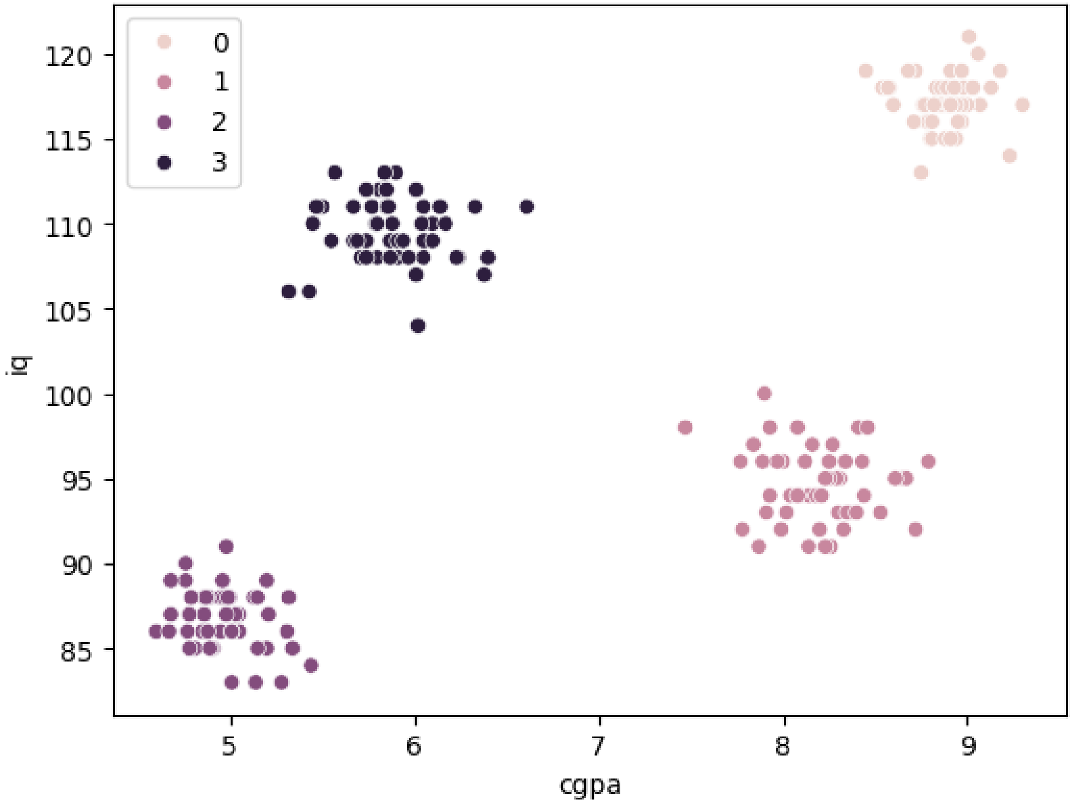
1. Perform K-means with scratch and with library.

