**K-Means Clustering**

* Esența problemelor de clustering
* K-Means: toate punctele trebuie să se includă în clustere și un punct poate fi inclus doar într-un singur cluster
* Etapele algoritmului K-Means:
* Selectarea numărului de clustere – K
* Se selectează aliator K puncte a datelor care se vor considera centre ale clusterelor
* Punctele rămase se distribuie clusterelor în funcție de distanța până la punctele centre ale clusterelor
* Conform punctelor fiecărui cluster se determina un nou centru al clusterului
* Se redistribuie punctele în clustere în funcție de distanța până la noile centre ale clusterelor
* Se repetă pașii de mai sus până când structura clusterelor nu se mai modifică la redistribuire
* Algoritmul de elaborare a modelului K-Means pentru clusterizare în Scikit-Learn:
* importul datelor
* vizualizarea informației despre date ***df.info()***
* analiza datelor din coloanele numerice ***sns.histplot(date=df, x=”col”)***
* analiza datelor din coloanele categoriale ***sns.countplot(date=df, x=”col”)***
* transformarea datelor categoriale în date numerice ***pd.get\_dummies(df)***
* importul modulului de scalarea ***from sklearn.preprocessing import StandardScaler***
* Crearea unui obiect de scalarea ***scaler = StandardScaler()***
* Scalarea datelor X ***X\_scalat = scaler.fit\_transform(X)***
* importul modulului KMeans ***from sklearn.cluster import KMeans***
* crearea modelului pentru 2 clustere ***model = KMeans(n\_clusters=2)***
* Ajustarea modelului și determinarea clusterului fiecarei date ***cluster\_labels = model.fit\_predict(X\_scalat)***
* adăugarea setului de indici ai clusterului ca o coloana label in datele X ***X['Cluster'] = cluster\_labels***
* determinarea corelatie dintre datele coloanei Cluster si celelalte coloane ***X.corr()['Cluster']***
* vizualizarea grafica a corelatiei atele coloanei Cluster si celelalte coloane ***X.corr()['Cluster'].iloc[:-1].sort\_values().plot(kind='bar')***
* vizualizarea coordonatelor centrelor clusterilor ***model.cluster\_centers\_***
* determinarea sumei patratelor distantelor dintre puncte și centru clusterilor ***model.inertia\_***
* Rolul sumei pătratelor distantelor dintre puncte și centru clusterelor în determinarea valorii optime a numărului de clustere
* Determinarea valoarii optime a numărului de clustere în Scikit-Learn:
* Crearea unei bucle for pentru determinarea valorilor sumei patratice a distantei pentru modele cu diferite valori a numarului de clastere
* Vizualizarea grafica valorilor sumelor pătratice a distantei **plt.scatter(range(2,10),suma)**
* vizualizarea diferentei dintre valorile sumei patratelor distantelor ***pd.Series(suma).diff()***
* vizualizarea grafica diferentei dintre valorile sumei patratelor distantelor ***pd.Series(suma).diff().plot(kind='bar')***
* Selectarea valori numărului de clustere acolo unde scade tendința de descreștere a valorilor acestei sume
* Aplicarea K-Means la gruparea culorilor imaginii în Scikit-Learn:
* importul modulului matplotlib.image ***import matplotlib.image as mpimg***
* importulimaginii ***mpimg.imread('palm\_trees.jpg')***
* vizualizarea imaginii ***plt.imshow(imagine\_array);***
* vizualizarea formei matricii imaginii ***imagine\_array.shape***
* citirea inaltimii, latimii si nuamrului de canale a imaginii ***h,w,c = imagine\_array.shape***
* transformarea matricii din 3d in 2d ***imagine\_array.reshape(h\*w,c)***
* crearea modelului K-Means pentru 6 clustere ***model = KMeans(n\_clusters=6)***
* Ajustarea modelului și determinarea clusterului fiecarei date ***labels = model.fit\_predict(imagine\_array2d)***
* determinarea coordonatelor centrelor clusterelor ***model.cluster\_centers\_***
* rotungirea si transformarea in valori intregi a valorilor coordonatelor centrelor*clusterilor* ***coduri\_rgb = model.cluster\_centers\_.round(0).astype(int)***
* Crearea matricii imaginii cu culorile indentice pentru fiecare cluster ***imagine\_cuatizata = np.reshape(coduri\_rgb[labels], (h, w, c))***
* vizualizarea imaginii cuantizate ***plt.imshow(imagine\_cuatizata)***