Group Students from marks of 3 subjects

Links
https://scikit-learn.org/stable/modules/clustering.html

#Libraries
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import metrics
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
```

#Data
```
X = [1,1,1,2,1,2,1,2]
Y = [4,2,4,1,1,4,1,1]
Z = [1,2,2,2,1,2,2,1]
df = pd.DataFrame({'X':X, 'Y':Y, 'Z':Z})
```
df

#Algorithm - kMeans
Kmeans - algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares (see below). This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

#Modeling
```
km3 = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
km3.fit_predict(df)
```

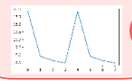
#Output
- km3.cluster_centers_
- km3.inertia_ — measure of how internally coherent clusters are
- km3.labels_ — #which row has goes to which cluster no
- km3.get_params
- km3.n_clusters
- pd.concat([df, pd.Series(km3.labels_)], axis=1)

#selecting No of Clusters

#manually
```
km1.fit(df)
km1.inertia_
km1 = KMeans(n_clusters = 1, init = 'k-means++', random_state = 42)

km2.fit(df)
km2.inertia_
km2 = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)

km3.fit(df)
km3.inertia_
km3 = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)

km4.fit(df)
km4.inertia_
km4 = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
```
(19.25, 4.53, 3.16, 2.33) Elbow point at Point 2 (Clusters)

```
km1.inertia_, km2.inertia_, km3.inertia_, km4.inertia_
```

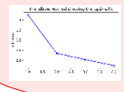
#inertia
```
wcss = [ ]
for i in range(1, 5):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)
```
19.25, 4.53 3.16, 2.33, 19.25, 4.53, 3.16, 2.33] — wcss



plt.plot(wcss)

#distortions
```
K = range(1,5)
distortions = [ ]
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(df)
    kmeanModel.fit(df)
    distortions.append(sum(np.min(cdist(df, kmeanModel.cluster_centers_, 'euclidean'), axis=1)) / df.shape[0])
```
distortions

Find the elbow point - it is at 2



```
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show();
```

#plot using 2 variables
```
km3b = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
X2 = np.random.randint(50,100, size=100)
Y2 = np.random.randint(60,90, size=100)
km3b.fit(df2)
df2 = pd.DataFrame({'X':X2, 'Y':Y2})
km3b.inertia_
centers
centers = np.array(km3b.cluster_centers_)
plt.scatter(x=df2.X, y=df2.Y)
plt.scatter(centers[:,0], centers[:,1], marker="x", color='r')
plt.show();
```