VJEŽBA 5: GRUPIRANJE PODATAKA. KMEANS. HIJERARHIJSKO GRUPIRANJE.

I. Cilj vježbe: *Upoznati se s osnovnim algoritmima za grupiranje podataka.*

II. Opis vježbe:

Nenadzirano učenje (engl. *unsupervised learning*) je vrsta strojnog učenja gdje je cilj izvući zaključke o raspoloživom skupu podataka koji se sastoji samo od ulaznih veličina, bez odgovarajuće izlazne veličine (npr. otkriti interesantna svojstva). Najčešće se analiza podataka provodi u skladu sa sljedećim pitanjima:

- Mogu li se otkriti grupe u podacima?
- Može li se otkriti skrivena struktura u podacima?
- Mogu li se podaci predstaviti na drugačiji način?
- Mogu li se podaci efikasno komprimirati

Dva najvažnija problema nenadziranog učenja:

- Grupiranje podataka (engl. *clustering*)
- Smanjivanje dimenzionalnosti (engl. *dimensionality reduction*)

II.1. Grupiranje podataka *Kmeans* algoritmom

Grupiranje podataka ili klaster analiza jedan je najčešćih problema nenadziranog strojnog učenja. Koristi se kako bi se pronašle grupe ili skrivene zakonitosti i obrasci u podacima odnosno pokušava se naučiti optimalna podjela podataka. Podaci su neoznačeni - jednom kada se pronađu grupe u podacima moguće je nove mjerne uzorke dodijeliti odgovarajućoj grupi. Primjene su raznolike: segmentacija korisničkog ponašanja (npr. prema povijesti kupovine, aktivnosti u aplikaciji i sl.), detektiranje "botova" i anomalija, *text mining*, obrada medicinskih slika, segmentacija slika, sustavi preporuka itd.

U ovoj vježbi razmatra se problem nenadgledanog učenja gdje je cilj grupirati podatke koji se sastoje *m* ulaznih veličina $X = [x_1, x_2, \dots, x_m]$. Stoga, svaki podatak je vektor vrijednosti ulaznih i zapisuje se u obliku:

$$\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}\right]^T. \tag{5-1}$$

Skup koji se sastoji od *n* raspoloživih mjernih podataka može se zapisati u matričnom obliku:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_m^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_m^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_m^{(n)} \end{bmatrix} \tag{5-2}$$

Algoritam *k* srednjih vrijednosti (engl. *Kmeans*) je jednostavan i često korišten algoritam grupiranja koji kao rezultat daje *K* klastera koji su zapisani kao *m* dimenzionalni vektori $\mathbf{c}^{(k)}, k = 1, \dots, K$. Algoritam se zasniva na dvije pretpostavke:

- Centar nekog *k*-tog klastera $\mathbf{c}^{(k)}$ je aritmetička sredina svih podataka $\mathbf{x}^{(i)}$ koji pripadaju tom klasteru
- Svaki podatak je bliže svom klasteru nego centrima ostalih klastera

Pomoću navedenih pretpostavki moguće je napisati kriterijsku funkciju algoritma *Kmeans*:

$$J(\mathbf{c}^{(k)}, k = 1, \dots, K; \mathbf{X}) = \sum_{i=1}^{n} \sum_{k=1}^{K} b_k^{(i)} \left\| \mathbf{x}^{(i)} - \mathbf{c}^{(k)} \right\|^2 \quad (5\text{-}3)$$

pri čemu je $b_k^{(i)}$ jednak 1 ili 0 ovisno pripada li podatak $\mathbf{x}^{(i)}$ centru $\mathbf{c}^{(k)}$.

U nastavku je dan pseudokod *Kmeans* algoritma koji sa svakom iteracijom smanjuju vrijednost kriterijske funkcije (5-3). Postoje razni načini inicijalizacije centara odnosno određivanje početnih vrijednosti centara. Npr. kao početne vrijednosti cenatara uzima se nasumično *K* podataka iz dostupnog skupa podataka **X**. Kao kriterij zaustavljanja obično se uzima promjena centara u dvije uzastopne iteracije ili se unaprijed zadaje broj iteracija.

Kmeans algoritam

1. odredi broj centara *K*. Odredi početne vrijednosti centara klastera $\mathbf{c}^{(k)}, k = 1, \dots, K$.