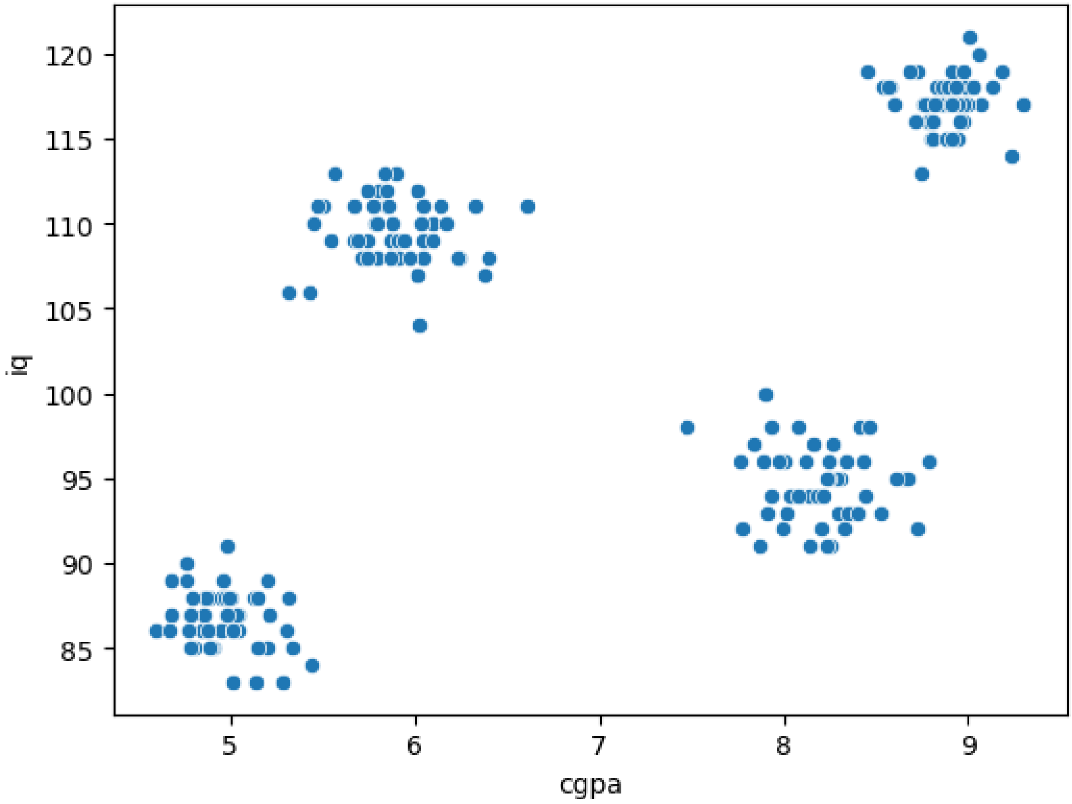
K-Means With Scratch

```
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("student_clustering.csv")
class Kmeans:
    def __init__(self,n_clusters=2,max_iter=100):
        self.n_clusters= n_clusters
        self.max_iter= max_iter
        self.centroids= None

    def fit_predict(self,x):
        rand_index=random.sample(range(0,x.shape[0]),self.n_clusters)
        self.centroids=x[rand_index]
        #print(self.centroids)
        for i in range(self.max_iter):
            cl_group=self.assign_clusters(x)
            old_centroids=self.centroids
            self.centroids=self.move_centroids(x,cl_group)
            if(old_centroids==self.centroids).all():
                break
        return cl_group
    def assign_clusters(self,x):
        cluster_group=[]
        distance=[]
        for row in x:
            for centroid in self.centroids:
                distance.append(np.sqrt(np.dot(row-centroid,row-centroid)))
        #distance.append(np.linalg.norm(row-centroid,axis=1))
      min_dist = min(distance)
      index_pos = distance.index(min_dist)
      cluster_group.append(index_pos)
      distance.clear()
    return np.array(cluster_group)

  def move_centroids(self,x,cluster_group):
    new_centroids=[]
    c_type=np.unique(cluster_group)
    for type in c_type:
      new_centroids.append(x[cluster_group==type].mean(axis=0))
    return np.array(new_centroids)

k = Kmeans(4,500)
y_means=k.fit_predict(df.values)
sns.scatterplot(x=df.iloc[:,0],y=df.iloc[:,1],hue=y_means)
```