2. Sve dok nije zadovoljen kriterij zaustavljanja

3. Za svaki podatak $\mathbf{x}^{(i)}$odredi kojem centru (klasteru pripada)

$$b_k^{(i)} = \begin{cases} 1 \; ako \; je \; \left\| \mathbf{x}^{(i)} - \mathbf{c}^{(k)} \right\| \text{ najmanja od svih udaljenosti za } k = 1, \dots, K \\ \text{u suprotnom } 0 \end{cases}$$

4. Osvježi vrijednosti svih centara $k = 1, \dots, K$

$$\mathbf{c}^{(k)} \leftarrow \frac{\sum_{i=1}^{n} b_k^{(i)} \mathbf{x}^{(i)}}{\sum_{i=1}^{n} b_k^{(i)}}$$

Kmeans je iterativna procedura jer jednom kad se izračunaju novi centri $\mathbf{c}^{(k)}$ mijenjaju se i pripadnosti pojeding podatka $\mathbf{x}^{(i)}$ pa se njihovim preračunavanjem opet utječe na centre. Ova dva koraka se izmjenjuju sve dok se vrijednosti centara ne stabiliziraju.

II.2. Hijerarhijsko grupiranje podataka

Postoje metode grupiranja koje za razliku od *Kmeans* uzimaju u obzir samo sličnosti među podacima, bez ikakvog drugog uvjeta na podatke. Cilj je pronaći grupe na način da su podaci koji pripadaju jednoj grupi više slični jedni drugima nego podaci iz drugih grupa. Ovakav pristup koristi hijerarhijsko grupiranje. Dvije osnovne grupe ovakvih algoritama su: *agglomerative* i *divise*.

Agglomerative clustering algoritam počinje s *n* grupa pri čemu svaka inicijalno sadrži samo jedan podatak iz skupa za učenje. Zatim se slične grupe spajaju u veće grupe ili klastere sve dok ne nastane jedna velika grupa. U svakoj iteraciji *agglomerative* algoritma, odabiru se dvije najbliže grupe za spajanje. Hijerarhijsko grupiranje se često prikazuje dendogramom.

III. Priprema za vježbu:

Ponovite gradivo vezano za grupiranje podataka.

IV. Rad na vježbi:

1. Klonirajte vaš repozitorij `rusu_lv_2019_20` na računalo pomoću gitbash. Zatim povucite moguće promjene iz izvornog repozitorija pomoću naredbi:

```
git remote add upstream https://gitlab.com/rgrbic/rusu_lv_2019_2020
git fetch upstream
git merge upstream/master
```

2. Riješite dane zadatke, pri čemu Python skripte trebaju imati naziv `zad_x.py` (gdje je `x` broj zadatka) i trebaju biti pohranjene u direktorij `rusu_lv_2019_20/LV5/solutions/`. Svaki zadatak rješavajte u zasebnoj *git* grani koju spojite s glavnom granom kada riješite pojedini zadatak. Pohranite skripte u lokalnu *git* bazu kao i u `rusu_lv_2019_20` repozitorij na vašem korisničkom računu. Svaki puta kada načinite promjene koje se spremaju u *git* sustav napišite i odgovarajuću poruku prilikom izvršavanja `commit` naredbe.
3. Nadopunite postojeću tekstualnu datoteku `rusu_lv_2019_20/LV5/Readme.md` s kratkim opisom vježbe i kratkim opisom rješenja vježbe te pohranite promjene u lokalnu bazu. Na kraju pohranite promjene u udaljeni repozitorij.

Zadatak 1

U prilogu vježbe nalazi se funkcija 5.1. koja služi za generiranje umjetnih podataka kako bi se demonstriralo grupiranje podataka. Funkcija prima cijeli broj koji definira željeni broju uzoraka u skupu i cijeli broj (od 1 do 5) koji definira na koji način će se generirati podaci, a vraća generirani skup podataka u obliku numpy polja pri čemu su prvi i drugi stupac vrijednosti prve odnosno druge ulazne veličine za svaki podatak.

Generirajte 500 podataka i prikažite ih na slici. Pomoću scikit-learn ugrađene metode za *kmeans* odredite centre klastera te svaki podatak obojite ovisno o njegovoj pripadnosti pojedinom klasteru (grupi). Nekoliko puta pokrenite napisani kod. Što primjećujete? Što se događa ako mijenjate način kako se generiraju podaci?

Zadatak 2

Scikit-learn *kmeans* metoda vraća i vrijednost kriterijske funkcije (5-3). Za broj klastera od 1 do 20 odredite vrijednost kriterijske funkcije za podatke iz Zadatka 1. Prikažite dobivene vrijednosti pri čemu je na x-osi broj klastera (npr. od 2 do 20), a na y-osi vrijednost kriterijske funkcije. Kako komentirate dobivene rezultate? Kako biste pomoću dobivenog grafa odredili optimalni broj klastera?

Zadatak 3

Primijenite hijerarhijsko grupiranje na podatke korištene u Zadatku 1 pomoću funkcije *linkage* koja je ugrađena scipy metoda za *agglomerative clustering*:

```
from scipy.cluster.hierarchy import dendrogram, linkage
```

Prikažite pripadni dendogram. Mijenjajte korištenu metodu (argument *method*). Kako komentirate postignute rezultate?

Zadatak 4

Primijenite scikit-learn *kmeans* metodu za kvantizaciju boje na slici. Proučite kod 5.2. iz priloga vježbe te ga primijenite za kvantizaciju boje na slici `example_grayscale.png` koja se nalazi u `rusu_lv_2019_20/LV5/resources/`. Mijenjajte broj klastera. Što primjećujete? Izračunajte kolika se kompresija ove slike može postići ako se koristi 10 klastera.

Pomoću sljedećeg koda možete učitati sliku:

```
import matplotlib.image as mpimg

imageNew = mpimg.imread('example_grayscale.png')
```

Zadatak 5

Primijenite scikit-learn *kmeans* metodu za kvantizaciju boje na slici `example.png` koja se nalazi u `rusu_lv_2019_20/LV5/resources/`. Prikažite originalnu i kvantiziranu sliku.

Zadatak 6

Na temelju izraza pseudokoda iz Opisa vježbe implementirajte *kmeans* algoritam. Primijenite implementirani algoritam na podatke kao u Zadatku 1 te zaključite jeste li pravilno napravili implementaciju algoritma.

Modificirajte programski kod tako da u svakoj iteraciji algoritma u polje pohranite i izračunate vrijednosti centara klastera. Primijenite programski kod na podatke kao u Zadatku 1 te prikažite u ravnini podatke i izračunate centre u svakoj iteraciji algoritma (npr. spojite izračunate vrijednosti centara linijama kako bi vizualizirali „kretanje“ centara kroz iteracije *kmeans* algoritma).

V. Izvještaj s vježbe

Kao izvještaj s vježbe prihvaća se web link na repozitorij pod nazivom `rusu_lv_2019_20` koji sadrži rješenja unutar direktorija `rusu_lv_2019_20/LV5/solutions/`.

VI. Dodatak

Funkcija 5.1. – generiranje podataka

```
from sklearn import datasets
import numpy as np

def generate_data(n_samples, flagc):

    if flagc == 1:
        random_state = 365
        X,y = datasets.make_blobs(n_samples=n_samples, random_state=random_state)

    elif flagc == 2:
        random_state = 148
        X,y = make_blobs(n_samples=n_samples, random_state=random_state)
        transformation = [[0.60834549, -0.63667341], [-0.40887718, 0.85253229]]
        X = np.dot(X, transformation)

    elif flagc == 3:
        random_state = 148
        X, y = make_blobs(n_samples=n_samples,
                          cluster_std=[1.0, 2.5, 0.5, 3.0],
                          random_state=random_state)

    elif flagc == 4:
        X, y = datasets.make_circles(n_samples=n_samples, factor=.5, noise=.05)

    elif flagc == 5:
        X, y = datasets.make_moons(n_samples=n_samples, noise=.05)

    else:
        X = []

    return X
```

Kod 5.2. – kvantizacija boje

```
import scipy as sp
from sklearn import cluster, datasets
import numpy as np
import matplotlib.pyplot as plt

try:
    face = sp.face(gray=True)
except AttributeError:
    from scipy import misc
    face = misc.face(gray=True)

X = face.reshape((-1, 1)) # We need an (n_sample, n_feature) array
k_means = cluster.KMeans(n_clusters=5,n_init=1)
k_means.fit(X)
values = k_means.cluster_centers_.squeeze()
labels = k_means.labels_
face_compressed = np.choose(labels, values)
face_compressed.shape = face.shape

plt.figure(1)
plt.imshow(face,  cmap='gray')

plt.figure(2)
plt.imshow(face_compressed,  cmap='gray')
```