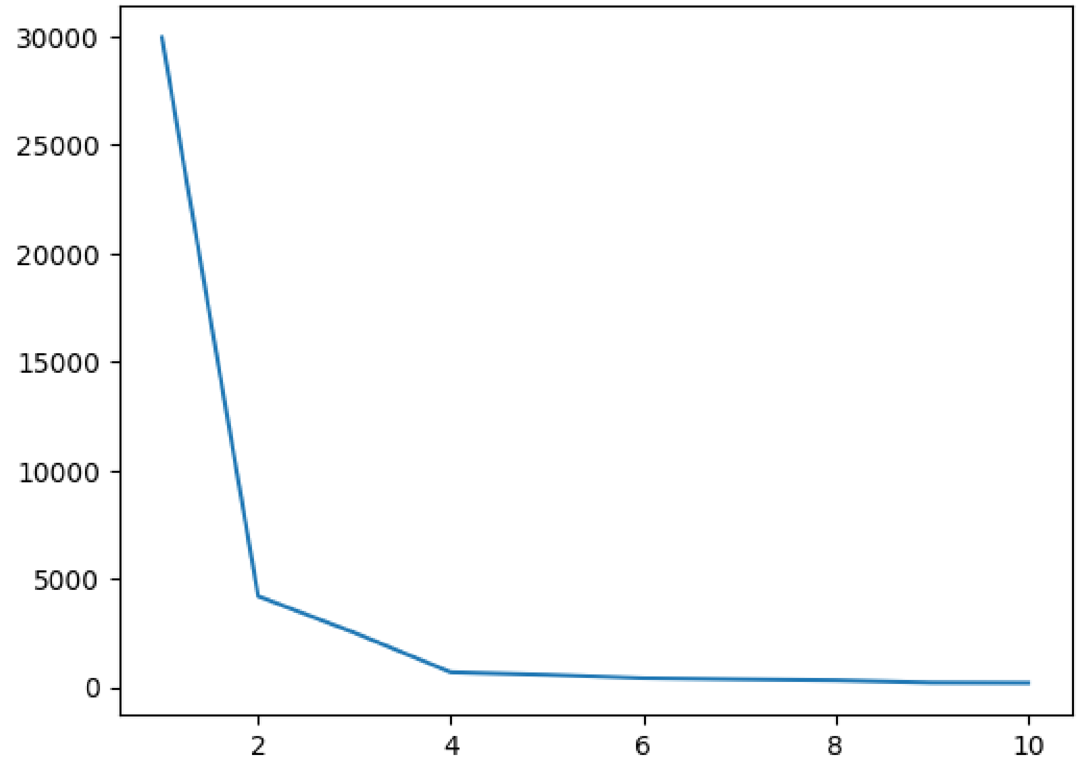


K-Means with Library

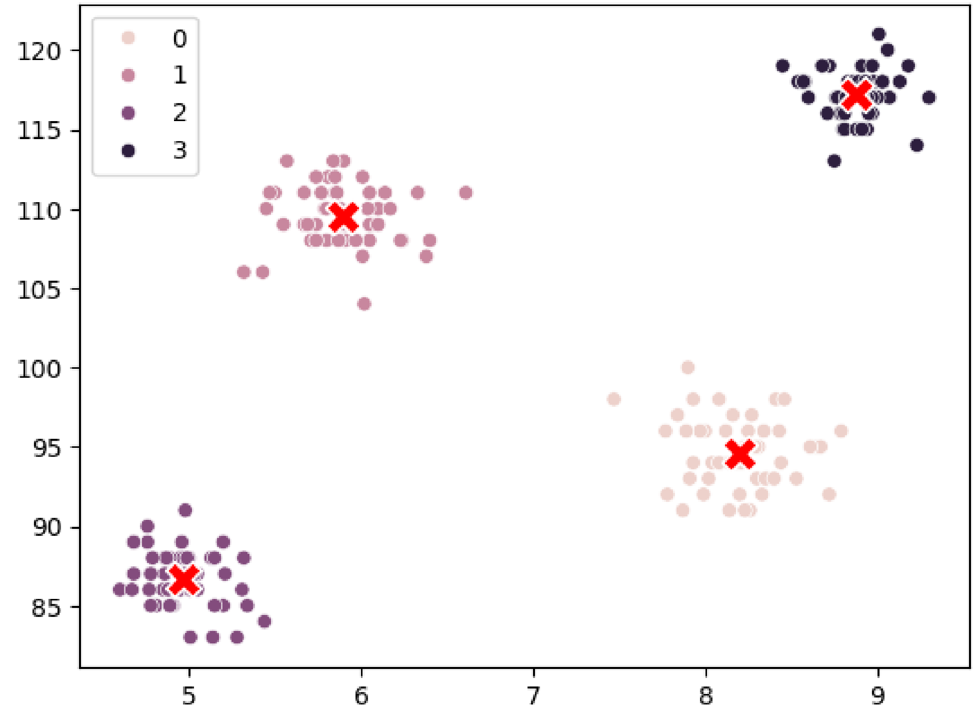
```
import pandas as pd

import numpy as np

import seaborn as sns

df=pd.read_csv("student_clustering.csv")

df.shape

sns.scatterplot(data=df,x=df["cgpa"],y=df["iq"])
```



```
from sklearn.cluster import KMeans

wcss=[]

for i in range(1,11):

   k=KMeans(n_clusters=i)

   k.fit_predict(df)

   wcss.append(k.inertia_)

sns.lineplot(x=range(1,11),y=wcss)
```



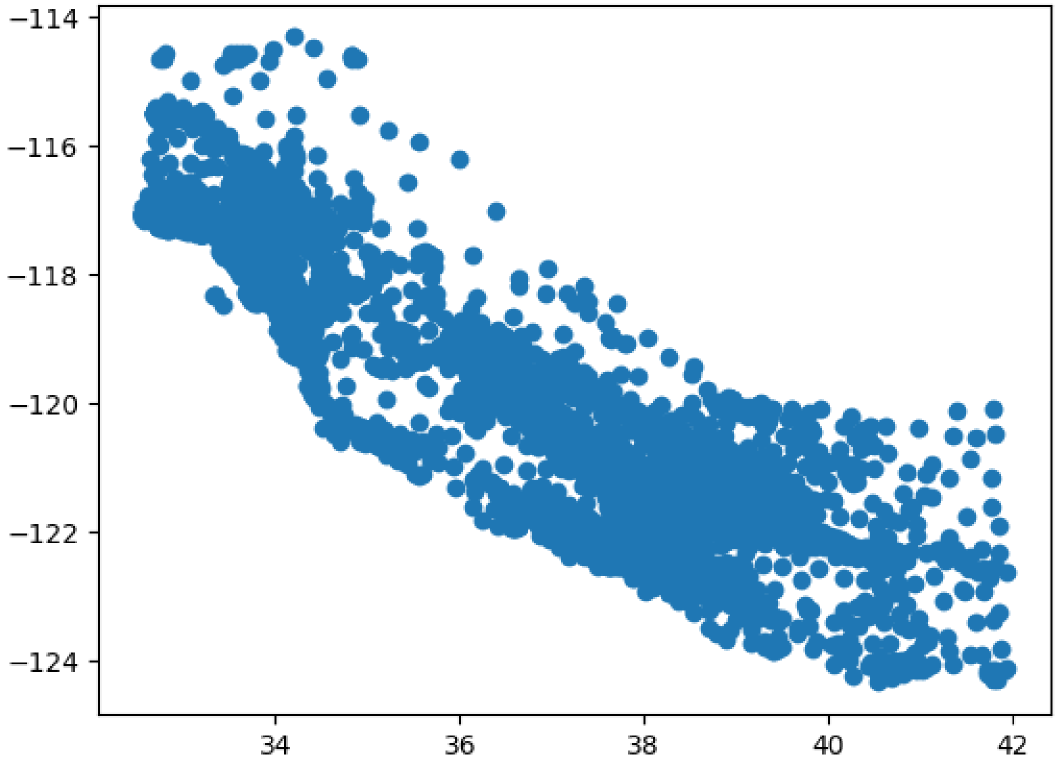
```
x=df.values

k=KMeans(n_clusters=4)

y_means=k.fit_predict(x)

sns.scatterplot(data=df,x=x[:,0],y=y[:,1],hue=y_means)

sns.scatterplot(x=k.cluster_centers_[:, 0], y=k.cluster_centers_[:, 1], marker='X', s=200, c='red')
```



2. Perform Fuzzy c means with scratch and with library.

Fuzzy c means with Scratch

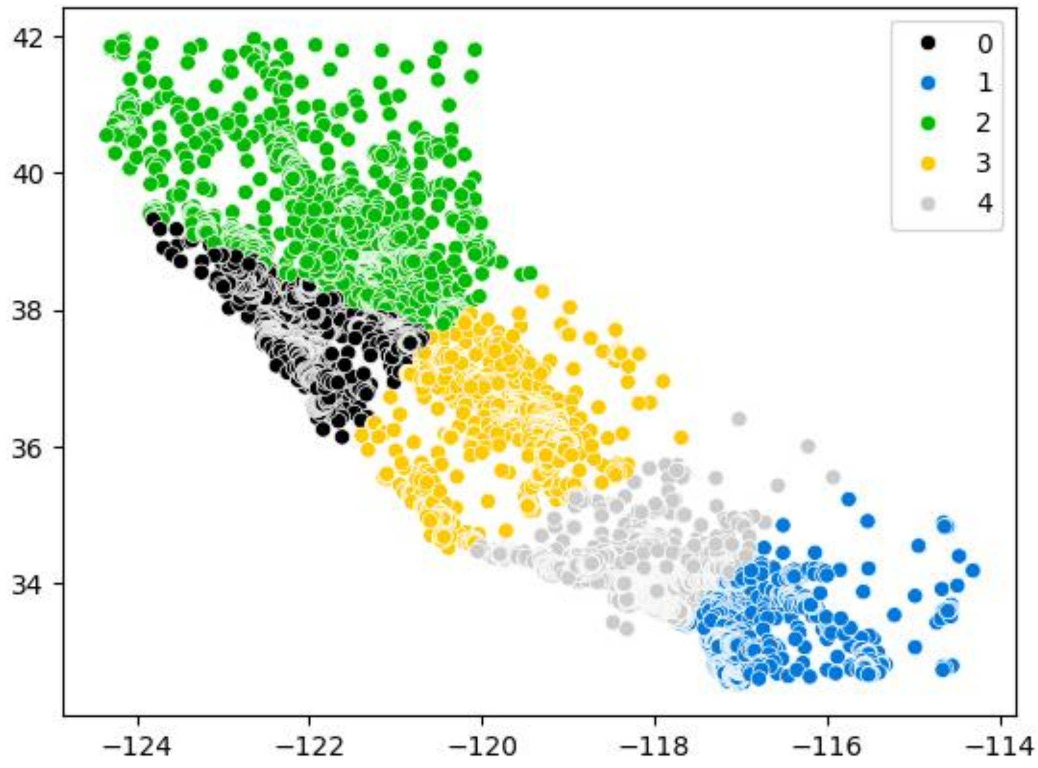
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


df=pd.read_csv("Downloads/housing.csv")


data=df.iloc[:,:2]

plt.scatter(data["latitude"],data["longitude"])
```



```
data.shape
```

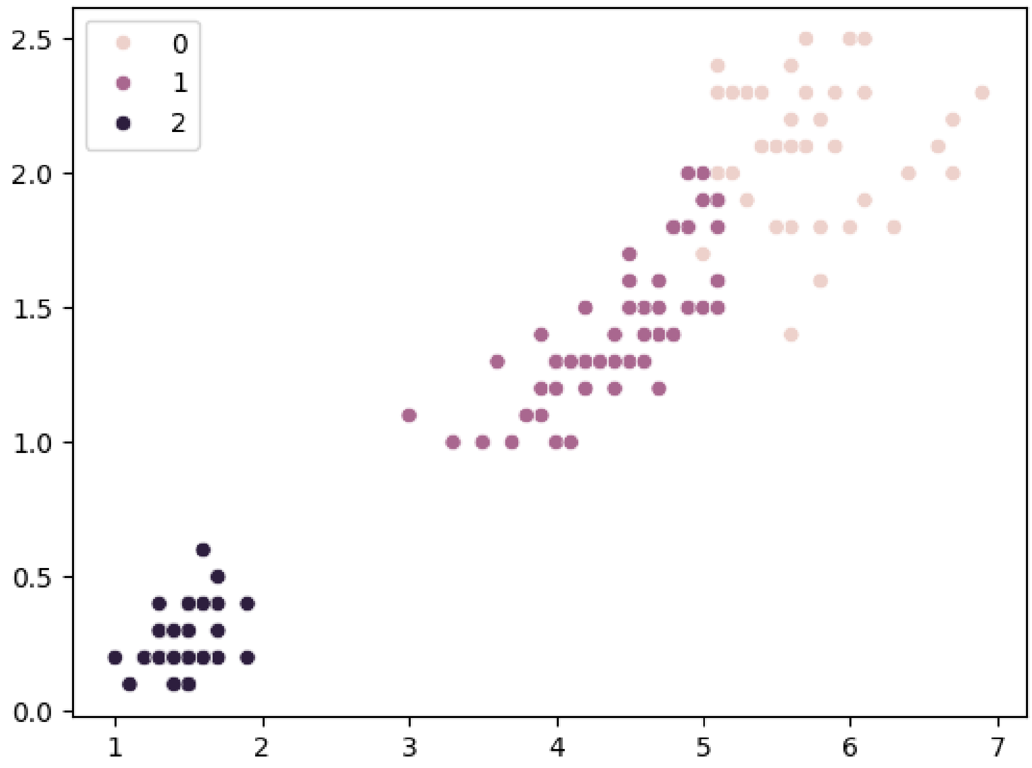
```
k=5

u=np.random.rand(data.shape[0],5)
u/=np.sum(u,axis=1)[:,np.newaxis]

def calculate_centroid(data,k,u,m):
    centroids=np.zeros((k,data.shape[1]))
    #print(data)
    for i in range(k):
        centroids[i,:]=np.sum((u[:,i]**m)[:,np.newaxis]*data.values,axis=0)/np.sum(u[:,i]**m)
    print(centroids)
    return centroids


def cal_membership(data,centroids,k,m):
    u_new=np.zeros((data.shape[0],k))
    #print(centroids)
    for i in range(k):
        u_new[:,i]=np.linalg.norm(data.values-centroids[i,:],axis=1)
    u_new=1/ (u_new ** 2 * np.sum((1/u_new) ** 2 , axis=1 )[:,np.newaxis] )
    return u_new


for i in range(100):
    centroids = calculate_centroid(data,5,u,2)
    u_new=cal_membership(data,centroids,5,2)
    if np.linalg.norm(u_new-u)<=0.00001:
        break
    u=u_new
labels=np.argmax(u,axis=1)
```

```
sns.scatterplot(data=data, x=data.values[:,0],y=data.values[:,1], hue=labels, palette='nipy_spectral')
```



Fuzzy c means with library

```
import pandas as pd

import seaborn as sns

import numpy as np

from sklearn.datasets import load_iris

iris=load_iris()

x=pd.DataFrame(iris.data,columns=iris.feature_names)

y=pd.DataFrame(iris.target,columns=["Species"])

from fcmeans import FCM

f=FCM(n_clusters=3)

f.fit(x.values)

ylab=f.predict(x.values)

f.centers

sns.scatterplot(x=x.values[:,2],y=x.values[:,3],hue=ylab)

sns.scatterplot(x=x.values[:,2],y=x.values[:,3],hue=ylab)
```



3. Perform KNN with the library and plot the results. Print the Accuracy score, classification report and plot the confusion matrix. Use Diabetes and wine dataset for this.

```
import pandas as pd

import numpy as np

import seaborn as sns


from sklearn.datasets import load_wine


b=load_wine()


df=pd.DataFrame(data=b.data,columns=b.feature_names,)
```

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

```
df=pd.DataFrame(data=ss.fit_transform(df),columns=b.feature_names)
```

```
df["quality"]=b.target
```

```
df.shape
```

```
(178, 14)
```

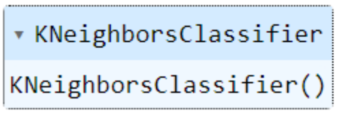
```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(df.iloc[:,:30],df.iloc[:,-1],test_size=0.2,random_state=42)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
knn.fit(x_train,y_train)
```



```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred=knn.predict(x_test)
```

```
from sklearn.metrics import accuracy_score,r2_score
```

```
accuracy_score(y_test,y_pred)
```

```
0.9444444444444444
```

```
r2_score(y_test,y_pred)
```

```
0.9047619047619048
```

```
from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      1.00      0.97        14
           1       1.00      0.86      0.92        14
           2       0.89      1.00      0.94         8

    accuracy                           0.94        36
   macro avg       0.94      0.95      0.94        36
weighted avg       0.95      0.94      0.94        36
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test,y_pred)
```

```
array([[14,  0,  0],
       [ 1, 12,  1],
       [ 0,  0,  8]], dtype=int64)
```